Decision Trees: Equivalence and Propositional Operations

Hans Zantema

Department of Computer Science, Utrecht University,
Padualaan 14, P.O. box 80.089, 3508 TB Utrecht, The Netherlands,
e-mail: hansz@cs.ruu.nl

Abstract. For the well-known concept of decision trees as it is used for inductive inference we study the natural concept of equivalence: two decision trees are equivalent if and only if they represent the same hypothesis. We present a simple efficient algorithm to establish whether two decision trees are equivalent or not. The complexity of this algorithm is bounded by the product of the sizes of both decision trees.
The hypothesis represented by a decision tree is essentially a boolean function, just like a proposition. Although every boolean function can be represented in this way, we show that disjunctions and conjunctions of decision trees can not efficiently be represented as decision trees, and simply shaped propositions may require exponential size for representation as decision trees.

1 Introduction

The problem of *inductive inference*, or shortly *induction*, in machine learning ([7]) can be described as follows. Roughly speaking, a number of observations each having an outcome, has to be used to predict the outcome of new observations. More precisely, given a finite set of *attributes* A, a finite *instance space* $X = \prod_{a \in A} X_a$ and a finite *training set* of observations $(x_i, y_i) \in X \times \{\mathsf{true}, \mathsf{false}\}$ for $i = 1, 2, \ldots, n$, we have to find a suitable total function $h : X \to \{\mathsf{true}, \mathsf{false}\}$, called the *hypothesis*, such that $h(x_i) = y_i$ for all $i = 1, 2, \ldots, n$. The objective is to find a hypothesis reflecting underlying unknown regularities in the observations as much as possible. A standard way is by means of a *decision tree* ([8]): a directed tree in which

- every internal node including the root is labelled by an attribute $a \in A$;
- every internal node labelled by an attribute $a \in A$ has exactly $\#X_a$ outgoing edges, each of them labelled by a unique element of X_a;
- every leaf is labelled by true or by false.

Such a decision tree T represents a hypothesis in the following way: to compute the result of the hypothesis for a particular instance start in the root of the tree and follow a path by choosing in every node the edge labelled by the corresponding value. Then the result of the hypothesis is the label of the leaf at the end of this path.

The standard way of decision tree induction is now as follows: for a training set of observations find a corresponding decision tree T using some greedy algorithm, and yield the corresponding hypothesis. For this purpose two decision trees behave equally if the two corresponding hypotheses are equal. This is called *decision equivalence* ([2, 3]). This notion of equivalence is central in this paper. It is a natural notion that has various practical implications. For instance, the standard greedy algorithms for growing decision trees often yield a decision tree for which an equivalent smaller decision tree exists. As an example consider the following training set of 16 observations with three binary attributes p, q, r and one binary 'result' :

frequency	p	q	r	result
5	true	true	true	true
1	true	true	false	false
1	true	false	true	false
1	false	true	true	true
4	false	true	false	false
4	false	false	true	false

Let T be the decision tree $q(r(\mathsf{true},\mathsf{false}),\mathsf{false})$, or $r(q(\mathsf{true},\mathsf{false}),\mathsf{false})$ by symmetry between q and r, in the notation to be introduced in Section 2. The decision tree T represents the conjunction of q and r and is consistent with all 16 observations. However, the standard greedy algorithm for growing decision trees based on the information gain criterion as in [8] yields the decision tree $p(T,T)$, which is equivalent to the much smaller decision tree T. A usual objective, motivated by Occam's Razor, is trying to find a decision tree that is as small as possible. Finding the smallest tree consistent with some training set is NP-hard ([6]); from this example we see that replacing the resulting tree by a smaller equivalent tree may already be helpful. In [3] an efficient algorithm is presented for this goal.

Instead of elaborating this kind of practical applications, in this paper we concentrate on basic theoretical issues. We present a new quadratic algorithm for deciding whether two decision trees are equivalent or not, more precisely, the complexity of the algorithm is bounded by the product of the sizes of both decision trees.

Finally we compare propositions and decision trees; in fact the hypothesis represented by a decision tree is a function yielding a boolean value, just like a proposition. Since establishing equivalence of propositions is known to be NP-complete and establishing equivalence of decision is done by a quadratic algorithm we expect that efficient representation of propositions as decision trees will not be possible. Indeed we prove that in worst case for two given decision trees the smallest decision tree yielding the disjunction of the hypotheses of both given trees has a size exceeding the product of the sizes of the given trees, and similar for conjunction. Surprisingly, the upper and lower bounds we can give for these sizes exactly coincide. Moreover, for arbitrary n we give a simple proposition of size $O(n)$ for which we prove that the size of the smallest decision tree representing this proposition is exponential in n.

To keep the treatment concise we assume all attributes to be binary, i.e., $X_a = \{\mathsf{true},\mathsf{false}\}$ for all $a \in A$. All our results straightforwardly generalize to arbitrary finite sets X_a; we do not do it here because of a lot of extra notation needed. In this way our treatment covers all decision trees with nominally valued attributes; numeric attributes handled by binary splits are not covered.

In Section 2 we present the basic definitions. In Section 4 deciding equivalence of two decision trees is discussed. The relation between propositions and decision trees is elaborated in Section 5.

2 Basic definitions

We consider a finite set A of binary *attributes* of which typical elements are denoted by $p, q, r, \ldots$. An *instance* s over A is defined to be a map from A to $\{\mathsf{true},\mathsf{false}\}$; intuitively for an attribute p and an instance s the value $s(p)$ represents whether the boolean attribute p holds for the instance s or not. Note that in the introduction we considered instances as being elements of $\prod_{a\in A} X_a$; by the assumption that $X_a = \{\mathsf{true},\mathsf{false}\}$ for all $a \in A$ this difference is only a matter of notation.

