International Journal on Software Tools for Technology Transfer
https://doi.org/10.1007/s10009-021-00635-x

EXPLANATION PARADIGMS LEVERAGING ALGEBRAIC INTUITION

Special Section: Introducing ExPLAIn

Algebraic aggregation of random forests: towards explainability and rapid evaluation

Frederik Gossen[1] · **Bernhard Steffen**[1]

Accepted: 25 June 2021

Abstract
Random Forests are one of the most popular classifiers in machine learning. The larger they are, the more precise the outcome of their predictions. However, this comes at a cost: it is increasingly difficult to understand why a Random Forest made a specific choice, and its running time for classification grows linearly with the size (number of trees). In this paper, we propose a method to aggregate large Random Forests into a single, semantically equivalent decision diagram which has the following two effects: (1) minimal, sufficient explanations for Random Forest-based classifications can be obtained by means of a simple three step reduction, and (2) the running time is radically improved. In fact, our experiments on various popular datasets show speed-ups of several orders of magnitude, while, at the same time, also significantly reducing the size of the required data structure.

Keywords Random forest · Algebraic decision diagram · Aggregation · Explainability · Interpretability · Running time optimisation · Memory optimisation

1 Introduction

Random[1] Forests are one of the most widely known classifiers in machine learning [2,19]. The method is easy to understand, to implement, and at the same time achieves impressive classification accuracies in many applications. In contrast to a single decision tree, Random Forests—a collection of many trees – do not overfit as easily on a dataset and their variance decreases with size. On the other hand, their running time for classification linearly grows with the number of trees, which is critical as forests may well consist of hundreds, if not thousands of trees—a problem especially for applications with a high throughput [9].

In this paper, we present an optimisation method that is based on algebraic aggregation: Random Forests are transformed into a single decision diagram in a semantics-preserving fashion, which, in particular, also preserves the learner's variance and accuracy. Being a post-process, the ease of Random Forest training is also maintained.

The great advantage of the resulting decision diagrams is their absence of redundancy: during classification every predicate is considered at most once, and only if its evaluation is required. This allows one to obtain concise *explanations* and *evaluation times* that are optimal.[2]

Key to our approach are Algebraic Decision Diagrams (ADDs) [28]. Their algebraic structure supports compositional aggregation, abstraction, and reduction operations that lead to minimal normal forms. In combination with a reduction that exploits the infeasibility of paths in decision diagrams, this results in a three stage aggregation process:

1. **Faithful Aggregation.** Using basic algebraic operations, such as concatenation and addition, allows us to aggregate a Random Forest into a single ADD that faithfully maintains the individual results of each tree in the forest.
2. **Abstraction.** Abstracting results (i.e. the leaf structure of the decision diagrams) to the essence, in this case the

[1] The paper is based on the sketch of our approach presented in [15].

✉ Bernhard Steffen
bernhard.steffen@tu-dortmund.de

Frederik Gossen
frederik.gossen@tu-dortmund.de

[1] Chair for Programming Systems, TU Dortmund University, Dortmund, Germany

[2] Up to an underlying predicate ordering

Published online: 29 September 2021

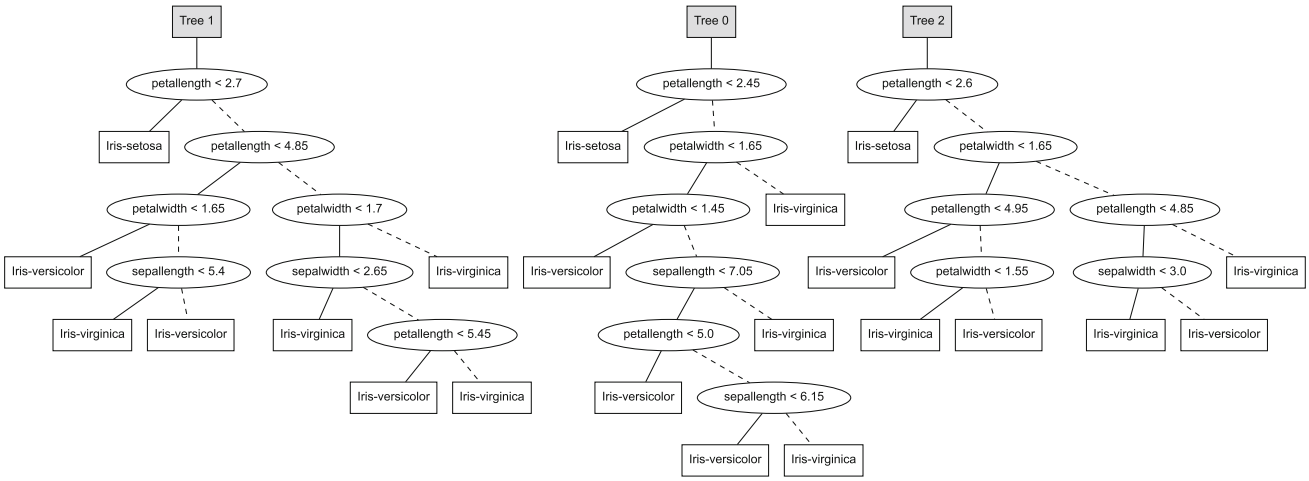

Fig. 1 Random forest learned from the Iris dataset [8,11].

outcome of a majority vote, maintains the classification function of the original Random Forest while drastically reducing the size of the representing decision structure. In fact, given an ordering of the involved predicates, this step, which is realised just by playing with the underlying algebra, results in a minimal normal form.

3. **Infeasible Path Reduction.** Eliminating infeasible paths (path with contradicting predicates) addresses the redundancies imposed by the semantic dependencies between the involved predicates. This reduction would be vacuous for the original trees, but has a radical impact after the aggregation. In contrast to the previous reduction steps, infeasible path reduction, which can be considered a *don't care* optimisation, does not yield normal forms.

The result of this three-stage aggregation process is a minimal representation of the original Random forest in terms of an ADD. Being as easy to understand as common decision trees, the resulting ADD may indeed be considered a precise solution to the *model explanation problem* [16]. In addition, it is a good basis for deducing concise explanations for individual classifications and for optimising the evaluation performance:

Outcome Explanation. Given a Random Forest RF and some sample profile σ,[3] a minimal, sufficient explanation for the corresponding classification $RF(\sigma)$ can be obtained from its explanation model A by means of three further reduction steps, which follow the same pattern used for the aggregation to obtain the previous ADDs:

- Changing the majority vote algebra underlying A to a Boolean algebra allows us to distinguish the chosen class $RF(\sigma)$ from the rest. The result is a Binary Decision Diagram (BDD) [3] B with terminal nodes $RF(\sigma)$ and $\neg RF(\sigma)$, which precisely characterises the set of sample profiles σ which RF would classify as $RF(\sigma)$.
- Building the conjunction of the choices[4] in B for the path leading σ to $RF(\sigma)$.
- Removing choices from the conjunction as long as redundant choices exist.[5]

The resulting conjunction is then a sufficient minimal explanation for the Random Forest's choice (see Sect. 8 for more details).

Please note the importance of the reduction of A to a BDD that characterizes the class $RF(\sigma)$: It eliminates all the choices in A that are required for distinguishing two classes that are different from $RF(\sigma)$. This has already a significant effect for the three class example used for illustration in this paper. Of course, the effect grows with the number of distinct classes. We are not aware of any work that achieves a similar effect.

For the ease of notation we will abbreviate *sample profile* by *sample* in the rest of the paper.

Rapid Evaluation. The evaluation time is radically improved. In fact, it is even provably optimal for a chosen predicate ordering in the sense that each predicate is considered at most once, and only if its evaluation is required. Our experiments with popular data sets from the UCI Machine Learning Repository [8] showed performance gains of several orders of magnitude (see Fig. 10 and Table 1).

A potential problem of our aggregation method is only an explosion in size which can, in principle, be exponential for

[3] Sample profiles are typically certain measurements of real world objects.

[4] These are either predicates in B or their negations.

[5] Please note that there may be redundant predicates on a path that are all necessary in the BDD as a whole due to the considered predicate ordering.

decision diagrams. However, this problem did not arise in most of our experiments. On the contrary, we even observed drastic size reductions (see. Fig. 11 and Table 2).

Please note that these results are achieved in a very generic, algebraic fashion using a common classifier on standard datasets. In particular, no scenario-specific heuristics have been applied. We are therefore convinced that our aggregation approach has the potential to be applied in a wide range of related scenarios.

The Iris Cast Study. Figure 1 shows a small Random Forest that was learned from the Iris flower dataset [8,11]. The task is to predict a flower's species based on its sepal and petal dimension. This dataset is a popular choice in machine learning, both, for test cases and also as a running example to present new methods.

For classification, all three trees must be evaluated individually. Only then, we can derive the most common answer among the trees (also known as the *majority vote*) and promote it to be the overall decisions. This effort clearly grows linearly with the number of trees, i.e. the size of the forest. In the following, we use this example to illustrate our approach to forest aggregation and its great effects on running time, size, and explainability.

The following Sections provide the foundations for Algebraic Decision Structures (Sect. 2), Random Forests (Sect. 3), and Algebraic Decision Diagrams (Sect. 4). Section 5, subsequently, defines transformations on simple co-domains which are then lifted to semantics-preserving transformations of Random Forests (Sect. 6), before Sect. 7 addresses the heuristic treatment of infeasible paths. The impact of our transformational approach is then illustrated in Sect. 8, which presents solutions to three explainability problems, and in Sect. 9, which shows our experimental performance evaluation. The paper closes after a discussion of related work in Sect. 10) with our conclusion and directions to future work in Sect. 11.

2 Algebraic decision structures

Core ingredients of decision structures, in particular decision trees and decision diagrams, are predicates that we assume to come in a linearly ordered fashion:

Definition 1 (Predicate Systems)
A *Predicate System* is linearly ordered set of predicates

$$\mathcal{P} = (\{p_0, p_1, \ldots, p_{n-1}\}, \prec_{\mathcal{P}}).$$

The (concrete) semantics of predicates $p \in \mathcal{P}$ is defined relative to a semantic domain Σ. In particular, for the ease of notation, we will avoid explicit reference to Σ, $\mathcal{P}$, and $\prec_{\mathcal{P}}$ whenever they are clear from the context.

Definition 2 (Predicate Semantics)
A function $[\![\cdot]\!]_{\mathcal{P}} : \mathcal{P} \to (\Sigma \to \mathbb{B})$ is called a *predicate semantic function*. This function naturally extends to the set of logical formulas $\mathcal{L}$ comprising $\wedge$, $\vee$ and $\neg$.

We consider also the concrete semantic function $[\![\cdot]\!]_{\mathcal{P}}$ as a given for the rest of the paper. Interesting is also the following *symbolic semantics*, which considers predicates as Boolean variables and thus, in particular, as mutually independent.

Definition 3 (Symbolic Predicate Semantics)
Let $\Sigma^s = \mathcal{P} \to \mathbb{B}$ be a predicate assignment. The symbolic predicate semantic function $[\![\cdot]\!]^s_{\mathcal{P}} : \mathcal{P} \to (\Sigma^s \to \mathbb{B})$ is then defined as

$$[\![p]\!]^s_{\mathcal{P}} = \sigma \mapsto \sigma(p).$$

The essence of this paper focuses on Algebraic Decision Diagrams (ADDs), which typically live in the symbolic world. In other words, they are typically based on Boolean variables and not on predicates which may have dependencies. Sects. 4–6 concern this symbolic setting. Dependencies are only considered starting with Sect. 7.

