POSEIDON: A Graphical Editor for Item Selection Rules Within Feature Combination Rule Contexts

Daniel Bischoff[1(✉)], Wolfgang Küchlin[2], and Oliver Kopp[1]

[1] Mercedes-Benz AG, Böblingen, Germany
{daniel.bischoff,oliver.kopp}@mercedes-benz.com
[2] Universität Tübingen, Tübingen, Germany
wolfgang.kuechlin@uni-tuebingen.de

Abstract. A car line can offer more than 10^{100} variants, and for each customer order a concrete selection of features and parts needs to be done. The respective selection rules are interconnected and are subject to constraints imposed by different car lines. We address the problem of finding and fixing logical errors in these interconnected selection rules within different contexts of allowed feature combinations. Previous work has focused on text-based or matrix-like representations which presented challenges regarding cognitive complexity, size of the representation, and usability. We present an integrated visualization of the combined practical effects of item selection and feature combination rules. The implemented tool detects logical errors and supports user workflows to fix the data visually.

Keywords: Product configuration · Configuration systems · Configuration rules · Data quality · Visualization

1 Introduction

Customization of cars increases the space of available variants. The customer can select multiple equipment options (features). Based on this selection, the parts (mechanical, electronic, or software functions) to assemble the car have to be chosen. These parts are documented in the Bill-of-Materials (BoM). Typically, this is not a 1:1 feature/part relation, but there are complex rules in place [8]. Each rule specifies a Boolean condition on the chosen features and outputs whether the associated part has to be chosen. In this paper, we will refer to them as *item selection rules*. A general so called 150% BoM, which contains all possible parts of all possible variants, is converted into a specific 100% BoM by evaluating all rules. From this 100% BoM, a car can be produced.

The distance between the headlights of a car, for instance, might differ between a limousine and a convertible. When producing one of these cars, the

© IFIP International Federation for Information Processing 2023
Published by Springer Nature Switzerland AG 2023
F. Noël et al. (Eds.): PLM 2022, IFIP AICT 667, pp. 3–14, 2023.
https://doi.org/10.1007/978-3-031-25182-5_1

same Electronic Control Unit (ECU) could still be used but needs to be configured. This ECU has to be informed which headlight distance is an accurate description of the ECU's context, i.e. the car. To derive that information from the equipment options, item selection rules for the configuration of the ECU software are defined.

In general, a customer can order from different product lines (e.g., gasoline or electric line). In certain product lines a combination of convertible with a sports package might be permitted, whereas other lines forbid this combination. The context constraining the allowed combinations is called *feature combination rule context*.

Item selection rules for identical parts in different contexts might be similar and are often used as the basis for new rules. Thus, item selection rules share similar parts but also differ in other parts. In general, a product line can also evolve over time. Thus, an item selection rule can be applicable in, and be subject to, multiple contexts. The main challenge is the creation of the initial rules and recontextualization for new contexts.

In this paper we address the challenges during initial creation and recontextualization by a) visualizing item selection rules, while b) being able to switch seamlessly between different contexts, and c) enabling interactions on the visual data itself. For that we start by presenting background and related work (Sect. 2). Then we provide a running example (Sect. 3). Afterwards, we discuss an algorithm to calculate the visualization (based on the given context; Sect. 4) and its implementation (Sect. 5). Subsequently, the implementation is evaluated and discussed (Sect. 6). Finally, we summarize our findings and provide an outlook on future work (Sect. 7).

2 Background and Related Work

In the automotive context, a product line defines which vehicle variants are possible. The term *feature combination rule context*, or *context* for short, entails the declaration of features and the constraints that make feature combinations valid or invalid. Different professional domains have different names for this concept. Examples are: High Level Configuration, Feature Tree, Product Overview, Vehicle Description Summary, and Model Description [2,7]. Automotive product lines have been encoded in Boolean logic at least since the 1990s s and formal validation methods based on Satisfiability Solving (SAT) have been used at least since 2000 [8], whereas feature trees were first transformed to Boolean logic three years later [9].

Each equipment option is represented as a Boolean variable indicating equipment option presence. Thus, a concrete vehicle is modeled as an assignment of the Boolean variables defined by the context which indicates presence or absence of equipment options. During assembly, concrete vehicles are built. The *item selection rules* define for which vehicle variants which of the items have to be selected. In the automotive context, examples of an item are a physical component to build a car, a software version, or a configuration parameter value.

