Compliance Checking for Decision-Aware Process Models

Stephan Haarmann(✉), Kimon Batoulis, and Mathias Weske

Hasso Plattner Institute, University of Potsdam,
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
{stephan.haarmann,kimon.batoulis,mathias.weske}@hpi.de
https://bpt.hpi.uni-potsdam.de/

Abstract. The business processes of an organization are often required to comply with domain-specific regulations. Such regulations can be checked based on the models of the respective processes. These models' main focus is on the operational part of the process. However, also decisions play a major role in the execution behavior of processes, and they are expressed in separate decision models. In this paper, we investigate the influence of decision models on business process compliance checking. To this end, we formalize decision-aware processes as colored Petri nets, extract the state space, and check compliance rules using temporal logic model checking. The approach improves the quality of existing compliance checking by reducing the risk of false negatives. We provide a prototype and discuss advantages and disadvantages.

Keywords: Business process management · Business process compliance · Decisions

1 Introduction

Business process compliance is the topic of ensuring that an organization's business processes comply with internal and external domain-specific regulations. Such regulations often refer to the sequence of certain activities that must or must not occur. Given that the organization documented its processes in respective models, it is possible to uncover potential violations of the regulations at design time already to ensure a compliant execution of the process.

Compliance checking for business process models has been given a lot of attention in the literature in recent years [1–3,18,19]. These approaches focus on process models specifying aspects such as control flow and high level data dependencies, which may be subject to internal and external domain-specific regulations. However, particular instances of these processes often depend on additional decision logic defining fine-grained data dependencies, which are not specified in the process model, but in a separate decision model.

This paper presents a semi-automated approach to design-time compliance checking of decision-aware process models. To this end, we formally capture the

F. Daniel et al. (Eds.): BPM 2018 Workshops, LNBIP 342, pp. 494–506, 2019.
https://doi.org/10.1007/978-3-030-11641-5_39

execution semantics of decision-aware processes (i.e., the logic of the process and the decision as well as the data dependencies between the two). Subsequently, we show that considering decisions increases the size of the processes' state space by adding more information to the data objects. However, at the same time, the amount of traces is limited, since there may be interdependencies between decisions that rule out certain traces. Therefore, compliance checking with decision-aware processes may actually lead to more accurate results since in the decision-unaware process rules were violated by traces that could actually never occur.

The remainder of this paper is structured as follows. Section 2 presents related work. In Sect. 3, we provide definitions of the structures used and an example. The paper presents the decision-aware compliance checking approach in Sect. 4. We evaluate the approach using a prototypical implementation in Sect. 4.4. Finally, Sect. 5 concludes the paper and discusses future work.

2 Related Work

Business Process Compliance received increasing attention of BPM research from 2000 to 2007 and is still actively researched [15]. Recently, Hashmi, Governatori, and Lam summarized the developments and gave possible directions for future work in a survey paper [15]. They distinguish between design-time, run-time, and auditing approaches—based on their application during the process life cycle. This paper contains a design-time approach that checks models for potential violations using a model checking approach [4,11]. It addresses an open issue [15]: consideration of activities' effects.

Various approaches for design-time compliance checking have been developed. Awad et al. introduce *BPM-Q* (and its visual counterpart *BPMN-Q*) to formally (and visually) model queries for process models by reusing BPMN elements and annotating them with additional information [1]. Later they used it for modeling compliance rules [2], and they added data support [3]. The compliance rules are formalized using temporal logic and checked with the model checker NuSMV[1]. Awad et al. only investigate compliance rules based on control-flow relationships and data; however, data based rules can only constrain the state of the data object.

In contrast, *(Extended) Compliance Rule Graphs (eCRGs)* are capable of expressing fine grained data conditions and additional perspectives such as time and resources [19,20,26]. While most eCRG based approaches are used for run-time or auditing compliance approaches, Knuplesch et al. present an approach for checking data-aware rules on process models [18], which is strongly related to this paper. In contrast to Awad, fine grained data conditions can be evaluated. Knuplesch et al. infer respective knowledge from arc-conditions and derive *abstraction predicates* (contraints for possible values) for all data attributes.

