Context-Aware Verification of DMN

Simon Vandevelde *
KU Leuven, De Nayer Campus
Dept. of Computer Science
Leuven.AI, Belgium
s.vandevelde@kuleuven.be

Benjamin Callewaert
KU Leuven, De Nayer Campus
Dept. of Computer Science
Leuven.AI, Belgium
benjamin.callewaert@kuleuven.be

Joost Vennekens
KU Leuven, De Nayer Campus
Dept of Computer Science
Leuven.AI, Belgium
joost.vennekens@kuleuven.be

Abstract

The Decision Model and Notation (DMN) standard is a user-friendly notation for decision logic. To verify correctness of DMN decision tables, many tools are available. However, most of these look at a table in isolation, with little or no regards for its context. In this work, we argue for the importance of context, and extend the formal verification criteria to include it. We identify two forms of context, namely in-model context and background knowledge. We also present our own context-aware verification tool, implemented in our DMN-IDP interface, and show that this context-aware approach allows us to perform more thorough verification than any other available tool.

1. Introduction

The Decision Model and Notation (DMN) [1] standard is a notation standard for decision logic. It was designed by the Object Management Group (OMG), who aimed at creating a notation that is readable, user-friendly and executable. Because of these features, the standard has quickly gained popularity in both industry [2, 3, 4] and academia [5, 6].

An important part of the DMN standard is the decision table, which is a modular representation of a definition of "output" variables in terms of "input" variables, as shown in Fig 1. This table has one input variable, *Body Mass Index (BMI)*, which is used to define one output variable, *BMILevel*. The value of the output variable(s) is determined by the rows of the tables which match the values of these input variables. These are the rows that are said to *fire*. For example, if *BMI* is a value higher than 25, then only the third row fires and the *BMILevel* is defined as "Overweight".

The behavior of a decision table is defined by its hit policy. There are two types of hit policies: single hit (the value of the output variable(s) is determined by a single row) and multiple hit (the value of the output variable(s) is determined by a set of rows). As our work focuses exclusively on the single hit policies, we will not discuss the multiple hit policies further. In total, there are three single hit policies. In tables with the U(nique) policy, rows should be mutually exclusive to ensure that only one is applicable for each set of input values. In A(ny) hit tables, overlap between the rows is allowed as long as they specify the same output value. Lastly, in F(irst) hit tables the rows are evaluated top-to-bottom, and the first one that is applicable determines the value of the output variable(s).

A DMN model consists of one or more decision tables, linked together by their input and output variables. The Decision Requirements Diagram (DRD) shows the connection between these different decision tables, their inputs, their knowledge sources and more. In this way, the DRD provides a clear overview of the entire model, allowing users to better see the *flow* of information throughout the decisions. Figure 2 shows an example of a DRD for a model with three decision tables and four input variables.

An important aspect of decision tables is that they should be both *complete* and *sound*. A table is complete if it contains an applicable row for every possible set of input values. The soundness of a table depends on its hit policy: U tables should not contain overlapping rows of any kind, while A tables allow them as long as they do not have conflicting outputs. Tables with the F hit policy are allowed to have overlap, and as such, are always sound. Tables lacking these correctness properties are considered erroneous. On top of soundness and completeness, tables should also be without *unfireable* rules, i.e., rules that can be omitted without changing the meaning of the table.

Efficient tools have been created to perform automated table verification. However, as we will show in this paper, most of these tools are unable to (sufficiently) reason on the *context* of the table and the model. Indeed, most approaches verify each table

*This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme

URI: https://hdl.handle.net/10125/80096
978-0-9981331-5-7
(CC BY-NC-ND 4.0)

BMILevel		
U	BMI	BMILevel
1	< 18.5	Underweight
2	$[18.5..25]$	Normal
3	> 25	Overweight

Figure 1: Example of a decision table.