In order to prepare the algebraic treatment of decision structures, we focus on decision structures whose leafs are labelled with the elements of an algebra $A = (\mathcal{A}, O)$. This subsumes the classical case of sets, which are simply algebras where O is empty.

Definition 4 (Algebraic Decision Tree)
Let $A = (\mathcal{A}, O)$ be an algebra with the carrier set $\mathcal{A}$ and set O of operations. An Algebraic Decision Tree (ADT) over the algebra A is inductively defined by the following BNF:

$$T ::= a \mid (p, T, T) \text{ with } a \in \mathcal{A} \text{ and } p \in \mathcal{P}.$$

Let $\mathcal{T}_A$ denote the set of all such ADTs.

We can merge nodes in these ADTs, which leads to the more general Algebraic Decision Structures (ADS):

Definition 5 (Algebraic Decision Structure)
Let t be an ADT and t' and t'' be two nodes in t such that t'' is not reachable from t'. Then, the two step transformation of t

- re-route the incoming edges of t'' to t' and
- eliminate all unreachable nodes of t.

is called a t'' into t' merge. A rooted directed acyclic graph (DAG) that results from an ADT by a series of node merges is called an Algebraic Decision Structure (ADS). Let $\mathcal{S}_A$ denote the set of all such ADSs.

We can define their semantics inductively.

Definition 6 (Decision Structure Semantics) The (concrete) semantic function

$$[\![\cdot]\!]_{\mathcal{S}_A} : \mathcal{S}_A \to (\Sigma \to \mathcal{A})$$

for ADSs is inductively defined as

$$[\![a]\!]_{\mathcal{S}_A}(\sigma) := a$$
$$[\![(p,t,u)]\!]_{\mathcal{S}_A}(\sigma) := \begin{cases} [\![t]\!]_{\mathcal{S}_A}(\sigma) & \text{if } [\![p]\!](\sigma) = 1 \\ [\![u]\!]_{\mathcal{S}_A}(\sigma) & \text{if } [\![p]\!](\sigma) = 0 \end{cases}$$

The symbolic semantic function $[\![\cdot]\!]^s_{\mathcal{S}_A} : \mathcal{S}_A \to \mathcal{A}$ for ADSs only differs from the concrete semantics in that it uses the abstract predicate semantics $[\![\cdot]\!]^s$.

The following notion of Ordered Algebraic Decision Structures (OADS) is essential for the intended lifting of the algebra to the decision structure.

Definition 7 (Ordered ADS)
An Ordered Algebraic Decision Structure (OADS) is an ADS t whose predicates respect the order $\prec_{\mathcal{P}}$ of the associated predicate system:

- if $t = (p, (q, \cdot, \cdot), \cdot)$ then $p \prec_{\mathcal{P}} q$ and
- if $t = (p, \cdot, (q, \cdot, \cdot))$ then $p \prec_{\mathcal{P}} q$.

Let $\mathcal{U}_A \subset \mathcal{S}_A$ denote the set of all OADSs.

We can transform any given ADS into an OADS simply by reordering its nodes according to $\prec_{\mathcal{P}}$.

Definition 8 (OADT Transformation)
Let $(\mathcal{P}, \prec_{\mathcal{P}})$ be a predicate system and let $p \mapsto 0$ and $p \mapsto 1$ assign a predicated $p \in \mathcal{P}$ the value of 1 and 0, respectively. Then we can partially evaluate any ADS as follows:

$$a_\alpha := a, \text{ where } \alpha \in \{p \mapsto 0, p \mapsto 1\}$$
$$(p,t,u)_{p\mapsto 1} := t_{p\mapsto 1}$$
$$(p,t,u)_{p\mapsto 0} := u_{p\mapsto 0}$$

The function $\Delta_{\prec_{\mathcal{P}}} : \mathcal{S}_A \to \mathcal{U}_A$ with

$$\Delta_{\prec_{\mathcal{P}}}(a) := a$$
$$\Delta_{\prec_{\mathcal{P}}}(p,t,u) := (q, \Delta_{\prec_{\mathcal{P}}}((p,t,u)_{q\mapsto 1}), \Delta_{\prec_{\mathcal{P}}}((p,t,u)_{q\mapsto 0}))$$

where q is the smallest predicate appearing in the child ADSs t, u and the predicate p according to $\prec_{\mathcal{P}}$ then defines the transformation from any ADS to an OADS that respects the order $\prec_{\mathcal{P}}$.

Note that the result of this transformation will always be an ADT. The following theorem guarantees that ADSs can be arbitrarily (re)ordered in a semantics-preserving fashion [28]:

Theorem 1 *((Re)Ordering)*
For any $t \in \mathcal{S}_A$ and any predicate order $\prec_{\mathcal{P}}$, the following holds:

- *$\Delta_{\prec_{\mathcal{P}}}(t)$ is an OADS respecting $\prec_{\mathcal{P}}$,*
- *$[\![t]\!]_{\mathcal{S}_A} = [\![\Delta_{\prec_{\mathcal{P}}}(t)]\!]_{\mathcal{S}_A}$, and*
- *$[\![t]\!]^s_{\mathcal{S}_A} = [\![\Delta_{\prec_{\mathcal{P}}}(t)]\!]^s_{\mathcal{S}_A}$.*

Theorem 1 says that an ADS and any of its (re)ordered variants are semantically equivalent according to the following definition:

Definition 9 (Semantic Equivalence)
Two ADSs t and u are semantically equivalent iff their semantic functions coincide

$$t \sim u \text{ iff } [\![t]\!]_{\mathcal{S}_A} = [\![u]\!]_{\mathcal{S}_A}$$

Analogously, we define symbolic semantic equivalence based on the abstract semantic functions

$$t \sim^s u \text{ iff } [\![t]\!]^s_{\mathcal{S}_A} = [\![u]\!]^s_{\mathcal{S}_A}$$

The following theorem states that one of two different nodes of an OADS that are semantically equivalent is redundant:

Theorem 2 *(Semantic Reduction)*
Let t be an OADS with two nodes t' and t'' that are semantically equivalent, i.e., $t' \sim^s t''$, and such that t'' is not reachable from t'. Moreover, let u be the t'' into t' merge of t. Then t and u are semantically equivalent, i.e, $t \sim^s u$.

Theorem 2 can be proven by induction over the ADS structure of t'. Algebraic Decision Diagrams are now defined as OADS without such redundancy:

Definition 10 (Algebraic Decision Diagrams)
OADSs without redundant nodes are called Algebraic Decision Diagrams (ADDs). We denote the set of a all ADDs for an algebra A with $\mathcal{D}_A$.

ADDs are popular, because of the following uniqueness property [3,28]:

Theorem 3 *(Normal Form)*
Every function $\mathbb{B}^n \to \mathcal{A}$ has a canonical representation (a minimal normal form) as an ADD that respects the predicate ordering $\prec_{\mathcal{P}}$. In particular, for every OADS there exists a unique ADD that is symbolically semantically equivalent and preserves $\prec_{\mathcal{P}}$.

We call the corresponding *normalising* function

$$\Delta_\sim : \mathcal{U}_A \to \mathcal{D}_A$$

which can straightforwardly be realised by successive elimination of redundant nodes $\sim^s$-quotienting transformation. Given a fixed predicate ordering, ADDs are canonical. The size of such canonical representations can, however, be very sensitive to the underlying ordering. The function $\Delta_{\prec_\mathcal{P}}$ allows us to flexibly switch between different orderings in a semantics-preserving fashion. There exist efficient algorithms that directly work on ADDs and allow one to find good predicate orderings heuristically [32].

That ADDs are not necessarily minimal for the (concrete) predicate semantics is due to possible dependencies between different predicates in $\mathcal{P}$, which induces so-called infeasible paths in the corresponding ADDs. Such dependencies impact the minimisation of ADDs in a similar fashion as the well-known don't care the minimisation of Boolean formulas. As a consequence, given $\prec_\mathcal{P}$, the results of our approach are (only) optimal relative to the abstract predicate semantics, and we deal with further dependency-based optimisations in a heuristic fashion (Sect. 7).

Section 4 presents the essence and impact of lifting the algebraic structure of $A = (\mathcal{A}, O)$ to the ADD-level in the classical case, i.e., with abstract semantics. The treatment of infeasible paths is somewhat independent and treated in Sect. 7.

It is straightforward to establish that the symbolic semantics allows one to separate the predicate evaluation step from the subsequent classification step in a way that the latter can be regarded to work on independent predicates:

Theorem 4 *(Early predicate evaluation)*
Let $I : \Sigma \to \Sigma^s$ be the function that evaluates each predicate according to its concrete semantics and returns a predicate assignment $I(\sigma) := p \mapsto [\![p]\!](\sigma)$. Then the following relation between (concrete) semantics and symbolic semantics holds for all ADSs $t \in \mathcal{S}_A$:

$$[\![t]\!]_{\mathcal{S}_A} = [\![t]\!]^s_{\mathcal{S}_A} \circ I.$$

After introducing forests in the next section, Sects. 4, 5, and 6 will focus on the classification step, before Sect. 7 will deal with the phenomenon of predicate dependency.

3 Random forests

Random Forests are one of the most widely known classifiers in machine learning for classifying the elements of a domain space Σ. The classification algorithm is relatively simple and yields good results for many real-world applications [2]. Its decision model generalises from a training dataset $\Sigma' \subset \Sigma$ that holds examples of input data labelled with the desired output, also called the *class*. In the following, $\mathcal{C}$ denotes the set of considered classes, which is assumed to be linearly ordered by some precedence $\prec_\mathcal{C}$.

ADTs, with the classes $\mathcal{C}$ for their underlying algebra, form the classification components of a Random Forest:

Definition 11 (Random Forest, ADT Forests)
Let $\mathcal{C}$ be the set of classes and let C be the corresponding algebra with no operations. A Random Forest (for C) is a finite list of ADTs (for C). [6] Let $\mathcal{T}^*_C$ denote the set of all Random Forests.

In practice, the ADTs forming a Random Forest are learned from randomly selected samples of a training dataset[7]. Consequently, all trees are pairwise different in structure, represent different decision functions, and can yield different decisions for the same input data.

For classifying (previously unseen) input data, Random Forests

- evaluate each of their component ADTs t separately to determine the class $c = [\![t]\!]_{\mathcal{T}_C}(\sigma)$ for the considered input $\sigma \in \Sigma$, and
- determine the overall result via (weighted) majority vote according to the following definition. Please note that we need two versions of the majority vote function, one just for election based on a given frequency function $f' : \mathcal{C} \to \mathbb{N}$ and a *lifted* version $f : \Sigma \to (\mathcal{C} \to \mathbb{N})$, which is parameterised by samples $\sigma \in \Sigma$:

Definition 12 (Majority Vote)
The Σ domain-independent majority vote function $\mu : (\mathcal{C} \to \mathbb{N}) \to \mathcal{C}$ is defined by:

$$\mu(f') := \arg\max_{c \in \mathcal{C}} f'(c).$$

and its *lifted* version

$$\mathsf{M} : (\Sigma \to (\mathcal{C} \to \mathbb{N})) \to (\Sigma \to \mathcal{C})$$

by

$$\mathsf{M}(f) := \sigma \mapsto \arg\max_{c \in \mathcal{C}} f(\sigma)(c)$$

where, in both cases, possible ties are resolved by the class precedence order $\prec_\mathcal{C}$.