A *decision tree* over A is a binary tree in which every internal node is labelled by an attribute and every leaf is labelled either true or false. More formally, the set D of decision trees is defined to be the smallest set of strings satisfying

- $\mathsf{true} \in D$, and
- $\mathsf{false} \in D$, and
- if $p \in A$ and $T, U \in D$ then $p(T,U) \in D$.

Introducing the convention that in a decision tree the left branch of a node p corresponds to p taking the value true and the right branch corresponds to false, a boolean value $\phi(T,s)$ can be assigned to every decision tree T and every instance s, inductively defined as follows

$$\begin{aligned}
\phi(\mathsf{true}, s) &= \mathsf{true}\\
\phi(\mathsf{false}, s) &= \mathsf{false}\\
\phi(p(T,U), s) &= \phi(T,s) \text{ if } s(p) = \mathsf{true}\\
\phi(p(T,U), s) &= \phi(U,s) \text{ if } s(p) = \mathsf{false}.
\end{aligned}$$

Alternatively, in propositional notation the last two lines can be written as

$$\phi(p(T,U),s) \;=\; (s(p) \wedge \phi(T,s)) \vee (\neg s(p) \wedge \phi(U,s)),$$

or equivalently as $\phi(p(T,U),s) \;=\; (s(p) \to \phi(T,s)) \wedge (\neg s(p) \to \phi(U,s))$. The function $\phi(T,-)$ is the hypothesis corresponding to T.

For any decision tree T let $\mathsf{attr}(T)$ be the set of attributes occurring in T, defined inductively by

$$\mathsf{attr}(\mathsf{true}) = \mathsf{attr}(\mathsf{false}) = \emptyset,$$

$$\mathsf{attr}(p(T,U)) = \{p\} \cup \mathsf{attr}(T) \cup \mathsf{attr}(U) \quad \text{for all } p \in A \text{ and all } T,U \in D.$$

For any decision tree T let $\mathsf{depth}(T)$ be the depth of T, defined inductively by

$$\mathsf{depth}(\mathsf{true}) = \mathsf{depth}(\mathsf{false}) = 0,$$

$$\mathsf{depth}(p(T,U)) = 1 + \max(\mathsf{depth}(T), \mathsf{depth}(U)) \quad \text{for all } p \in A \text{ and all } T,U \in D.$$

3 Decision equivalence

Two decision trees T and U are called *decision equivalent*, or shortly *equivalent*, denoted as $T \simeq U$, if

$$\phi(T,s) = \phi(U,s) \;\text{ for all } s : A \to \{\mathsf{true},\mathsf{false}\}.$$

Decision equivalence can be described by a finite equational axiomatization as follows. Let $\mathcal{E}$ consist of the equations

$$\begin{array}{lc} (1) & p(x,x) = x \\ (2) & p(q(x,y),q(z,w)) = q(p(x,z),p(y,w)) \\ (3) & p(p(x,y),z) = p(x,z) \\ (4) & p(x,p(y,z)) = p(x,z) \end{array}$$

for all $p,q \in A$. Here x,y,z,w are variables from a set X of variable symbols for which decision trees have to be substituted. More precisely, if $\sigma : X \to D$, and t is a term built from attributes from A and variables from X, then t^σ is defined inductively as follows

$$\begin{array}{rl} x^\sigma &= \sigma(x) \qquad \text{for all } x \in X, \\ p(t,u)^\sigma &= p(t^\sigma, u^\sigma) \text{ for all } p \in A. \end{array}$$

Let $\equiv_\mathcal{E}$ be the congruence generated by $\mathcal{E}$, i.e., $\equiv_\mathcal{E}$ is the smallest binary relation on D satisfying

- $t^\sigma \equiv_\mathcal{E} u^\sigma$ for every equation $t = u$ in $\mathcal{E}$ and every $\sigma : X \to D$, and
- $\equiv_\mathcal{E}$ is reflexive, symmetric and transitive, and
- if $T \equiv_\mathcal{E} U$ then $p(T,V) \equiv_\mathcal{E} p(U,V)$ and $p(V,T) \equiv_\mathcal{E} p(V,U)$ for all $p \in A$ and all $V \in D$.

We prove that $\mathcal{E}$ is a sound and complete axiomatization for decision equivalence, i.e., the relations $\equiv_\mathcal{E}$ and $\simeq$ on decision trees coincide. This means that two decision trees are equivalent if and only if this can be derived by only applying the four types of equations in $\mathcal{E}$. Our first theorem states soundness of $\mathcal{E}$.