A concrete vehicle is segmented into alternative bundles called *positions*, which need to be filled. For each position, there needs to be an item selected. Thus, each position lists respective $<item, rule>$ pairs. In our BoMs, positions can contain roughly up to 30 alternative items, while up to 25 variables are used in some item selection rules.

For each position, there are quality criteria in place [8]: 1. Each item selection rule is statisfiable in its context (satisfiability). 2. For each variable assignment, there is at most one item hit (uniqueness). 3. For each variable assignment, there is at least one item hit (completeness). Identical data quality goals have also been found by Astesana et al. [2]. The quality criterion of satisfiability has been identified by Berndt et al. [3]: Multi-valued decision diagrams (MDDs) have been there for computation only. The quality criteria of uniqueness and completeness also have been identified by Voronov et al. [14].

Tidstam et al. [12] present an approach for item selection rules captured in matrices. One dimension captures the product families, the other dimension captures the items to be selected. An element in the matrix indicates whether an item is always selected, not selected, and sometimes selected. The case "sometimes" appears, because the context cannot be fully captured in the matrix. In contrast, our approach presents an explicit visualization of selections, the implications of a context, and the visualization is not bound to a matrix representation. Tidstam and Malmqvist [13] compared their matrix-based approach with a list-based approach. It turned out that the matrix-based approach was preferred by their target audience.

Shafiee et al. [11] conducted a survey on visual representation techniques for product configuration systems in industrial companies. Their result indicates that companies using visual knowledge representation techniques tend to have (i) higher quality of the product configuration system's knowledge base and (ii) higher quality with respect to communications with domain experts [11]. This supports our approach to use visual representations instead of text-based ones.

Amilhastre et al. [1] encoded context validity in a Constraint Satisfaction Problem (CSP), compiled the CSP into an automaton and represented it graphically. The automaton accepts valid feature selections, but, in contrast to our approach, does not encode which part selection follows, i.e. which and how many parts are involved in a potential collision.

POSEIDON is the first decision-diagram-based item selection rule editor, that has build-in quality checks and while allowing for seamless context switches.

3 Running Example

The example we will use throughout the rest of this paper centers around the configuration of an ECU that is used in two vehicle contexts called Electric (E-Series) and Gasoline (G-Series), respectively. We will focus on a single configuration value that represents the distance between the headlights called *headlight_distance_mm*.

Among the variables we use are *LIMOUSINE, KOMBI, COUPE*, and *CABRIO* to indicate body types and *FR, LU, BE, NL, DK, PL, CZ, AT, LI*,

and *CH* to name distribution countries. The variables *SPORT, NIGHT*, and *ELEGANCE* are used to represent extras.

The context of E (γ_E) and G-Series (γ_G) is defined in Eq. (1) and Eq. (2), respectively. Here, pseudo Boolean constraints are used for brevity and can be transformed into pure Boolean constraints [5]. They are generally of the form $\Sigma(v) \leq 1$, which we refer to as an *at-most-one* (AMO) constraint, or $\Sigma(v) = 1$, an *exactly-one* (EXO) constraint. Some of these are shared between the contexts. We generally use the following operator precedence: $\neg, \wedge, \vee, \implies, \iff$ (from strongest to weakest binding).

$$\begin{aligned}\gamma_E = &[LIMOUSINE + KOMBI + COUPE + CABRIO = 1] \wedge \\ &[SPORT + NIGHT + ELEGANCE \leq 1] \wedge \\ &[FR + LU + BE + NL + DK + PL + CZ + AT + LI + CH \leq 1] \wedge \\ &((LIMOUSINE \vee KOMBI) \implies \neg SPORT \wedge \neg NIGHT) \wedge \\ &(NL \implies NIGHT) \wedge \\ &(CABRIO \implies \neg DK \wedge \neg PL \wedge \neg CZ)\end{aligned} \tag{1}$$

$$\begin{aligned}\gamma_G = &[LIMOUSINE + KOMBI + COUPE + CABRIO = 1] \wedge \\ &[SPORT + NIGHT + ELEGANCE \leq 1] \wedge \\ &[FR + LU + BE + NL + DK + PL + CZ + AT + LI + CH \leq 1] \wedge \\ &(NL \implies \neg SPORT \wedge \neg ELEGANCE) \wedge \\ &(CABRIO \implies SPORT) \wedge \\ &\neg(KOMBI \wedge (BE \vee NE \vee LU) \wedge SPORT)\end{aligned} \tag{2}$$

Table 1. The position for `headlight_distance_mm` used in the example.

