

A Tool for Checking Soundness of Decision-Aware Business Processes

Kimon Batoulis and Mathias Weske

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
 {Kimon.Batoulis, Mathias.Weske}@hpi.de

Abstract. Based on the Decision Model and Notation (DMN), decision logic can be outsourced from process models to standardized decision models. This requires considerations of the sound integration of the two types of models. In this paper, we describe an open-source tool that checks the soundness of such decision-aware process models, inspired by the traditional workflow soundness notion. This includes ensuring that the process can continue after a decision has been taken, and that all activities following the decision can be executed. Our tool is scalable and its relevance has been established based on two empirical evaluations.

Keywords: BPMN, DMN, Soundness, Camunda

1 Introduction

The Business Process Model and Notation (BPMN) [5] is the industry standard for representing process models. This standard was complemented recently by a standard for modeling decisions: the Decision Model and Notation (DMN) [6]. Designing decisions in separate models associated with process models leads to the notion of *decision-aware process models* [7], separating the concerns of process and decision logic.

Consider the fragment of a business process about online advertisement in Fig. 1. This fragment begins with a BPMN business rule task, referred to as decision task

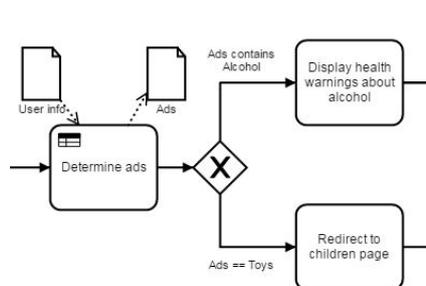


Fig. 1. Decision fragment of a process model calling a decision model

Determine ads			
adDecision			
R	Input +		Output +
	Age	Has Children	Ads
	age	hasChildren	ads
	integer	boolean	string
1	>= 18	true	Toys
2	> 12	-	Videogames
3	>= 18	-	Alcohol
4	< 18	-	Toys

Fig. 2. Decision table for determining a set of ads to serve. This table is associated with the decision task in Fig. 1.

from now on. Based on some user info the decision task determines a set of ads that are relevant for the customer. Afterwards, depending on the outcome of the decision,

measures such as displaying warnings about alcohol must be taken, or if only toys should be advertised the user is redirected to a designated webpage for children. We call these fragments of process models decision fragments, defined by their structure of beginning with a decision task, followed by a split gateway, followed by two or more tasks. The tasks are enabled via the conditions of the split gateway's outgoing edges, which refer to the outcome of the decision. The DMN decision table associated with the decision task *Determine ads* is shown in Fig. 2. This table is the top-level decision table of a DMN decision model, consisting of just this table with two inputs, *Age* and *Has Children*. The corresponding DMN decision requirements diagram is not shown for simplicity.

Complementing process models with decision models requires considerations about the correctness of the resulting decision-aware process models. The presented tool checks two behavioral criteria that must be fulfilled for a correct integration, namely decision deadlock freedom and dead branch absence [2]. Validating these two criteria ensures that the traditional soundness criteria for workflow nets [1] are still fulfilled even for decision-aware processes. On the one hand, decision deadlock freedom requires every table of the decision model to be complete and that for each output of the top-level decision at least on edge condition of the decision fragment evaluates to true. This makes sure that the decision fragment does not contain deadlocks. On the other hand, dead branch absence demands that for each edge condition there is at least one decision output such that the condition becomes true. This ensures that the decision fragment does not contain dead branches. The presented tool implements an efficient and scalable algorithm to check these behavioral criteria.

2 Tool

The tool to check the soundness of decision-aware business processes is implemented in *dmn-js*, a DMN decision table editor developed by Camunda¹. Parts of our implementation rely on functionality provided in a tool to verify DMN decision tables [4]. Our tool is available for download together with exemplary models, the pseudocode of the algorithms, a user guide and two screencasts at <https://bpt.hpi.uni-potsdam.de/Public/BpmnDmnSoundness/WebHome>.