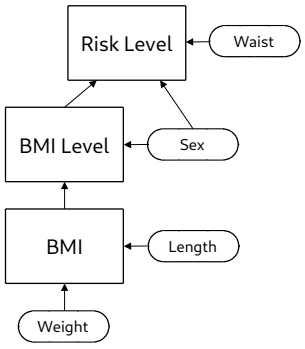

Figure 2: Example of a Decision Requirements Diagram

in isolation, i.e., without any regard to the rest of the model. We are aware of two approaches that do take context into account, but they either do not make enough use of context, or are unable to sufficiently pin-point specific errors, as we will discuss later.

The contributions of this paper are threefold: (a) demonstrating the importance of context via concrete use cases, (b) formally defining context-aware verification, and (c) building a tool capable of verification *within context*. It is structured as follows. We first look at the tools that are already available in Section 2. Afterwards in Section 3, we give concrete examples in which context is important. In Section 4 we formally explain the correctness criteria, and extend them to include context. These definitions are then used in our implementation, as explained in Section 5. We briefly compare and evaluate our tool in Section 6, and finally we conclude in Section 7.

2. Related Work

Smit et al. [7] conducted a study on DMN verification in a real-life context. In total, they identified eight different verification capabilities for decision tables. As shown in Table 1, five of these capabilities can be considered as specific kinds of soundness. Indeed, all these errors are caused by the same root cause: two or more (fully) overlapping rules. While they do differ in the actions required to fix them, we will nevertheless combine these five capabilities together in this work, as

Table 1: Translation between verification capability terminology.

Smit et al. terminology	Our terminology
Identical rules Equivalent rules Subsumed rule Indeterminism Overlapping fact value range	Soundness
Missing Rules	Completeness
—	Unfireable rule
Unnecessary fact verification	—
Specific partial reduction	—

this already showcases the benefit of our context-aware approach. The sixth verification capability, "Missing Rules", corresponds to table completeness. The two remaining verification capabilities, "Unnecessary fact verification" and "Specific partial reduction", are not discussed in this paper as these are beyond our scope. There is no counterpart for unfireable rules in the framework of Smit et al.

Table 2 shows an overview of previous works on decision table verification, and which verification capabilities they support. With the exception of Hasic et al. [8] and Calvanese et al. [9], all of the proposed algorithms verify a decision table in isolation from the rest of the model. For example, when verifying the *BMILevel* table in Fig. 2, the information in the (upstream) *BMI* table or in the (downstream) *Risk Level* table would simply be ignored.

The tool by Hasic et al. [8] is one of the exceptions that does incorporate some context from the rest of the model in the verification process. Indeed, when verifying a table, it also checks whether every output value of the directly upstream table(s) appears as input value in the current table, and check for every output value of the current table whether it appears as input value in the directly downstream table(s). For example, when verifying the *Risk Level* table, they check whether every possible output for *BMILevel* (Underweight, Normal, Overweight) appears at least once as an input.

In [9], Calvanese et. al outline a conceptual framework for *semantic DMN*, as a way to reason on DMN tables together with background knowledge. Together with this framework, they also extend their verification methods from [10] to include forms of background knowledge. However, when verifying table completeness for example, their algorithm can only tell *if* rules are missing, but not *which* rules are missing.

Table 2: Verification tools and their capabilities.
(X = full support, o = partial support, * = does not distinguish between types of soundness, † = boolean result)

Work	Soundness	Completeness	Unfireable rules	Context
Calvanese et al. (2016) [11]	o	X		
Laurson et al. (2016) [12]	o	X		
Batoulis et al. (2017) [13]	o	X		
Calvanese et al. (2018) [10]	o	X		
Corea et al. (2019) [14]	X	X		
Calvanese et al. (2019) [9]	o†	X†	X†	X
Hasic et al. (2020) [8]	X	X		o
Our tool	X*	X	X	X

Depending on the size of the DMN model and the amount of background knowledge, this can drastically reduce the actual usability of the verification method. Indeed, if a large table is found to be incomplete, manually finding all the missing rules can be both difficult and time-consuming.

3. Types of Context

In this paper, we identify two types of context, each with a different meaning and scope.

1. *In-model context*: the information contained in the rest of the model, i.e., in all tables apart from the one currently being verified.
2. *Background knowledge*: additional knowledge about the domain that is not part of the DMN model itself.

This section elaborates on both types, and gives concrete examples of cases where this context matters.

