Discovery of Fuzzy DMN Decision Models from Event Logs

Conference Paper · May 2017

DOI: 10.1007/978-3-319-59536-8_39

CITATIONS
7

READS
479

5 authors, including:

Ekaterina Bazhenova
Hasso Plattner Institute
14 PUBLICATIONS **167** CITATIONS
SEE PROFILE

Stephan Haarmann
Hasso Plattner Institute
14 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Sven Ihde
Hasso Plattner Institute
9 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Andreas Solti
Wirtschaftsuniversität Wien
56 PUBLICATIONS **1,144** CITATIONS
SEE PROFILE

Discovery of Fuzzy DMN Decision Models from Event Logs

Ekaterina Bazhenova[1], Stephan Haarmann[1*], Sven Ihde[1*], Andreas Solti[2], and Mathias Weske[1]

[1] Hasso Plattner Institute at the University of Potsdam, Germany
{firstname.lastname}@hpi.de
*{firstname.lastname}@student.hpi.de
[2] Vienna University of Economics and Business, Austria
{solti@ai.wu.ac.at}

Abstract. Successful business process management is highly dependent on effective decision making. The recent Decision Model and Notation (DMN) standard prescribes decisions to be documented and executed complementary to processes. However, the decision logic is often implicitly contained in event logs, and "as-is" decision knowledge needs to be retrieved. Commonly, decision logic is represented by rules based on Boolean algebra. The formal nature of such decisions is often hard for interpretation and utilization in practice, because imprecision is intrinsic to real-life decisions. Operations research considers fuzzy logic, based on fuzzy algebra, as a tool dealing with partial knowledge. In this paper, we explore the possibility of incorporating fuzziness into DMN decision models. Further, we propose a methodology for discovering fuzzy DMN decision models from event logs. The evaluation of our approach on a use case from the banking domain shows high comprehensibility and accuracy of the output decision model.

Keywords: Fuzzy Logic, Decision Models, Decision Mining, DMN, Event Logs

1 Introduction

Decisions play an important role in enterprise's business processes. This has become a premise for an increased interest of academia and industry towards different aspects of integrated process and decision management. Recently, the Decision Model and Notation (DMN) was released [17], which prescribes modeling decisions complementary to processes. The combined usage of the DMN standard and Business Process Model and Notation (BPMN) [16] is recommended in order to encapsulate process and decision logic, and to increase reuse of decisions in processes.

Decision logic is normally represented by *crisp* decision rules based on Boolean algebra. This imposes certain limitations on its application, as imprecision is often involved in real-life decisions. Operations research considers *fuzzy* logic, based on fuzzy algebra, as a tool dealing with partial input knowledge [22,6]. Fuzzy rules represent strings encoding the semantic meaning of a certain probability behind a value range, e.g., "If loan duration is long, then risk is very high". Since the meaning can be derived directly from the representation, it is generally considered that fuzzy rules are more

comprehensible compared to crisp rules [12]. Moreover, using literals across rules increases decision flexibility, as it allows a consistent adaptation of all rules by adjusting only the underlying mappings. In this paper, we incorporate fuzziness into DMN decision models by utilizing fuzzy decision tables, and introduce fuzzy hit policies.

Further, to assist enterprises with efficient decision management, knowledge about "as-is" decision making needs to be retrieved. This can be done by analysing process event logs and discovering decision rules from them. Existing approaches to decision discovery consider only either mining of independent fuzzy rule sets [12], or crisp decision models [18,5,9]. In order to provide useful insights into the discovery of fuzzy DMN decision models from event logs, we propose a methodology which consists of five steps: (1) Identification of fuzzy subsets and corresponding membership functions corresponding to the data attributes from the event log; (2) Discovery of process decisions and dependencies between them; (3) Application of fuzzy learners for fuzzy rules discovery; (4) Construction of fuzzy decision tables based on discovered rules; (5) Identification of fuzzy hit policy for decision tables. In order to test the validity of the proposed approach, we implement it and conduct validation experiments on a simulated event log for the use case of loan assessment in banks. The evaluation results show good interpretability and accuracy of the output fuzzy DMN decision model.

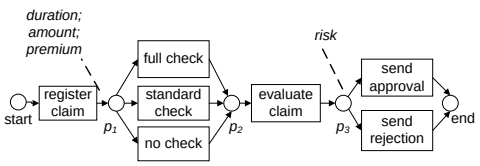
The remainder of this work is structured as follows. Section 2 introduces the motivation for deriving fuzzy DMN models on an example, followed by related work. Section 3 presents our formal framework. Section 4 introduces the methodology for mining fuzzy decision models, which is followed by Evaluation and Conclusion sections.

2 Background

In this section, we motivate the need for our approach and describe related work.

2.1 Motivating example

We consider a business process of credit-risk assessment (Fig. 1a). This use case is inspired by a real data set from a major Benelux financial company, also used in [3].



(a) Petri Net of the process

Event ID	Trace ID	Activity	Attributes
1	1	register claim	duration=14 [mths] amount=9700 [EUR] premium=FALSE
2	1	full check	-
3	2	register claim	duration=28 [mths] amount=2000 [EUR] premium=TRUE
4	2	no check	-
5	1	evaluate claim	risk=high
6	1	send rejection	-

(b) Process event log

Fig. 1: Example business process of credit-risk assessment

The process starts with the registration of the user's claim details such as *duration, amount*, and *premium* in the bank information system. There are two decisions involved in this process, which according to our interviews with a partner bank are performed by loan experts in about 20% of cases. At p_1 the type of application check is chosen, and at p_3 the claim is evaluated for approval which happens alongside with assigning *risk* to the claim. An example execution log is given in Fig. 1b.

The problem is that this process model does not explicitly contain the decision rules, so the decisions done by experts are only implicitly contained in the event log. There exist many works using statistical means for discovery of decision rules [21], but direct application of them can lead to the unjust treatment of an individual applicant, e.g., judging creditability just by the applicant's nationality [7]. Thus, for companies it is important to have interpretive models explaining how a certain decision is made. To achieve this, we propose to derive DMN decision models, explaining and representing explicitly process decisions, from event logs. An example of a DMN model consisting of a decision requirements diagram (DRD) and decision logic layers is shown in Fig. 2.