Item	Item selection rule ϕ
1650 mm	$CABRIO \vee COUPE \vee$ $SPORT \wedge (\neg KOMBI \wedge \neg LIMOUSINE \vee KOMBI \wedge$ $(\neg(BE \vee CZ \vee DK \vee FR \vee NL \vee PL) \vee LI \vee AT \vee CH \vee LU)) \vee$ $NIGHT \wedge (\neg LIMOUSINE \vee KOMBI)$
1700 mm	$KOMBI \wedge \neg NIGHT \wedge \neg SPORT \vee$ $NIGHT \wedge (LIMOUSINE \vee \neg CABRIO \wedge \neg COUPE \wedge \neg KOMBI)$
1750 mm	$LIMOUSINE \wedge \neg NIGHT \vee$ $ELEGANCE \wedge \neg(CABRIO \vee COUPE \vee KOMBI \vee LIMOUSINE) \vee$ $KOMBI \wedge SPORT \wedge (BE \vee CZ \vee DK \vee FR \vee NL \vee PL)$

The position configuring `headlight_distance_mm` is given in Table 1. To produce simpler diagrams, the rules used in this example initially meet all quality criteria (Sect. 2).

4 Visualizing Item Selection Rules

Algorithm 1 describes a procedure how to generate a decision diagram data structure based on a context, items, and item selection rules. This data structure can then be used as input for a graph layout algorithm. We will describe the most important aspects of Algorithm 1 in the following sections. The algorithm described here extends our previously published technique [4] by a) adding grouping of variables and b) the capability to visualize under different contexts.

Algorithm 1. Generate Visualization Data Structure

1: Given Context γ
2: Given Map of <Item i, Item Selection Rule ϕ_i > of length n
3: configVars ← Variables that occur in $\{\phi_0, \ldots, \phi_{n-1}\}$
4: γ^* ← project(γ, configVars)
5: groups ← group(γ^*, configVars)
6: $\psi \leftarrow \gamma^* \wedge \bigwedge_{i=0}^{n-1} (\phi_i \iff H_i)$
7: result ← bdd(ψ, groups)
8: result ← terminals(result)
9: result ← mdd(result, groups)
10: result ← labeling(result)
11: return result;

4.1 Projection

Projection aims to provide focus on the visualization in terms of the configuration variables: Variables not used in the configuration, but defined in the context should be removed. However, they still might have an effect on the configuration. Thus, their effects on valid combinations of the configuration variables are kept. To implement the projection, we use the model enumeration-based quantifier elimination as described and evaluated by Zengler and Küchlin [15]. In the running example, the contexts (γ_E and γ_G, Sect. 3) are already projected to the set of configuration variables for better readability.

4.2 Grouping

The grouping step (Line 5) partitions the configuration variables such that the variables in each group exclude each other. The context is the main source of potential groups, since it contains explicit at-most-one (AMO) and exactly-one (EXO) constraints. To generate more potential groups we implemented functionality which allows our users to propose groups, which we then validate against the context by checking whether $context \implies AMO(proposal)$ is a tautology. If the proposal turns out to be valid, we store it in our database, which realizes a

user-defined group storage for each context. This shared and growing common knowledge is used during future groupings. In our example the groups are 1. the four body types, 2. the options SPORT, NIGHT and ELEGANCE, and 3. the distribution countries.

4.3 Constructing a BDD

The first step of constructing our visualization is to build a Binary Decision Diagram. For this purpose, we create a set of Boolean helper variables $H = \{H_0, \ldots, H_{n-1}\}$ representing the n items. Together with the n item selection rules $\phi = \{\phi_0, \ldots, \phi_{n-1}\}$ and the projected context γ^* they are used to create the core formula

$$\psi = \gamma^* \wedge \bigwedge_{i=0}^{n-1} (\phi_i \iff H_i). \tag{3}$$

We call H the set of *item variables*, in contrast to the configuration variables that occur in the selection rules ϕ.