[1] NuSMV's web page: http://nusmv.fbk.eu/ (retrieved 4/10/2018).

They embedded their approach in the SeaFlow compliance checker, which models rules as *Compliance Rule Graphs (CRGs)* [23]. However, our method additionally considers decision models and supports operations on data.

The approach of this work uses a *colored Petri net (CPN)* based formalism for process models. The translational semantics are based on Dijkman's et al. approach of mapping processes to Petri nets [13] and Lee's et al. method for modeling decisions as Petri nets [22]. Instead of classical Petri nets, we use CPNs [17]. CPNs have been used for analyzing data aware processes in [27]. In our paper, CPNTools[2] is used for implementation: the formalization, the state space extraction, and the compliance checking using the ASK-CTL extension [10].

Decisions and decision models receive increasing attention from BPM research. Recently Jansen et al. and Batoulis et al. defined criteria for consistent integration of process and decision models [7,16]. Further, Batoulis et al. investigates soundness notions for decision-aware processes and thereby domain independent correctness criteria [6,8]. Compliance is based on domain specific rules. Therefore, our approach complements existing correctness criteria for decision-aware processes.

3 Foundations

A decision-aware process model contains imperative and declarative parts of a business process. The imperative parts are captured by a traditional process model (e.g., a BPMN model [24]) while the declarative parts are captured by decision models (e.g., a DMN model [25]). Process models link decision models through decision tasks, which refer to a decision in a decision model. This section contains a description of decision-aware processes, an running example, and a brief description of compliance checking.

3.1 Decision-Aware Process Models

A business process consists of a set of tasks that contribute to a common business goal and are executed in a technical and organizational environment [28]. A process model describes these tasks and their temporal and causal dependencies. Further, it has one start event and one end event. The model contains gateways to express exclusiveness (XOR gateways) and concurrency (AND gateways) of tasks. Additional dependencies can be expressed by using data objects and data flow: an activity can read data objects in specific states and write data objects in specific states. Each activity has, therefore, a set of input sets and a set of output sets. At least one input set must be available to enable the activity. One output set is chosen and its elements are written by the activity.

Consider the sample process model in Fig. 1. It depicts the inquire process of a car rental company. If an order is received, then the company automatically checks if discounts apply and grants them. Afterwards, additional fees are

[2] CPNTools' web page: http://cpntools.org (accessed 4/10/2018).

determined and calculated. Eventually, the final price is set and the updated order is sent to the customer. Throughout the process, various data objects are used: *Offer* comprises the key information, *Special Offer* contains information about potential discounts, *Fees* contains the determined fees which are saved in separate objects when calculated (*Young Driver Fee* and *Last Minute Fee*).