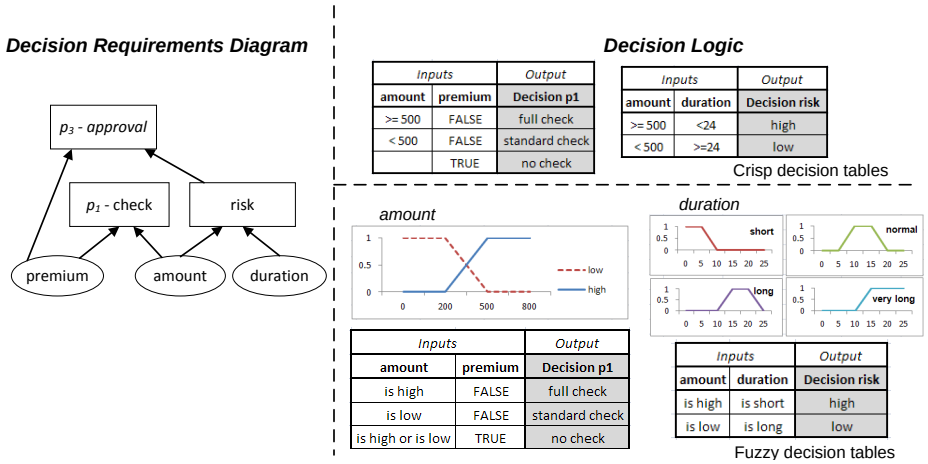


Fig. 2: DMN decision requirements diagram and two types of decision logic

The decision logic corresponding to the DRD can be designed in the form of either crisp, or fuzzy decision tables, as shown in the figure. Fuzzy rules are considered to be more comprehensible, as a user has an intuitive understanding of the linguistic concepts to which fuzzy rules refer. An example of such mapping is a trapezoidal membership function representing the membership grade of attribute *amount* in fuzzy sets "low", and "high". Thereby, flexibility of fuzzy decision models is higher as the change of rules can be executed just by adjusting the membership functions. To the best of our knowledge, the question of deriving fuzzy DMN decision models from event logs has not received attention yet. In the next section, we provide an overview of related works.

2.2 Related work

The presented problem was addressed in the literature from different perspectives. The first category of related work deals with mining of independent fuzzy rule sets. In [8,11,15], a concept of fuzzy rule bases and techniques to design them are introduced. Fuzzy rules learning as introduced in works [12,13] can be applied for solving our problem, but process context and metrics were not taken into account. Fuzzy mining of rules represent a more general case of mining crisp rule bases from data [21,3]. The incorporation of fuzziness within the concept of decision tables was done in [19], but this work only accounts for the design of decision tables, and no knowledge discovery is applied.

With respect to process context, one of the first works in this direction was done in [18] for discovery of control flow decisions in processes from event logs. Mining

of complex decision logic for process branching conditions with the help of decision tree learning was done in [9], and in [14]. These two works do not mine dependencies between process decisions, and they also do not consider fuzziness in decisions. Discovery of DMN decision models from control-flow structures of process models was done in [4], but this work does not consider the execution level. In [5], the discovery of DMN decision models from event logs was done, but only crisp decision logic was considered. The derived DMN decision models are supposed to be used complementary to process models which raises novel research questions, e.g., which elements should belong to the process, and which elements should be a part of the decision model. Exploration of issues connected to the separation of concerns is presented in [20,10].

3 Formal Framework

In this section, we introduce needed definitions as follows.

Definition 1 (Event Instance, Attribute, Trace, Log). Let $E=\{e_1, ..., e_n\}, n\in\mathbb{N}^+$ be a finite set of n event instances and $A=\{a_1, ..., a_v\}, v\in\mathbb{N}^+$ a finite set of v attributes. Each attribute $a\in A$ is associated with a domain $D(a)$, which represents a set of either numeric, or nominal values. Each event instance $e\in E$ has tuples $(a, d(a))$, $a\in A$, $d(a)\in D(a)$ assigned to it. A trace is a finite sequence of event instances $e\in E$, such that each event instance appears in it once. An event log L is a multi-set of traces over E. $\diamond$

Fig. 1b shows an example event log where event instances are labeled by names of executed activities associated with values of data attributes attached to them.

Definition 2 (Decision Requirements Diagram). A *decision requirements diagram* *DRD* is a tuple (D_{dm}, ID, IR) consisting of a finite non-empty set of decision nodes D_{dm}, a finite set of input data nodes *ID*, and a finite non-empty set of directed edges *IR* representing the information requirements such as $IR \subseteq D_{dm}\cup ID \times D_{dm}$, and $(D_{dm}\cup ID, IR)$ is a directed acyclic graph. $\diamond$

A decision may additionally reference the decision logic level where its output is determined through an undirected association. One of the most widely used representation for decision logic is a decision table, which we utilize for the rest of the paper. Fig. 2 demonstrates an example DRD and corresponding decision tables.

Facilitation of fuzzy logic implies that the crisp values of data attributes from the event log are to be perceived as a matter of truth value ranging between completely true and completely false. In order to facilitate handling of decision rules in a fuzzy manner, below we introduce a set of necessary definitions. Given is an event log L, and a corresponding set of v attributes of the event instances $A=\{a_1, \ldots, a_v\}, v\in\mathbb{N}^+$.

Definition 3 (Fuzzy Subset, Membership Function). A domain set $D(a)$ for an attribute $a\in A$ has K^a *fuzzy subsets* characterized by tuples $FS_i^a = \{(d, l_i^a, \mu_{l_i}^a) \mid d\in D(a)\}$, if for each $i \in [1;\ K^a]$ there exist:

- linguistic terms l_i^a labelling fuzzy subsets FS_i^a;
- *membership functions* $\mu_{l_i}^a : D(a) \longrightarrow [0, 1]$ which represent the grade of membership of attribute values from $D(a)$ in a fuzzy subset FS_i^a by mapping each value $d\in D(a)$ to a real number in the interval $[0, 1]$. $\diamond$

We identify a fuzzy set with its membership function. An example of a membership function is presented with the help of a graph in Fig. 2 for the attribute *amount*. In this case, $K^{amount}=2$, and the domain set $D(amount)$ has two fuzzy subsets FS_1^{amount} and

FS_2^{amount} characterized correspondingly by membership functions μ_{low}^{amount} and μ_{high}^{amount}. This example is illustrated with the help of *trapezoidal membership function*, as one of the most widely used types of membership functions. However, the approach proposed in our paper is general and can be used for the other types of membership functions, e.g., triangular-shaped, Gaussian curve, etc. [6].

Fuzzy logic of decision models can be expressed through fuzzy decision rules and fuzzy decision tables. Fuzzy decision tables consist of fuzzy rules which represent "if-then" mapping between a subset of event log attributes associated with corresponding linguistic terms, and an output process variable associated with a set of labels.