Theorem 1. *If $T \equiv_\mathcal{E} U$ then $T \simeq U$.*

Proof. Since $\simeq$ is a congruence, it suffices to prove that $\phi(t^\sigma, s) = \phi(u^\sigma, s)$ for every equation $t = u$ in $\mathcal{E}$, every $\sigma : X \to D$, and every instance s. We prove this for all four types of equations.
(1). $\phi(p(x,x)^\sigma, s) = \phi(p(x^\sigma, x^\sigma), s)$
$= (p(s) \wedge \phi(x^\sigma, s)) \vee (\neg p(s) \wedge \phi(x^\sigma, s))$
$= \phi(x^\sigma, s)$.
(2). If $p(s)$ and $q(s)$ then

$$\phi(p(q(x,y),q(z,w))^\sigma, s) = \phi(x^\sigma, s) = \phi(q(p(x,z),p(y,w))^\sigma, s),$$

if $p(s)$ and $\neg q(s)$ then

$$\phi(p(q(x,y),q(z,w))^\sigma, s) = \phi(y^\sigma, s) = \phi(q(p(x,z),p(y,w))^\sigma, s),$$

if $\neg p(s)$ and $q(s)$ then

$$\phi(p(q(x,y),q(z,w))^\sigma, s) = \phi(z^\sigma, s) = \phi(q(p(x,z),p(y,w))^\sigma, s),$$

and if $\neg p(s)$ and $\neg q(s)$ then

$$\phi(p(q(x,y),q(z,w))^\sigma, s) = \phi(w^\sigma, s) = \phi(q(p(x,z),p(y,w))^\sigma, s),$$

hence for all four cases the required equality holds.
(3). If $p(s)$ then $\phi(p(p(x,y),z)^\sigma, s) = \phi(x^\sigma, s) = \phi(p(x,z)^\sigma, s)$, otherwise $\phi(p(p(x,y),z)^\sigma, s) = \phi(z^\sigma, s) = \phi(p(x,z)^\sigma, s)$, hence in both cases the required equality holds.
(4). If $p(s)$ then $\phi(p(x,p(y,z))^\sigma, s) = \phi(x^\sigma, s) = \phi(p(x,z)^\sigma, s)$, otherwise $\phi(p(x,p(y,z))^\sigma, s) = \phi(z^\sigma, s) = \phi(p(x,z)^\sigma, s)$, hence in both cases the required equality holds. □

In order to prove completeness we need a few extra definitions and lemmas. For any decision tree T let $\mathsf{attr}(T)$ be the set of attributes occurring in T, defined inductively by

$$\mathsf{attr}(\mathsf{true}) = \mathsf{attr}(\mathsf{false}) = \emptyset,$$

$$\mathsf{attr}(p(T,U)) = \{p\} \cup \mathsf{attr}(T) \cup \mathsf{attr}(U) \quad \text{for all } p \in A \text{ and all } T, U \in D.$$

For any decision tree T let $\mathsf{depth}(T)$ be the depth of T, defined inductively by

$$\mathsf{depth}(\mathsf{true}) = \mathsf{depth}(\mathsf{false}) = 0,$$

$$\mathsf{depth}(p(T,U)) = 1 + \max(\mathsf{depth}(T), \mathsf{depth}(U)) \quad \text{for all } p \in A \text{ and all } T, U \in D.$$

The first lemma states that one attribute may be forced to occur only at the root.

Lemma 2. *Let T be a decision tree and let $p \in A$. Then there exist decision trees T_1 and T_2 satisfying $T \simeq p(T_1, T_2)$ and $\mathsf{attr}(T_i) \subseteq \mathsf{attr}(T) \setminus \{p\}$ for both $i = 1, 2$.*

Proof. A straightforward proof can be given by induction on $\mathsf{depth}(T)$. Alternatively, this lemma follows from Lemma 8 for which we give an independent proof. □

The next lemma states that attributes not occurring in a decision tree do not affect the corresponding hypothesis.

Lemma 3. *Let T be a decision tree and let $p \in A$ for which $p \notin \mathsf{attr}(T)$. Let s, s' be instances such that $s(q) = s'(q)$ for all $q \in A \setminus \{p\}$. Then $\phi(T, s) = \phi(T, s')$.*

Proof. Straightforward by induction on $\mathsf{depth}(T)$. □

The next lemma is not only a key lemma for the completeness result, also for the results of the next sections.

Lemma 4. *Let $p \in A$ and $T_1, T_2, T_3, T_4 \in D$ satisfying $p \notin \mathsf{attr}(T_i)$ for $i = 1, 2, 3, 4$. Then $p(T_1, T_2) \simeq p(T_3, T_4)$ if and only if $T_1 \simeq T_3$ and $T_2 \simeq T_4$.*

Proof. The 'if'-part is immediate from the definition of ϕ. For the 'only if'-part assume $p(T_1, T_2) \simeq p(T_3, T_4)$ and let s be an arbitrary instance. Define instances s', s'' by $s'(p) = \mathsf{true}$, $s''(p) = \mathsf{false}$ and $s'(q) = s''(q) = s(q)$ for all $q \in A \setminus \{p\}$. Since $p(T_1, T_2) \simeq p(T_3, T_4)$ we have $\phi(p(T_1, T_2), s') = \phi(p(T_3, T_4), s')$ and $\phi(p(T_1, T_2), s'') = \phi(p(T_3, T_4), s'')$ by definition. Applying Lemma 3 we obtain

$$\phi(T_1, s) = \phi(T_1, s') = \phi(p(T_1, T_2), s') = \phi(p(T_3, T_4), s') = \phi(T_3, s') = \phi(T_3, s)$$

and

$$\phi(T_2, s) = \phi(T_2, s'') = \phi(p(T_1, T_2), s'') = \phi(p(T_3, T_4), s'') = \phi(T_4, s'') = \phi(T_4, s).$$

This holds for arbitrary s, hence $T_1 \simeq T_3$ and $T_2 \simeq T_4$. □

Now we are ready for proving completeness.

Theorem 5. *If $T \simeq U$ then $T \equiv_{\mathcal{E}} U$.*

Proof. We proceed by induction on $\#\mathsf{attr}(T) + \#\mathsf{attr}(U)$. If $\#\mathsf{attr}(T) + \#\mathsf{attr}(U) = 0$ then both T and U are either true or false. Since $\mathsf{true} \not\simeq \mathsf{false}$ we conclude from $T \simeq U$ that $T = U$, hence $T \equiv_{\mathcal{E}} U$.