With this we can define the semantics of a Random Forest as follows:

[6] Essential is here the *multi-set facet* of lists. We use the ordering only to identify individual trees.

[7] In this paper we use Weka [37] as a reference implementation for this purpose. However, our approach does not depend on specific implementation details and can be easily adapted to other implementations.

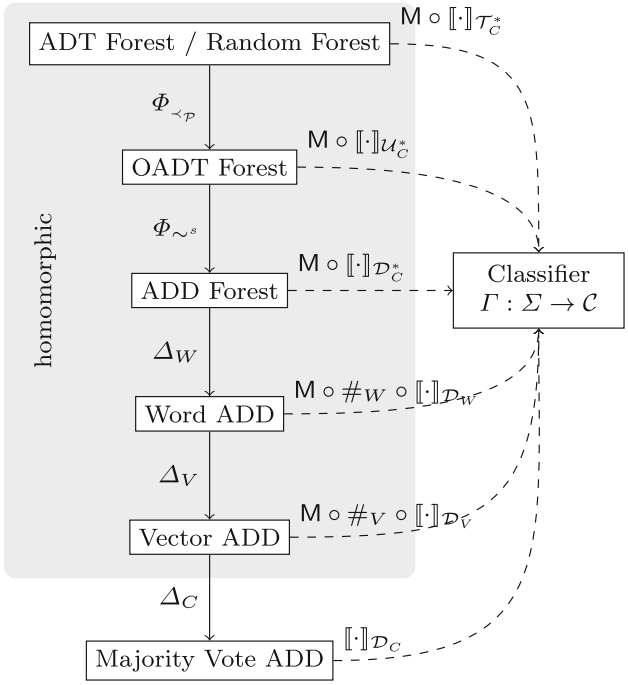

Fig. 2 Overview of the random forests transformations

Definition 13 (Random Forest Semantics)
The semantic function $[\![\cdot]\!]_{\mathcal{T}_C^*} : \mathcal{T}_C^* \to (\Sigma \to (\mathcal{C} \to \mathbb{N}))$ for a Random Forest $t_0 \cdot t_1 \cdots t_{n-1} \in \mathcal{T}_C^*$ is defined as

$$[\![t_0 \cdots t_{n-1}]\!]_{\mathcal{T}_C^*}(\sigma)(c) := \sum_{i=0}^{n-1} \mathbb{I}([\![t_i]\!]_{\mathcal{T}_C}(\sigma) = c).$$

where $\mathbb{I}$ evaluates to 1 if the argument condition holds and to 0 otherwise.

The composition $\mathsf{M} \circ [\![\cdot]\!]_{\mathcal{T}_C^*}$ defines the corresponding majority vote-based *classifier*.

Key advantage of this approach is the, compared to single decision trees, reduced variance. A detailed introduction to Random Forests, decision trees, and their learning procedures can be found in [2,19,27].

In the following, we aim at aggregating Random Forest into a single, concise ADD. This requires to

- transform the component ADTs into ADDs, followed by
- algebraic aggregation (Sect. 6).

In the remainder of this section, we will focus on the first step which allows us to transform Random Forests into ADD Forests. Figure 2 gives an overview of the complete transformation with all its steps. The algebraic aggregation, the main technical contribution of this paper, will follow later.

With the semantics-preserving transformations from ADTs to OADTs (Definition 8) and from OADTs to ADDs (Theorem 3) we can already treat the forests' individual trees. The generalisation to the entire forests is just a straightforward element-wise application of the respective transformations.

Definition 14 (OADT Forest)
An OADT Forest is an ADT Forest whose individual trees are OADTs. For a given ADT Forest

$$(t_0, t_1, \ldots, t_{n-1}) \in \mathcal{T}_C^*$$

the corresponding OADT Forest

$$\Phi_{\prec_\mathcal{P}}(t_0, t_1, \ldots, t_{n-1}) \in \mathcal{U}_C^*$$

is derived by component-wise application of $\Delta_{\prec_\mathcal{P}}$.

As classification components of an OADT Forest are still ADTs, the semantic function of Random Forests (Definition 13) naturally carries over as $[\![\cdot]\!]_{\mathcal{U}_C^*}$.

In the same fashion, we can derive ADD Forests.

Definition 15 (ADD Forest)
An ADD Forest is an OADT Forest whose individual trees are ADDs. For a given OADT Forest

$$(u_0 \cdot u_1 \cdots u_{n-1}) \in \mathcal{U}_C^*$$

the corresponding ADD Forest

$$\Phi_{\sim^s}(u_0 \cdot u_1 \cdots u_{n-1}) \in \mathcal{D}_C^*$$

is derived by element-wise application of $\Delta_{\sim^s}$.

Again, the semantic function of Random Forests (Definition 13) carries over as $[\![\cdot]\!]_{\mathcal{D}_C^*}$.

With that, we can establish semantic equivalence of all forest representations seen so far, i.e. both $\Phi_{\prec_\mathcal{P}}$ and $\Phi_{\sim^s}$ are semantics-preserving.

Theorem 5 *(Forest Equivalence)*
For any ADT Forest $(t_0 \cdot t_1 \cdots t_{n-1}) \in \mathcal{T}_C^*$*, the derived OADT Forest is semantically equivalent:*

$$[\![t_0 \cdot t_1 \cdots t_{n-1}]\!]_{\mathcal{T}_C^*} = [\![\Phi_{\prec_\mathcal{P}}(t_0 \cdot t_1 \cdots t_{n-1})]\!]_{\mathcal{U}_C^*}$$

For any OADT Forest $(u_0 \cdot u_1 \cdots u_{n-1}) \in \mathcal{U}_C^*$ *the derived ADD Forest is semantically equivalent:*

$$[\![u_0 \cdot u_1 \cdots u_{n-1}]\!]_{\mathcal{U}_C^*} = [\![\Phi_{\sim^s}(u_0 \cdot u_1 \cdots u_{n-1})]\!]_{\mathcal{D}_C^*}$$

The proof of Theorem 5 is essentially a repeated application of Theorem 1.

From here on, we will leave all unconstrained ADSs and ADTs behind and focus only on the ADD Forests which

allows us to take full advantage of the properties that are unique to ADDs.

At the top level, $\mathcal{T}_C^*$, $\mathcal{U}_C^*$, and $\mathcal{D}_C^*$ are all just word-like structures, which form monoids together with the (polymorphic) word concatenation $\bullet$[8]. In this sense, $\Phi_{\prec_\mathcal{P}}$ and $\Phi_{\sim^s}$ are both just homomorphisms, elegantly defining the forest transformation steps.

In the next section, we will lift the algebraic structures to the *classical* ADD level.

4 The essence of ADDs

It is well-known that algebraic structures $A = (\mathcal{A}, O)$ can be lifted to an algebraic functional space

$$F_{\Sigma\to A} := (\{f \mid f : \Sigma \to \mathcal{A}\}, O_F)$$

where O_F is the result-wise extension of O and Σ is a domain set. Also every homomorphism $\alpha : A \to A'$ can be lifted to a homomorphism

$$\alpha_F : F_{\Sigma\to A} \to F_{\Sigma\to A'}$$

in a similar fashion.

Semantically, ADDs $\mathcal{D}_A$ live in the world of these function spaces, where $\Sigma = \mathcal{P} \to \mathbb{B}$, but they are special in the following sense: Due to their canonicity (Theorem 3) it is is straightforward to define a set of operations O_D such that $D_A = (\mathcal{D}_A, O_D)$ and its corresponding semantic domain $F_{(\mathcal{P}\to\mathbb{B})\to A}$ are isomorphic. In particular, D_A also *inherits* the algebraic structure of A, as well as possible homomorphisms and functions[9] from its co-domain:

Theorem 6 *(Lifting)*
Let $A = (\mathcal{A}, O)$ and $A' = (\mathcal{A}', O')$ be two algebras, $\alpha : A \to A'$ a homomorphism, $\mathcal{C}$ a set, and $g : \mathcal{A} \to \mathcal{C}$ and $g' : \mathcal{A}' \to \mathcal{C}$ two functions: Then there exists a unique homomorphism

$$\alpha_D : \mathcal{D}_A \to \mathcal{D}'_A$$

such that

$$\alpha \circ [\![\cdot]\!]_{\mathcal{D}_A} = [\![\cdot]\!]_{\mathcal{D}_{A'}} \circ \alpha_D.$$

In particular, $g = g' \circ \alpha$ implies

$$g \circ [\![\cdot]\!]_{\mathcal{D}_A} = g' \circ [\![\cdot]\!]_{\mathcal{D}_{A'}} \circ \alpha_D.$$

[8] If the distinction of the structures is important, we write $\bullet$ with an index, e.g. in Fig. 4.

[9] Functions are just homomorphisms with no underlying algebraic structure.

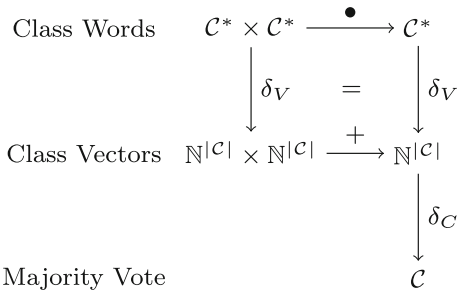

Fig. 3 Co-domain algebras and their relationships

Also the following theorem is a direct consequence of the fact that the algebraic structure of a codomain is inherited by its corresponding functional space:

Theorem 7 *(Aggregation)*
Let $A = (\mathcal{A}, O)$ be an algebra, $t, u \in \mathcal{D}_A$ be two ADDs, and $\bullet \in O$ be an operation of A. Then we have

$$[\![t]\!]_{\mathcal{D}_A} \bullet [\![u]\!]_{\mathcal{D}_A} = [\![t \bullet_{D_A} u]\!]_{\mathcal{D}_A}.$$

This allows us to define aggregation functions on $\mathcal{A}$ and apply their lifted variant to ADDs of $\mathcal{D}_A$. Even better, it generally allows us to focus on the structure of the underlying algebra A when proving properties of homomorphisms between ADDs, in particular, that they preserve the majority vote. In the following, we will denote operations of an algebra A and the operations of the corresponding ADD algebra D_A with the same symbols for brevity.

We exploit this property for the aggregation of Random Forests by partial evaluation with respect to three algebraic co-domains: the class word co-domain $\mathcal{C}^*$, the class vector co-domain $\mathbb{N}^{|\mathcal{C}|}$, and the co-domain of classes $\mathcal{C}$ themselves.