The core formula ψ consists of two components – the context and a representation of the position. The construction of the position representation $\bigwedge_{i=0}^{n-1} (\phi_i \iff H_i)$ assures that no assignment of only the configuration variables can falsify this second part of ψ. There is always an otherwise unconstrained helper variable H_i to match the value of ϕ_i, thereby providing a trivial solution to all constraints. We can therefore conclude that any assignment of the configuration variables falsifying ψ represents a violation of the context constraints, and thus an invalid vehicle. As a result, the item selection rules determine the resulting item for each variant, while the context defines which variants are considered.

Subsequently, we compute a BDD for ψ. Thereby, the item variables H are always last in the BDD's ordering, while the configuration variables that belong to the same group are next to each other. For this purpose we adapted the sifting algorithm [10], which turned out best [6]. For our example this yields the result shown in Fig. 1 for the E-Series context.

4.4 Terminal Generation

To compress the visualization further, we traverse the graph top-down. Every path eventually assigns all configuration variables. At this point, a terminal node is generated that contains exactly those item variables H_i which must be true in order to reach the final $\$true$ node of the BDD. This path is unique at this point, because the core formula (3) constrains all H_i to be equivalent to their already fixed ϕ_i counterparts. For our example this yields Fig. 2.

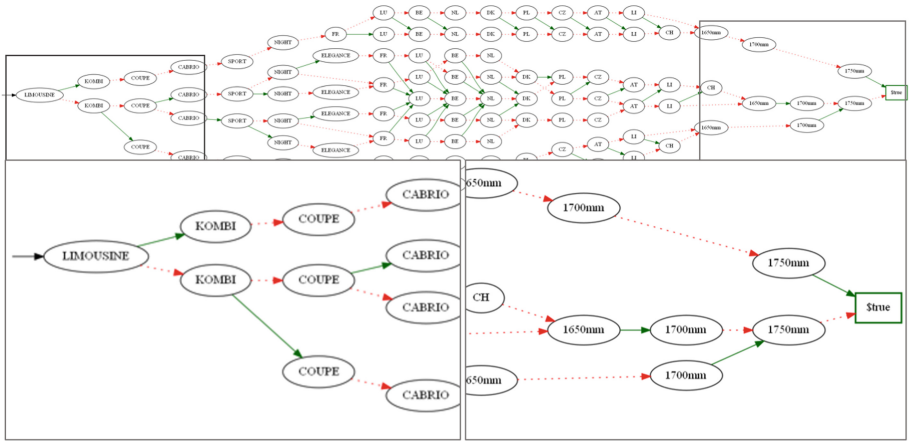

Fig. 1. The BDD for `headlight_distance_mm` in the E-Series context. The *$false* node is hidden with all its incoming edges for better readability.

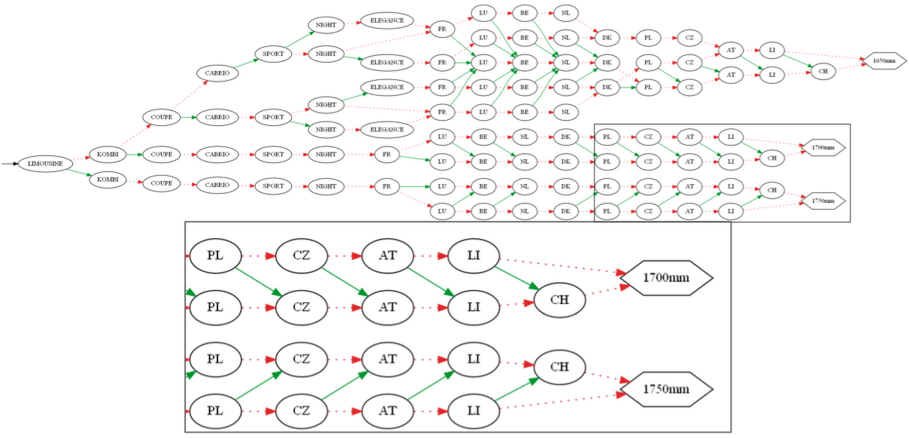

Fig. 2. The BDD for headlight_distance_mm in the E-Series context with terminals. The *$false* node is hidden with all its incoming edges for better readability.

