Learning Planning Model for Semantic Process Compensation

Ahmad Alelaimat, Metta Santipuri, Yingzhi Gou, and Aditya Ghose

Decision Systems Lab, School of Computing and Information Technology University of Wollongong, Wollongong, NSW 2522 Australia
{aama963,ms804,yg4524,adity}@uow.edu.au

Abstract. Recent advancements in business process conformance analysis have shown that the detection of non-conformance states can be learned with discovering inconsistencies between process models and their historical execution logs, despite their real behaviour. A key challenge in managing business processes is compensating non-conformance states. The concentration of this work is on the hardest aspect of the challenge, where the process might be structurally conformant, but it does not achieve an effect conform to what is required by design. In this work, we propose learning and planning model to address the compensation of *semantically non-conformance states.* Our work departs from the integration of two well-known AI paradigms, Machine Learning (ML) and Automated Planning (AP). Learning model is divided into two models to address two planning problems: learning predictive model that provides the planner with the ability to respond to violation points during the execution of the process model, and instance-based learning model that provides the planer with a compensation based on the nearest class when there are no compensations perfectly fit to the violation point.

Key words: semantic process compensation, learning model, automated planning

1 Introduction

The problem of business process monitoring has received considerable recent attention in the literature. Much of the work done on process monitoring involves conformance checking, which seeks to ensure that the task sequence being executed is, in fact, a task sequence mandated by the operative process model. We shall refer to this conception of conformance as *structural conformance.* This paper builds on a more sophisticated notion of conformance *semantic conformance* [1] that seeks to ensure that the observed effects of a process at every step correspond to the expected post-conditions at those steps.

To provide comprehensive support for exception handling and run-time adaptation in executing process instances, this paper addresses the question of what can be done to fix non-conformant process instances. The notion of conformance used here is semantic non-conformance, but that notion subsumes structural non-conformance. When a process instance is found to be non-conformant, two possible strategies might be adopted: (1) aborting the process instance and starting again from scratch and (2) continuing the execution of the process instance by deploying an appropriate fix. The former strategy can be problematic, since some of the transactions involved might be impossible to roll back. Our focus, therefore, is on the latter strategy. We shall refer to the fix as a *compensation*, i.e., a suffix of the current task sequence that is distinct to the one originally mandated by the process design that (eventually) restores to the process instance to a conformant state.

We offer a novel technique for computing compensations in this paper. Computing a compensation can be viewed, in the first instance, as a planning problem. We know the current state of the process, and we also have a specification of the goals of the process (and hence, a goal state). The planning operators [2] are the enterprise capabilities that appear as tasks either in the currently deployed process design, or in other designs in the organizations process repository. The output generated by a planner will therefore be a task sequence that will restore the process to a conformant state, or, at the very least, a goal-satisfying state.

The planning problem is not as straightforward as the account above suggests. There are trade-offs involved in terms of choosing between compensations that achieve full goal-compliance but delayed restoration of conformance (i.e., the process executes for a period of time in a non-conformant fashion). There might be a gap between *violation time*, when the non-conformance is detected, and *compensation time*, when the compensation is deployed. This raises questions about the trade-off between deliberation and action.

The paper innovates further by viewing the computation of compensations as a *learning* problem [3]. Given a history of past executions, it is possible to learn from past instances of non-conformance the compensations that were deployed and how effective they were. The problem can be viewed, for instance, as an instance-based learning problem [4], where we search for the most similar past instance and then deploy the compensation used in that case.

In the reminder of this paper, we describe learning planning semantic process compensation, which extends the idea of semantic monitoring and compensation in socio-technical processes [1]. The rest of the paper is structured as follows. In section 2, we introduce some preliminaries. Section 3 represents learning planning semantic process compensation model. We describe the implementation and empirical evaluation of this model in section 4. Then, in section 5 we present some related literature about learning planning models and semantic process compensation. We conclude the work in section 5.

2 Preliminaries

This section introduces the key concepts used in the reminder of this paper. First, we introduce process model notations, then we outline annotated strategies for computing semantic process compensation [1].

Definition 1. A semantically annotated process model $\mathcal{P}$ is a process model in which each activity or event is associated with a set of effect scenarios. Each effect scenario *es* is a 4-tuple $\langle ID, S, Pre, Succ\rangle$, where S is a set of sentences in the background language, *ID* is a unique identification for each effect scenario, *Pre* is a set of IDs of effect scenarios that can be valid predecessors in $\mathcal{P}$ of the current effect scenario, while *Succ* is a set of IDs of effect scenarios that can be valid successors in $\mathcal{P}$ of the current effect scenario.