5 Co-domain algebras and their relationships

The co-domain algebra for forests is just the (linearly ordered) set of classes $\mathcal{C}$. There are no operations that may support the aggregation of the votes of the individual trees or ADDs. Such an aggregation operation must maintain enough information that the subsequent majority vote is not affected. The algebra which most directly mimics the standard evaluation process of forests simply records each vote in their respective order:

Definition 16 (Class Word Monoid)
The *Class Word Monoid* is defined as $W := (\mathcal{C}^*, \bullet, \epsilon)$ with concatenation $\bullet$ and the empty word ϵ as its neutral element.

With that, we can finally aggregate the individual trees' decisions:

Definition 17 (Aggregation)
Let $a_0, a_1, \ldots, a_{n-1} \in \mathcal{C}$ be class labels. Since $\mathcal{C} \subset \mathcal{C}^*$, we can define a aggregation function

$$\delta_W : \mathcal{C} \times \cdots \times \mathcal{C} \to \mathcal{C}^*$$

directly on $\mathcal{C}$, simply by concatenating the component classes:

$$\delta_W(a_0, a_1, \ldots, a_{n-1}) := a_0 \bullet a_1 \bullet \cdots \bullet a_{n-1}.$$

Obviously, W maintains much more information than is required: For the majority vote evaluation, we do not need to know which tree voted for which class. The following algebra therefore abstracts from this information:

Definition 18 (Class Vector Monoid)
The *Class Vector Monoid* is defined as $V := (\mathbb{N}^{|\mathcal{C}|}, +, \mathbf{0})$ with component-wise addition $+$ and the 0-vector $\mathbf{0}$ as its neutral element.

The corresponding transformation can now be defined as a function directly from class words to class vectors. To this aim, let us first define a function to count class frequencies in the obtained class words:

Definition 19 (Class Word Frequencies)
The class word frequency function $\#'_W : \mathcal{C}^* \to (\mathcal{C} \to \mathbb{N})$ is defined as

$$\#'_W(\epsilon) := c \mapsto 0$$
$$\#'_W(a \cdot \mathbf{w}) := c \mapsto \mathbb{I}(a = c) + \#'_W(\mathbf{w})$$

where $\mathbb{I}$ is the indicator function which evaluates to 1 if the condition holds and to 0 otherwise. Analogously, we can define the correspondingly *lifted* Σ domain-dependent variant

$$\#_W : (\Sigma \to \mathcal{C}^*) \to (\Sigma \to (\mathcal{C} \to \mathbb{N}))$$

that we will later need when reasoning about ADD semantics.

The abstraction from class words to class vectors can now be defined based on $\#'_W$. It is essentially its vector-embedded representation:

Definition 20 (Class Vector Abstraction)
The class vector abstraction function $\delta_V : \mathcal{C}^* \to \mathbb{N}^{|\mathcal{C}|}$ is defined as

$$\delta_V(\mathbf{w}) := (n(c_0), n(c_1), \ldots, n(c_{|\mathcal{C}|-1}))$$

with $n(c) := \#'_W(\mathbf{w}(c))$.

To define the final majority vote abstraction (and also to reason about semantic equivalences later), let us define another function that determines class frequencies, this time simply via a projection of class vectors.

Definition 21 (Class Vector Frequencies)
The class vector frequency function

$$\#'_V : \mathbb{N}^{|\mathcal{C}|} \to (\mathcal{C} \to \mathbb{N})$$

is defined as

$$\#'_V(n_0, n_1, \ldots, n_{|\mathcal{C}|-1})(c) := n_i$$

where i is the index associated with class c. Analogously, we can define the correspondingly *lifted* Σ domain-dependent variant

$$\#_V : (\Sigma \to \mathbb{N}^{|\mathcal{C}|}) \to (\Sigma \to (\mathcal{C} \to \mathbb{N}))$$

again to later reason about ADD semantics.

Based on this, the final majority vote abstraction yields the most frequent class:

Definition 22 (Majority Vote Abstraction)
The majority vote abstraction $\delta_C : \mathbb{N}^{|\mathcal{C}|} \to \mathcal{C}$ is defined based on $\#'_V$ and μ (Definition 12):

$$\delta_C := \mu \circ \#'_V.$$

The commuting[10] diagram shown in Fig. 3 indicates that δ_V is a homomorphism. This is important to support an incremental aggregation and abstraction for growing forests. δ_C is no homomorphism, meaning that it has to be applied at the very end.

6 Correctness and optimality

We are now prepared to explain the ADD hierarchy displayed in Fig. 2 and 4. In fact, only Δ_W, Δ_V, and Δ_C still need to be defined. They all arise from lifting (Theorem 6):

Definition 23 (Lifted aggregations and abstractions]
With Theorem 6, we can derive the aggregations and abstractions on ADDs, Δ_W, Δ_V, and Δ_C, from their definitions on the underlying algebras, δ_W, δ_V, and δ_C. Let

- Δ_W be the lifted version of δ_W (Definition 17),
- Δ_V be the lifted version of δ_V (Definition 20), and
- Δ_C be the lifted version of δ_C (Definition 22).

Figure 4 shows the lifted ADD algebras and their relationships. All transformations, except Δ_C, are homomorphisms. This guarantees that the five classifying function compositions that end in $\Sigma \to \mathcal{C}$ are all semantically equivalent, a

[10] indicated by $=$.

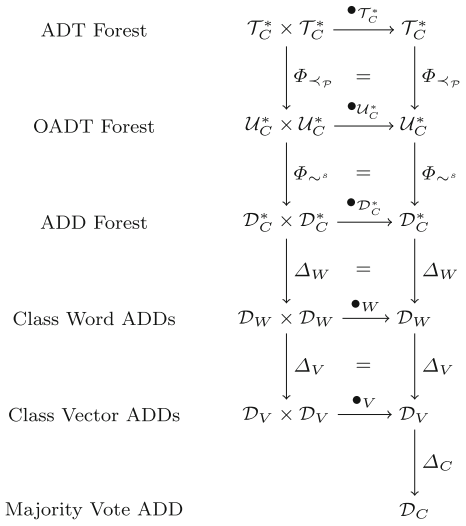

Fig. 4 ADD algebras and their relationships

precondition for the correctness of our incremental aggregation/abstraction approach.

Theorem 8 *(Correctness)*
The majority vote semantics of Random Forests / ADT Forests / OADT Forests / ADD Forests / Class Word ADDs / Class Vector ADDs and Majority Vote ADDs are equivalent:

$$\begin{aligned} M \circ [\![\cdot]\!]_{\mathcal{T}_C^*} &= M \circ [\![\cdot]\!]_{\mathcal{U}_C^*} \circ \Phi_{\prec_{\mathcal{P}}} \\ &= M \circ [\![\cdot]\!]_{\mathcal{D}_C^*} \circ \Phi_{\sim^s} \circ \Phi_{\prec_{\mathcal{P}}} \\ &= M \circ \#_W \circ [\![\cdot]\!]_{\mathcal{D}_W} \circ \Delta_W \circ \Phi_{\sim^s} \circ \Phi_{\prec_{\mathcal{P}}} \\ &= M \circ \#_V \circ [\![\cdot]\!]_{\mathcal{D}_V} \circ \Delta_V \circ \Delta_W \circ \Phi_{\sim^s} \circ \Phi_{\prec_{\mathcal{P}}} \\ &= [\![\cdot]\!]_{\mathcal{D}_C} \circ \Delta_C \circ \Delta_V \circ \Delta_W \circ \Phi_{\sim^s} \circ \Phi_{\prec_{\mathcal{P}}} \end{aligned}$$

While Random Forests necessarily use predicates with potential dependencies, ADDs are typically thought of as using Boolean variables, which are mutually independent. It is worth noting that Theorem 8 makes no assumption about the underlying predicate semantics. It holds for both, the concrete semantics (Definition 2) as well as for the symbolic semantics (Definition 3).

As all the forest and ADD transformations ($\Phi_{\prec_{\mathcal{P}}}$, $\Phi_{\sim^s}$, Δ_W, Δ_V, and Δ_C) can be statically evaluated, the following ADDs can pre-computed:

Definition 24 (Partial evaluation)
For every input forest $\mathbf{t} \in \mathcal{T}_C^*$ we define:
$d_W := \Delta_W \circ \Phi_{\sim^s} \circ \Phi_{\prec_{\mathcal{P}}}(\mathbf{t})$
$d_V := \Delta_V \circ \Delta_W \circ \Phi_{\sim^s} \circ \Phi_{\prec_{\mathcal{P}}}(\mathbf{t})$
$d_C := \Delta_C \circ \Delta_V \circ \Delta_W \circ \Phi_{\sim^s} \circ \Phi_{\prec_{\mathcal{P}}}(\mathbf{t})$

The effect of this partial evaluation is, in particular, that $[\![d_C]\!]_{\mathcal{D}_C}$ can be regarded as a very efficient realisation of the classifier defined by the original Random Forest $\mathbf{t} \in \mathcal{T}_C^*$.

Figures 5, 6, and 7 show the aggregated ADDs for the exemplary Random Forest (Fig. 1). The aggregated structures are not only faster to evaluate (shallower), with increasing size of the forest they are also the smaller representation. This effect can already be observed when comparing the exemplary ADDs to each other. We will analyse these effects in more detail in Sect. 9.

Due to the canonicity (Theorem 3), we know that the resulting ADDs are minimal for representing their decision function for a given predicate ordering $\prec_{\mathcal{P}}$ when considering the predicates as independent. In particular, we have:

Theorem 9 *(Optimality)*
Let $\mathbf{t} \in \mathcal{T}_C^*$ be a Random forest and assume that all predicates in $\mathbf{t}$ are independent, then

$$d_C = \Delta_C \circ \Delta_V \circ \Delta_W \circ \Phi_{\sim^s} \circ \Phi_{\prec_{\mathcal{P}}}(\mathbf{t})$$

is the smallest, $\prec_{\mathcal{P}}$-respecting decision structure satisfying $[\![d_C]\!]_{\mathcal{D}_C} = [\![\mathbf{t}]\!]_{\mathcal{T}_C^*}$.

This result leaves us with two aspects for optimisation:

- **Exploiting the dependencies between predicates:** Dependencies between predicates may lead to infeasible path in the resulting ADDs. Conceptually, eliminating infeasible paths reminds of the well-known *don't care* optimisation of Boolean formulas: There are no normal forms but we can heuristically eliminate all imposed redundancies as long as the underlying logic is decidable. We will discuss such heuristics, which turned out to have quite some impact, in the next section.
- **Finding good predicate orderings**: The reordering Theorem 1 guarantees that we can dynamically adjust the variable ordering at any time and without affecting the ADD semantics. ADD implementations typically come with good heuristics for this purpose [14,32] and they usually aim to reduce the size. While this is the adequate goal for explainability (Sect. 8) and to reduce the memory footprint, it may be a secondary concern in other contexts: To reduce evaluation time (Sect. 9), e.g., reducing the depth of ADDs is more important. Both goals also dependent on the treatment of predicate dependencies which may affect a *good* ordering. Luckily, size-reduced ADDs are typically also shallower and so we can rely on common heuristic implementations, which will not be discussed further in this paper.