In the following, we will apply the tool to the example from Section 1. Fig. 3 shows the interface of the tool after the process model was imported to the decision table editor via the *import BPMN* button and the *check soundness* button was clicked to check for the two behavioral soundness criteria. The results are presented in the table *Analysis results*. First of all, the output set of the decision table is computed and displayed. This is defined to be the set of outputs the table can produce given the domains of the input variables, the rules, and the hit policy. The table in the example has four rules that operate on an integer and a boolean input. Furthermore, it is a multi-hit table with a rule order policy, denoted by the letter *R* in the upper left corner. Consequently, if two rules match for the same input, the output of both rules will be returned as a list ordered by the order of the rules. For example, only rule 4 matches for the input $(10, false)$ (or $(10, true)$), so the only output will be *Toys*. However, for input $(13, false)$ both rules 2 and 4 match, such that the outputs of both rules are returned in a list: $[Videogames, Toys]$.

¹<https://camunda.org>

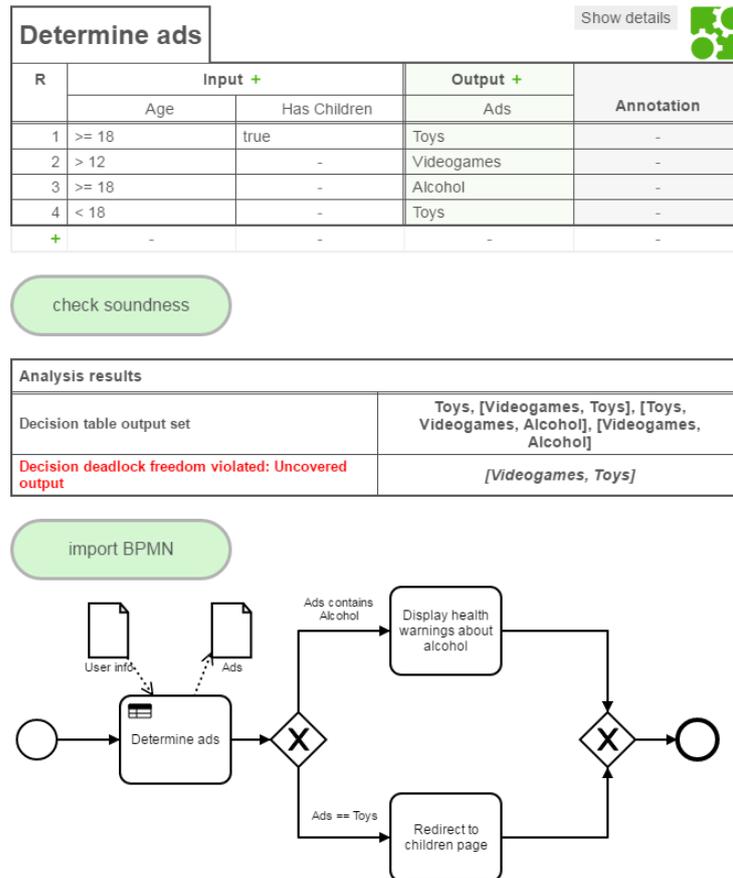


Fig. 3. View of the tool after checking soundness for the running example

After the entire table output set was computed, the behavioral criteria are checked. In the example in Fig. 3, the decision deadlock freedom criterion is violated because no edge condition of the decision fragment evaluates to true for the output $[Videogames, Toys]$. All other outputs fulfil at least one edge condition. For example, the output *Toys* perfectly matches the lower branch condition and the upper condition is satisfied for the last two outputs because they contain the element *Alcohol*.

To get rid of the error, we could now either change the table so that every output is covered by an edge condition or adapt the decision fragment so that the uncovered output is accounted for. For instance, we could try to add another branch with the condition $Ads == Videogames$ and check soundness again. This will lead to the result displayed in Fig. 4. Unfortunately, the output $[Videogames, Toys]$ is still uncovered. Additionally, by adding the new branch we violated the dead branch absence criterion since the condition $Ads == Videogames$ is actually unreachable. There is no output of the decision table that is exactly equal to *Videogames*. To eliminate both errors the branch condition should be changed to $Ads\ contains\ Videogames$. In this way, the uncovered output will match this condition, which in turn is not unreachable anymore.