Normally, business process models are associated with a set of **normative traces** [1], each normative trace *nt* represents one possible way in which the process might be executed. However, the actual execution of process models is not necessarily be normative. Thus, we introduce **semantic execution trace** to semantically annotate the execution of process model $\mathcal{P}$ at run time.

Definition 2. A normative trace *nt* is a sequence $\langle \tau_1, es_1, \tau_2, ...es_{n-1}, \tau_n, es_n\rangle$, where

- $es_i, ..., es_n$ are effect scenarios, and for each $es_i = \langle ID_i,\ S_i,\ Pre_i,\ Succ_i\rangle$, $i \in [2, .., n]$, it is always the case that $ID_{i-1} \in Pre_i$ and $ID_i \in Succ_{i-1}$;
- $es_n = \langle ID_n, S_n, Pre_n, \emptyset\rangle$ is the final effect scenario, normally associated with the end event of the process;
- $es_1 = \langle ID_1, S_1, \emptyset, Succ_1\rangle$ is the initial effect scenario, normally associated with the start event of the process;
- Each of $\tau_1, \tau_2, ..., \tau_n$ is either an event or an activity in the process.

We shall refer to the sequence $\langle \tau_1, \tau_2, ..., \tau_n\rangle$ as the identity of the trace *nt*.

To simplify the presentation later on, the *es* in the trace, from now, refers to S in the 4-tuple $\langle ID, S, Pre, Succ\rangle$ because *ID*, *Pre*, and *Succ* are meta information used only to construct normative traces.

Definition 3. A semantic execution trace of a process $\mathcal{P}$ is a sequence $et = \langle \tau_1, o_1, \tau_2, o_2, ..., \tau_m, o_m\rangle$, where each τ_i is either a task or an event, and o_i is a set of sentences in the background language that we shall refer to as an observation that describes the process context after each τ_i. We shall refer to the sequence $\langle \tau_1, \tau_2, ..., \tau_m\rangle$ as the identity of the execution trace.

Definition 4. Semantic non-conformance execution trace an execution trace $et = \langle \tau_1, o_1, ..., \tau_m, o_m\rangle$ is said to be **non-conformant** with respect to a semantically annotated process $\mathcal{P}$ if and only if any of the following hold: (1)

there exists an o_i in et such that for all normative traces $nt' = \langle \tau'_1, es_1, \tau'_2, ...\rangle$ for which the identity of $\langle \tau_1, o_1, ..., \tau_i, o_i\rangle$ is a prefix of its identity and $o_j \models es_j$ for each $j = 1, ..., i-1$, $o_i \not\models es_i$ (we shall refer to this as weak semantic non-conformance). (2) If we replace non-entailment with inconsistency in condition (1) above, i.e., $o_i \cup es_i \models \perp$, we obtain strong semantic non-conformance. In each case, we shall refer to τ_i as the violation point in the process.

Definition 5. Semantically compensated instance is a process instance $et = \langle \tau_1, o_1, ..., \tau_m, o_m\rangle$ will be referred to as a semantically compensated instance of a (semantically annotated) process $\mathcal{P}$ if there exist τ_i and τ_j in et, with $i < j$, such that τ_i is a violation point, and there exists a normative trace $nt = \langle \tau_1, es_1, \tau_2, ...es_{h1}, \tau_h, es_h, ..., \tau_n, es_n\rangle$ of $\mathcal{P}$ with an identity for which $\langle \tau_1, ..., \tau_{j-1}\rangle$ serves as a prefix, such that $o_k \models es_l$ for $k = j, ..., m$ and $l = h, ..., n$. As well, it must be the case that $o_m \models g$. We shall refer to τ_j as the compensation point. The compensation point must be a task and not an event.

Definition 6. A compensation given a semantically compensated process instance $et = \langle \tau_1, o_1, ..., \tau_m, o_m\rangle$ of $\mathcal{P}$ with a compensation point τ_j, a **compensation** is a process design $\mathcal{P}'$ for which the completion of τ_{j-1} serves as the start event and $\langle \tau_j, o_j, ..., \tau_m, o_m\rangle$ is a valid normative trace. Every normative trace associated with $\mathcal{P}'$ must end in an effect scenario es such that $es \models g$, where g is the goal associated with the original process $\mathcal{P}$.