4.5 Multi-decision Diagram Generation

We traverse the graph top-down. For every node that contains a configuration variable, we collect the variables of its children nodes recursively until we find the first variable of another group or a terminal. For those collected variables we form a *multi-decision node* representing the possible selections from the group. For our example this yields Fig. 3.

The multi-decision nodes replace the binary ones and have several outgoing edges. For each group member an outgoing edge exists, which may be shared with other members if they lead to the same result. One additional edge represents the *no selection* option for that group, where all variables are assigned *$false*. It

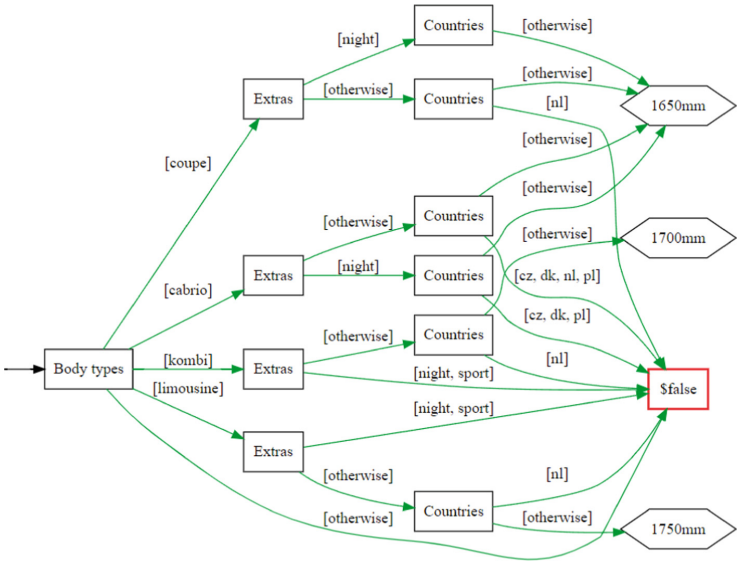

Fig. 3. The MDD for headlight_distance_mm in the E-Series context. The text in square brackets represents the decision in the previous MDD node.

may happen that *no selection* leads to the same item as choosing some member of the group, so that the edges may be shared.

The *no selection* edge is called *otherwise* in our figures, representing all options that are not explicitly noted on other outgoing edges of the same MDD node.

For example the topmost *Countries* node in Fig. 3 has a single outgoing edge *otherwise* which represents all possible countries or no country at all since they all result in the same item selection of 1650 mm. The *Countries* node right below has two outgoing edges, where NL leads to *$false* and therefore represents a configuration in contradiction with the context constraints and the *otherwise* edge representing all other or no country selection resulting in the 1650 mm item being chosen.

4.6 Label Creation and Propagation

We introduce labels to reduce the size of the graph. For this, we traverse the graph top-down. If a node has the *$false* node as a child, a *label* is generated to indicate that this assignment contradicts the context. The label contains the opposite variable assignment and is placed on all incoming edges of the node. The edge to the *$false* node is then removed (cf. Fig. 4). If a node has the same label on all its outgoing edges, the label is propagated up from all outgoing to all incoming edges. Labels on the same edge are combined using the $\wedge$ operator. This may result in nodes with only a single unlabeled outgoing edge, which are then removed. For our example this yields Fig. 5.

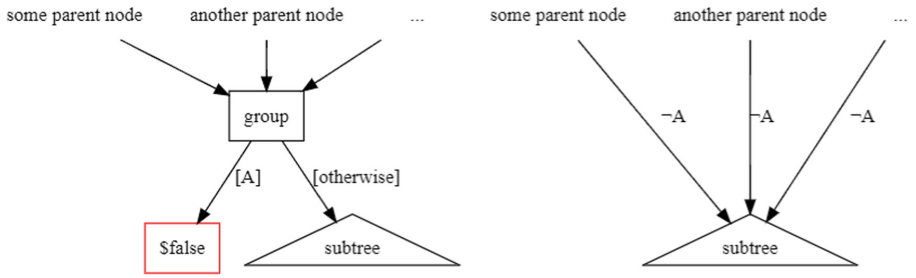

Fig. 4. Selecting A leads to a contradiction, thus A needs to be assigned *$false*. We force $\neg A$ by adding labels and (since the group now has a single unlabeled child) remove the decision node by rerouting its incoming edges to its valid subtree.