Next assume $\#\mathsf{attr}(T) + \#\mathsf{attr}(U) > 0$ and $T \simeq U$. Then at least one of the values $\#\mathsf{attr}(T)$ and $\#\mathsf{attr}(U)$ is positive, by symmetry we may assume it is $\#\mathsf{attr}(T)$. Then $T = p(T_1, T_2)$ for some $p \in A$ and $T_1, T_2 \in D$. Due to Lemma 2 there are decision trees T_1' and T_2' satisfying $T \simeq p(T_1', T_2')$ and $\mathsf{attr}(T_i') \subseteq \mathsf{attr}(T) \setminus \{p\}$ for both $i = 1, 2$, hence $\#\mathsf{attr}(T_i') < \#\mathsf{attr}(T)$ for both $i = 1, 2$. Again according to Lemma 2 there are decision trees U_1 and U_2 satisfying $U \simeq p(U_1, U_2)$ and $\mathsf{attr}(U_i) \subseteq \mathsf{attr}(U) \setminus \{p\}$ for both $i = 1, 2$, hence $\#\mathsf{attr}(U_i) \leq \#\mathsf{attr}(U)$ for both $i = 1, 2$.

From $T \simeq U$ we conclude $p(T_1', T_2') \simeq p(U_1, U_2)$. Since p is not contained in any of the sets $\mathsf{attr}(T_i'), \mathsf{attr}(U_i)$ for $i = 1, 2$, we may apply Lemma 4, yielding $T_i' \simeq U_i$ for $i = 1, 2$. Since $\#\mathsf{attr}(T_i') + \#\mathsf{attr}(U_i) < \#\mathsf{attr}(T) + \#\mathsf{attr}(U)$ we may apply the induction hypothesis for both $i = 1$ and $i = 2$, yielding $T_i' \equiv_{\mathcal{E}} U_i$ for $i = 1, 2$. Hence

$$T = p(T_1, T_2) \equiv_{\mathcal{E}} p(T_1', T_2') \equiv_{\mathcal{E}} p(U_1, U_2) \equiv_{\mathcal{E}} U.$$

□

Summarizing Theorems 1 and 5 we have proved that two decision trees are equivalent if and only if this can be derived by only applying the four types of equations in $\mathcal{E}$. A some sloppier proof of the same observation has been given in [2].

4 Deciding equivalence

Given two decision trees T and U, how can we decide whether they are equivalent or not? A trivial algorithm for this is computing and comparing $\phi(T, s)$ and $\phi(U, s)$ for all $2^{\#A}$ values for s. It is not efficient: in worst case it is exponential in the size of the decision trees. Here the *size* $\mathsf{size}(T)$ of a decision tree T is defined to be the number of (internal) nodes of T. A more efficient algorithm follows from observations in [2], as we will discuss later on.

Here we present a very simple quadratic algorithm for deciding equivalence of decision trees. More precisely, for two decision trees having n and m nodes respectively, our algorithm requires $O(m * n)$ steps worst case to decide whether the two decision trees are equivalent or not. In the algorithm we need two auxiliary functions $\mathsf{strip} : A \times \{\mathsf{true}, \mathsf{false}\} \times D \to D$ and $\mathsf{clean} : D \to D$ inductively defined by

$$\begin{aligned}
\mathsf{strip}(p, b, \mathsf{true}) &= \mathsf{true} \\
\mathsf{strip}(p, b, \mathsf{false}) &= \mathsf{false} \\
\mathsf{strip}(p, b, q(T, U)) &= q(\mathsf{strip}(p, b, T), \mathsf{strip}(p, b, U)) \text{ if } q \neq p, \\
\mathsf{strip}(p, \mathsf{true}, p(T, U)) &= \mathsf{strip}(p, \mathsf{true}, T), \\
\mathsf{strip}(p, \mathsf{false}, p(T, U)) &= \mathsf{strip}(p, \mathsf{false}, U),
\end{aligned}$$

$$\begin{aligned}
\mathsf{clean}(\mathsf{true}) &= \mathsf{true},\\
\mathsf{clean}(\mathsf{false}) &= \mathsf{false},\\
\mathsf{clean}(p(T,U)) &= p(\mathsf{clean}(\mathsf{strip}(p,\mathsf{true},T)),\mathsf{clean}(\mathsf{strip}(p,\mathsf{false},U)))
\end{aligned}$$

for all $p,q \in A, b \in \{\mathsf{true},\mathsf{false}\}, T,U \in D$. Hence $\mathsf{strip}(p,\mathsf{true},T)$ removes all p's and its right branches from T. Similarly $\mathsf{strip}(p,\mathsf{false},T)$ removes all p's and its left branches from T. By clean a decision tree is made *clean*, where a decision tree is called *clean* if no attribute occurs more than once on any path from the root to a leaf. More precisely, for every attribute p and every subtree of the shape $p(U,U')$ the attribute p is neither contained in U nor in U'. Roughly speaking, clean removes a particular kind of redundancy from a decision tree.