3 Learning Planning Semantic Process Compensation

Constructing semantically compensated instance with learning planning model is described in term of data mining. This model aims to describe a way in which process model returns to a semantically conformant state after the occurrence of a violation point. The first part of the learning model is compensation description algorithm, where a semantic solution suggested to fix semantic non-conformance state. Given a process execution log, normative trace, and execution trace holds a violation point, compensation description algorithm will be able to produce compensated process instances. Compensation description algorithm generates compensated process instances based on three features: execution violation point, normative desired effect, and goal associated with the original process $\mathcal{P}$.

The learning model is divided into two models to address two planning problems. The first problem is prediction problem [5], where predictive model provides the planner with the ability to respond to violation point during the execution of the process model [6]. In the predictive model, the predicted target is an instance; one way in which the detected violation point might be compensated. The second problem is instance-based problem [4], where instance-based learning model provides the planner with experiences that are solved with the same compensation, thereby violation points can be compensated based on their classification. The reason behinds using instance-based learning model is to provide planers with a compensation based on the nearest class when there are no compensations fit exactly to the violation point. Figure 1 shows the detailed framework of learning planning semantic process compensation .

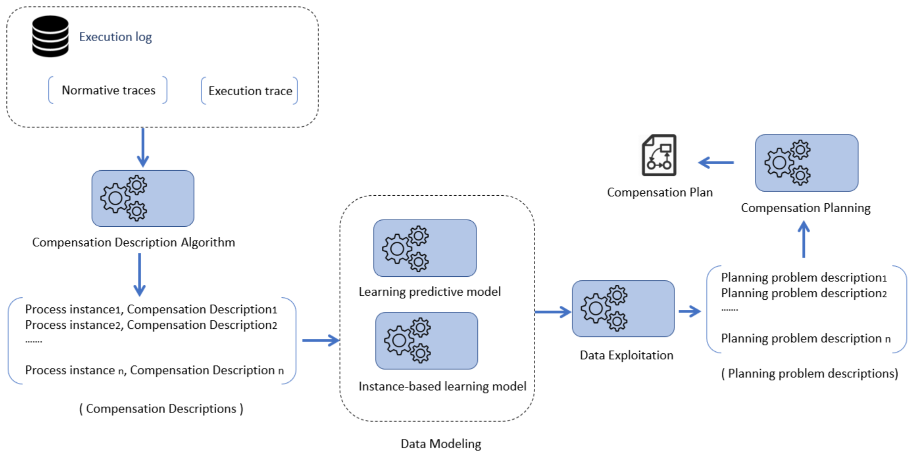
Execution log
Normative traces
Execution trace
Compensation Description Algorithm
Process instance1, Compensation Description1
Process instance2, Compensation Description2
....
Process instance n, Compensation Description n
(Compensation Descriptions)
Learning predictive model
Instance-based learning model
Data Modeling
Data Exploitation
Compensation Plan
Compensation Planning
Planning problem description1
Planning problem description2
....
Planning problem description n
(Planning problem descriptions)

Fig. 1. Learning planning semantic process compensation model

In figure 1, process execution log, normative traces and execution log are model inputs. Compensation description algorithm takes the role of selecting relevant features [7] for data modeling which are violation point, compensation point, and process association goal. The output of compensation description algorithm is a set of descriptions that illustrate potential fixes of the detected violation point. Data modeling generates a prediction or classification classes based on selecting relevant criteria. When all was said and done, automated planning befits from the learned knowledge through the using of exploitation learned knowledge algorithm, where it relies on employing the learned knowledge in planning problem description.

3.1 Compensation Description

Compensation description algorithm is an algorithm-based software that has been designed to describe semantic compensation instances. Compensation description algorithm takes execution log, normative traces, execution trace, and a goal associated with the original process $\mathcal{P}$ as inputs. Standing on definition 5 and 6, compensation description algorithm produces semantic process compensation. Algorithm outputs illusteat selecting relevant features [7], where the output might be singular task or sequence of tasks that describes how to execute the rest of the process model in which such violation point can be compensated.