Definition 4 (Elemental Fuzzy Decision Rule, Fuzzy Decision Table). An *elemental fuzzy decision rule (FDR)* is a mapping:

$$R_j : a_1 \text{ is } l_{x_1}^{a_1}, \ldots, a_w \text{ is } l_{x_w}^{a_w} \longrightarrow c_{j,z}, \tag{1}$$

where the attributes $\{a_1, \ldots, a_w\} \subseteq A$ are inputs, $l_{x_1}^{a_1}, \ldots, l_{x_w}^{a_w}$ are linguistic terms labeling fuzzy subsets associated with the inputs, and $c_{j,z}$ is the $z-th$ label of a process variable serving as a rule output, such that $z \in \mathbb{N}^+,\ 1 \leq w \leq v, |A| = v, 1 \leq x_1 \leq K^{a_1}, \ldots, 1 \leq x_w \leq K^{a_w}$. A *fuzzy decision table (FDT)* is a set of fuzzy decision rules $\bigcup R_j,\ j \in \mathbb{N}^+$. $\diamond$

Below are examples of elemental fuzzy decision rules (see also FDT in Fig. 2):

$$\textit{amount is high},\ \textit{premium is FALSE} \longrightarrow p_1 = \textit{full check} \tag{2}$$

$$\textit{amount is low},\ \textit{duration is long} \longrightarrow \textit{risk} = \textit{low} \tag{3}$$

The definition of elemental fuzzy rule sets can be extended to *conjunctive normal form of fuzzy rules (CNF)*, where each attribute $a_i, 1 \leq i \leq v$ can be associated with a set of $p \in \mathbb{N}^+$ several linguistic terms $\{l_{x_i,1}^{a_i}, \ldots, l_{x_i,p}^{a_i}\}$ which are joined by a disjunctive operator. Fuzzy rule bases in CNF form are also called Mamdani rules [6]. An example of CNF fuzzy rule is following (see also FDT in Fig. 2):

$$\textit{amount is high or low},\ \textit{premium is TRUE} \longrightarrow p_1 = \textit{no check} \tag{4}$$

Mapping a runtime input for a set of fuzzy rules to an output is called *inference*. There exist multiple inference techniques, and the method that is most closely matching the real-world problem should be chosen [6]. We address this question further in the paper.

4 Methodology

DMN decision model consists of a DRD representing decisions and its dependencies, and of a decision logic layer expressed by sets of rules. Fuzziness is not relevant for DRDs, as they store names of decisions and dependencies between them (see Fig. 2). In contrast, decision rules might incorporate fuzziness. Such a view does not violate the DMN standard, so we adopt it and use the term Fuzzy DMN (FDMN) for such kind of decision models. This section consists of five subsections representing consequent steps in our methodology of mining FDMN from event logs, see Fig. 3:

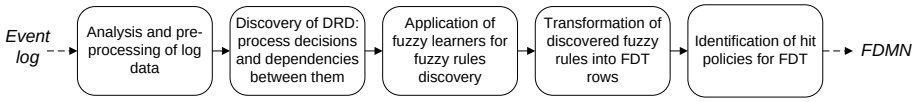


Fig. 3: Our approach for discovering Fuzzy DMN (FDMN) models from event logs

4.1 Analysis and Preprocessing of Log Data

Given is a an event log L, and a corresponding set of attributes of the event instances $A=\{a_1,\ldots,a_v\}, v\in\mathbb{N}^+$. Firstly, we preprocess log data in such a way that fuzzy learning of rules can be applied on it. For that, for each attribute there should exist membership functions that describe them in linguistic terms, e.g., that the value of an attribute is "low" or "high". Specifically, we need to find $\mu^a_{l_i} : D(a)\longrightarrow[0,1]$ for each attribute $a\in A$ from the event log, which we propose to do by the procedure CONSTRUCTMF from Algorithm 1. In the procedure, we iterate over all attributes from the event log. If

Algorithm 1 Preprocessing of Data Attributes from Event Log

```
1:  procedure CONSTRUCTMF(eventLog L, attributes A)
2:      for all a ∈ A do
3:          if D(a) is a set of nominal values then
4:              if d(a) = l then
5:                  μ^a_l = 1
6:              else
7:                  μ^a_l = 0
8:          else
9:              K^a ← a number of fuzzy subsets                    ▷ Expert input
10:             for all i ∈ [1; K^a] do
11:                 l^a_i ← assign a linguistic term               ▷ Expert input, e.g., "low", "high"
12:                 μ^a_l = BUILDNUMATTRIBUTESMF()                 ▷ Either expert input or FCM
```

the attribute domain is nominal by itself, the membership function is constructed as a characteristic function (Lines 4–7). Otherwise, if the attribute domain is numerical, an expert should set the number K^a of fuzzy subsets for this domain, that are to be generated, and assign corresponding linguistic terms. For our use case (see Fig. 2), the expert would set $K^a=2$ for attribute *amount*, and assign two linguistic terms: *low* and *high*.

Function BUILDNUMATTRIBUTESMF can be realized using two approaches. The first approach involves experts which express their opinions on how well attributes $a\in A$ can be associated with fuzzy subsets $FS^a_i,\ i\in[1;\ K^a]$. The details of construction of membership functions based on expert opinions are out of scope of this current paper, as they can be found in [23]. The expert approach can be recommended when there is not enough input data allowing to derive the membership function automatically. As the input event log in our case is supposed to be large, we propose to utilize the second approach – the *Fuzzy C-Means (FCM)* algorithm described in [8], as it allows to automatically derive membership functions for data attributes from event logs. Applicable to our case, the FCM-algorithm is able to calculate the degree of membership $\mu^a_{l_i}$ of data attribute value a from the event log in each of i clusters, such that $1\leq i\leq K^a$. FCM requires the number of clusters as an input, which is equal to the number of fuzzy subsets obtained from an expert in Line 9. Example membership functions for our use case are visualized in Fig. 2 for variables *amount* and *duration* with respective values.

4.2 Discovery of DRD

Discovery of DRDs from event logs represents a classification problem over crisp data. We distinguish between two types of process decisions: (1) *control flow decisions* represented in process models by split gateways (e.g., decision from Equation 2), and (2) *data decisions* reflecting dependencies between values of data attributes in the event log

(e.g., decision from Equation 3). We use the C4.5 classifier for learning both types of decisions taking event log attributes and transition labels as features, because it generally delivers a computationally inexpensive classification based on few comprehensible business rules, with a small need for customization [21]. For finding dependencies between decisions, we utilize the same classifier, taking the event log attributes and found decision outcomes as features. Each mined decision is added to the set of decision nodes D_{dm} of the output DRD. Each attribute influencing these decisions which is not a decision by itself is added to the set of input data nodes *ID* of the output DRD. All dependencies are added to the set of information requirements *IR* of the output DRD.