Since for all p and all b the complexity of computing $\mathsf{strip}(p,b,T)$ is $O(\mathsf{size}(T))$, and $\#\mathsf{attr}(\mathsf{strip}(p,b,T)) < \#\mathsf{attr}(T)$, we conclude that the complexity of computing $\mathsf{clean}(T)$ according to its inductive definition is $O(\mathsf{size}(T) * \#\mathsf{attr}(T))$. By using a more sophisticated data structure it is even possible to compute $\mathsf{clean}(T)$ in $O(\mathsf{size}(T) * \log(\#\mathsf{attr}(T)))$ steps, but for the quadratic bound on our algorithm we will not use this.

```
S := {(clean(T), clean(U))};
while S ≠ ∅ do
      begin
      choose (T, U) ∈ S;
      S := S \ {(T, U)};
      if T ∈ {true, false} then
                        begin
                        if b ≠ T for some leaf b of U then return('NO')
                        end
      else
                        begin
                        let T = p(T1, T2);
                        S := S ∪ {(T1, strip(p, true, U)), (T2, strip(p, false, U))};
                        end
      end;
return('YES')
```

In this algorithm T and U are initially the two decision trees for which equivalence has to be established. The type of S is a set of pairs of decision trees.

Theorem 6. *Applying the algorithm on two decision trees T and U returns 'YES' if and only if $T \simeq U$ and 'NO' if and only if $T \not\simeq U$. The complexity of the algorithm is $O(\mathsf{size}(T) * \mathsf{size}(U) + \mathsf{size}(T) * \#\mathsf{attr}(T) + \mathsf{size}(U) * \#\mathsf{attr}(U))$.*

Crucial in the proof of this theorem is the observation that the assertion $\forall (T,U) \in S : T \simeq U$ remains invariant through all changes of S in the algorithm. The full proof is given below.

If both $\mathsf{size}(T)$ and $\mathsf{size}(U)$ are bounded by n, the time complexity of the algorithm is $O(n^2)$. If choose (the first statement in the body of the loop) is implemented according to the principle: *last in, first out*, then it can be seen that during the execution of the algorithm always $\#S \leq a$ holds, for a being the number of occurring attributes, yielding a space complexity of $O(n * a)$.

In [2] the following result is proved. Two decision trees T and U are equivalent if and only if $\mathsf{clean}(T)$ and $\mathsf{clean}(T')$ are equal up to idempotence (rule (1) in $\mathcal{E}$), where T' is obtained from T by replacing all leaves by a copy of U. Since checking equality up to idempotence is easily checked by only applying rule (1) from left to right, this provides an alternative algorithm for establishing equivalence.

The rest of this section consists of the proof of Theorem 6, starting by some lemmas.

Lemma 7. *Let p be an attribute, $b \in \{\mathsf{true},\mathsf{false}\}$, T a decision tree and s an instance. Then*

$$\phi(\mathsf{strip}(p,b,T),s) = \phi(T,s')$$

for the instance s' defined by $s'(p) = b$ and $s'(q) = s(q)$ for all $q \neq p$.

Proof. We apply induction on the depth of T. If $\mathsf{depth}(T) = 0$ then $T = \mathsf{true}$ or $T = \mathsf{false}$ and $\mathsf{strip}(p, b, T) = T$.

Let $\mathsf{depth}(T) > 0$. Then $T = q(T_1, T_2)$ for some attribute q. The induction hypothesis may be applied on T_1 and T_2. If $q \neq p$ and $s(q) = \mathsf{true}$ then

$$\phi(\mathsf{strip}(p, b, T), s) = \phi(q(\mathsf{strip}(p, b, T_1), \mathsf{strip}(p, b, T_2)), s) =$$

$$\phi(\mathsf{strip}(p, b, T_1), s) = \phi(T_1, s') = \phi(q(T_1, T_2), s') = \phi(T, s'),$$

and similar for $q \neq p$ and $s(q) = \mathsf{false}$. If $q = p$ and $b = s'(p) = \mathsf{true}$ then

$$\phi(\mathsf{strip}(p, b, T), s) = \phi(\mathsf{strip}(p, b, T_1), s) = \phi(T_1, s') =$$

$$\phi(p(T_1, T_2), s') = \phi(T, s'),$$

and similar for $q = p$ and $b = s'(p) = \mathsf{false}$. □

Lemma 8. *Let T be any decision tree and $p \in A$. Then*

$$T \simeq p(\mathsf{strip}(p, \mathsf{true}, T), \mathsf{strip}(p, \mathsf{false}, T)).$$

Proof. Let s be an arbitrary instance. If $s(p) = \mathsf{true}$ then

$$\phi(p(\mathsf{strip}(p, \mathsf{true}, T), \mathsf{strip}(p, \mathsf{false}, T)), s) = \phi(\mathsf{strip}(p, \mathsf{true}, T), s) = \phi(T, s)$$

by Lemma 7. If $s(p) = \mathsf{false}$ then

$$\phi(p(\mathsf{strip}(p, \mathsf{true}, T), \mathsf{strip}(p, \mathsf{false}, T)), s) = \phi(\mathsf{strip}(p, \mathsf{false}, T), s) = \phi(T, s)$$

again by Lemma 7. In both cases we are done. □

Lemma 9. *Let T be any decision tree. Then $T \simeq \mathsf{clean}(T)$ and $\mathsf{clean}(T)$ is clean.*