3.1. In-model context

The first type of context is the in-model context, which consists of all decision tables in the model that are not the target table to be verified. For example, consider the model in Fig. 3 reproduced from [15]: if we want to perform table verification on the "Risk Level" table, its context would consist of the "BMI" and the "BMILevel" tables.

When verified in isolation, the "Risk Level" table passes our three tests: it is sound, complete, and has

BMI

U	Weight	Length	BMI
1	—	—	Weight/(Length*Length)

BMI Level

U	BMI	Sex	BMILevel
1	< 18.5	Female	Underweight
2	< 25	Male	Underweight
3	$[18.5..25]$	Female	Normal
4	$(25..30]$	Male	Normal
5	$(25..30]$	Female	Overweight
6	> 30	—	Obese

Risk Level

U	BMILevel	Sex	Waist	Risk Level
1	Normal	—	—	Low
2	Underweight	—	—	High
3	Overweight	Male	≤ 102	Increased
4	Overweight	Male	> 102	High
5	Overweight	Female	≤ 88	Increased
6	Overweight	Female	> 88	High
7	Obese	Male	≤ 102	High
8	Obese	Male	> 102	Very High
9	Obese	Female	≤ 88	High
10	Obese	Female	> 88	Very High

Figure 3: Decision tables defining a patient's BMILevel

no unfireable rules. However, when taking the in-model context into consideration, the table has two rows that are actually *unfireable*. Indeed, both row 3 and 4 of the table can never fire, as the input combination *BMILevel* = *Overweight* and *Sex* = *Male* is not possible, according to the "BMILevel" table; this table has no rule for which the input value of *Sex* = *Male* leads to output value *Overweight* for *BMILevel*.

This is an example in which the error cannot be found by considering either table in isolation: it can only be detected by looking at both tables together. There are two possible root causes for this error. Either rule 3 and 4 are indeed redundant and should be removed, or, perhaps more likely, the "BMILevel" table is missing a rule in which a man can be overweight.

3.2. Background knowledge

The second type of context is the *background knowledge*: information about the domain that is not explicitly present in the model. Typically, a DMN model contains only the knowledge needed to make certain decisions in a domain. However, the domain experts may have a lot more knowledge that, even though it plays no direct role in the decision process, can be useful when verifying the model.

Consider for instance the decision table shown in Fig. 4, derived from a real-life use case at a company that develops information systems for trains.

Sequence ID				
U	Station Type	Location	Status	ID
1	—	origin	departure	s1a
2	minor	intermediate	departure	s1b
3	major	intermediate	departure	s1c
4	airport	intermediate	departure	s1d
5	—	—	in between	s2
6	—	intermediate	arrival	s3a
7	minor	terminating	arrival	s3b
8	major	terminating	arrival	s3b
9	airport	terminating	arrival	s3c

Figure 4: Decision table defining a sequence ID

Sequence ID: missing rules?				
U	Station Type	Location	Status	ID
1	—	terminating	departure	?
2	—	origin	arrival	?

Figure 5: Missing rules (?) of the table in Fig. 4

Based on some train and station information, the table defines a sequence ID that can then be used to decide what information should be announced in the train. Concretely, the table has three inputs: the type of the station (major, minor, airport), the location of the station in the train's route (origin, intermediate, terminating) and the status of the train (departing, in between stations, arriving).

The following background knowledge is obvious to the domain experts:

- If the station is the terminating station, the train's status can never be "departure".
- If the station is the station of origin, the train's status can never be "arrival".

Because of this background knowledge, some input combinations are not needed in the decision table. For instance, there is no rule for *Location* = *origin* and *Status* = *arrival*.

If the table is verified without background knowledge, it would seem to be incomplete, and a verification tool might suggest to add at least two new rules, shown in the table in Fig. 5. However, from a modeller's point of view, such rules of course do not make much sense.

4. Formal correctness criteria

In this section, we formally define correctness of decision tables, taking into account context. First, we explain the semantics of single hit decision tables and their completeness criterion as described by Calvanese et al. [10]. Following that, we describe the criterion for unfireable rules, and extend the correctness criteria with both types of context. We will only consider tables with the U hit policy in this section for the sake of simplicity, but the semantics are trivial to extend to the A and F hit policy.