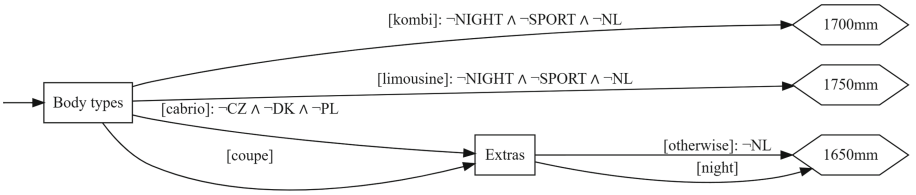

Fig. 5. The final MDD for headlight_distance_mm in the E-Series context with labels. The text in square brackets represents the decision in the previous MDD node.

4.7 Roundtrip

We allow users to manipulate the graphical representation, e.g., by introducing new nodes or redirecting edges. Effectively, this allows for a complete (re-) definition of item selection rules merely through the GUI. The resulting graph can then be transformed back into individual rules by reading off the possible necessary assignments to reach a terminal. E.g., we can read the resulting item selection rule for 1700 mm from Fig. 5, as $KOMBI \wedge \neg NIGHT \wedge \neg SPORT \wedge \neg NL$ for the top-most path. Hence, we can transform a position and a context into an MDD representation, let the user change the MDD visually, and transform the result back into the updated position for the used context.

5 Implementation

We implemented Algorithm 1 in a web-based tool which layouts the decision diagram and makes it editable. In order to demonstrate the use cases of our application and to be able to show how we deal with questionable data quality, we now change the position by adding another item 1800 mm with its selection rule $CABRIO \wedge NIGHT$. Since this case was already covered by the rule for 1650 mm, this leads to a data quality issue which we visualize to the user as shown in Fig. 6. The ambiguity highlighted in red indicates that there are vehicle configurations in this context that select multiple items. The path from the

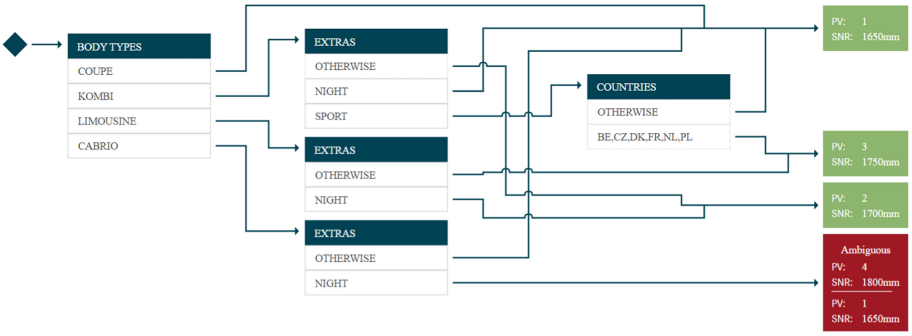

Fig. 6. A position containing an ambiguity in the context shared by E and G-Series.

diamond-shaped root node to the ambiguity identifies the vehicle(s) affected, in this case all *Cabrios* with the *Night* extra need correction.

Our user interface allows users to rearrange the ordering of groups in which case we simply skip automatic ordering in the algorithm. If a user thus has a certain ordering in mind they can manually force this ordering. This works when creating new positions as well as when working with preexisting positions. We implemented editing functionality which allows the user to work with the graph while keeping it valid until they want to retrieve the item selection rules (based on the roundtrip functionality described in Sect. 4.7). Therefore, a user can manipulate the graphical representation and retrieve the resulting rule set after the change.

6 Evaluation and Discussion

Using the implementation, we evaluated a model series from 2019 and took positions with 10 or more items resulting in 535 positions with sizes from 10 to 100 and a median of 17. Thereby, the runtime of the complete algorithm ranged from 0.8 s to 12.3 s, whereby the median runtime was 1.4 s. Within this, projection (Sect. 4.1) took 40% of the total runtime (in the median) with a minimum of 3% and a maximum 87%. As a consequence, we implemented caching for the projection: When a context had been projected to a given configuration variable set previously, the result is reused. At a cold start with caching activated, the runtime of the complete algorithm changed to the range of 0.02 s to 13 s, with a median of 1 s.