Proof. We apply induction on the depth of T. For $T \in \{\mathsf{true}, \mathsf{false}\}$ both assertions hold. Next let $T = p(T_1, T_2)$, as the induction hypothesis we assume that both assertions hold for $\mathsf{strip}(p, \mathsf{true}, T_1)$ and $\mathsf{strip}(p, \mathsf{false}, T_2)$. Applying this induction hypothesis, the definitions of strip and clean and Lemma 8 we obtain

$$T \simeq p(\mathsf{strip}(p, \mathsf{true}, T), \mathsf{strip}(p, \mathsf{false}, T)) = p(\mathsf{strip}(p, \mathsf{true}, T_1), \mathsf{strip}(p, \mathsf{false}, T_2))$$

$$\simeq p(\mathsf{clean}(\mathsf{strip}(p, \mathsf{true}, T_1)), \mathsf{clean}(\mathsf{strip}(p, \mathsf{false}, T_2))) = \mathsf{clean}(p(T_1, T_2)) = \mathsf{clean}(T).$$

It remains to show that

$$\mathsf{clean}(T) = p(\mathsf{clean}(\mathsf{strip}(p, \mathsf{true}, T_1)), \mathsf{clean}(\mathsf{strip}(p, \mathsf{false}, T_2)))$$

is clean. By the induction hypothesis both $\mathsf{clean}(\mathsf{strip}(p, \mathsf{true}, T_1))$ and $\mathsf{clean}(\mathsf{strip}(p, \mathsf{false}, T_2))$ are clean; it suffices to show that

$$p \notin \mathsf{attr}(\mathsf{clean}(\mathsf{strip}(p, \mathsf{true}, T_1))) \quad \text{and} \quad p \notin \mathsf{attr}(\mathsf{clean}(\mathsf{strip}(p, \mathsf{false}, T_2))).$$

This is immediate since $\mathsf{strip}(p, -, -)$ removes all occurrences of p and clean does not introduce new occurrences of p. □

Lemma 10. *Let $T = p(T_1, T_2)$ be a clean decision tree. Then both T_1 and T_2 are clean.*

Proof. By definition. □

Lemma 11. *Let T be a clean decision tree and let $b \in \{\mathsf{true}, \mathsf{false}\}$. Then $T \simeq b$ if and only if all leaves of T are equal to b.*

Proof. The 'if'-part is trivial. For the 'only if'-part we prove by induction on $\mathsf{depth}(T)$ that

If T is clean and contains a leaf unequal to b then $T \not\simeq b$.

If $T \in \{\mathsf{true}, \mathsf{false}\}$ then this trivially holds. For the induction step let $T = p(T_1, T_2)$. Since T contains a leaf unequal to b there is $i \in \{1, 2\}$ such that T_i contains a leaf unequal to b. By Lemma 10 T_i is clean, and by the induction hypothesis applied on T_i there is some instance s satisfying $\phi(T_i, s) \neq b$. Define s' by $s'(q) = s(q)$ for $q \neq p$, and $s'(p) = \mathsf{true}$ if $i = 1$ and $s'(p) = \mathsf{false}$ if $i = 2$. Since T is clean we obtain $p \notin \mathsf{attr}(T_i)$. Applying Lemma 3 we obtain

$$\phi(T, s') = \phi(p(T_1, T_2), s') = \phi(T_i, s') = \phi(T_i, s) \neq b,$$

proving $T \not\simeq b$. □

Lemma 12. *Let T be a clean decision tree and let $b \in \{\mathsf{true}, \mathsf{false}\}$ and $p \in A$. Then $\mathsf{strip}(p, b, T)$ is clean.*

Proof. An arbitrary path from the root to a leaf in $\mathsf{strip}(p, b, T)$ is obtained from a path from the root to a leaf in T by stripping away occurrences of p. If the original path does not contain double occurrences of any attribute q, then the same holds for the stripped path. □

Now we are prepared for proving Theorem 6.

Proof. Write T_0, U_0 for the original decision trees on which the algorithm is applied. First we prove that

$$T_0 \simeq U_0 \iff \forall (T, U) \in S : T \simeq U. \qquad (*)$$

is an invariant of the while loop. Initially it holds by Lemma 9 and the initialization $S := \{(\mathsf{clean}(T), \mathsf{clean}(U))\}$. It remains to show that if the invariant holds and the body of the while loop is executed, then after successful termination the invariant holds again. Here successful termination means that no $\mathsf{return}(\text{'NO'})$ is executed. This occurs in two cases. In the first case $(T, U) \in S$ has been chosen satisfying $T \in \{\mathsf{true}, \mathsf{false}\}$ and $b = T$ for all leaves b of U. Since for every $(T', U') \in S$ the decision tree U' is originally clean by lemma 9, and remains clean at every step of the while loop by lemma 12, we conclude that U is clean. Hence $T \simeq U$ by lemma 11, and (T, U) may be removed from S without affecting the invariant $(*)$.

In the second case we have $T = p(T_1, T_2)$. Since for every $(T', U') \in S$ the decision tree T' is originally clean by Lemma 9, and remains clean at every step of the while loop by Lemma 10, we conclude that T is clean. Hence $p \notin \mathsf{attr}(T_i)$ for $i = 1, 2$. Moreover $p \notin \mathsf{attr}(\mathsf{strip}(p, b, U))$ for $b \in \{\mathsf{true}, \mathsf{false}\}$, and $U \simeq p(\mathsf{strip}(p, \mathsf{true}, U), \mathsf{strip}(p, \mathsf{false}, U))$ by lemma 8. Hence by Lemma 4 we conclude

$$T \simeq U \iff T_1 \simeq \mathsf{strip}(p, \mathsf{true}, U) \wedge T_2 \simeq \mathsf{strip}(p, \mathsf{false}, U),$$

by which $(*)$ remains unaffected. Hence $(*)$ is indeed an invariant of the while loop.