Compensation Description Algorithm

```
1: $EL \leftarrow$ execution_log Data set
2: $nt \leftarrow$ normative_trace Array
3: $et \leftarrow$ semantic_execution_trace Array
4: $de \leftarrow$ desired_effect String
5: $g \leftarrow$ associated_goal String
6: $V_p \leftarrow$ violation_point String
7: for j=1 to size(nt)
8: if $similar(V_p, nt[j]) == 1$ then
9:     $de \leftarrow nt[j+1]$
10: end for
11: for k=1 to size(EL)
12: if (EL[k,end] != g) then
13:     continue
14: if $ismember(EL(k), et) == 1$ && $ismember(EL(k), de) == 1$ then
15:     $Compensation \leftarrow EL(k, index(V_p) : index(end))$
16:     else
17:     Print(No compensation found)
18: end for
```

Compensation description algorithm starts with discovering an $es \in nt$ that serves as a desired effect to the violation point (τ_i, o_i). A semantically compensated process instance is an instance that holds an observation entails g, (τ_i, o_i) such violation point and observation such the desired effect. When relevant process instance found, compensation description algorithm starts recording all activities positioned between (τ_i, o_i) and g. Thus, compensation might be singular task or sequence of tasks. Compensation description algorithm can be seen as a pre-processing phase, where each description will be used as nominal class [8] appended at the end of its instance, that way learning models are able to learn only from relevant experiences.

3.2 Data Modeling

Data modeling is divided into two models to address two planning problems. First problem is prediction problem, where the learning predictive model provides the planner with the ability to respond to violation points during the execution of the process model [6]. In the predictive model, the predicted compensation is one way in which the violation point might be compensated. The second problem is instance-based problem, where instance-based learning model provides the planner with experiences that are solved with the same compensation, thereby violation points can be compensated based on their classification.

The predictive model is a decision tree created using J48 prediction algorithm [5]. In an abstract sense, a compensation can be seen as a prediction of a singular task or a sequence of tasks that returns the process model to semantically conformant state. After the occurrence of a violation point, J48 prediction algorithm predicts a compensation based on compensation description algorithm.

Instance-based learning model is a description of instances generality created using IBK classification algorithm [4]. The reason behind selecting IBK is to design a learning model that is able to provide planers with a compensation based on the nearest class when there are no compensations fit exactly to the violation point. modeling of J48 prediction algorithm and IBK classifier has been implemented using Waikato Environment for Knowledge Analysis (WEKA) [9]

3.3 Data Exploitation

Exploitation of the learned knowledge [3] can be leveraged in two orientations: (1) an execution trace that has process compensation instance in *EL* can be planned using the predictive model. (2) an execution trace that has no process compensation instance in *EL* can be planned using instance-based learning model. The following shows exploitation of the learned knowledge algorithm.

Exploitation of the learned knowledge

```
1: (:initial ← violation point
2: predicted compensation ← predicted compensation based on J48
3: nearest compensation ← nearest compensation based on IBK
4: if (∃ compensation ∈ EL | compensation is relevant to (: initial ) then
5:     (:goal ← predicted compensation
6:     else
7:     (:goal ← nearest compensation
```

In term of automated planning [2], semantic process compensation problem description consists of `(:init` state that represents the structural design of the process and the violation point, and `(:goal` state represents what is the fact that we would to be true.

3.4 Semantic Process Compensation Planning

In the context of automated planning represention, exploited knowledge has been achived using Planning Domain Definition Language (PDDL) [10]. PDDL domain has been designed based on the logic of Petri-net [11]. In an abstract sense, the `EXECUTE` of τ_i enables the transition of data flow from the current `Event` into an `Event` satisfies both `Output function` and `Input function`. In PDDL problem domain, detecting `(:initial` state and reasoning about `(:goal` state are considered in exploitation of the learned knowledge. Figure 2 shows PDDL representation for action `EXECUTE` from emergency department process example [12].

```
(:action EXECUTE
 :parameters (?exe - Task ?eve - Event)
 :precondition (forall (?e - Event)
                 (imply (input_function ?exe ?e)(> (Patient_at ?e) 0)))
 :effect (and (forall (?e - Event)
             (when (input_function ?exe ?e)(decrease (Patient_at ?e) 1)))
             (forall (?e - Event) (when (output_function ?exe ?e)
             (increase (Patient_at ?e) 1)))
```

Fig 2: PDDL representation for `EXECUTE` action

In order to solve semantic process planning problem, off-the-shelve domain independent planner has been used. SGPlan6 planning system [13] used to solve the problem domain shown in the running example (section 4.1) through the plan shown in figure 2.