4.1. Decision table semantics, completeness and soundness

Each cell of a decision table (i, j) corresponds to a formula $F_{ij}(x)$ in one free variable x. For example, a cell containing "< 18.5" translates to the formula "$x < 18.5$". As such, a table T with rows R, n input columns I and m output columns O is represented by a conjunction of material implications:

$$\bigwedge_{i \in R} \left(\bigwedge_{j \in I} F_{ij}(x_j) \Rightarrow \bigwedge_{k \in O} F_{ik}(y_k) \right)$$

We use $Sem_T(V_1, \ldots, V_{m+n})$ to represent this table semantics, with V_i representing the DMN variable in the heading of the ith table column. where H_j is the header of column j.

For example, the table in Fig. 3 can be translated as:

$$\begin{aligned} &(BMI < 18.5 && \Rightarrow BMILevel = Underweight) \\ \wedge&(18.5 \leq BMI \leq 25 && \Rightarrow BMILevel = Normal) \\ \wedge&(BMI > 25 && \Rightarrow BMILevel = Overweight) \end{aligned}$$

Besides this table semantics, Calvanese et al. define a few table correctness criteria, including table completeness and soundness.

Completeness: a decision table is complete if it contains an applicable rule for every legal configuration of input values. In other words, it is not possible that, for a given input, no rule fires.

$$\forall \vec{x} : \left(\bigwedge_{i \in I} Legal_i(x_i) \right) \Rightarrow \bigvee_{r \in R} \left(\bigwedge_{i \in I} F_{ir}(x_i) \right)$$

with $\vec{x}$ representing a set of input values, and $Legal_i$ a predicate that denotes all legal values for the variable x_i.

Soundness: a decision table with the U hit policy is sound whenever the rules of the table are mutually exclusive. As the soundness of a decision table does not change when taking context into account, we refer to the work of Calvanese et al. [10] for the formal criterion.

4.2. Unfireable rules

While Calvanese et al. [10] also specify a criterion for unfireable rules, they only do so for F tables. Indeed,

Sequence ID		
U	Station	ID
1	city	0
2	major	1

(a)

Stop		
U	ID	Stop
1	0	No
2	1	Yes

(b)

Figure 6: Example incorrect DMN tables

in F tables certain rules can be *masked* by others, and thus prevented from firing. However, it is also possible (though unlikely) to have unfireable rules in both U and A tables. As such, we propose our own criterion for such rules: for every row in the decision table, there should be a legal set of input values that satisfies it.

$$\bigwedge_{r \in R} \left(\exists \vec{x} : \bigwedge_{i \in I} \left(Legal_i(x_i) \wedge F_{ir}(x_i) \right) \right)$$

Note that this criterion will only rarely be violated, namely only if a condition rules out all possible values for an input variable, such as a condition "< 0" for a natural number variable.

4.3. Correctness criteria with context

Before extending the correctness criteria of completeness and unfireable rules with context, we elaborate on both types of context. In-model context consists of all tables that are not the table currently being verified.

Background knowledge, in the form of FO($\cdot$) formulae, is contained in the knowledge base *KB*. For example, the knowledge base of the *Sequence ID* example given in Section 3.2 consists of the following two implications:

$$Location = terminating \Rightarrow Status \neq departure.$$
$$Location = origin \Rightarrow Status \neq arrival.$$

The "variables" of DMN are represented by constants (e.g., *Location* and *Station*) in the FO KB and in the FO semantics Sem_T of a DMN table. In order to define our context-sensitive correctness criteria, however, we will need to be able to quantify over the DMN variables. Therefore, in the correctness criteria, the DMN variables will need to be represented by FO variables. Let $\vec{V}$ be the set of all FO constants that correspond to the DMN variables of the model. We introduce for each $V \in \vec{V}$ a unique FO variable x_V and define $\vec{x} = (x_V)_{V \in \vec{V}}$ and denote by $\phi[\vec{x}]$ the result of replacing each constant V in the formula ϕ by the variable x_V. For example, if ϕ is the formula

$$Location = terminating \Rightarrow Status \neq departure$$

then $\phi[\vec{x}]$ is the formula

$$x_1 = terminating \Rightarrow x_2 \neq departure$$

where $x_1 = x_{Location}$ and $x_2 = x_{Status}$. We now extend the completeness and unfireable rules criteria with both types of context.