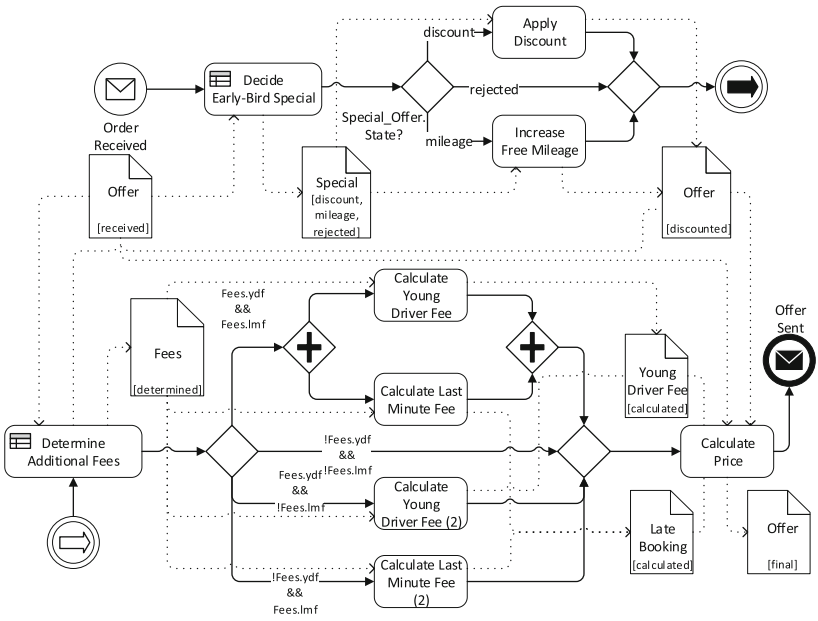


Fig. 1. Sample process model (BPMN) of a car rental company

A decision model consists of two layers: the decision requirements and the decision logic. The former comprises high-level information about the necessary inputs the data and the preceding decisions that are required to execute a certain decision. The logic level contains a specification of how decisions are made. These can be informal or formal. We assume that all decisions are formally specified by a decision table. A decision table comprises a set of rules, which consist of conditions for the inputs and expressions for producing outputs. Although rules can, in general, be overlapping, we only consider unique decision tables where only one rule matches an input. For unique tables, the order of rules is irrelevant. It has been shown, that all DMN decision tables can be transformed into unique ones [9].

The car rental scenario contains two decisions, the logic of which is given by Table 1a and b. A rule is represented horizontally. The first decision (Table 1a) has the input *Offer.Lead Time*, which is the time in weeks between booking and picking the rental up. The output is either *rejected* (no discount), *mileage* (increased free mileage), or *discount* (monetary discount). The decision

in Table 1b considers the lead time and the driver's age to determine the fees: a driver younger than 25 must pay a young driver's fee. If the booking occurs less than one week in advance, then a last minute fee is due.

Table 1. Decision tables for the sample process

(a) Decision logic for *Decide Early-Bird Special*

U	Offer.Lead Time	Special Offer.State
	Number	{*rejected, discount, mileage*}
1	< 2	rejected
2	$[2..4)$	mileage
3	≥ 4	discount

(b) Decision logic for *Determine Additional Fees*

U	*Offer*		*Fees*	
	Lead Time	Driver Age	Young Driver	Last Minute
	Number	$[18..99]$	*bool*	*bool*
1	< 1	< 25	true	true
2	≥ 1	< 25	true	false
3	< 1	≥ 25	false	true
4	≥ 1	≥ 25	false	false

Decisions are linked to processes via decision tasks. Whenever a decision task is reached, the process provides the required inputs to the decision, the logic is executed, and the process handles the decision output. We make the following assumption about the input-output behavior of decision tasks: the inputs of the decision task directly correspond to the inputs of the linked decision table. The decision's outputs are reflected in the task's output sets. If a decision task has multiple output sets, the decision logic chooses an output set. To do so, the attribute *State* is set. The sample process model has two decision tasks linking the respective decisions. *Decide Early-Bird Special* links to Table 1a so that the decision determines the task's output set. *Determine Additional Fees* refers to Table 1b. The decision sets attributes of the only output *Fees*.

3.2 Compliance Checking

A decision-aware process model is a blueprint for process instances: it describes the possible behavior. The model structures the process and constrains it (e.g., limits the order of activities). The process can be implemented, for example by using a process engine, to support and control instances. However, real world process instances are subject to laws, guidelines, and regulations, which might or might not be captured in the process model. Violating these constraints can carry penalties and jurisdictional consequences. Thus, it is important to assert compliant behavior.

One step towards business process compliance is the verification of process models with respect to compliance regulations. The compliance regulations are expressed as so called compliance rules (properties that must not be violated). Table 2 contains rules for the car rental process. In general, we consider the occurrence and order of activities (rules c1, c2, c3) and might use data conditions for further restrictions (c4, c5).

Table 2. Compliance rules for the example process

c1 Every received order will eventually be sent back to the customer
c2 The price will only be calculated after additional fees have been determined
c3 If a discount applies, the order must not be subject to a last minute fee
c4 The young driver fee must be calculated if the driver is younger than 25
c5 If an early-bird special applies, calculate price reads the offer in state *discounted*

4 Decision-Aware Compliance Checking

Compliance checking can take place during different phases of the business process lifecycle. The decision-aware compliance checking approach of this paper takes place during design-time, i.e., models are checked for compliance violations. Although different methods for verifying models (e.g., theorem-proofing and simulation) exist [15], *model checking* is the most common one for compliance checking [15]. Therefore, our approach follows the general model checking paradigm depicted in Fig. 2 [21]. The decision-aware process model describes the system and defines a state space. We use colored Petri nets (CPNs) to assign formal behavioral semantics to such models. The compliance rules are properties that must not be violated, and we formalize them as *Computational Tree Logic (CTL)* formulas. The formal rules and the formal model are then consumed by a model checker, which verifies the properties.