Termination of the while loop follows from the observation that the value of $\sum_{(T,U) \in S} \mathsf{size}(T)$ strictly decreases at every step of the while loop, hence the total number of steps is bounded by $\mathsf{size}(T_0)$. Hence the algorithm always terminates in either returning 'YES' or returning 'NO'. In case 'NO' is returned, then by lemma 11 this is caused by an element $(T, U) \in S$ satisfying $T \not\simeq U$. Hence by the invariant $(*)$ we conclude that $T_0 \not\simeq U_0$. In the remaining case the algorithm returns 'YES', which is only done in case $S = \emptyset$. In that case we conclude from the invariant $(*)$ that $T_0 \simeq U_0$, proving correctness of the algorithm.

It remains to show the bound on the complexity of the algorithm. As remarked the computation of $\mathsf{clean}(T_0)$ and $\mathsf{clean}(U_0)$ according to the inductive definition of clean requires $O(\mathsf{size}(T_0) * \#\mathsf{attr}(T_0))$ and $O(\mathsf{size}(U_0) * \#\mathsf{attr}(U_0))$ steps respectively. Next we prove that the while loop requires $O(\mathsf{size}(T_0) * \mathsf{size}(U_0))$ steps. As remarked the total number of steps is bounded by $\mathsf{size}(T_0)$, hence it remains to prove that the complexity of the body of the while loop is $O(\mathsf{size}(U_0))$. This is immediate since the only computation in the body is checking all leaves of U or computing $\mathsf{strip}(p, \mathsf{true}, U)$ and $\mathsf{strip}(p, \mathsf{false}, U)$ for U satisfying $\mathsf{size}(U) \leq \mathsf{size}(U_0)$.

This concludes the proof of Theorem 6. □

5 Decision trees versus propositions

Every boolean function represented by a proposition can be represented by a decision tree too. Since establishing logical equivalence of propositions is known to be NP-complete, Theorem 6 is quite surprising. Combining these observations we expect that efficient representation of propositions as decision trees will not be possible. In this section we indeed prove this: we derive sharp bounds on the efficiency of the representation of propositions as decision trees.

Write B for $\{\mathsf{true}, \mathsf{false}\}$ and $(A \to \mathsf{B})$ for the set of maps from A to B. Defining $\mathsf{true}, \mathsf{false}, p : (A \to \mathsf{B}) \to \mathsf{B}$ by $\mathsf{true}(s) = \mathsf{true}$, $\mathsf{false}(s) = \mathsf{false}$ and $p(s) = s(p)$ for all $s \in (A \to \mathsf{B})$ and all $p \in A$, and defining $(P \vee Q)(s) = P(s) \vee Q(s)$, $(P \wedge Q)(s) = P(s) \wedge Q(s)$ and $(\neg P)(s) = \neg(P(s))$ for all $s \in (A \to \mathsf{B})$ and all $P, Q : (A \to \mathsf{B}) \to \mathsf{B}$, we can identify propositions with functions from $(A \to \mathsf{B})$ to B built from $\mathsf{true}, \mathsf{false}, p, \neg, \vee$ and $\wedge$. We define the *size* of a proposition to be the number of applications of $\neg, \vee$ and $\wedge$ used to construct it.

We extend the equivalence on decision trees to an equivalence on the union of the sets of decision trees and propositions: a proposition P and a decision tree T are called equivalent if $\phi(T, s) = P(s)$ for all $s \in (A \to \mathsf{B})$, and two propositions P and Q are called equivalent if $P(s) = Q(s)$ for all $s \in (A \to \mathsf{B})$, corresponding to the usual notion of logical equivalence. One easily checks that this extended equivalence $\simeq$ is an equivalence relation on the union of decision trees and propositions.

From the definition of ϕ it is clear that any decision tree T of size n is equivalent to a proposition of size $4n$. Conversely, for every proposition P there exists an equivalent decision tree T, for instance by choosing a decision tree of size $2^{\#A} - 1$ coding the truth table of P. It is a natural question to ask whether this can be done more efficiently: given a proposition of size n, what is the minimal size of an equivalent decision tree T? By the observations above we expect that in worst case this will be exponential in n. Indeed we will show in this section that if p_i, q_i are pairwise distinct attributes for $i = 1, \ldots, n$, for the proposition $\bigvee_{i=1}^{n}(p_i \wedge q_i)$ of size $2n - 1$ the minimal size of an equivalent decision tree is exactly $2^{n+1} - 2$. Moreover, we will show that the minimal size of a decision tree equivalent to $(\bigwedge_{i=1}^{n} p_i) \vee (\bigwedge_{i=1}^{k} q_i)$ is $n * k + \min(n, k)$. Since $\bigwedge_{i=1}^{n} p_i$ and $\bigwedge_{i=1}^{k} q_i$ are equivalent to decision trees of sizes n and k, respectively, this shows that in this case describing the disjunction of two decision trees as a decision tree requires a size exceeding the product of the sizes of the original decision trees.

First we derive some upper bounds on sizes of decision trees.

Proposition 13. *Let T, U be decision trees of sizes n, k, respectively, $n, k \geq 1$. Then*

- *$\neg\phi(T, -)$ is equivalent to a decision tree of size n;*
- *$\phi(T, -) \vee \phi(U, -)$ is equivalent to a decision tree of size at most $n * k + \min(n, k)$;*
- *$\phi(T, -) \wedge \phi(U, -)$ is equivalent to a decision tree of size at most $n * k + \min(n, k)$;*

Proof. A decision tree equivalent to $\neg\phi(T, -)$ of size n is simply obtained from T by replacing all occurrences of true in T by false and replacing all occurrences of false in T by true.