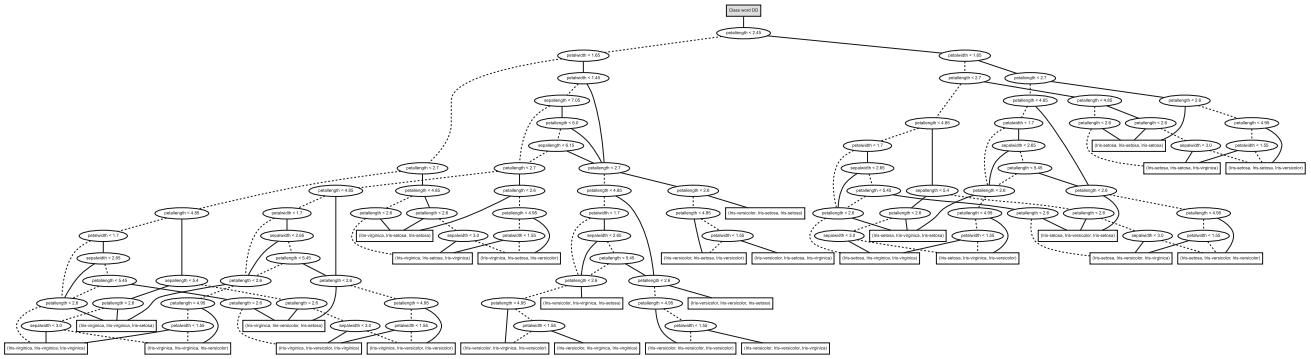

Fig. 5 Class word ADD d_W (108 nodes)

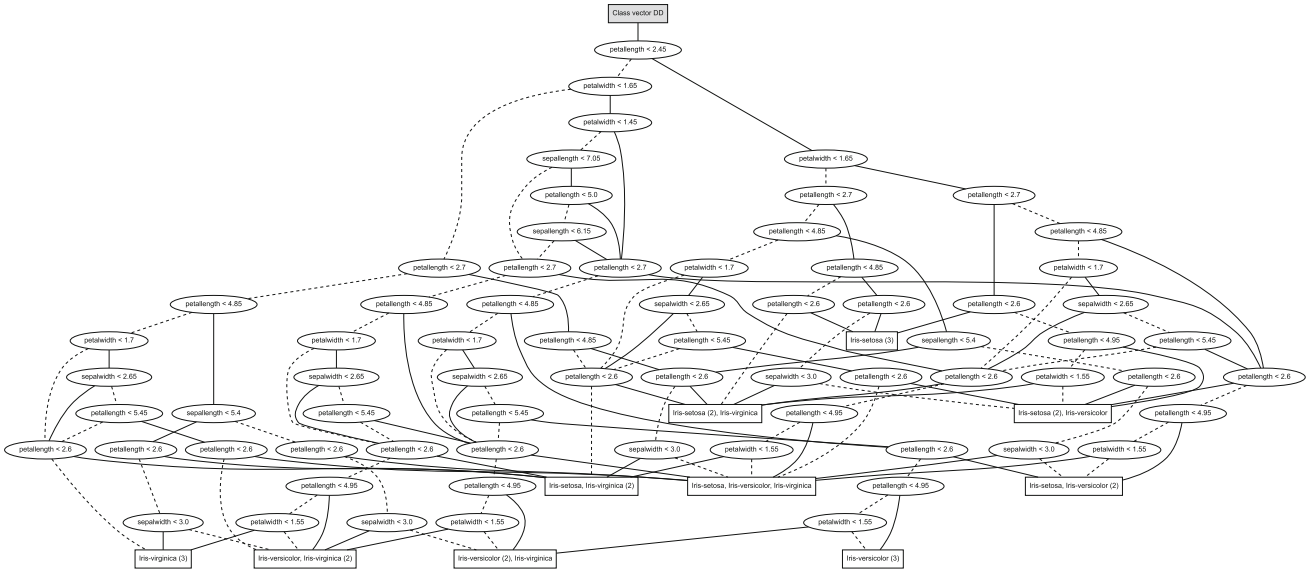

Fig. 6 Class vector ADD d_V (79 nodes)

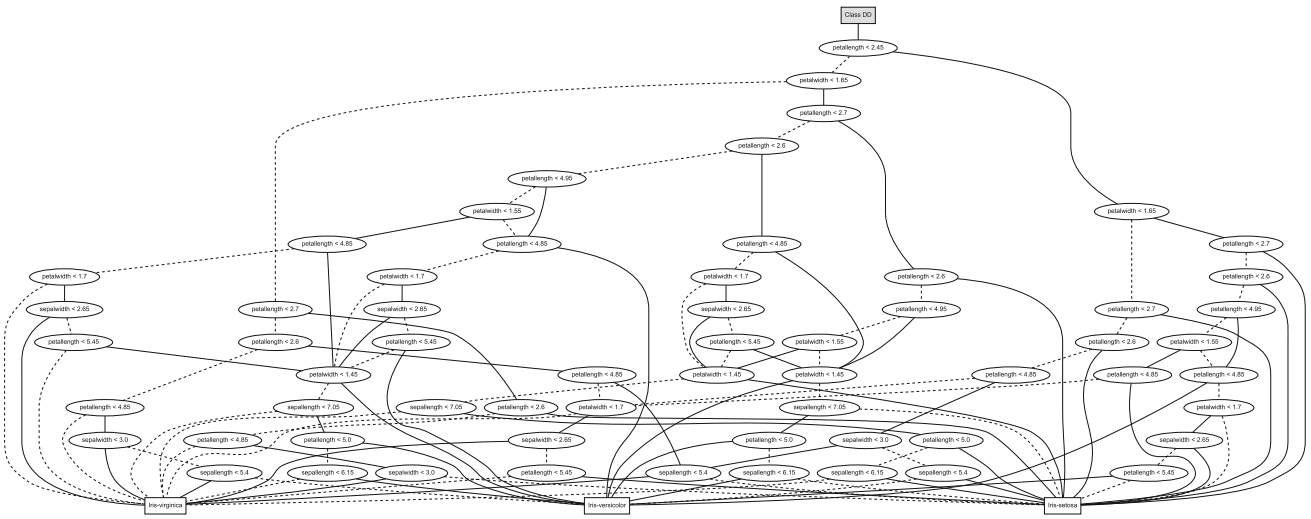

Fig. 7 Majority vote ADD d_C (64 nodes)

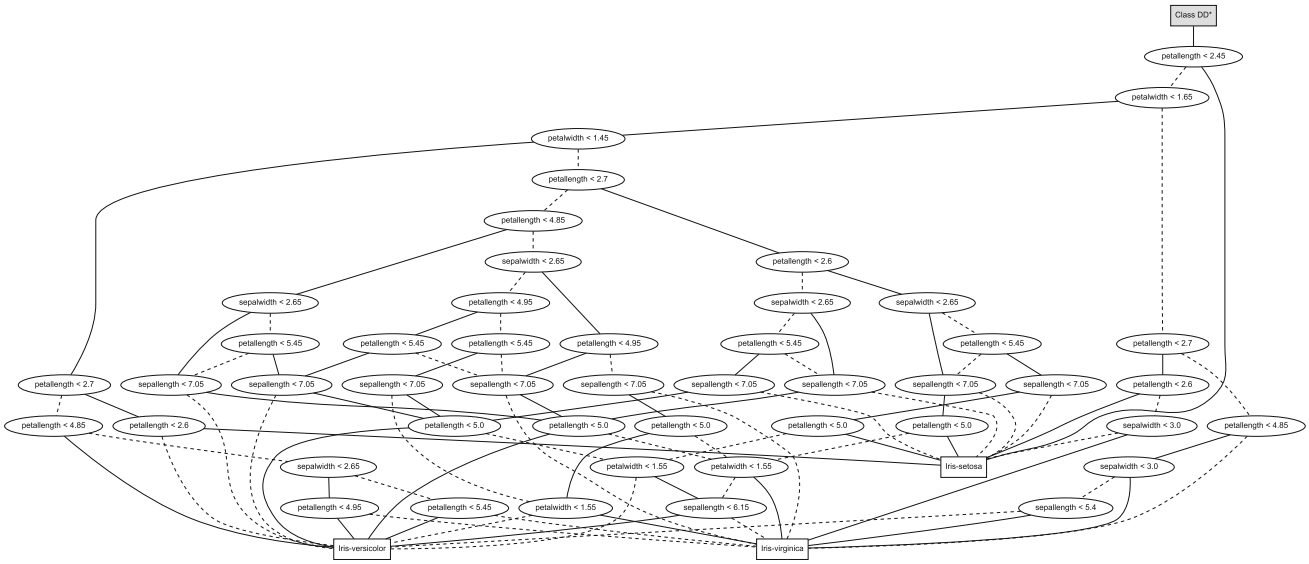

Fig. 8 Redundancy-free majority vote ADD (50 nodes)

7 Infeasible path reduction

Moving from symbolic semantics (Definition 3) to concrete semantics (Definition 2) allows for dependencies between predicates. This means, in particular, that predicates can appear in the aggregated ADDs that are necessary with regard to symbolic semantics but redundant with regard to their concrete semantics.

Definition 25 (Vacuity)
Let $\bar{\mathcal{P}}$ be the set of negated predicates of $\mathcal{P}$. Then we call $\pi = p_0 \cdots p_{m-1} \in (\mathcal{P} \cup \bar{\mathcal{P}})^*$ a predicate path.

- π is called a predicate path of a decision structure $d \in \mathcal{D}_A$ iff there exists a path $\pi' = p'_0 \ldots p'_{m-1} \in \mathcal{P}^*$ from the root of d to one of its leaves such that $p_i = p'_i$ in case that π' follows the left/true branch at p_i in d and $p_i = \bar{p}'_i$ otherwise. We denote the node of d that is associated with the predicate p_i by $node(p_i)$.
- A predicate p_i is called *vacuous* in π iff the conjunction of the preceding predicates $p_0 \cdots p_{i-1}$ in π implies p_i.
- A decision structure $d \in \mathcal{D}_A$ is called *vacuity-free* iff there exists no predicate path with a vacuous predicate.

This allows us to define the following optimisation step.

Definition 26 (Vacuity Reduction)
Let $d \in \mathcal{D}_A$ be a decision structure and let

$$\pi = p_0 \ldots p_{m-1} \in (\mathcal{P} \cup \bar{\mathcal{P}})^*$$

be a predicate path in d on which p_i is vacuous. Then, rerouting the edge from $node(p_{i-1})$ to $node(p_{i+1})$ is called a *vacuity reduction* step.

ADDs, being DAGs, only have finitely many predicate paths which can be effectively analysed for vacuous predicates. As the elimination vacuous predicates is a simple semantics-preserving transformation we have:

Theorem 10 *(Minimality)*
Every ADD can be effectively transformed into a semantically equivalent, vacuity-free ADD that is minimal in the sense that any further reduction would change its semantics.

This guarantees that the following transformation is effective:

Definition 27 (Redundancy Elimination)
A transformation that eliminates all redundancies of a given ADD is called a Redundancy Elimination transformation.

For the simple predicates considered here transformations for eliminating all partial redundancies are easy to implement, e.g., using some SMT solver like Z3 [5]. Please note, however, that the result of redundancy elimination very much depends on the order and how in which partially redundant predicates are treated. Fig. 8 shows a redundancy-free majority vote ADD for our running example.