The output of this step is a DRD (see Fig. 2 for an example) consisting of a set of $PDN \in \mathbb{N}^+$ decisions $\{(A_1, C_1), ..., (A_{PDN}, C_{PDN})\}$, where *PDN* stands for process decisions number. Herewith, $A_{pdn} = \{a_1, ..., a_g\} \subseteq A,\ 1 \leq g \leq v, pdn \in [1, ..., PDN]$, are attributes influencing these decisions. A more detailed description of mining DRDs from event logs can be found in our previous work [5].

4.3 Application of fuzzy learners for fuzzy rules discovery

After discovering process decisions $(A_{pdn}, C_{pdn}),\ 1 \leq pdn \leq PDN$ from the event log, we aim at discovering of fuzzy rules corresponding to these decisions. Thus, we iterate over the set of *PDN* decisions and solve for each of them the classification problem presented in Fig. 4. Here the training data is comprised in the event log subset $L_{pdn} \subseteq L : A_{pdn} \subseteq A$ containing only values of attributes that are influencing the given process decision.

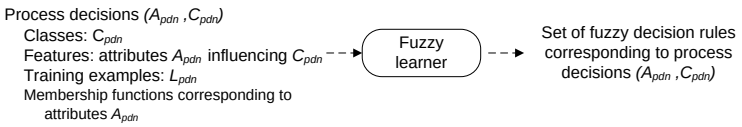


Fig. 4: Classification problem of discovering fuzzy decision rules from event logs

As our goal is to output the explanatory models describing decisions made in the past and recorded in the event log, we take into account that the fuzzy learners should provide good *accuracy* and *interpretability* of results. To explore the appropriateness of known fuzzy classification algorithms in achieving our goals, we did experiments on applying the genetic [12] and NEFCLASS [15] algorithms, both of which are well-known fuzzy classifiers that infer fuzzy classification rules. Below we provide short descriptions of algorithms taking into account modifications needed for solving our problem of deriving fuzzy rule sets from event logs.

4.3.1 Genetic algorithm (GA) Genetic algorithm (GA) is successfully applied for solving fuzzy classification tasks [8,11,12]. Also, GA suits us because it is applicable for training sets with large dimensions, which is typical for event logs. The heuristic character of GA does not guarantee optimality of solution, but it provides approximate solutions close to optimal, which is appropriate for business environments. Below we consider the adaptation of GA applicable to our problem.

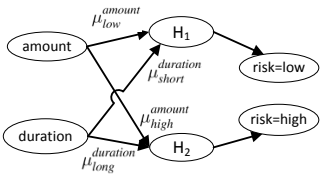
GA0. For utilizing GA, we firstly need to represent our problem in a genetic form. For that, we view fuzzy decision rules (FDR) discovered from event logs as so called chromosomes. Given is a process decision (A_{pdn}, C_{pdn}) where $A_{pdn} = \{a_1, ..., a_g\} \subseteq A$, $1 \leq g \leq v$ is a set of influencing attributes from the corresponding event log subset L_{pdn}.

Definition 5 (Chromosome). A *chromosome* corresponding to the event log subset L_{pdn} is a string G_t, $t \in \mathbb{N}^+$, which is a concatenation of two bit strings S_t and B_t, where:
- $S_t = \{s_1, ..., s_v\}$ is a string of bits $s_i, 1 \leq i \leq g$ indicating the presence of an attribute $a \in A_{pdn}$ in a rule antecedent of the chromosome;
- $B_t = \{b_1, ..., b_u\}$ is a string of bits b_k denoting the presence of a linguistic term l_i^a labeling fuzzy subsets $FS_i^a, i \in [1;\ K^a]$ in the rule antecedent of the chromosome, where $u, k \in [1;\ \sum_{j=1}^{g} K^{a_j}]$. ◇

Fuzzy rules are composed of a chromosome serving as the rule antecedent, and of a consequent that is chosen according to the majority class of the training instances covered by the rule antecedent. Example fuzzy rules are presented in Fig. 5a.

	s^{amount}	$s^{duration}$	$s^{premium}$	l^{amount}_{low}	l^{amount}_{high}	$l^{duration}_{short}$	$l^{duration}_{long}$	$l^{premium}$	p1
Rule from Eq.2	1	0	1	0	1	0	0	0	full check
Rule from Eq.4	1	0	1	1	1	0	0	1	no check

(a) Example chromosomes with corresponding consequents for rules from Equations 2, 4



(b) Example perceptron

Fig. 5: Example representation of rules for the GA and NF classifiers

GA1. At this step of GA, in each iteration $t \in \mathbb{N}^+$ a chromosome consisting of a fuzzy rule G_t is randomly generated. Next, the rule error $E(G_t)$ is computed by checking its degree of matching between the rule antecedent of instances from the event log subset L_{pdn} and the generated rule antecedent:

$$E(G_t) = \frac{\sum_{a|c_a \neq c_t} w_a \, \mu^a_{G_t}}{\sum_a w_a \, \mu^a_{G_t}}, \tag{5}$$

We utilize a boosting approach from [12] which modifies iterative fuzzy rule learning in an incremental fashion. The idea is to repeatedly train a weak classifier on various training data distributions, which shows a considerable improvement in accuracy of the results. The weights of event instances $e_k, 1 \leq k \leq n$ from the current distribution that are correctly classified by the rule G_t are reduced by a factor reflecting the error rate of the generated rule: $w_k(t+1) = w_k[(1 - E(R_t))/E(G_t)]^{\mu^a_{R_t}}$. Misclassified or uncovered examples keep their original weights. Thereby, boosting increases the relative weight of those examples which are "hard to learn". We calculate the chromosome consequent c_t as an output of the boosting classifier considering the vote of the rule G_t, weighted by logarithmic accuracy, on event instances from the event log L_{pdn} by t-norm (cf. [12]):

$$c_t = \underset{c_m}{\operatorname{argmax}} \sum_{G_t|c_t=c_m} \log \frac{1 - E(G_t)}{E(G_t)} \min_{j=1}^{w(t)} \mu^a_{l_j} \tag{6}$$

