# Semantics and Analysis of DMN Decision Tables (Extended Abstract)

Marlon Dumas* Diego Calvanese† ´Ulari Laurson‡ Fabrizio Maria Maggi§
Marco Montali¶ Irene Teinemaa‖

Business rules play a central role in modern information systems. Among other things, business rules are routinely used to automate operational decisions during the execution of business processes. Business rules are used for example to automatically determine how purchase orders, shipments, invoices and approval requests are routed throughout an organization. In this setting, it is critical to explicitly capture business rules in a way that is unambiguous and understandable by a wide range of stakeholders – including managers, analysts and process workers.

For already several decades, decision tables are one of the tools of choice for capturing complex business rules. In recent years, a standard for decision tables, namely the Decision Model and Notation (DMN), has brought much-needed consolidation to the field. The DMN notation provides mechanisms for capturing decisions at two levels of abstraction: first at an architectural level by allowing decisions to be modularized, inter-linked, and connected to their corresponding data inputs and knowledge sources via so-called Decision Requirements Diagrams (DRDs), and second by allowing atomic decisions to be specified by means of decision tables enhanced with metadata properties such as "hit policies", "priority indicators" and "completeness indicators". Decision tables in DMN can be of different levels of sophistication, depending on whether the expressions in the table are specified using the so-called FEEL (Friendly Enough Expression Language) or its simplified version, namely S-FEEL (Simplied FEEL).

A DMN table consists of columns corresponding to input or output attributes, and rows corresponding to rules. Each column has a type (e.g., a string, a number, or a date), and optionally a more specific domain of possible values, which we hereby call a *facet*. Each row has an identifier, one expression for each input column (a.k.a. the *input entries*), and one specific value for each output column (the *output entries*). For example, Table 1 shows a DMN table with two input columns, one output column and four rules.

Given an input configuration consisting of a vector of values (one entry per column), if every input entry of a row holds true for this input vector, then the vector *matches* the row and the output entries of the row are evaluated. For example, vector $\langle 500, 4230 \rangle$ mat-

*University of Tartu, Estonia marlon.dumas@ut.ee
†Free University of Bozen Bolzano, Italy calvanese@inf.unibz.it
‡University of Tartu, Estonia ulaurson@ut.ee
§University of Tartu, Estonia fm.maggi@ut.ee
¶Free University of Bozen Bolzano, Italy montali@inf.unibz.it
‖University of Tartu, Estonia irene.teinemaa@ut.ee

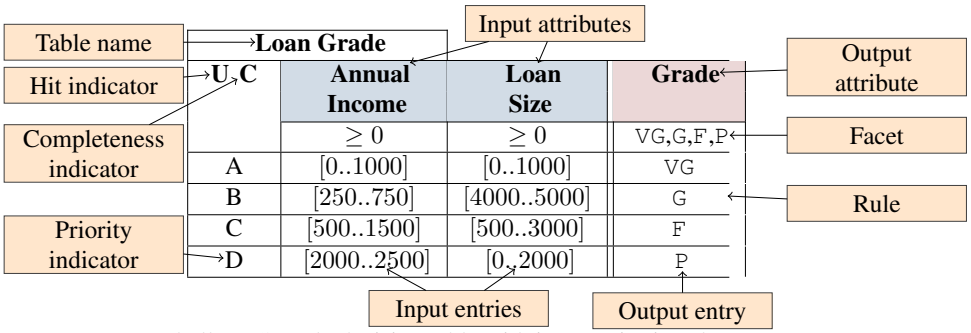

Tabelle 1: Sample decision table with its constitutive elements

ches rule $B$ in Table 1, thus yielding G in the output configuration. To specify how output configurations are computed from input ones, a DMN table has a *hit indicator* and a *completeness indicator*. The hit indicator specifies whether only one or multiple rows of the table may match a given input, and if multiple rules match an input, how should the output be computed. The completeness indicator specifies whether every input must match at least one rule or potentially none. If an input configuration matches multiple rules, this may contradict the hit indicator. Similarly, if no rule matches an input configuration, this may contradict the completeness indicator. The former contradiction leads to *overlapping rules* while the latter leads to *missing rules*.

The increasing use of DMN decision tables to capture complex and critical business decisions raises the need to support the analysis and refactoring of these tables, for example to ensure their correctness and to enhance their comprehensibility. In this respect, two common analysis tasks are the detection of overlapping rules and of missing rules as discussed above. Meanwhile, a common refactoring task is to simplify a decision table by merging pairs of rules into a single more general rule capturing the same business logic as the two original rules.

The analysis of classic (pre-DMN) decision tables has been widely studied in the literature for already several decades. However, different analysis and optimization tasks are approached differently, and the proposed analysis algorithms are usually restricted to categorical data types, as opposed to combinations of numerical and categorical (or string) datatypes as supported in DMN decision tables. Moreover, DMN decision tables lack a formal semantics to unambiguously reason about their properties.

To address the above gaps, we have specified a formal semantics of DMN decision tables and we have designed a versatile approach to analyze decision tables by means of geometric algorithms, specifically sweep-line algorithms. The key idea of our proposal is that a DMN decision table in the S-FEEL notation can be mapped to a set of iso-oriented hyper-rectangles (one per rule) in an N-dimensional space (where N is the number of attributes in the table). Under this geometric interpretation, the problem of detecting overlapping rules is mapped to that of detecting maximal sets of overlapping hyper-rectangles; the problem of finding missing rules is mapped to that of determining if the union of the rules in the table (viewed as hyper-rectangles) covers the N-dimensional space identified by the

cartesian product of the domains of the input columns of the table; and finally, the problem of simplifying a decision table is mapped to that of finding maximal sets of adjacent hyper-rectangles and merging them into a minimal set of hyper-rectangles covering the same space. We contend that the same geometric approach can be used to tackle other analysis and refactoring tasks such as for example that of transforming a DMN table with a multi-hit policy into an equivalent unique-hit table.

Based on the above ideas, we have proposed algorithms to tackle the problems of detecting overlapping and missing rules in unique-hit policy tables as well as an algorithm for simplifying a decision table via rule merging. We have implemented these algorithms atop the dmn-js open-source DMN editor. Our dmn-js extension with verification and simplification features can be found at `https://github.com/ulaurson/dmn-js` and a deployed version is available at `http://dmn.cs.ut.ee`.

Based on this implementation, we have experimentally compared the proposed approach against classic decision table analysis algorithms, using a collection of decision tables of varying sizes derived from a credit lending dataset.

The work summarized in this extended abstract has been published in [CDL+16]. In more recent follow-up work, we have also studied the problems of detecting overlapping and missing rules when a decision table is combined with additional background knowledge captured by means of a decription logic ontology equipped with datatypes [CDMM17].

**Acknowledgement.** This research was partly funded by the Estonian Research Council (Grant IUT20-55).

## Literatur

[CDL+16] Diego Calvanese, Marlon Dumas, Ülari Laurson, Fabrizio Maria Maggi, Marco Montali und Irene Teinemaa. Semantics and Analysis of DMN Decision Tables. In *Proc. of BPM*, Seiten 217–233. Springer, 2016.

[CDMM17] Diego Calvanese, Marlon Dumas, Fabrizio Maria Maggi und Marco Montali. Semantic DMN: Formalizing Decision Models with Domain Knowledge. In *Proc. of RuleML+RR*. Springer, 2017. To appear.