In the following let $\Delta_v : \mathcal{D}_C \to \mathcal{D}_C$ (polymorphically) denote a (heuristic) transformation that eliminates all vacuities from it argument ADD.

In practice, applying Δ_v on intermediate ADDs, not only between the different phases, but also already when aggregating the ADD Forest, is vital to improve scalability. It should be noted, however, that applying Δ_v does not preserve $\sim_s$ and may therefore have a tangible effect on future reductions.

8 Towards explainability

While the unprocessed Random Forest is impossible to understand in its entirety and is, therefore, considered a black box model in the literature [16], the aggregated ADDs presented above are semantically equivalent representations of the same classification functions in a form that is as easy to understand as usual decision trees. Therefore, we consider the most concise ADDs as illustrated in Fig. 8 as an ideal form of *Model Explanation* [16]. It can be achieved via the following functional:

Definition 28 (Model Explanation)
The function $\Xi_M : \mathcal{T}_C^* \to \mathcal{D}_C$ defined by

$$\begin{aligned}\Xi_M &:= \Delta_v \circ \Delta_C \circ \Delta_V \circ \Delta_W \circ \Phi_{\sim^s} \circ \Phi_{\prec_\mathcal{P}} \\ &:= \Delta_v \circ d_C\end{aligned}$$

is called *Model Explanation Functional*.

Before discussing our solution to *Outcome Explanation*, let us consider the new notion of *Class Characterisation*. It is important to achieve minimal Outcome Explanations, but it is also interesting in its own right.

Class Characterisation is based on a transformation of the Model Explanation Model into a Binary Decision Diagram that characterises a chosen class.

Definition 29 (Binary Decision Diagram)
Binary Decision Diagrams (BDDs) are ADDs over the standard Boolean algebra $B = (\mathbb{B}, \wedge, \vee, \neg)$. Let $\mathcal{D}_B$ denote the set of all BDDs.

BDD-based Class Characterisation can be defined via the following simple transformation function:

Definition 30 (Class Projection)
Given a class $c \in \mathcal{C}$, we define a corresponding projection function $\delta_B(c) : \mathcal{C} \to \mathbb{B}$ on the co-domain as

$$\delta_B(c)(c') := \begin{cases} 1 & \text{if } c' = c \\ 0 & \text{otherwise.} \end{cases}$$

for $c' \in \mathcal{C}$. Again, the function $\delta_B(c)$ can be lifted to operate on ADDs, yielding $\Delta_B(c) : \mathcal{D}_C \to \mathcal{D}_\mathbb{B}$.

Moving from the majority vote algebra to the standard Boolean logic via class projection continues our line of abstraction. The resulting BDDs are explanations of the Random Forest's behaviour that provide a precise and very focused explanation of when a certain class is chosen (Fig. 9, where *Iris-setosa* stands for 1). Like Model Explanation, the resulting Class Characterisation can be obtained via a corresponding functional:

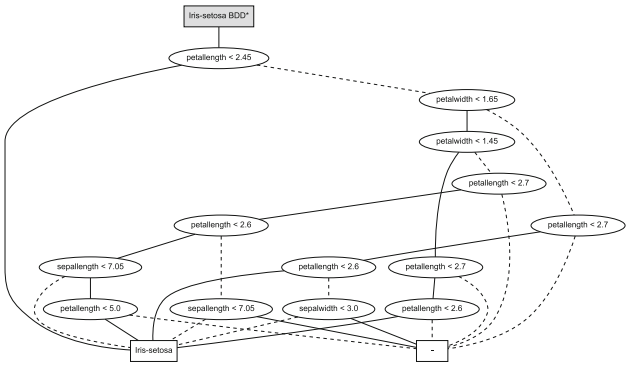

Fig. 9 Explanation BDD for the class *Iris-setosa* (15 nodes)

Definition 31 (Class Characterisation)
The function $\Xi_C : \mathcal{C} \to (\mathcal{T}_C^* \to \mathcal{D}_\mathbb{B})$ defined by

$$\Xi_C(c) \;=\; \Delta_B(c) \circ \Xi_M$$

is called *Class Characterisation Functional*.

Figure 9 shows the Class Characterisation for the class *Iris-setosa* which has only 15 nodes, in comparison to the 50 nodes of the Model Explanation. In fact, the sum of the number of nodes in the Class Characterisations for the three classes is smaller than 50, which indicates the potential for a corresponding *semantic decomposition* of the Model Explanation Model.

Class Characterisation is particularly interesting because it allows one to *reverse* the classification process: Instead of determining a class for a certain sample, one obtains a characterisation of the set of all samples that will be classified as the given class. This change of perspective may have an important impact, e.g., in marketing contexts for switching from a customer to a product perspective [12].

For a responsible use of automatically derived classifications, the *Outcome Explanation Problem* is essential [16]. Class Characterisations allow us to solve this problem in two further steps:

- **Path-based explanation.** Focusing on just one input at a time allows us to further refine the obtained explanation. When evaluating the aggregated BDD resulting from the first step, the conjunction of the components of the corresponding predicate path, i.e., of the predicates (if required negated) along the classification trace, provides a sufficient condition for the decision made. E.g., considering the sample

$$\begin{aligned} petallength &= 2.6, \\ petalwidth &= 1.5, \\ petallength &= 2.65, \\ sepallength &= 6,9, \end{aligned}$$

we obtain

$$\begin{aligned} & petallength \geq 2.45 \\ \wedge\ & petalwidth < 1.65, \\ \wedge\ & petalwidth \geq 1.45, \\ \wedge\ & petallength < 2.7, \\ \wedge\ & petallength \geq 2.6, \\ \wedge\ & sepallength < 7.05 \end{aligned}$$

- **Simplifying conjunctions.** The collected predicates along a trace may yield redundancies—even when unsatisfiable paths were eliminated from the ADD. Removing choices from the conjunction as long as redundant choices exist[11] yields a minimal explanation of the finally predicted class.[12] In our example, $petallength \geq 2.45$ is redundant relative to the stricter predicate $petallength \geq 2.6$, which overall leads to a conjunction with five predicates.

This two step transformation

$$\Omega : \mathcal{D}_{\mathbb{B}} \to (\Sigma \to \mathcal{L})$$

yields the desired Outcome Explanation, which can be obtained via the following functional:

Definition 32 (Outcome Explanation)
The function $\Xi_O : \mathcal{T}_C^* \to (\Sigma \to \mathcal{L})$ defined by

$$\begin{aligned} \Xi_O(\mathbf{t})(\sigma) &:= \Omega(\Xi_C(c)(\mathbf{t}))(\sigma) \\ \text{with } c &:= \Xi_M(\mathbf{t})(\sigma) \end{aligned}$$

for a given ADT Forest $\mathbf{t}$ and $\sigma \in \Sigma$ is called *Outcome Explanation Functional*.

Please note that it is important to go via the Class Characterisation. Otherwise the Outcome Explanation may be significantly larger because of predicates separating detail not required for the explanation of the outcome of considered class.

The three forms of explanation are precise (in the sense of representing the same classification function) and optimal up to variable re-ordering and infeasible path elimination, which are both known to have canonical solutions:

[11] Please note that there may be redundant predicates on a path that are all necessary for the BDD due to the considered predicate ordering.

[12] Please note that, in contrast to the vacuity elimination during infeasible path reduction, the order in which the predicates appear in the predicate path does not matter here.

Theorem 11 *(Explanation)*
Given an ADD-Forest $\mathbf{t}$ *and a class* $c \in \mathcal{C}$ *we have:*
Precision of Explanation:

- $\Xi_M(\mathbf{t}) = M \circ [\![\mathbf{t}]\!]_{\mathcal{T}_C^*}$
- $\forall \sigma \in \Sigma.\ \Xi_C(c)(\mathbf{t})(\sigma) = 1$ *iff* $M \circ [\![\mathbf{t}]\!]_{\mathcal{T}_C^*}(\sigma) = c$
- $[\![\Xi_O(\mathbf{t})(\sigma)]\!]_{\mathcal{P}}(\sigma') = 1 \Rightarrow \Xi_M(\mathbf{t})(\sigma) = \Xi_M(\mathbf{t})(\sigma')$ *for all* $\sigma, \sigma' \in \Sigma$.

Conciseness of Explanation: The Explanation models $\Xi_M(\mathbf{t})$, $\Xi_c(c)(\mathbf{t})$, *and* $\Xi_O(\mathbf{t})(\sigma)$ *are size minimal up to two heuristics: (1) variable re-ordering and (2) infeasible path elimination.*

The proof of this theorem follows straightforwardly from the results of the previous section and the canonicity of the ADD construction.

Explainability is, however, not the only effect of our ADD-based aggregation. As illustrated in the next section, it leads to drastic reductions of the classification time.

9 Experimental performance evaluation

ADD-based aggregation radically improves the evaluation time. In fact, it is even provably optimal for chosen predicate ordering in the sense that each predicate is considered at most once, and only if its evaluation is required. Our experiments with popular data sets from the UCI Machine Learning Repository [8] showed performance gains of several orders of magnitude (cf. Fig. 10 and Table 1).

A potential problem of our aggregation method is an explosion in size which can, in principle, be exponential for decision diagrams. However, this problem did not arise in our experiments. On the contrary, we even observed drastic size reductions (cf. Fig. 11 and Table 2), an effect we cannot generally expect.

It should be noted, however, that all results reported in this section are achieved in a very generic, algebraic fashion using a common classifier on standard datasets, which indicates at least some generalisation potential.

Our three-tree accompanying example is useful to explain the concepts but inadequate to illustrate the impact of our radical aggregation technology. This section, therefore, provides a careful quantitative analysis on the basis of a number of popular data sets that illustrate the performance differences between the semantically equivalent representations of the original Random Forest.