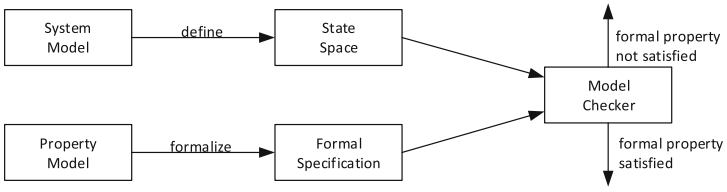


Fig. 2. Schematic description of the general model checking approach (cf. [21])

4.1 Requirements and Challenges for Formalizing Decision-Aware Process Models

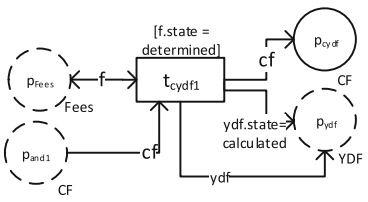
A formalism for a decision-aware process comprises the behavior of the process as well as the logic of the decisions, which is why we chose CPNs for this task. CPNs can model conditions and operations on data. Since we assume decision tables to be unique—i.e., its rules are non-overlapping—the set of rules of a table extensionally define a function, mapping an input to exactly one output. Therefore, decisions are just data operations. Further, the structure and temporal and casual constraints of process models can be captured in a (colored) Petri net [13]. Consequently, CPNs are suited for formalizing decision-aware process models.

To model check a Petri net for compliance rules, one needs to investigate its state space which in case of Petri nets is called occurrence graph. This graph must be complete in order to find all possible violations of a compliance rule, i.e., *every* possible trace has to be represented. However, decisions operate on data attributes, which may have large domains (such as the integers). Representing every possible instance explicitly leads to large state spaces (i.e., $2^{32} \times |states|$, if a single integer is involved). Thus, model checking can become infeasible and a proper abstraction is required.

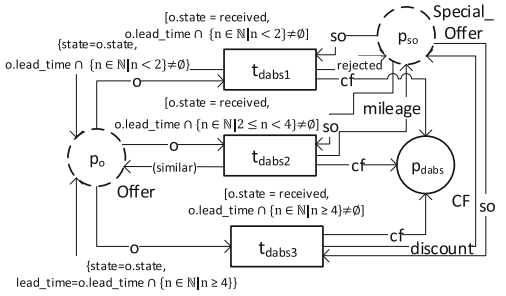
Some model checkers, such as NuSMV, support symbolic abstraction [12]. Instead of considering each possible instance separately, an attribute is represented by a symbol (i.e., *Offer.Lead_Time*$= \mathbb{N}$). Whenever the state space extraction reaches a condition or operation, it is applied to the abstract symbol (i.e., *Offer.Lead_Time*$= \{n \in \mathbb{N} | n < 2\}$ after executing the first rule the decision). If alternative conditions exist, the state space branches. Each branch considers one alternative [5]. But no symbolic model checker for CPNs exists. To overcome this, we incorporate the abstraction into our formalization and implemented required operations and conditions for symbolic execution [12].

4.2 Mapping Decision-Aware Process Models to CPNs for Symbolic Execution

This paper's mapping of decision-aware process models to CPNs builds upon the mapping of process models to Petri nets given by Dijkman et al. in [13]. In this section, we highlight major differences especially those caused by the data-awareness of CPNs and the use of symbolic abstraction. For one, a data object is represented by *exactly one* colored token on *exactly one* place. When the state of the object (or an attribute) is updated, the color of the token changes, but the location remains the same. A formal mapping is described in [14].