Completeness When verifying the completeness criterion for a table T_j, we require that the set of variable values $\vec{x}$ satisfies all other tables T_i, $i \neq j$ and the background knowledge in the KB. The completeness property for table T_j with input variables $W_1, \ldots, W_m$ and rows R is defined as:

$$\forall \vec{x} : \bigwedge_{i \neq j} Sem_{T_i}[\vec{x}] \wedge KB[\vec{x}] \wedge \bigwedge_{k \in 1..m} Legal_i(x_{W_k})$$
$$\Rightarrow \bigvee_{r \in R} \bigwedge_{k \in 1..m} F_k r(x_{W_k}).$$

For example, consider the completeness verification of the table in Fig. 6a, with variables for *Station*, *ID* and *Stop*.

$$\forall p,q,r : \Big((q = 0 \Rightarrow r = No) \wedge (q = 1 \Rightarrow r = Yes) \Big)$$
$$\wedge Legal_p(minor) \wedge Legal_{ID}(q) \wedge Legal_{Stop}(r)$$
$$\Rightarrow (p = major \vee p = city))$$

We can conclude that the table is incomplete, as there is no row for *Station* = *minor* or for *Station* = *airport*.

Unfireable rules We extend the unfireable criterion in a similar manner to the completeness criterion, by adding the KB together with the other tables. In other words, for every row of table T_j, there should be a set of variable assignments $\vec{x}$ that satisfies all other tables, the KB, and the inputs of the row itself.

$$\bigwedge_{r \in R} \Bigg(\exists \vec{x} : \bigwedge_{i \neq j} Sem_{T_i}[\vec{x}] \wedge KB[\vec{x}]$$
$$\wedge \bigwedge_{k \in 1..m} Legal_i(x_{W_k}) \wedge \bigwedge_{i \in I} \Big(F_{ir}(x_i) \Big) \Bigg)$$

For example, consider verifying if the first rule of the table in Fig. 6b is fireable, using variables for *Station* and *ID*.

$$\exists p, q : (p = city \Rightarrow q = 0) \wedge (p = major \Rightarrow q = 1)$$
$$\wedge\ Legal_{Station}(p) \wedge Legal_{ID}(q) \wedge q = 0$$

Because the *city* is not a legal value for the variable *Station*, q can never be 0. As such, this first rule is unfireable.

5. Implementation

To show the practical applicability of this work, we have created a tool capable of decision table verification with context. Concretely, each of the three verification capabilities has been implemented using the IDP (Imperative Declarative Programming) system [16], a state-of-the-art logical reasoning engine. As an implementation of the Knowledge Base Paradigm (KBP) [17], the IDP system stores its knowledge in a Knowledge Base (KB), which can then be put to different uses by applying powerful inference tasks to it. As will be shown later on, this separation between knowledge and its use facilitates the reuse the same knowledge for different purposes. In IDP, the KB is created in FO($\cdot$), an extension of First Order (FO) logic adding types, aggregates and inductive definitions. To reason on the KB, IDP supports multiple forms of inferences, of which we use three in this work: propagation, model expansion and abstract model expansion.

A partial interpretation $\mathcal{I}$ for a vocabulary assigns a value to some, but not necessarily all, of the symbols in this vocabulary. Given such a partial interpretation $\mathcal{I}$ for the vocabulary of a theory T, the **propagation inference** computes the consequences of $\mathcal{I}$ according to T. To be more precise, the result of this inference is a new partial interpretation $\mathcal{I}'$ that extends $\mathcal{I}$ (i.e., $\mathcal{I}'$ interprets at least all symbols interpreted by $\mathcal{I}$ and, moreover, it interprets them in the same way) such that all models I of T that extends $\mathcal{I}$ still extend $\mathcal{I}'$.