GA2. For building next generation of chromosomes, the rules that have a good fitness are kept. We calculate fitness function as a product of normalized values of such functions as class coverage *CC*, rule coverage *RC*, and rule consistency *RCS*:

$$f(G_t) = \overline{CC(G_t)} \times \overline{RC(G_t)} \times \overline{RCS(G_t)} \tag{7}$$

The class coverage *CC* is defined as the ratio of the number of training instances covered by the rule G_t to the overall number of training instances carrying the same class label c_i: $CC(G_t) = \sum_{a|c_a=c_t} w_a \, \mu^a_{G_t} \,/\, \sum_{a|c_a=c_t} w_a$.

$$RC(G_t) = \begin{cases} 1, & n{>}k_{cov}; \\ \dfrac{1}{k_{cov}} \dfrac{\sum_{a|c_a=c_t} w_a\, \mu^a_{G_t}}{\sum_a w_a}, & o/w. \end{cases} \qquad RCS(G_t) = \begin{cases} 0, & n_c^+ \times \varepsilon < n_c^-; \\ \dfrac{n_c^+ - \dfrac{n_c^-}{\varepsilon}}{n_c^+}, & o/w. \end{cases} \tag{8}$$

The rule coverage *RC* reflects the fraction of instances covered by the rule k_{cov}, irrespective of the class label. If $M{\in}\mathbb{N}^+$ is a number of classes having the same number of instances in the event log, a reasonable choice for fraction of covered instances is $k_{cov}{=}1/M$, as no rule can cover more than such fraction of instances without covering other (false) instances. For example, the number of classes for decision p_1 (see Fig. 2) is equal to 3, therefore, k_{cov} is chosen as 0.33. The rule consistency *RCS* demands that a rule covers a large number of correctly classified weighted instances $n_c^+ = \sum_{a|c_a=c_t} w_a\, \mu^a_{G_t}$, and a small number of incorrectly classified weighted instances $n_c^- = \sum_{a|c_a\neq c_t} w_a\, \mu^a_{G_t}$. Herewith, parameter $\varepsilon{\in}[0;\,1]$ determines the maximal tolerance for the error made by an individual rule (see Eq. 8). Specific for our use case, as it contains not so many decision classes, we choose $\varepsilon{=}0.2$. If classes from the log have unevenly distributed number of instances, it is advisable to choose smaller values of ε.

GA3. The GA algorithm iterates $t{=}t{+}1$ and repeats steps *GA1, GA2* until a given condition is fulfilled. The termination conditions can be chosen by a process analyst, depending on the requirements stemming from the business environment. For example, the algorithm can stop if a prespecified minimal rule coverage is reached. Also, if the number of rules, that are *not* added to the rule set, reaches a prespecified threshold, the algorithm can be stopped in order to avoid obtaining only redundant or low-performing rules. We discuss in the evaluation section on the example of our use case the possibility of combining termination criteria related both to properties of rule sets, and resulting fuzzy decision tables into which the discovered rules are transformed later.

4.3.2 Neurofuzzy algorithm NEFCLASS (NF)

Neural networks are also broadly applied to solve fuzzy classification problems, and the NEFCLASS algorithm (NF) is one of the most widely used [12]. NF is also known for producing simple and comprehensible fuzzy rules [15], which is appropriate for the business environment.

NF0. For using NF, we firstly need to represent our problem in a neuro-fuzzy terminology. In such a way, we view the system assisting with mining fuzzy decision rules as a *3-layer fuzzy perceptron*, or simply *perceptron*. Again, given is a process decision (A_{pdn}, C_{pdn}) where $A_{pdn} = \{a_1, ..., a_g\}{\subseteq}A$, $1{\leq}g{\leq}v$ is a set of influencing attributes from the corresponding event log subset L_{pdn}. An example perceptron is visualized in Fig. 5b.

Definition 6 (Perceptron). A *perceptron* is a network representation of fuzzy classification problem in the form of a neural network $\mathit{\Pi} = (U, \Omega, \Upsilon)$, where

- $U{=}U_1{\cup}U_2{\cup}U_3$ is a set consisting of a set $U_1{=}\{a_1, ..., a_g\}$ of attributes influencing the process decision (input neurons), a set $U_2{=}\{H_1, ..., H_r\}, r{\in}\mathbb{N}^+$ of fuzzy rules (hidden neurons), and a set $U_3{=}\{c_1, ..., c_z\}, z{\in}\mathbb{N}^+$ of the process decision classes (output neurons);
- $\Omega(a,h){=}\,\mu^a_h$ is a fuzzy weight defined as the membership grade of the value of input neuron $a{\in}U_1$ in a fuzzy subset provided by the hidden rule unit $h{\in}U_2$;
- Υ is a mapping that assigns activation functions as follows: $\Upsilon_h{=}\min_{a\in U_1}\Omega(a,h)$ if $u \in U_1{\cup}U_2$, and $\Upsilon_c = \dfrac{\sum_{h\in U_2}\Omega(H,c)\Upsilon_h}{\sum_{h\in U_2}\Omega(H,c)}$ if $u{\in}U_3$. ◇

NF1. At each step of the algorithm, an event instance from the event log subset L_{pdn} is chosen consequently. Based on this instance, a rule $H_t : a_1 \; is \; l_{x_1}^{a_1}, \ldots, a_{pdn} \; is \; l_{x_{pdn}}^{a_{pdn}} \longrightarrow c_{t,z}$ is generated, where A_{pdn} is a subset of influencing attributes, and $c_{t,z}$ is the label of the $z-th$ class of the rule $H_t, t \in \mathbb{N}^+$, of the process decision (A_{pdn}, C_{pdn}) with the values corresponding to the records in the event log. Further, initialization of the perceptron should be done by creating g nodes in input layer $U_1 = \{a_1, ..., a_{pdn}\}$ corresponding to each attribute in the generated rule. Each input neuron $a \in U_1$ is characterized by K^a fuzzy sets $FS_i^a, \; i \in [1; \; K^a]$. Thus, at the iteration t, for each input neuron $a_j \in U_1$ the membership function is found such that $\mu_{l_j}^a(a) = max_{i \in \{1,\ldots,K^a\}}\{\mu_i^a(a_j)\}$. If the hidden layer U_2 of the perceptron does not have a hidden rule node $h \in H$ such that $\Upsilon(a_1, h) = \mu_{l_1}^{a_1}, ..., \Upsilon(a_g, h) = \mu_{l_g}^{a_g}$, then such node is created. Hereby, the class $c_{t,z}$ is assigned as the consequent of the rule.