4 Implementation and Evaluation

In this section, we outline an implementation of learning planning semantic process model described previously and present empirical results. The implementation of proposed model starts with compensation description, running in Matlab. It is useful to note, that we omit some details but these can be found in [12]. On the other hand, we use a machinery to semantically simulate process instances. After compensation description, process instances is tagged with a tag that represents nominal class (i.e., discrete class) [8], in which it serves as a target for prediction and classification.

As indicated in advance, modeling of J48 prediction algorithm and IBK classifier has been implemented using WEKA [9]. Predictive and instance-based models take tagged process instances as an input. Learning predictive model employed after the generation of compensation descriptions based on compensation description algorithm (see section 3.1), while instance-based mode employed when we need to capture the nearest way in which such violation point could be compensated. In term of relevance measurement, the ideal k-*nearest neighbors* is *k=3*.

In term of automated planning, we used PDDL to illustrate the running example in the following section. Planning starts with an off-the-shelf planner to plan a compensation for 5 randomly-chosen process instances base on compensation description algorithm and learning predictive model. In order to solve the given problem, SGPlan6 [13] has been used.

4.1 Running Example

Figure 3 illustrates a process from health care domain. The figure exemplifies the motivation of learning planing semantic process compensation. In a process model taken by [12], at a hospital equipped with a process-aware information system, when patients arrive they assigned a triage priority (i.e., an assignment of urgency degrees), registered and then assigned to a responsible nurse. The assigned nurse checks patient condition in parallel, but not simultaneous, with doctor visit, X-Ray and then Final visit. In the proposed example, there are set of possible observation of (check, X-Ray, visit, final visit) in which patient condition is represented and accordingly appropriate treatment.

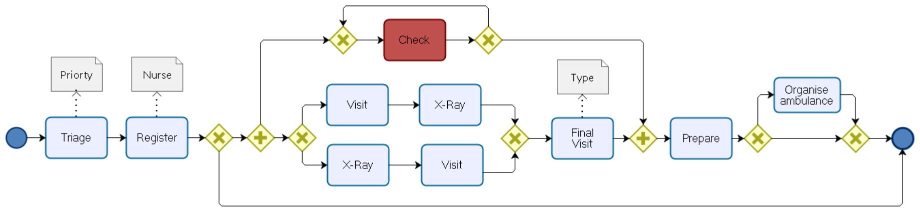

Fig 3: A BPMN notation for emergency department process

For instance, a violation point appears when *observation(visit)= Patient blood pressure expected readings are lower* at execution trace holds *observation(check)= Patient pressure check reveals elevated readings.* Table 3 represents a fragment of execution trace terminated after the occurrence of a violation point.

execution trace			*observations*	
case	*activity*	*timesatmps*	*observation*	*timestamps*
1000	*Triage*	*02:17:00*	*Patient urgency degree is orange*	*02:19:00*
1000	*Register*	*02:25:00*	*Patient registered*	*02:28:00*
1000	*Visit*	*02:47:00*	*Patient blood pressure expected readings are lower*	*02:50:00*
1000	*Check*	*02:53:00*	*Patient pressure check reveals elevated readings*	*02:56:00*

Table 1. An execution trace of emergence department process holds a violation point

In table 1, despite the structure of the execution trace until the violation point conforms to process model and vice versa, but it does not semantically. Non-conformance states might be much complicated and require deep models to detect them such as [14] and [15].

4.2 Learning Planning Model Evaluation

In this section, we aim to establish that the proposed model able to generate reliable throughput. Evaluation of learning model is helpful in achieving the following:

1. An accurate description of process instance compensations through compensation description algorithm.
2. A correct prediction of target variables based on learning predictive model.

3. An efficient generalization of process instances using instance-based learning model.

For learning model, we considered a synthetic process log consistences of 1000 instances. In table 2, compensation description, learning predictive model, and instance-based learning model performance measures are illustrated.