The **model expansion inference**, given the KB theory T and a partial structure $\mathcal{I}$, searches for an interpretation I that extends $\mathcal{I}$ and that satisfies the KB theory T ($I \models T$). If such an expansion is found, it's returned as output, else, if no expansion can be found the system returns "Unsatisfiable".

Given a theory T and partial interpretation $\mathcal{I}$, the **abstract model generation** (AMG) inference searches for a set C of simple constraints that imply the theory, i.e., such that for all I that extend $\mathcal{I}$, $I \models C \rightarrow T$. Here, the simple constraints are conjunctions of literals, in which each literal is an (in-)equality on a single DMN variable, such as $18.5 < BMI \wedge BMI \leq 25$. Each interpretation that satisfies all of the constraints C is a model of T; in this sense, the constraints form an abstract representation of a class of models of T.

To verify a table j, we now convert the DMN model into an FO($\cdot$) theory that consists of $KB \cup \bigcup_{i \neq j} Sem_{T_i}$ (both types of context) and the formula

$$\bigwedge_{r \in R} Row(r) \Leftrightarrow \bigwedge_{i \in I} F_i r(H_i)$$

to represent table j, with *Row* a new predicate to represent if a row has fired. We will now go over every verification capability and explain how the IDP system can perform them.

Completeness To verify a table's completeness, we look for a set of assignments for which no row fires. If the IDP system is then unable to find any solution, we can conclude that the table is complete. We perform this check by adding a constraint to the KB stating that no row is allowed to fire.

$$\forall r : \neg Row(r).$$

We then run IDP's *model expansion* inference, to find a structure that satisfies the adapted KB theory T'. Because this KB includes the representations of the other tables and the background knowledge in the form of FO($\cdot$) formulae, we are effectively able to verify completeness w.r.t. both types of context. If no solution can be found by model expanding, the table is complete.

One weakness of this verification is that, in the case of an incomplete table, it is not easy to pin-point the exact gaps in its rules. We cannot use model expansion to generate all value assignments for which no rule is applicable, because there can be an infinite number of them when reasoning with integers and floats. However, we overcome this issue by using AMG. Indeed, this inference allows us to find the gaps in a decision table, with every abstract model representing a different gap in the rules.

Soundness To detect overlapping rules, we use the IDP system to find a set of value assignments that results in multiple rules of a table firing. Similarly to table completeness, we can do so by adding a new constraint:

$$\#\{r : Row(r)\} > 1.$$

This constraint can be read as "The number of rows for which *Row* is satisfied should be greater than 1". If IDP's model expansion is then incapable of finding a solution (i.e., no two rows can fire at the same time), the table is sound. Else, the solution will contain which specific rows overlap, allowing us to present this information.

Unfireable rules The IDP system's propagation inference task, can be used to detect unfireable rules. If the propagation derives that an atom of the form $Row(i)$ must be False, we know that row i cannot fire. If it does not derive any atoms of this form, the table is free of unfireable rules.

All of these described table verification algorithms have been implemented in DMN-IDP, a DMN modeller combined with an IDP-based interface [18]. It is available for demonstration online[1], and includes all the examples discussed in this paper. Internally, the conversion from DMN to FO(·) is done automatically using the same tool as used by Aerts et al. [19], which has been extended to generate the FO(·) representation for table verification. The specific version of the IDP system used in this work is IDP-Z3 [20].

6. Comparison and Evaluation

In this section, we give a brief comparison between our tool, the verification algorithms described by Calvanese et al. [9] and the state-of-the-art verification tool by Hasic et al. [8]. When compared to the approach by Calvanese et al., our work distinguishes itself in two ways. Firstly, our completeness verification does not only return a boolean output, but is also capable of identifying the missing rules. Similarly, our overlap detection and unfireable rule detection are also able to pin-point the specific rules causing the error. Secondly, we have created a concrete implementation of our verification methods, which has also been integrated in a DMN tool. In this way, we show that our approach is also practically feasible.