NF2. When adding the generated rules to the outcome rule base, only the rules that have good fitness are kept. For tuning the perceptron weights, we apply the rule H_t on each instance $e_w = (A^w, c^w), w \in \mathbb{N}^*$ from the event log L_{pdn} with the overlapping antecedent and compare the factual class assigned from the event log c^w, and class predicted by the perceptron. We identify the rule fitness as the ratio of the number of the correctly classified instances by the rule $TP(H_t)$ to the sum of number of the correctly and incorrectly classified instances $FP(H_T)$ in the event log subset L_{pdn}:

$$f(H_t) = \frac{TP(H_T)}{TP(H_T) + FP(H_T)} \tag{9}$$

For example, in our further implementation we only keep rules with fitness $f(H_t) \geq 0.1$.

NF3. The algorithm iterates $t = t+1$ and repeats steps *NF1, NF2* until a given condition happens. Again, different termination conditions can be chosen with respect to the business environment. As a new rule is generated for each event instance, the algorithm can be stopped when complete event log is processed. As this might produce a large amount of rules, the number of first rules to be generated can be prespecified. More complex termination criteria for the NF classifier can be found, e.g., in [12].

4.4 Transformation of discovered fuzzy rules into FDT rows

The outcome of application of fuzzy learners described at the previous step are fuzzy rule sets corresponding to each process decision $(A_{pdn}, C_{pdn}), \; 1 \leq pdn \leq PDN$, where PDN is the number of discovered process decisions, and $A_{pdn} = \{a_1, ..., a_g\} \subseteq A, \; 1 \leq g \leq v$ are attributes influencing them. For each of the process decisions (A_{pdn}, C_{pdn}), a corresponding set of frn fuzzy rules is discovered: $R_{pdn} = \{R_1, ..., R_{frn}\}, frn \in \mathbb{N}^+$. Next, the discovered fuzzy rule sets need to be transformed into fuzzy decision tables (FDTs), which are to be used further during process execution. Therefore, the FDT *interpretability* is of the highest importance. However, direct application of discovered fuzzy rule sets might provide low FDT interpretability because of duplications and overlapping of rules and attributes. Below we describe these issues in more detail and propose the ways to overcome them. Some of the steps are similar to the stages of manual designing of fuzzy decision tables described in [19].

Step 1: Removal of duplications. Duplicated rules will lead to FDT with duplicated rows, so the first step is to remove all duplicate rules from the base.

Step 2: Splitting CNF Rules. Some fuzzy rule learners, as the genetic one, might generate the rule bases consisting of fuzzy rules represented by CNF. As the CNF

rules can represent very complex structures, to improve the linguistic interpretability, we propose to replace them by a set of equivalent elemental rules. In particular, each discovered rule $R_y^{pdn} \subseteq R^{pdn}, 1 \leq y \leq frn$ can be represented in the following form: $R_y^{pdn} = (\bigcup_{j,i}(a_j, \mu_{l_i}^{a_j}(a_j), \mu_{l_i}^{a_j}), c_{y,z}),\ 1 \leq j \leq g, 1 \leq i \leq K^{a_j}$. Then, the corresponding set of elemental rules RS_y^{pdn} can be identified as the Cartesian product of (1) all possible subsets of the rule antecedent containing all attributes of the rule in a couple with the single value of the corresponding fuzzy subset; and (2) the value of the rule class:

$$RS_y^{pdn} = \bigcup_{j,i}(a_j, \mu_{l_i}^{a_j}(a_j)) \times c_{y,z} \tag{10}$$

For example, let the following rule R be given in natural language:

$$R\text{: } amount\ is\ high\ or\ low,\ \ duration\ is\ short\ or\ long \longrightarrow risk = high$$

Then the corresponding set of simple rules is $RS = \bigcup_i R_i, 1 \leq i \leq 4$,

$$R_1 : amount\ is\ high,\ \ duration\ is\ short \longrightarrow risk = high$$
$$R_2 : amount\ is\ low,\ \ duration\ is\ short \longrightarrow risk = high$$
$$R_3 : amount\ is\ high,\ \ duration\ is\ long \longrightarrow risk = high$$
$$R_4 : amount\ is\ low,\ \ duration\ is\ long \longrightarrow risk = high$$

Step 3: Mapping simple rules to FDT rows. Further, each elemental rule $RS_y^{pdn},\ 1 \leq y \leq frn$ is mapped to a corresponding row in a FDT which represents a union of all the fuzzy elemental rules $\bigcup_y RS_y{}^{pdn}$. If a rule has no linguistic literal for an attribute, the corresponding cell is left blank, meaning that this attribute has no influence on the outcome.
Step 4: FDT optimization. There can be rules represented by multiple rows, which differ only in the irrelevant attributes' values. If we find such rules, we aggregate them into one row by removing these attributes, because they do not impact the outcome. For example, see Fig. 6 where the contracted FDT is derived from the expanded FDT by combining logically adjacent rows that leads to the same action configuration.

Inputs			*Output*
amount	duration	premium	risk
low	short	false	low
low	short	true	low

Inputs			*Output*
amount	duration	premium	risk
low	short	-	low

Fig. 6: Example of rule reduction in a FDT

4.5 Identification of hit policies for generated FDT

Each discovered fuzzy rule, for which its antecedent matches the runtime input, contributes to an output through the compositional rule called *inference*. In DMN, inference is described by hit policies which specify how the table output is obtained, if there are multiple rule matches for a given set of inputs. A *single hit policy* returns the output of one rule; a *multiple hit policy* returns the output of multiple rules, or an aggregation of rules. The activation of a hit policy corresponds to the phase of process execution. During instantiation of processes, the process activity that invokes a corresponding decision model supplies the decision-making system with input data which can be crisp or fuzzy. Below we propose a hit policy formula for FDT.