(a) Mapping for the task *calculate young driver fee (cydf)* writing the object *young driver fee (ydf)*



(b) Mapping for the decision task *decide early bird special* and the respective logic

Fig. 3. Mapping of a sample task and a sample decision task linking a decision table

An activity has a set of input sets and a set of output sets. Consequently, the chosen output set is reflected in the process state. For this reason, we map

an activity to a set of transitions—one for each output set. The precondition given by the input set is checked by the transitions' guards. Figure 3a shows the mapping of the task *calculate young driver fee*. It has exactly one input set (*Fees* in state *determined*) and one output set (*Young Driver Fee* in state *calculated*) and is therefore mapped to one transition.

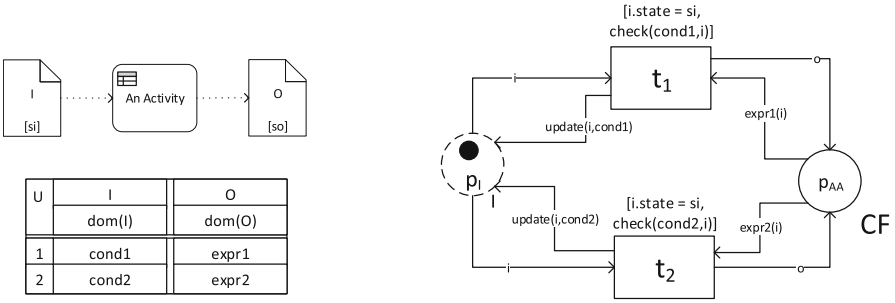


Fig. 4. Generic mapping of a decision task including the decision logic

The outcome of a decision task depends on the decision logic. At this point, we assume that the decision logic determines the output set and may also set other attributes. Figure 4 contains the generic mapping: we create a CPN transition for each row. The transitions can fire if the condition of the rule is fulfilled. Since data objects are described by symbols, it is enough if only one instance fulfills the condition. However, we need to update the input's symbolic abstraction respectively. We denote this by *update(i,cond)* where i is an input and *cond* the corresponding condition. Further, we update the symbols for the outputs according to output value of the corresponding rules (*expr(i)* where i is the input and *expr* the output expression).

Consider the decision task *Decide Early-Bird Special* and its corresponding decision table (Table 1a). The formalization is depicted in Fig. 3b: for the three rules we create three transitions. Each transition corresponds to one rule and has a respective guard. For the first rule, the symbol for *Offer.Lead_Time* must comprise at least one value that is less than two. If the transition fires, the token o for *Offer* is updated so that *Offer.Lead_Time* describes only the values less than two. Further, the output data objects are updated. For the first rule, the transition sets the state of *Special_Offer* to *rejected*.

Finally, also XOR splits are treated differently than in [13], namely similar to decision tasks: they are like decisions with no output. Thus, they must read all the data required for the branching and may update the respective symbols. In contrast to decision tasks, each transition created for the XOR split has a separate control flow place.

4.3 Compliance Checking for CPN Formalism

Since CPNs have formal semantics, it is possible to extract their state spaces, given that they are finite. In case of (colored) Petri nets, the state space corresponds to the net's occurrence graph. That is to say, the current state of the net is given by its current marking, and by firing a transition in the net a state transition in the state space is performed. Our approach checks the compliance of the decision-aware process by verifying temporal logic queries against the state space.

Compliance rules can originate in laws, guidelines, or regulations. Table 2 lists some compliance rules for the sample process, and Table 3 the corresponding CTL formulas. A rule can restrict the occurrence or order, and a rule can be conditional: its restriction must only be satisfied given certain data conditions. So, how does decision-awareness influence the compliance of a process model?

Decisions encode information about the data attributes, which are represented as abstract symbols in our CPN and accordingly in our state space. In decision-aware compliance checking, we consider these attributes; consequently, the state space grows (it is larger than the decision-independent one). However, decisions also encode instance-level dependencies. These dependencies restrict the possible traces of the process further to the existing control flow. For this reason, a decision-independent state space can have more traces than its decision-aware counterpart.