On the language level, the description logic $\mathcal{ALC}$ used by Calvanese et al. can be seen as a fragment of the FO(·) language, making our approach at least as expressive in theory. However, because FO(·) is not decidable (unlike $\mathcal{ALC}$), the IDP system can actually only reason with theories that obey certain restrictions on the domains over which the variables range. Variables that range over a finite domain are not a problem, and neither are certain uses of numerical variables within infinite domains. In particular, the theories that result from translating DMN variables with an infinite numerical domain can be handled. However, in general, IDP cannot perform all of the Open World reasoning of typical description logic reasoning engines over infinite domains. The relation between FO(·), IDP and description logics has been further described in [21].

To compare our implementation to the one by Hasic et al., we used both tools to verify the DMN models in Fig. 3 and Fig. 4. In the first model (3 tables, 17 rows, 4 inputs), as mentioned in Section 3.1, the *Risk Level* table contains two unfireable rules. The tool by Hasic et al. is unable to detect this. Our tool on the other hand is able to correctly identify the two unfireable rules, and considers the table error-free after their removal. The tool by Hasic et al., on the other hand generates a false positive for the fixed table, reporting missing rules for the cases that cannot occur.

Table 3: Comparison between table verification time in milliseconds.

	BMI (Risk Level)	Train sequence
Hasic et al. [8]	118	97
Our tool	1245	287

The second example (1 table, 9 rows, 3 inputs), as described in Section 3.2, is considered to be incomplete by the tool of Hasic et al., which states that it is missing the rules listed in Fig. 5. In contrast, our tool is able to use the background knowledge and conclude that this table is indeed correct.

The increased functionality of our tool comes at a computational cost, however. As shown in Table 3, the verification time of our tool is a magnitude higher compared to the tool by Hasic et al. when verifying the examples given in this paper. These timings were measured as the time it took for the servers to respond with the verification results. The reasons for the difference in efficiency are twofold: firstly, by keeping in mind the context of a table, we increase the verification complexity, as more information has to be verified. Secondly, we employ a general logic-based solver for our verification instead of highly optimized procedures.

7. Conclusion

Most state-of-the-art verification DMN verification algorithms tend to verify a table "in isolation", without regards to the rest of the model or to background knowledge.

In this paper, we have first explained the importance of context, extended the formal correctness criteria, and then proceeded to present a *context-aware* DMN verification tool. Instead of verifying isolated tables, this tool always keeps the other tables in mind. On top of this, the tool allows the addition of background knowledge in the form of FO(·), thus ensuring a more correct verification.

As we discussed in Section 6, our tool offers more functionality than the context-aware methods of

[1] https://dmn-idp.herokuapp.com/

Hasic et al. and Calvanese et al.

The verification tool has also been implemented as part of an existing DMN editor, allowing anyone to freely test it.

References

[1] Object Modelling Group, "Decision Model and Notation v1.3," February 2021.

[2] N. J. Car, "Using decision models to enable better irrigation decision support systems," *Computers and Electronics in Agriculture*, vol. 152, pp. 290–301, 2018.

[3] L. J. Sooter, S. Hasley, R. Lario, K. S. Rubin, and F. Hasić, "Modeling a clinical pathway for contraception," *Applied clinical informatics*, vol. 10, p. 935—943, October 2019.

[4] F. Hasic and J. Vanthienen, "From decision knowledge to e-government expert systems: the case of income taxation for foreign artists in belgium," *Knowledge and information systems*, vol. 62, no. 5, pp. 2011–2028, 2020.

[5] I. Dasseville, L. Janssens, G. Janssens, J. Vanthienen, and M. Denecker, "Combining DMN and the knowledge base paradigm for flexible decision enactment," vol. 1620 of *Supplementary Proceedings of the RuleML 2016 Challenge*, CEUR-WS.org, 2016.

[6] B. Aerts, S. Vandevelde, and J. Vennekens, "Tackling the dmn challenges with cdmn: A tight integration of dmn and constraint reasoning," in *Rules and Reasoning*, Lecture Notes in Computer Science, pp. 23–38, Cham: Springer International Publishing, 2020.