Determine ads				Hide Details	
adDecision					
R	Input +		Output +	Annotation	
	Age	Has Children	Ads		
	age	hasChildren	ads		
	integer	boolean	string		
1	>= 18	true	Toys	-	
2	> 12	-	Videogames	-	
3	>= 18	-	Alcohol	-	
4	< 18	-	Toys	-	
+	-	-	-	-	

check soundness

Analysis results	
Decision table output set	Toys, [Videogames, Toys], [Toys, Videogames, Alcohol], [Videogames, Alcohol]
Decision deadlock freedom violated: Uncovered output	[Videogames, Toys]
Dead branch absence violated: Unreachable condition	Ads == Videogames

import BPMN

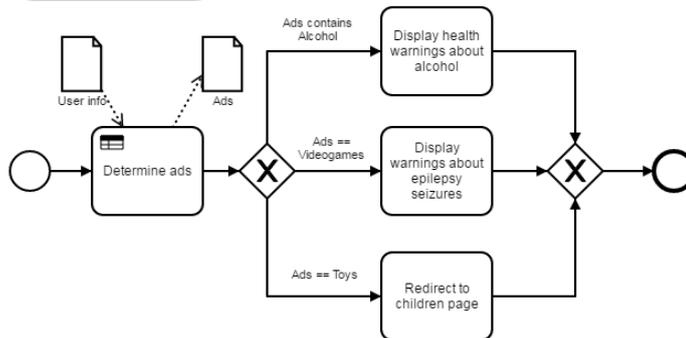


Fig. 4. View of the tool after checking soundness with the adapted decision fragment

3 Maturity

In this section, we discuss three aspects regarding our tool. First, we report our findings from two empirical studies we conducted regarding the practical relevance of our notion of soundness of decision-aware business processes. Second, we discuss the scalability of our algorithm. Lastly, we describe our tool’s maturity.

In one study regarding the practical relevance of our notion we asked participants of an online course on process and decision modeling to design a BPMN process and an associated DMN decision model given a textual description of a billing process. We received 70 valid submissions, 43 (61%) of which violated at least one of the behavioral criteria. In another study, we analyzed a process management project of a large German

insurance company that serves to handle invoices of insured customers automatically. This project contains 86 decision-aware business processes. In 26 (30%) of those we detected violations of our soundness criteria.

With respect to scalability, our implementation is divided into two parts, one for computing the output set of a decision table, and one for checking the two behavioral criteria. The former part extends the code for finding overlapping rules in a tool for DMN decision table verification [4], described in detail in [3]. We reuse this code to implement a function to find all sets of rules that match for some input. Given this information and the hit policy of the table, the output set can be computed. The algorithm in [3] is described to be scalable. Our code does not introduce additional complexity. Our modification adds simple set intersection operations and sorts the matching rules according to their priority if demanded by the hit policy. For the second part of our implementation—checking the behavioral criteria—the complexity is given by the number of possible outputs of the table times the number of branches of the decision fragment. Therefore, our algorithm has the same scalability as the one in [3].

Our tool is able to check the sound integration of decision fragments with top-level decision tables with any number of output variables (a DMN decision table may have more than one output variable column) and any kind of hit policy. Regarding the decision fragment, its split gateway can be an exclusive, inclusive or complex gateway. Furthermore, the edge conditions can be made up of conjunctions or disjunctions of unary tests. Each unary test is an expression that compares the output of the table to another value. For the comparison, in case of single value outputs the usual arithmetic comparison operators appropriate for the type of the variable can be used. In case of list outputs, equality and containment operators are available. In the end, the result of the comparison will either be true or false so that it can be easily decided whether or not an output is covered or a branch is reachable. Note that we assume that a decision table has finitely many rows and therefore also finitely many outputs. Similarly, a decision fragment is assumed to have a finite number of branches each consisting of finitely many unary tests. Therefore, our soundness check is guaranteed to terminate.

References

1. Aalst, W.M.P., Hee, K.M., Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.: Soundness of workflow nets: classification, decidability, and analysis. *Formal Aspects of Computing* 23(3), 333–363 (2010)
2. Batoulis, K., Weske, M.: Soundness of decision-aware business processes. In: *BPM Forum* (2017)
3. Calvanese, D., Dumas, M., Ülari Laurson, Maggi, F.M., Montali, M., Teinemaa, I.: Semantics and analysis of dmn decision tables. In: *BPM* (2016)
4. Ülari Laurson, Maggi, F.M.: A tool for the analysis of dmn decision tables. In: *BPM Demo Track*. pp. 56–60 (2016)
5. OMG: *Business Process Model and Notation, Version 2.0.2* (January 2014)
6. OMG: *Decision Model and Notation, Version 1.1* (May 2016)
7. Von Halle, B., Goldberg, L.: *The Decision Model: A Business Logic Framework Linking Business and Technology*. Taylor and Francis Group (2010)