If compliance rules only constrain the occurrence and order of activities (with or without data conditions), each trace contributes to the result. If traces are removed, a rule that previously held will still hold, but a rule that did not hold must be reevaluated because all violations could be part of the removed traces. As an effect, decision-awareness can reduce the number of false negatives (compliance rules that are violated in the model, but not in reality).

For example, consider rule c3: if a discount applies, a last minute fee must not apply. The CTL formula says that in all traces, if *Apply Discount* is executed, *Calculate Last Minute Fee* must not be executed. However, if we ignore decisions it is impossible to infer whether the two XOR-splits are independent. Hence, we have to consider the trace in which $Special\_Offer.State = discount$ and *Fees.lmf=true*. In that trace, the rule is violated. If, however, we take decisions into consideration, this would imply that $Offer.Lead\_Time \geq 4$ and $Offer.Lead\_Time < 1$. This is a conflict, and such a trace is not part of the state space. Hence, the decision-aware process model is compliant to c3.

Furthermore, based on decision-aware processes, we are able to check rules based on data object attribute values (cf. rule c4). This was previously not possible, because the process model does not reference this attribute and every knowledge about its valuation is based on the decision model. Since our mapping of decision-aware processes includes decision models, we can now verify rules involving data attribute conditions. We only need to check if one value contained in the symbolic abstraction satisfies the data condition. For instance, the sample process is compliant to rule c4.

Table 3. Compliance rules for the sample process expressed as CTL formulas

c1	**AF**(end)
c2	**A**(**NOT**(Calculate Price) **U** Determine Additional Fees) ∨ **AG**(**NOT**(Calculate Price) ∧ **NOT**(Determine Additional Fees))
c3	**AG**(Apply Discount ⟹ **AG**(**NOT**(Calculate Last Minute Fee)))
c4	**AG**(Calculate Young Driver Fee ⟹ dataCondition(Offer.Driver Age < 25))
c5	**AG**((writes(Early Bird-Special,Special Offer, discount) ∨ writes(Early Bird-Special,Special Offer, mileage)) ⟹ **AF**(reads(Calculate Price,Offer,discounted)))

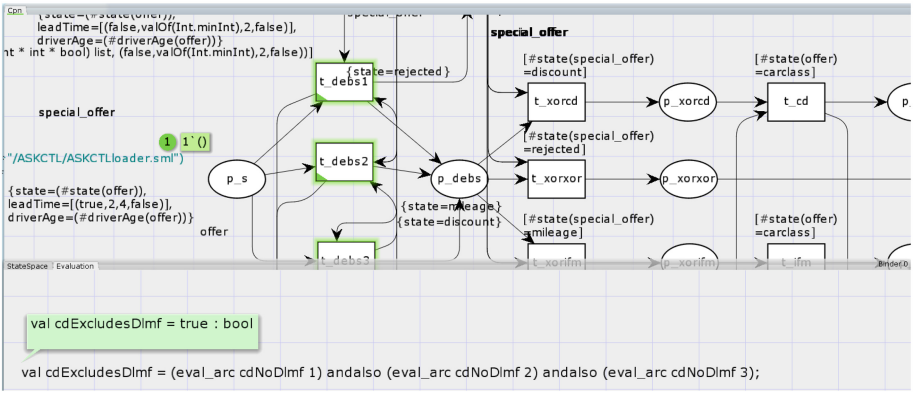


Fig. 5. Screenshot showing partly the colored Petri net and a compliance rule including the compliance checking result

4.4 Prototype

This paper's approach is a CPN based method for compliance checking of decision-aware processes. We applied this approach using CPNTools. Process models were manually translated to CPNs to extract the state space, and formal compliance rules were specified as ASK-CTL queries to be checked (cf. Fig. 5). ASK-CTL is an extension of CPNTools, that allows to evaluate CTL formulas on the state spaces of CPNs.

CPNTools does not support symbolic execution. Therefore, respective data types, comparisons, and operations need to be defined. We added the functionality for *int*, *bool*, and *real* to show the feasibility of the approach. Since large sets (such as *int*) cannot be represented explicitly, we use intervals to describe the current abstraction of a symbol. Examples (including compliance rules) are provided online[3].