The diagrams in this section show the results of our experiments with the Iris flower data set, which was previously also used for our small running example. Tables 1 and 2 summarise the results for other popular data sets to indicate the generalisation potential of our approach. All the reported classification time and size results were determined as an average over the complete corresponding data set. For the

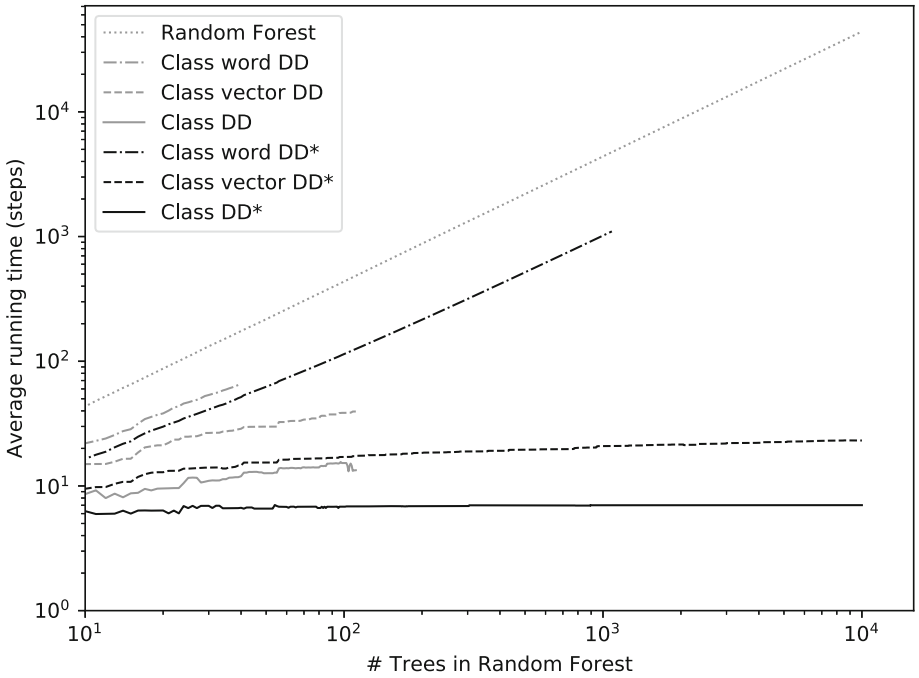

Fig. 10 Average running time for classification over all examples in the Iris dataset [10]

Table 1 Running time improvements for classification with Random Forests of size 10.000 for other datasets [8]

Dataset	100		1,000		10,000	
	Random forest	Final DD	R. forest	Final DD	R. forest	Final DD
Balance Scale	802.21	7.71 (– 99.04%)	8,014.12	7.73 (– 99.90%)	80,277.03	8.16 (– 99.99%)
Breast Cancer	1,298.72	17.12 (– 98.68%)	13,020.03	17.11 (– 99.87%)	130,361.20	17.73 (– 99.99%)
Lenses	452.50	3.67 (– 99.19%)	4,431.42	3.67 (– 99.92%)	43,883.79	3.67 (– 99.99%)
Iris	436.11	6.82 (– 98.44%)	4,395.77	6.97 (– 99.84%)	44,043.89	7.01 (– 99.98%)
Tic-Tac-Toe	1,066.66	14.25 (– 98.66%)	10,733.68	14.22 (– 99.87%)	107,300.69	14.18 (– 99.99%)
Vote	693.57	9.02 (– 98.70%)	6,921.56	8.33 (– 99.88%)	69,216.62	8.30 (– 99.99%)

Iris flower example, these are 150 records, a number that also explains the quite smooth result graphs.

Our implementation relies on the standard Random Forest implementation in Weka [37] and on the ADD implementation of the ADD-Lib [13,14,33]. Please note that the considered data sets have been developed with evaluations of this kind in mind by independent parties, and that we are not using any additional data for our transformation. Thus, our analysis can be considered unbiased.

Optimising the classification time was the original goal when we started to develop our approach. As wall-clock time measurements are very sensitive to implementation details and machine profiles, we decided for the—in our eyes more objective—step count measure for performance analysis. We consider here the steps through the corresponding data structures, and in cases where the most frequent class must be computed at runtime, we account one additional step per read. For both, the original Random Forest and the word-based decision diagram these are n additional steps and the class vector variant needs $|\mathcal{C}|$ additional steps.

Figure 10 shows the average evaluation times of the decision models for Random Forests of up to 10,000 trees. The evaluation time of the original Random Forest grows linearly as expected: Every new tree contributes approximately the same running time. Due to the large number of trees relative to their individual sizes our measurements appear as an almost straight line.

Already, the word-based diagrams (see Class word DD in Fig. 10) significantly reduce the classification time in comparison to the original Random Forest. This is due to the suppression of redundant predicate evaluations. In fact, the overall classification time is dominated by the linearly growing time to compute the most frequent class in each terminal word.

The reduction to just $|\mathcal{C}|$ terminal nodes of the class vector-based variants has two effects:

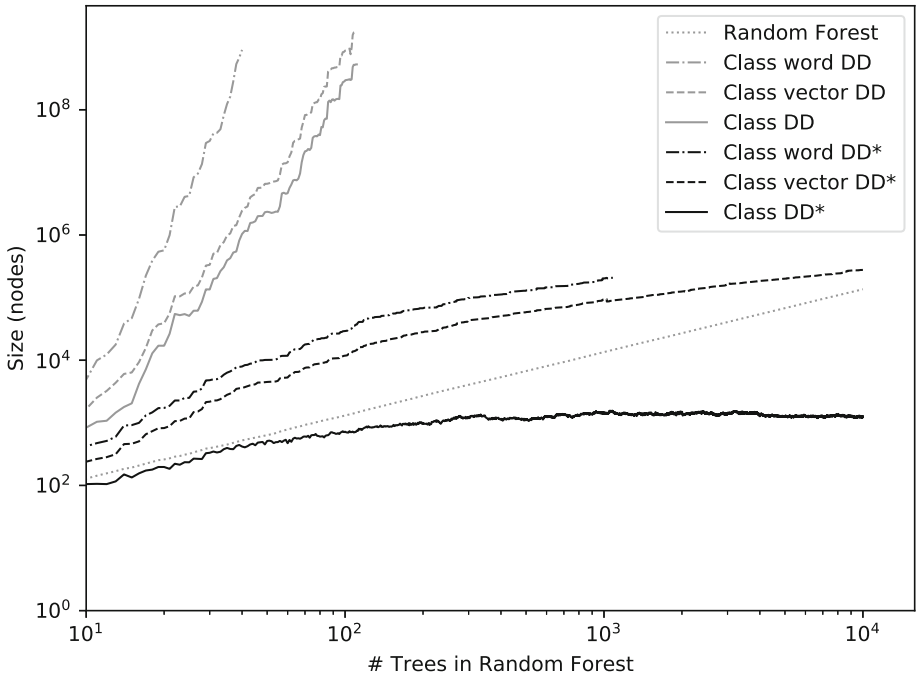

Fig. 11 Sizes of the random forest and its semantically equivalent decision diagrams

Table 2 Decision diagram sizes for Random Forests of size 10.000 for other datasets [8]

Dataset	100		1,000		10,000	
	Random forest	Final DD	R. forest	Final DD	R. forest	Final DD
Balance Scale	21,720	137 (– 99.37%)	214,844	139 (– 99.94%)	2,158,330	144 (– 99.99%)
Breast Cancer	55,172	3,501 (– 93.65%)	546,504	3,647 (– 99.33%)	5,494,682	3,760 (– 99.93%)
Lenses	1,518	11 (– 99.28%)	14,132	11 (– 99.92%)	136,986	11 (– 99.99%)
Iris	1,312	722 (– 44.97%)	13,492	1,458 (– 89.19%)	135,952	1,267 (– 99.07%)
Tic-Tac-Toe	55,232	1,563 (– 97.17%)	570,976	1,593 (– 99.72%)	5,670,532	1,529 (– 99.97%)
Vote	9,768	1,337 (– 86.31%)	97,770	1,168 (– 98.81%)	988,358	1,148 (– 99.88%)

- A partial collapse of the decision diagram: It is no longer essential which tree proposes which class, unifying all cases where the various classes are equally often proposed.
- Reduction to a constant overhead for the final aggregation step, in this case $|\mathcal{C}|$.

The evaluation time reductions are again quite significant. Only the space requirement got (like for the word-based variant) out of hand very soon (see Fig. 11), explaining the cut-off in Fig. 10.

Whereas the previous two model structures can directly be computed compositionally, the most frequent label abstraction, i.e. the evaluation of the *majority vote* at compile time, can only be applied at the very end. Thus its construction has the same limitation as the class vector variant, and its impact on the size of the corresponding decision model is comparatively moderate (see Fig. 11). Its impact on the evaluation time is, however, quite substantial (see Fig. 10): Many of the internal decision nodes have become redundant by just focusing on the results of the majority vote.

Exploiting the dependencies between the predicates by unsatisfiable path elimination overcomes the scalability problems that are due to the enormous space requirements. In fact, in our experiments it avoids the exponential blow-up in size in all three variants, with DD* even becoming significantly smaller than the original Random Forest (see Fig. 11).

Moreover, the classification times are drastically reduced in all three cases (see Fig. 1). In fact, the classification times eventually stabilise for DD^*, illustrating the key feature of Random Forests, the reduction of the learner's variance. [13] As sketched in Tables 1 and 2, these observations carry over to other popular data sets in the UCI Machine Learning Repository [8].

[13] Remember, being just a different representation of the original Random Forest, DD^* has the same variance.

10 Related work

Related work splits into two categories, explainability of automatically generated classifications and the corresponding classification performance.

In machine learning, explainability is a topic of increasing importance, as learned structures, be it, e.g., neural networks or, as in the focus of this paper, Random Forests, are more and more used to replace/support human decisions [30]. In particular in cases where the proposed classification is apparently counter-intuitive, explanation is important.

Being based on a well-understood predicate structure, Random Forests are easier to control than, e.g., neural networks. Nevertheless, because of their diverse structure with its high degree of parallelism they are considered black-box models [16].

Various methods for making Random Forests interpretable exist such as extracting decision rules from the considered black-box model [6], methods that are agnostic to the black-box model under consideration [21,29] or by deriving a single decision tree from the black-box model [4,7,18,35,38]. In this context, single decision trees are considered key to a solution to both

- the *model explanation problem* [16], i.e. the problem to make Random Forests as a whole interpretable and understandable by humans. Here, the tree itself is considered to be understandable. And
- the *outcome explanation problem* [16], i.e. explaining (the reasons for) a concrete classification to a human. Here the conjunction of the (if required negated) predicates on the path from the root to the classifying leaf are considered an adequate explanation.

State of the art solutions to derive a single decision tree from a Random Forest are approximative [4,7,18,35,38]. Thus, their derived explanations are not fully faithful to the original semantics of the considered Random Forest. This is in contrast to our ADD-based aggregation, which precisely reflects the semantics of the original Random Forest. This means:

- ADD-based aggregation provides a precise solution to the *model explanation problem* (Sect. 8). The fact that ADDs are not trees but acyclic graphs can either be accepted or easily be taken care of by expanding the ADD to a tree.
- Our formula derived in Sect. 8 is a precise and minimal explanation of the proposed classification. To our knowledge none of the existing approaches for explaining Random Forest in terms of decision trees are precise.

Runtime performance of Random Forests has been addressed, e.g., via optimising code generation with moderate success [9,20,34,37], and, with a greater performance impact, via model simplification which, however, changed the semantics [17]. Yet, others applied semantic aggregation [1,17,22,26] to Random Forests without explicitly addressing the runtime performance, while the authors in [24,25] focused solely on the memory footprint, all with moderate success.

The only paper on Random Forests we know of that uses decision diagrams similar our ADDs is [23]. However, it uses these diagrams only to compact the individual tree rather than to aggregate an entire forest. In fact, the reported speedup by a factor of up to 61 seems to rather rely on technical and even hardware details than on the use of decision diagrams. In contrast, our approach focuses on the decision diagram-based holistic aggregation of entire Random Forests, which, due to its globality, has a much greater impact. In fact, we obtain speed-ups already at the hardware-independent level that are orders of magnitude higher than in [23].