Given is a process decision $(A_{pdn}, C_{pdn}),\ 1 \leq pdn \leq PDN$, where PDN is a number of discovered process decisions, and $A_{pdn} = \{a_1, ..., a_g\} \subseteq A,\ 1 \leq g \leq v$ are attributes influencing them. For the process decision (A_{pdn}, C_{pdn}), a corresponding FDT consisting of

a set of discovered fuzzy decision rules $R^{pdn} = \bigcup_j R_j, j \in \mathbb{N}^*$ is processed according to the procedures from the previous section. Let an event instance e occur further during the process execution. Then, the activation $\mathcal{A}$ for a rule $R_j, j \in \mathbb{N}^*$ describes the probability of a value from the class C_{pdn} to be correct for the given instance e. For calculating the activation rule value, we propose to utilize an adapted "min-max" operator, one of the most widely used composition operators suitable for Mamdani rule bases [22]:

$$M(R_j, e) = \min_k [\bigcup_{k \in [1;\, g]} \max_i (\bigcup_{i \in [1;\, K^{a_k}]} \mu_{l_i}^{a_k})], \tag{11}$$

where $1 \leq k \leq g : a_k \in e,\ a_k \in A_{pdn}$, so only attributes of an event instance which influence the process decision are evaluated. Let a loan application process be executed with the instance data $amount = 200[EUR]$, $duration = 30[mths]$, $premium = TRUE$, and imagine that the activation rule from Equation 3 needs to be calculated. Here k=2, and, consulting the membership functions, we establish that $\mu_{low}^{amount} = 1,\ \mu_{high}^{amount} = 0$, and $\mu_{short}^{duration} = 0.6,\ \mu_{long}^{duration} = 0.4$. Then, $\mathcal{A}(R, e) = \min\ [\bigcup \max(\{1; 0\}, \{0.6; 0.4\})] = 0.6$. Analogously, activation values for all the rules in a set $R^{pdn} = \bigcup_j R_j, j \in \mathbb{N}^*$ are evaluated, and the rule maximizing the value of the activation rule is chosen.

In order to further improve the accuracy of FDT hit policies, we propose to additionally weight the discovered rules by their error rate on instances from the event log. Then, even if several different rules are evaluated to the same value by the "min-max" operator, the rule which best corresponds to decisions recorded in the event log is hit:

$$\mathcal{A}(R_j, e) = W(R_j, L_{pdn}) * M(R_j, e) \tag{12}$$

Here, the weight variable $W(R_j, L_{pdn})$ characterizes how good the rule classifies the instances from the event log subset L_{pdn} corresponding to the process decision (A_{pdn}, C_{pdn}). The calculation of the weight value depends on the applied fuzzy learner. For GA, this value is equal to the logarithmic accuracy of the rule $\log(1 - E(R_j))/E(R_j)$ from Equation 5. For the NF, this value is equal to the rule fitness $f(R_j) = TP(R_j)/(TP(R_j) + FP(R_j))$ from Equation 9. Assigning the weights to rules in such a way can also contribute to satisfying the condition of *exclusivity* of decision rules in FDT, if that is required by user. Then, the rows are sorted by their weights, and while iterating over the rows, a rule is removed if it overlaps with the previous one. The low quality rows can also be removed, if some threshold is established.

With respect to the DMN standard, all the DMN hit policies can be applied to FDMN. Fuzzy activation rules are not foreseen though. However, the standard recommends to implement custom post processing steps in combination with the *Collect Hit* policy, which can be used considering the activation function from Equation 12.

5 Evaluation

In order to validate the presented methodology, we developed an open-source prototype of it[3]. Our experiments consisted of applying the methodology from Fig. 3 on a test log, and evaluating the interpretability and accuracy of the output FDMN.

Test setting. For the test data, we considered the loan assessment process from Fig. 1a. Estimating the distributions of parameters from a real credit-risk data set [3], we simulated an event log consisting of 1000 process instances with the help of the simulation

[3] https://bpt.hpi.uni-potsdam.de/Public/MiningFuzzyDMN

software CPN Tools [1]. The generated log contained two numeric attributes *duration* and *amount*, and two nominal attributes *premium* and *risk*. Corresponding membership functions were derived using Matlab [2], as it has an implementation of FCM. The resulted membership functions for *amount* and *duration* can be seen in Fig. 2. With the help of the DMN Analysis plug-in designed by us in our previous work [5], three process decisions were discovered: (1) control flow decision p_1 with influencing attributes *amount* and *premium*; (2) control flow decision p_3 with the influencing attribute *risk*; and (3) data decision *risk* with influencing attributes *duration* and *amount*. Next, we applied both GA and NF classifiers for discovering FDTs. For fuzzy rules obtained by the GA classifier, the maximal rule error was set to 0.2, the minimal coverage was set to 0.42, and the minimal fitness was set to 0.1. The minimal fitness for NF was also set to 0.1. We ran the GA classifier until the number of low-performing rules, that were not added to the output rule sets corresponding to process decisions, reached a threshold of 50000 rules. The NF classifier was running until the complete event log was processed. Both algorithms also stopped if the output decision table was *complete*, which means that for all combinations of input values there was a rule that covered it. The average run time for discovering FDT per process decision was 44.2 s for the GA classifier, and 1.8 s the for the NF classifier.

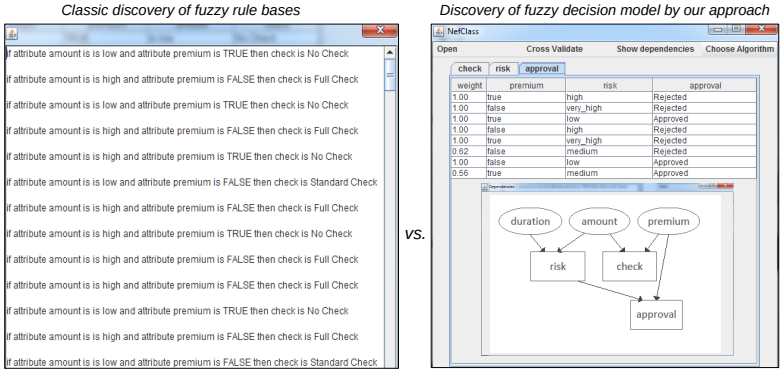


Fig. 7: Screenshots of the results from our prototype

Interpretability of the output FDMN. A screenshot of the FDMN, which our application outputs for the input log, is presented on the right side of Fig. 7. Corresponding to each decision in the DRD are the FDTs, which are obtained by the NF classifier, and post-processed according to our methodology. In the left side of the picture one can see the direct application of "state-of-the-art" NF classification for process decision *check*, which yields a large fuzzy rule base that is difficult for humans to interpret. In contrast, FDMN derived by our methodology (on the right side of Fig. 7) consists of compact FDTs incorporating rule weights and fuzzy hit policies, and it shows dependencies between decisions. It has to be noted that the interpretability of the output FDMN highly depends on the fuzzy algorithm applied for learning the model. According to our observations, NF often leads to a smaller amount of rules than GA.