5 Discussion and Conclusion

In comparison to other design-time compliance checking approaches, decision-aware compliance checking reduces the number of false alarms. Since decisions

[3] Example CPNs: https://owncloud.hpi.de/index.php/s/negAQyTLYPj45xH.

encode relationships that might not directly be visible in the process model, the possible traces are further restricted. If less traces exist, fewer violations of occurrence-based and order-based compliance rules can occur.

Although the knowledge about decision logic allows inferring attribute-level information on used data objects leading to more states, it is impossible that the decision-aware process model produces traces that are not part of the decision-independent model. Since we treat everything that is unknown using the open-world assumption, additional knowledge can be only equally restrictive or more restrictive. However, there are some edge-cases in which decision-aware compliance checking is too restrictive: violations can stay undiscovered if the environment changes the value between to occurrences in the process model (e.g., between a decision task and an XOR gateway).

Decision-awareness can also lead to problems in the model checking process. Translating a process model to a CPN that uses symbolic abstraction allows finding the right data abstraction during the state space extraction. In general, decision-aware process models are Turing-complete. As the execution of a Turing-complete program is undecidable, the symbolic execution is also undecidable. Research in symbolic abstraction presents methods (e.g., loop summarization) to support more models [5].

To summarize, this paper presents a decision-aware compliance checking approach. At design-time a process model and complementary decision-models are formalized as a CPN and model checking is applied to verify the model with respect to compliance rules. Thereby, symbolic abstraction is used to reduce the state space.

Tools, such as CPNTools, can model and analyze (e.g., apply model checking) to CPNs. However, it uses proprietary formats and requires expert knowledge. The manual formalization is an error-prone step. Future work should automate formalizing decision-aware process models and checking compliance, respectively.

CPNTools model checking extension provides only Boolean feedback. A rule holds or it is violated, but the cause of the violation is not exposed. Future work can overcome this by extending the model checking capabilities, using a different model checker, or integrating other approaches such as anti patterns. The latter finds all violating traces in a process model, which is a super set of the violations in a decision-aware setting [3].

Altogether, we showed that decision-aware compliance checking can improve the results compared to traditional design-time compliance checking.

References

1. Awad, A.: BPMN-Q: a language to query business processes. In: Reichert, M., Strecker, S., Turowski, K. (eds.) Enterprise Modelling and Information Systems Architectures - Concepts and Applications. Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2007), St. Goar, Germany, 8–9 October 2007. LNI, vol. P-119, pp. 115–128. GI (2007). http://subs.emis.de/LNI/Proceedings/Proceedings119/article1957.html

2. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85758-7_24
3. Awad, A., Weidlich, M., Weske, M.: Specification, verification and explanation of violation for data aware compliance rules. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC/ServiceWave -2009. LNCS, vol. 5900, pp. 500–515. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10383-4_37
4. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press, Cambridge (2008)
5. Baldoni, R., Coppa, E., D'Elia, D.C., Demetrescu, C., Finocchi, I.: A survey of symbolic execution techniques. CoRR (2016). http://arxiv.org/abs/1610.00502
6. Batoulis, K., Haarmann, S., Weske, M.: Various notions of soundness for decision-aware business processes. In: Mayr, H.C., Guizzardi, G., Ma, H., Pastor, O. (eds.) ER 2017. LNCS, vol. 10650, pp. 403–418. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69904-2_31
7. Batoulis, K., Meyer, A., Bazhenova, E., Decker, G., Weske, M.: Extracting decision logic from process models. In: Zdravkovic, J., Kirikova, M., Johannesson, P. (eds.) CAiSE 2015. LNCS, vol. 9097, pp. 349–366. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19069-3_22
8. Batoulis, K., Weske, M.: Soundness of decision-aware business processes. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNBIP, vol. 297, pp. 106–124. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65015-9_7
9. Batoulis, K., Weske, M.: Disambiguation of DMN decision tables. In: Abramowicz, W., Paschke, A. (eds.) BIS 2018. LNBIP, vol. 320, pp. 236–249. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93931-5_17
10. Cheng, A., Christensen, S., Mortensen, K.H.: Model checking coloured petri nets-exploiting strongly connected components. DAIMI Rep. Ser. **26**(519) (1997)
11. Clarke, E.M., Grumberg, O., Long, D.E.: Model checking and abstraction. In: Sethi, R. (ed.) Conference Record of the Nineteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Albuquerque, New Mexico, USA, 19–22 January 1992, pp. 342–354. ACM Press (1992)
12. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, Cambridge (2001). http://books.google.de/books?id=Nmc4wEaLXFEC
13. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Inf. Softw. Technol. **50**(12), 1281–1294 (2008)
14. Haarmann, S.: Decision-aware compliance checking. Master's thesis, University of Potsdam, March 2018. https://owncloud.hpi.de/index.php/s/GvZ3AJjdo6MJUr8
15. Hashmi, M., Governatori, G., Lam, H.P., Wynn, M.T.: Are we done with business process compliance: state-of-the-art and challenges ahead. Knowl. Inf. Syst. **57**, 79–133 (2018)
16. Janssens, L., Bazhenova, E., Smedt, J.D., Vanthienen, J., Denecker, M.: Consistent integration of decision (DMN) and process (BPMN) models. In: España, S., Ivanovic, M., Savic, M. (eds.) Proceedings of the CAiSE 2016 Forum, at the 28th International Conference on Advanced Information Systems Engineering (CAiSE 2016), Ljubljana, Slovenia, 13–17 June 2016. CEUR Workshop Proceedings, vol. 1612, pp. 121–128. CEUR-WS.org (2016). http://ceur-ws.org/Vol-1612/paper16.pdf
17. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer, Heidelberg (2009). https://doi.org/10.1007/b95112

18. Knuplesch, D., Ly, L.T., Rinderle-Ma, S., Pfeifer, H., Dadam, P.: On enabling data-aware compliance checking of business process models. In: Parsons, J., Saeki, M., Shoval, P., Woo, C., Wand, Y. (eds.) ER 2010. LNCS, vol. 6412, pp. 332–346. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16373-9_24
19. Knuplesch, D., Reichert, M.: A visual language for modeling multiple perspectives of business process compliance rules. Softw. Syst. Model. **16**(3), 715–736 (2017)
20. Knuplesch, D., Reichert, M., Ly, L.T., Kumar, A., Rinderle-Ma, S.: On the formal semantics of the extended compliance rule graph. Technical report, Ulm University (2013). http://dbis.eprints.uni-ulm.de/1147/1/TR_UIB_2013_05.pdf
21. Kunze, M., Weske, M.: Behavioural Models - From Modelling Finite Automata to Analysing Business Processes. Springer, Switzerland (2016). https://doi.org/10.1007/978-3-319-44960-9
22. Lee, S., O'Keefe, R.M.: Developing a strategy for expert system verification and validation. IEEE Trans. Syst. Man Cybern. **24**(4), 643–655 (1994)
23. Ly, L.T.: SeaFlows - a compliance checking framework for supporting the process lifecycle. Ph.D. thesis, University of Ulm (2013). http://vts.uni-ulm.de/docs/2013/8664/vts_8664_12857.pdf
24. OMG: Business process model and notation (BPMN), v2.0, January 2011. http://www.omg.org/spec/BPMN/2.0
25. OMG: Decision model and notation (DMN), v1.1, May 2016. http://www.omg.org/spec/DMN/1.1
26. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems: Challenges, Methods, Technologies. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30409-5
27. Wang, Z., Wang, J., Wen, L., Luo, G.: Formally modeling and analyzing data-centric workflow using WFCP-net and ASK-CTL. In: Zhang, R., Cordeiro, J., Li, X., Zhang, Z., Zhang, J. (eds.) ICEIS 2011 - Proceedings of the 13th International Conference on Enterprise Information Systems, Beijing, China, 8–11 June 2011, vol. 3, pp. 139–144. SciTePress (2011)
28. Weske, M.: Business Process Management: Concepts, Languages, Architectures, 2nd edn. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28616-2