Compensation Description		*Learning Predictive Model*		*Instance-based learning model*	
# of instances	*1000*	*Correctly Classified*	*995*	*# of nearest neighbors*	*3*
Precision	*0.974*	*Precision*	*0.997*	*Precision*	*0.997*
Recall	*1.00*	*Recall*	*0.995*	*Recall*	*0.995*
F-measure	*0.986*	*F-measure*	*0.995*	*F-measure*	*0.995*

Table 2. Learning model evaluation

In term of semantic process compensation planning, we evaluated five randomly-selected process instances, where they supplied first as a test set to the prediction algorithm. For comparison, we included two evaluation attribute: the number of required actions to reach to the compensation point and planning time.

Table 3 represents a modest evaluation. An off-the-shelf classical planner used to generate compensation plans according to five different scenarios. The right-most column shows the required time to compute the plan. Computing number of actions is important to identify where the earliest compensation is possible [1].

Process Instance ID	*Process compensation plan*	*# of actions*	*planning time*
1	*0.001: (EXECUTE X-RAY 2) [1]* *1.002: (EXECUTE VISIT 2) [1]* *2.003: (EXECUTE FINAL VISIT) [1]* *3.004: (EXECUTE PREPARE) [1]*	*4*	*0.019*
2	*0.001: (EXECUTE CHECK) [1]* *1.002: (EXECUTE X-RAY 1) [1]* *2.003: (EXECUTE VISIT 1) [1]* *3.004: (EXECUTE FINAL VISIT) [1]* *4.005: (EXECUTE PREPARE) [1]* *5.006: (EXECUTE ORGANIZE AMBULANCE) [1]*	*6*	*0.017*
3	*1.002: (EXECUTE X-RAY 1) [1]* *2.003: (EXECUTE CHECK) [1]* *3.004: (EXECUTE VISIT 1) [1]* *4.005: (EXECUTE FINAL VISIT) [1]* *5.006: (EXECUTE PREPARE) [1]*	*5*	*0.015*

4	*0.001: (EXECUTE FINAL_VISIT) [1]* *1.002: (EXECUTE PREPARE) [1]* *2.003: (EXECUTE ORGANIZE AMBULANCE) [1]*	*3*	*0.014*
5	*0.001: (EXECUTE CHECK) [1]* *1.002: (EXECUTE FINAL_VISIT) [1]* *2.003: (EXECUTE PREPARE) [1]*	*3*	*0.015*

Table 3. compensation plans of five randomly selected process instances

The evaluation of learning model shows that: compensation description algorithm able to produce an accurate description of semantic process compensation, learning predictive model is able to predict correct target variables, and instance-based learning model is able to generalize process instances correctly. The results obtained from the planning model are reasonable and encouraging. As a result, learning planning semantic process model is able to compute an accurate and correct compensation plan. Moreover, it beneficial in computing where the earliest compensation is possible.

5 Related Work

As far as we know, there are no literature illustrated the use of learning planning models as an aid for semantic process compensation. Thus, related work is divided into two subsections: learning planning models and semantic process compensation.

5.1 Learning Planning Models

The nearest research of departure for our work is learning planning portfolio [16], this model uses two shapes of machine learning: classification model (J48 decision tree) to solve the selection strategy based on planner ability to solve the problem, and classification model (IBK) to find the required time to compute the best plan. In [17], case-based planning approach for retrieve planning cases based on heuristically matching function is proposed, where similar reuse candidates can be chosen from plan libraries to solve similar planning problems in the future. Different from [16] and [17], compensation description algorithm reduces learning cost through allowing the learner to learn only from relevant experiences. The model taken in [18], provides rational learning to capture suitable action in different planing domains. The relational decision tree used as a guidance for ordering node evolutions which helps in limiting search tree, such guidance improves planner performance through controlling search knowledge. In [19], an architecture for integrating planning execution and learning (PELA) is presented, where PELA states the learning task with upgrading PDDL domain model which is executed initially with no prior sense of real life uncertainty. Relational learning task represents action performance patterns that can be compiled based on metric or probabilistic representation. When a decision has to be made, our model considers not only predictions, but also classifications.

5.2 Semantic Process Compensation

Learning planning semantic process compensation is strongly inspired by semantic monitoring and compensation [1]. The proposed approach in [1] introduces semantically annotated solution to detect and compensate semantically non-conformant state in socio-technical processes. In [20], compensation orchestrating for the semantics of long-running transaction is proposed. On the other side, [21] propose a framework for web services error-handling choreography. Many literature discussed semantic model checking. For example, [14] introduce semantic model checking algorithm to reason about web services behavior. In a similar way, [15] present semantic model checking for discovering bugs in cloud systems. Our approach is an assistance to these approaches, because learning past compensations allows to obtain effective plans to compensate semantically non-conformant states.