[7] K. Smit, M. Zoet, and M. Berkhout, "Verification capabilities for business rules management in the Dutch governmental context," in *2017 international conference on research and innovation in information systems (ICRIIS)*, pp. 1–6, 2017. tex.organization: IEEE.

[8] F. Hasic, C. Corea, J. Blatt, P. Delfmann, and e. Serral Asensio, "A Tool for the Verification of Decision Model and Notation (DMN) Models," in *Proceedings of the 2020 14th International Conference on Research Challenges in Information Science (RCIS)*, pp. 1–6, Springer, 2020.

[9] D. Calvanese, M. Montali, M. Dumas, and F. M. Maggi, "Semantic DMN: Formalizing and reasoning about decisions in the presence of background knowledge," *Theory and practice of logic programming*, vol. 19, no. 4, pp. 536–573, 2019.

[10] D. Calvanese, M. Dumas, U. Laurson, F. M. Maggi, M. Montali, and I. Teinemaa, "Semantics, analysis and simplification of DMN decision tables," *Information systems (Oxford)*, vol. 78, pp. 112–125, 2018.

[11] D. Calvanese, M. Dumas, Ü. Laurson, F. M. Maggi, M. Montali, and I. Teinemaa, "Semantics and analysis of dmn decision tables," in *Business Process Management* (M. La Rosa, P. Loos, and O. Pastor, eds.), (Cham), pp. 217–233, Springer International Publishing, 2016.

[12] U. Laurson and F. M. Maggi, "A Tool for the Analysis of DMN Decision Tables.," in *BPM (Demos)*, pp. 56–60, 2016.

[13] K. Batoulis and M. Weske, "A tool for checking soundness of decision-aware business processes.," in *BPM (demos)*, pp. 1–5, 2017.

[14] C. Corea, J. Blatt, and P. Delfmann, "A tool for decision logic verification in dmn decision tables.," in *BPM (PhD/Demos)*, pp. 169–173, 2019.

[15] V. Etikala, Z. V. Veldhoven, and J. Vanthienen, "Text2Dec: Extracting Decision Dependencies from Natural Language Text for Automated DMN Decision Modelling," in *Business Process Management Workshops* (A. Del Río Ortega, H. Leopold, and F. M. Santoro, eds.), (Cham), pp. 367–379, Springer International Publishing, 2020.

[16] B. De Cat, B. Bogaerts, M. Bruynooghe, G. Janssens, and M. Denecker, "Predicate logic as a modeling language: the IDP system," in *Declarative Logic Programming: Theory, Systems, and Applications* (M. Kifer and Y. A. Liu, eds.), pp. 279–323, ACM, Sept. 2018.

[17] M. Denecker and J. Vennekens, "Building a Knowledge Base System for an Integration of Logic Programming and Classical Logic," in *Logic Programming* (M. Garcia de la Banda and E. Pontelli, eds.), vol. 5366, pp. 71–76, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. Series Title: Lecture Notes in Computer Science.

[18] S. Vandevelde and J. Vennekens, "A multifunctional, interactive DMN decision modelling tool," in *Proceedings of BNAIC/BeneLearn 2020*, pp. 399–400, Leiden University, 2020.

[19] B. Aerts, S. Vandevelde, and J. Vennekens, "Tackling the dmn challenges with cdmn: A tight integration of dmn and constraint reasoning," in *Rules and Reasoning: 4th International Joint Conference*, vol. abs/2005.09998, pp. 23–38, Gutiérrez Basulto, Victor, Springer International Publishing, 2020.

[20] P. Carbonnelle, B. Aerts, M. Deryck, J. Vennekens, and M. Denecker, "An interactive consultant," in *Proceedings of the 31st benelux conference on artificial intelligence (BNAIC 2019) and the 28th belgian dutch conference on machine learning (benelearn 2019), brussels, belgium, november 6-8, 2019*, vol. 2491 of *CEUR workshop proceedings*, CEUR-WS.org, 2019.

[21] J. Vennekens, M. Denecker, and M. Bruynooghe, "FO(ID) as an extension of DL with rules," *Annals of Mathematics and Artificial Intelligence*, vol. 58, no. 1, pp. 85–115, 2010.