The scaleability primarily depends on the context. More precisely the number of combinations of the variables used in the item selection that are considered context-valid, is the primary driver of the visualization size. A decision diagram is always smaller than an analogous table, since inner nodes can be re-used.

The solution is applicable for other products and domains whenever the *context* and *item selection rules* are either written in or translatable to Boolean logic. This translation is possible for feature trees [9].

7 Conclusion and Outlook

We showed the theoretical foundation and a practical implementation of a tool for visualizing and editing item selection rules. It supports seamless context switches and direct visual data editing during initial creation and recontextualization. We showed that the tool can handle industry-grade sizes of item selection rules and contexts.

By allowing each branch to have its own ordering, we foresee that it is possible to reduce the size of the graph even further. Since the concrete impact on industry-sized item selection rules, especially when it comes to readability, is unclear, this investigation is our next research step.

Acknowledgements. This work has received funding by the BMWK-funded project SofDCar (19S21002D).

References

1. Amilhastre, J., Fargier, H., Marquis, P.: Consistency restoration and explanations in dynamic CSPS-application to configuration. Artif. Intell. **135**(1–2), 199–234 (2002)
2. Astesana, J.M., Cosserat, L., Fargier, H.: Constraint-based vehicle configuration: a case study. In: ICTAI. IEEE (2010)
3. Berndt, R., Bazan, P., Hielscher, K.S.: MDD-based verification of car manufacturing data. In: CIMSIM. IEEE (2011). https://doi.org/10.1109/CIMSim.2011.40
4. Bischoff, D., Küchlin, W.: Adapting binary decision diagrams for visualizing product configuration data. In: Eibl, M., Gaedke, M. (eds.) INFORMATIK 2017. GI e.V. (2017). https://doi.org/10.18420/in2017_149
5. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into sat. J. Satisfiability Boolean Model. Comput. **2**(1–4), 1–26 (2006)
6. Krawietzek, N.: Minimierung der graphischen Repräsentation von Coderegeln durch Variablensortierung. Master's thesis, SCMT, Filderstadt, Germany (2020)
7. Küchlin, W.: Logic and verification of product configuration in the automotive industry. In: Proof and Computation II, pp. 387–408. World Scientific (2021). https://doi.org/10.1142/9789811236488_0009
8. Küchlin, W., Sinz, C.: Proving consistency assertions for automotive product data management. J. Autom. Reason. **24**(1–2), 145–163 (2000)
9. Mannion, M., Camara, J.: Theorem proving for product line model verification. In: van der Linden, F.J. (ed.) PFE 2003. LNCS, vol. 3014, pp. 211–224. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24667-1_16
10. Rudell, R.: Dynamic variable ordering for ordered binary decision diagrams. In: Kuehlmann, A. (ed.) The Best of ICCAD, pp. 51–63. Springer, Boston (2003). https://doi.org/10.1007/978-1-4615-0292-0_5
11. Shafiee, S., Kristjansdottir, K., Hvam, L., Felfernig, A., Myrodia, A.: Analysis of visual representation techniques for product configuration systems in industrial companies. In: IEEM. IEEE (2016). https://doi.org/10.1109/ieem.2016.7797985
12. Tidstam, A., Bligård, L.O., Ekstedt, F., Voronov, A., Åkesson, K., Malmqvist, J.: Development of industrial visualization tools for validation of vehicle configuration rules. In: TMCE (2012)

13. Tidstam, A., Malmqvist, J.: Comparison of configuration rule visualizations methods. In: Bernard, A., Rivest, L., Dutta, D. (eds.) PLM 2013. IAICT, vol. 409, pp. 550–559. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41501-2_55
14. Voronov, A., Tidstam, A., Åkesson, K., Malmqvist, J.: Verification of item usage rules in product configuration. In: Rivest, L., Bouras, A., Louhichi, B. (eds.) PLM 2012. IAICT, vol. 388, pp. 182–191. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35758-9_16
15. Zengler, C., Küchlin, W.: Boolean quantifier elimination for automotive configuration – a case study. In: Pecheur, C., Dierkes, M. (eds.) FMICS 2013. LNCS, vol. 8187, pp. 48–62. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41010-9_4