Accuracy of the output FDMN. To evaluate the accuracy of the output FDMN, we used 10-fold cross validation over the test log [21]. Here, the accuracy is equal to the average number of event instances that are correctly classified by the rules in FDTs

divided by the total number of all instances. Further, we created incorrect log entries by randomly selecting from 0% to 10% of the event log instances, and randomly replacing their class labels with distinct class labels. The resulted accuracy of the classifiers with respect to the introduced noise is presented in Fig. 8:

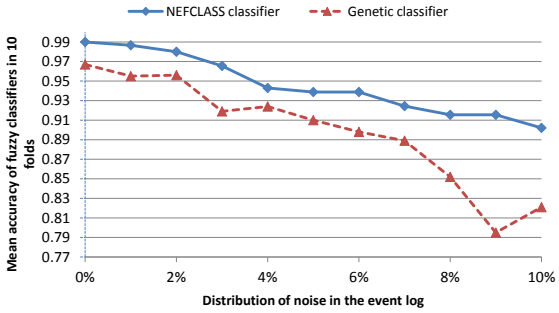


Fig. 8: Mean FDMN accuracy in 10 folds with respect to introduced noise in the log

Both classifiers achieved high accuracy for the event log without noise. Further, it can be seen, that NF shows a good stability, as it preserves a very good accuracy of ca. 90.21% in the presence of 10% input data noise. GA shows less stability as its accuracy reduces to ca. 82.10% in the presence of 10% input data noise. Setting the algorithms parameters can be used to adjust the desired user's output. For example, if a smaller amount of rules for better human interpretability is needed, the threshold has to be higher than 0.1 used in our case, although the accuracy might decrease. Further experiments on evaluating the accuracy with regards to tuning of termination criteria are planned for future work.

6 Conclusion

Recently, many efforts were made to standardize process decisions with respect to the separation of process and decision concerns [20,10]. Although the DMN standard is developed for this purpose, currently it is not well suited for managing fuzzy decisions. For example, the hit policies proposed by the standard can not handle overlapping fuzzy rules. Usage of FDMN models that consist of crisp DRDs referencing fuzzy decision tables, as proposed by us, is not yet foreseen by the standard, but is compliant with it.

To assist companies with automated construction of FDMN models, we introduced a methodology for mining them from event logs. Hereby, we explored application of the genetic and NEFCLASS classifiers for learning fuzzy rules in the process context. Further, we proposed a formula for the fuzzy activation rule as a FDT hit policy. Evaluation demonstrates that the interpretability of the mined FDMN is better than that of mined sets of fuzzy rule bases. Also, the accuracy of the output FDMN is high, so it can serve as an explanatory model based on historical data. Further, the derived FDMN can be automated and executed complementary to the process model.

With regards to the methodology limitations, for some steps of it, the domain knowledge is required. For example, an analyst has to describe the fuzzy subsets for the attributes from the event log. Also, it is recommended that an expert tunes the parameters of the fuzzy classifiers, such as termination criteria, or rules fitness threshold, depending on requirements of the business environment. FDMN mining, like other classification problems, often leads to the necessity to find a compromise between interpretability and accuracy of the output model.

We plan further experiments on evaluating the accuracy with regards to the termination criteria tuning for the future work. Additionally, we plan an evaluation of the amount of data needed for extraction of a reliable set of rules.

References

1. CPN Tools. `http://cpntools.org/`. [Online; accessed 29-November-2016].
2. Matlab. `https://de.mathworks.com`. [Online; accessed 29-November-2016].
3. B. Baesens, R. Setiono, C. Mues, and J. Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management science*, 49(3), 2003.
4. A. Batoulis, K.and Meyer, E. Bazhenova, G. Decker, and M. Weske. Extracting decision logic from process models. volume 9097 of *LNCS*, pages 349–366. Springer, 2015.
5. E. Bazhenova, S. Buelow, and M. Weske. Discovering decision models from event logs. volume 255 of *LNBIP*, pages 237–251. Springer, 2016.
6. B. Bede. *Mathematics of Fuzzy Sets and Fuzzy Logic*, volume 295 of *Studies in Fuzziness and Soft Computing*. Springer, 2013.
7. N. Capon. Credit scoring systems: A critical analysis. *Journal of Marketing*, 46(2), 1982.
8. S.L. Chiu. Fuzzy model identification based on cluster estimation. *J. Intell. Fuzzy Syst.*, 2(3):267–278, May 1994.
9. M. De Leoni, M. Dumas, and L. García-Bañuelos. Discovering branching conditions from business process execution logs. In *Fundamental Approaches to Software Engineering*, pages 114–129. Springer, 2013.
10. T. Debevoise and J. Taylor. *The MicroGuide to Process Modeling and Decision in BPMN/DMN*. CreateSpace Independent Publishing Platform, 2014.
11. A. Gonzalez and R. Perez. Completeness and consistency conditions for learning fuzzy rules. *Fuzzy Sets and Systems*, 96(1):37–51, 1998.
12. F Hoffmann, B. Baesens, C. Mues, T. Van Gestel, and J. Vanthienen. Inferring descriptive and approximate fuzzy rules for credit scoring using evolutionary algorithms. *European Journal of Operational Research*, 177(1):540–555, 2007.
13. E.M. Hubbard, I. Diester, J.F. Cantlon, D. Ansari, F. van Opstal, and V. Troiani. The evolution of numerical cognition: From number neurons to linguistic quantifiers. *The Journal of Neuroscience*, 28(46):11819–11824, 2009.
14. F. Mannhardt, M. De Leoni, H.A. Reijers, and W. Van der Aalst. Decision mining revisited - discovering overlapping rules. volume 9694 of *LNCS*, pages 377–392. Springer, 2016.
15. D. Nauck. Fuzzy data analysis with nefclass. *International Journal of Approximate Reasoning*, 32(2):103 – 130, 2003.
16. OMG. Business Process Model and Notation (BPMN), v. 2.0.2, 2013.
17. OMG. Decision Model And Notation (DMN), v.1.1, 2016.
18. A. Rozinat and W.M.P. van der Aalst. Decision mining in ProM. volume 4102 of *LNCS*, pages 420–425. Springer, 2006.
19. J. Vanthienen, G. Wets, and Guoqing Chen. Incorporating fuzziness in the classical decision table formalism. *International journal of intelligent systems*, 11(11):879–891, 1996.
20. B. Von Halle and L. Goldberg. *The Decision Model: A Business Logic Framework Linking Business and Technology*. Taylor and Francis Group, 2010.
21. I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
22. L.A. Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.
23. H.J. Zimmermann. *Fuzzy Set Theory and Its Applications (3rd Ed.)*. Kluwer Academic Publishers, Norwell, MA, USA, 1996.