6 Conclusion

This research represents two primary contribution. First, we designed an algorithm to select relevant features that helps learning model to discover potential compensations. Second, we showed how to exploit and employ learned knowledge for planning semantic process compensations. We have shown that learning planning model can be competitive with state-of-the-art process compensation models. As far as we know, no prior learning planning model has employed to handle semantic process compensation issue. A key challenge in applying semantic process compensations based on learning planning model is to accurately deal with choosing a robust fix among available compensations. One natural extension to the semantic process compensation introduced in this research is to consider the trade-off between compensation search-time and tolerable delays in terms of choosing between compensations.

References

1. Gou, Y., Ghose, A., Chang, C.F., Dam, H.K. and Miller, A.,: Semantic monitoring and compensation in socio-technical processes. In International Conference on Conceptual Modeling (pp. 117-126). Springer, Cham (2014)
2. Ghallab, M., Nau, D. and Traverso, P.: Automated Planning: theory and practice. Elsevier (2004)

3. Jimnez, S., De la Rosa, T., Fernndez, S., Fernndez, F. and Borrajo, D.: A review of machine learning for automated planning. The Knowledge Engineering Review, 27(4), pp.433-467 (2012)
4. Aha, D.W., Kibler, D. and Albert, M.K.: Instance-based learning algorithms. Machine learning, 6(1), pp.37-66 (1991)
5. Quinlan, J.R.: Learning with continuous classes. In 5th Australian joint conference on artificial intelligence. 92, 343-348 (1992)
6. Weber, B.G., Mateas, M. and Jhala, A.: Learning from Demonstration for Goal-Driven Autonomy. In AAAI (2012)
7. Blum, A.L. and Langley, P.: Selection of relevant features and examples in machine learning. Artificial intelligence. 97(1), 245-271(1997)
8. Kirkby, R., Frank, E. and Reutemann, P.: Weka explorer user guide for version 3-5-8. University of Waikato (2007)
9. Witten, I.H., Frank, E., Hall, M.A. and Pal, C.J.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2016)
10. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D. and Wilkins, D.: PDDL-the planning domain definition language (1998)
11. Peterson, J.L.: Petri net theory and the modeling of systems (1981)
12. Mannhardt, F., de Leoni, M., Reijers, H.A. and van der Aalst, W.M.: June. Data-Driven Process Discovery-Revealing Conditional Infrequent Behavior from Event Logs. In International Conference on Advanced Information Systems Engineering (pp. 545-560). Springer, Cham (2017)
13. Hsu, C.W. and Wah, B.W.: September. The SGPlan planning system in IPC-6. In Proceedings of IPC (2008)
14. Di Pietro, I., Pagliarecci, F. and Spalazzi, L.: Model checking semantically annotated services. IEEE Transactions on software engineering, 38(3), pp.592-608 (2012)
15. Leesatapornwongsa, T., Hao, M., Joshi, P., Lukman, J.F. and Gunawi, H.S.: SAMC: Semantic-Aware Model Checking for Fast Discovery of Deep Bugs in Cloud Systems. In OSDI , 399-414 (2014)
16. Cenamor, I., De La Rosa, T. and Fernndez, F., 2013, June. Learning predictive models to configure planning portfolios. In Proceedings of the 4th workshop on Planning and Learning. ICAPS-PAL, 14-22 (2013)
17. Serina, I.: Kernel functions for case-based planning. Artificial Intelligence, 174(16), 1369-1406 (2010)
18. De La Rosa, T., Celorrio, S.J. and Borrajo, D.: Learning Relational Decision Trees for Guiding Heuristic Planning. In ICAPS, 60-67 (2008)
19. Jimnez, S., Fernndez, F. and Borrajo, D.: The PELA architecture: integrating planning and learning to improve execution. In National Conference on Artificial Intelligence (2008)
20. Butler, M., Hoare, T. and Ferreira, C.: A trace semantics for long-running transactions. In Communicating Sequential Processes. The First 25 Years (pp. 133-150). Springer Berlin Heidelberg (2005)
21. Mazzara, M. and Lucchi, R.: A framework for generic error handling in business processes. Electronic Notes in Theoretical Computer Science, 105, 133-145 (2004)