11 Conclusions and future work

In this paper, we have presented an approach to aggregate large Random Forests into single and compact decision diagrams that faithfully reflects the semantics of the original Random Forest for a considered purpose. Here purpose comprises intents, like understandability and performance, but also more technical features like compositionality. Key to our approach is the combination of conceptual ideas:

- *Aggregation:* Decision trees can easily be translated into ADDs. In cases where the set of leaves of the original decision tree forms an algebra this translation can be used to lift the algebraic semantics to the ADD-level: The corresponding set of ADDs itself becomes an algebra which reflect all the algebraic operations of the leaf algebra. This is best illustrated by looking at the class vector ADDs. The majority vote underlying the Random Forest evaluation can be straightforwardly mimicked by aggregating the ADDs for the different decision trees according to addition.
- *Abstract Interpretation:* Abstracting the leaf algebra allows one to tailor the ADD-based representation according to the considered needs: While, e.g., the word algebra faithfully reflects the original Random forest, the class vector algebra anonymises the 'voters', but is still detailed enough to faithfully and compositionally model the majority vote. The other considered abstractions either trade compositionality (most frequent label abstraction) or introduce a special focus (class characterisation).
- *Predicate Evaluation:* The ADD algebra considers the underlying predicates as independent. Thus, it is not sensitive to relations between predicates, like inconsistency

or redundancy. It is, however, possible to take care of these aspects in a separate process called infeasible path reduction (cf. Sect. 7), which, in fact, does not even affect compositionality of the imposed semantics. Also the elimination of redundant predicates for obtaining concise outcome explanations (cf. Sect. 8) falls into the category of predicate evaluation.

We have seen that these three conceptual ideas in combination allow one to increase understandability and performance without any penalty concerning precision:
Explainability. Individual ADDs are intuitively as easy to understand as individual decision trees, a typical format used to explain Random Forests [16]. Our approach can therefore be considered to provide solutions to three black box explanation problems:

- *Model explanation problem:* The most frequent class abstraction (cf., e.g., Fig. 7) itself precisely reflects the classification semantics of the original Random Forest.
- *Class characterisation problem:* The most frequent class abstraction (cf., e.g., Fig. 9) which distinguishes a particular class from the rest is a concise and precise specification of all samples that the Random Forest will classify as this class. To our knowledge, our BBD construction is the first solution to this problem which, as a side-effect, allows one to reverse the original classifcation perspective from sample-to-class to class-to-samples.
- *Outcome explanation problem:* The reduced conjunction of the predicates characterising a sample's path through the class characterising BDD is a sufficient condition that explains the Random Forest's classification of the considered sample. In the considered example, this conjunction just comprises five comparisons.

To our knowledge, these solutions are all unique in precisely reflecting the semantics of the underlying Random Forest [4,16,18,38].
Rapid Evaluation. We have reported running time reductions by factors of thousands on multiple popular datasets (cf. Fig. 10 and Tab. 1). It is interesting to note both the quantitative performance gain and the clean semantics-oriented way in which it is achieved:

- ADD-based aggregation is canonical as soon as an order of predicates has been fixed. Thus the freedom of choice here reduces to the choice of an adequate variable ordering, a task heuristically taken care of by the corresponding frameworks [32].
- *Class frequency* abstraction is the coarsest, compositional abstraction that still allows one to faithfully represent the classification function of the original Random Forest.
- infeasible path reduction does not support normal forms, but the results are minimal, meaning that the resulting structures cannot be reduced further without changing the semantics of the classification function. In essence, the variability here is a consequence of the freedom of choice where to *root* infeasible paths. It can be seen as a generalisation of the classical problem of minimising Boolean functions with *don't cares*.
- *Most frequent class* abstraction reduces the final compositionally reduced decision diagrams to the smallest diagram that still represents the original classification function.

Thus our approach is optimal relative to two well-known conceptual hurdles, the choice of variable ordering for decision diagrams and the treatment of *don't cares*. In particular, our approach does not exploit any peculiarities of certain classifiers or data sets.

Of course, the impact of our approach may still strongly depend on the structure of the specific considered scenario. We are therefore currently investigating how easily these results can be adopted to other data sets and classifiers. An extension to weighted Random Forests [36], or even to forests that propose distributions[14] can be achieved simply by choosing an adequate leaf algebra for the ADDs.

Whereas introducing weights for the individual trees of a Random Forest should not cause any additional overhead, the treatment of distributions, even though being technically easy to realise, may come with an explosion of the size of the leaf algebra which we expect to require an approximative treatment.

An even more critical source of explosion is the number of predicates. They may lead to an exponential growths of the resulting ADDs, and will probably also require adequate approximation techniques. On the other hand, we do not expect any specific problems for treating slight structural variations like Decision Jungles [31].

The generalisation potential of ADD-based aggregation is large, as is the challenge to overcome or bypass the obstacles that characterise a new application scenario. Dealing with huge leaf algebras or with vast sets of predicates are just two obvious examples for such challenges, many of which will have to be dealt with using clever heuristics.

Funding Open Access funding enabled and organized by Projekt DEAL.

[14] With distribution we mean here an association of weights to the classes which sum up to 1.

References

1. Bonfietti, A., Lombardi, M., Milano, M.: Embedding decision trees and random forests in constraint programming. In: Michel, L. (ed.) Integration of AI and OR Techniques in Constraint Programming. pp. 74–90. Springer International Publishing, Cham (2015)
2. Breiman, L.: Random forests. Mach. Learn **45**(1), 5–32 (2001)
3. Bryant, R.E.: Graph-based algorithms for boolean function manipulation (1986)
4. Chipman, H.A., George, E.I., McCulloch, R.E.: Making sense of a forest of trees (1999)
5. De Moura, L., Bjørner, N.: Z3: An efficient smt solver. In: International conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 337–340. Springer (2008)
6. Deng, H.: Interpreting tree ensembles with intrees. Int. J. Data Sci. Anal. **7**(4), 277–287 (2019). https://doi.org/10.1007/s41060-018-0144-8
7. Domingos, P.M.: Knowledge discovery via multiple models. Intell. Data Anal. **2**(1–4), 187–202 (1998). https://doi.org/10.1016/S1088-467X(98)00023-7
8. Dua, D., Graff, C.: UCI machine learning repository. http://archive.ics.uci.edu/ml (2017)
9. Facebook: evaluating boosted decision trees for billions of users. https://code.fb.com/ml-applications/evaluating-boosted-decision-trees-for-billions-of-users (2017). Accessed: 11 June 2019
10. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugen. **7**(7), 179–188 (1936)
11. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugen. **7**(2), 179–188 (1936)
12. Gossen, F., Margaria, T., Steffen, B.: Towards explainability in machine learning: the formal methods way. IT Prof. **22**(4), 8–12 (2020). https://doi.org/10.1109/MITP.2020.3005640
13. Gossen, F., Margaria, T., Murtovi, A., Naujokat, S., Steffen, B.: Dsls for decision services: a tutorial introduction to language-driven engineering. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation. Modeling. pp. 546–564. Springer International Publishing, Cham (2018)
14. Gossen, F., Murtovi, A., Linden, J., Steffen, B.: The java library for algebraic decision diagrams. https://add-lib.scce.info. Accessed 1 Apr 2019
15. Gossen, F., Steffen, B.: Large random forests: optimisation for rapid evaluation. CoRR **abs/1912.10934** (2019). http://arxiv.org/abs/1912.10934
16. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 93:1–93:42 (2019). 10.1145/3236009,
17. H. Kargupta, B.P.: A fourier spectrum-based approach to represent decision trees for mining data streams in mobile environments. IEEE Trans. Knowl. Data Eng. **16** (2004)
18. Hara, S., Hayashi, K.: Making tree ensembles interpretable: a bayesian model selection approach. In: Storkey, A.J., Pérez-Cruz, F. (eds.) International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain. Proceedings of Machine Learning Research, vol. 84, pp. 77–85. PMLR (2018). http://proceedings.mlr.press/v84/hara18a.html
19. Ho, T.K.: Random decision forests. In: Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1). ICDAR '95, IEEE Computer Society, Washington (1995)
20. Browne, J., Mhembere, D., Tomita, T.M., Vogelstein, J.T., and Burns, R.: Forest packing: fast parallel, decision forests (2019)
21. Lou, Y., Caruana, R., Gehrke, J.: Intelligible models for classification and regression. In: Yang, Q., Agarwal, D., Pei, J. (eds.) The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, 12–16 Aug 2012, pp. 150–158. ACM (2012). https://doi.org/10.1145/2339530.2339556,
22. Mulvaney R., P.D.: A method to merge ensembles of bagged or boosted forced-split decision trees. IEEE Trans. PAM (2003)
23. Nakahara, H., Jinguji, A., Sato, S., Sasao, T.: A random forest using a multi-valued decision diagram on an fpga. In: 2017 IEEE 47th International Symposium on Multiple-Valued Logic (ISMVL), pp. 266–271 (2017)
24. Painsky, A., Rosset, S.: Lossless compression of random forests. J. Comput. Sci. Technol. **34**(2), 494–506 (2019)
25. Peterson, A.H., Martinez, T.R.: Reducing decision tree ensemble size using parallel decision dags. Int. J. Artif. Intell. Tools **18**(04), 613–620 (2009)
26. Philippe J. Giabbanelli, J.G.P.: An algebra to merge heterogeneous classifiers. CoRR (2015)
27. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
28. R. Iris Bahar, Erica A. Frohm, C.M.G.G.D.H.E.M.A.P.F.S.: Algebraic decision diagrams and their applications. In: Proceedings of the 1993 IEEE/ACM International Conference on Computer-aided Design. IEEE Computer Society Press (1993)
29. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should I trust you?": Explaining the predictions of any classifier. In: Krishnapuram, B., Shah, M., Smola, A.J., Aggarwal, C.C., Shen, D., Rastogi, R. (eds.) Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pp. 1135–1144. ACM (2016). 10.1145/2939672.2939778,
30. Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K. (eds.): Explainable AI: interpreting, explaining and visualizing deep learning. In: Lecture Notes in Computer Science, vol. 11700. Springer (2019). Doi: https://doi.org/10.1007/978-3-030-28954-6,
31. Shotton, J., Nowozin, S., Sharp, T., Winn, J., Kohli, P., Criminisi, A.: Decision jungles: compact and rich models for classification. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, Vol. 1 (2013)
32. Somenzi, F.: Efficient manipulation of decision diagrams. Int. J. Softw. Tools Technol. Transf. 3(2) (2001)
33. Steffen, B., Gossen, F., Naujokat, S., Margaria, T.: Language-Driven Engineering: From General-Purpose to Purpose-Specific Languages, pp. 311–344. Springer International Publishing, Cham (2019)
34. Treelite: Treelite: model compiler for decision tree ensembles. https://treelite.readthedocs.io (2017). Accessed 7 June 2019
35. Van Assche, A., Blockeel, H.: Seeing the forest through the trees: Learning a comprehensible model from an ensemble. In: Kok, J.N., Koronacki, J., Mantaras, R.L.d., Matwin, S., Mladenič, D., Skowron, A. (eds.) Machine Learning: ECML 2007. pp. 418–429. Springer, Berlin (2007)

36. Winham, S.J., Freimuth, R.R., Biernacka, J.M.: A weighted random forests approach to improve predictive performance. Stat. Anal. Data Min. ASA Data Sci. J **6**(6), 496–505 (2013)
37. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques, 4th edn. Morgan Kaufmann Publishers Inc., San Francisco (2016)
38. Zhou, Y., Hooker, G.: Interpreting models via single tree approximation (2016)