For $\phi(T, -) \vee \phi(U, -)$ assume by symmetry that $n \leq k$. If all $n + 1$ leaves of T are equal to false then $\phi(T, -) \vee \phi(U, -)$ is equivalent to U of size $k < n * k + \min(n, k)$. In the remaining case T has at most n leaves equal to false, and the required tree can be obtained from T by replacing each of these leaves by U.

For $\phi(T, -) \wedge \phi(U, -)$ a similar argument can be given by replacing each occurrence of true in T by U. □

Proposition 13 states upper bounds for minimal sizes of decision trees equivalent to negations, disjunctions and conjunctions of decision trees. Clearly the bound for negation is sharp; the next theorem states that the quadratic bounds for disjunctions and conjunctions are sharp too. Sharpness of all of these bounds is not only with respect to the order but even with respect to the exact value.

In order to be able to prove the theorem we first need a lemma.

Lemma 14. *Let a decision tree $p(T_1, T_2)$ be equivalent to a proposition P. Assume that $p \notin \mathsf{attr}(T_i)$ for $i = 1, 2$ and let P_1 and P_2 be two propositions in which p does not occur, and satisfying $p \wedge P \simeq p \wedge P_1$ and $(\neg p) \wedge P \simeq (\neg p) \wedge P_2$. Then $T_i \simeq P_i$ for $i = 1, 2$.*

Proof. Since p does not occur in P_1 and P_2 there are decision trees U_1, U_2 satisfying $U_i \simeq P_i$ and $p \notin \mathsf{attr}(U_i)$ for $i = 1, 2$, for instance by coding truth tables using only the attributes occurring in P_1 and P_2. We obtain

$$P(T_1, T_2) \simeq P \simeq (p \wedge P) \vee (\neg p \wedge P) \simeq (p \wedge P_1) \vee (\neg p \wedge P_2) \simeq p(U_1, U_2),$$

where the last step follows from $\phi(p(U_1, U_2), s) = (p \wedge \phi(U_1, s)) \vee (\neg p \wedge \phi(U_2, s))$ for every s. From Lemma 4 we conclude $T_i \simeq U_i \simeq P_i$ for $i = 1, 2$. □

Now we are ready to state and prove the theorem on the sharpness of the bounds.

Theorem 15. *Let p_i, q_j be pairwise distinct attributes for $i = 1, \ldots, n$ and $j = 1, \ldots, k$, where $n, k \geq 1$. Then the minimal size of a decision tree equivalent to $(\bigwedge_{i=1}^{n} p_i) \vee (\bigwedge_{i=1}^{k} q_i)$ is $n * k + \min(n, k)$, while $\bigwedge_{i=1}^{n} p_i$ and $\bigwedge_{i=1}^{k} q_i$ are equivalent to decision trees of sizes n and k, respectively.*

Conversely the minimal size of a decision tree equivalent to $(\bigvee_{i=1}^{n} p_i) \wedge (\bigvee_{i=1}^{k} q_i)$ is $n*k+\min(n, k)$ too, while $\bigvee_{i=1}^{n} p_i$ and $\bigvee_{i=1}^{k} q_i$ are equivalent to decision trees of sizes n and k, respectively,

Proof. We only give the proof of the first part; for the second part a similar proof can be given.

Let $T = p_1(p_2(\cdots(p_n(\mathsf{true}, \mathsf{false}), \mathsf{false}), \ldots), \mathsf{false})$. Then clearly $\bigwedge_{i=1}^{n} p_i$ is equivalent to the decision tree T of size n, and similar for $\bigwedge_{i=1}^{k} q_i$. Let $f(n, k)$ be the minimal size of a decision tree that is equivalent to $(\bigwedge_{i=1}^{n} p_i) \vee (\bigwedge_{i=1}^{k} q_i)$. From Proposition 13 we already know that $f(n, k) \leq n * k + \min(n, k)$; now it remains to prove $f(n, k) \geq n * k + \min(n, k)$. For $n = 1$ all $k + 1$ relevant attributes $p_1, q_1, \ldots, q_k$ have to occur as a node in the corresponding decision tree, yielding $f(1, k) \geq k + 1 = 1 * k + \min(1, k)$. Hence for $n = 1$ the required equality holds, and by symmetry also for $k = 1$. For $n, k > 1$ we proceed by induction on $n + k$. Let V be a decision tree of minimal size that is equivalent to $(\bigwedge_{i=1}^{n} p_i) \vee (\bigwedge_{i=1}^{k} q_i)$. We have to prove that $\mathsf{size}(V) \geq n * k + \min(n, k)$. We have either $V = p_a(V_1, V_2)$ for some $a = 1, \ldots, n$ or $V = q_b(V_1, V_2)$ for some $b = 1, \ldots, k$. First assume $V = p_a(V_1, V_2)$. From the minimal size of V we conclude $p_a \notin \mathsf{attr}(V_i)$ for $i = 1, 2$. Since

$$p_a \wedge ((\bigwedge_{i=1}^{n} p_i) \vee (\bigwedge_{i=1}^{k} q_i)) \simeq p_a \wedge ((\bigwedge_{i \in \{1,\ldots,n\} \setminus \{a\}} p_i) \vee (\bigwedge_{i=1}^{k} q_i))$$

and

$$(\neg p_a) \wedge ((\bigwedge_{i=1}^{n} p_i) \vee (\bigwedge_{i=1}^{k} q_i)) \simeq (\neg p_a) \wedge (\bigwedge_{i=1}^{k} q_i)$$

we conclude from Lemma 14 that

$$V_1 \simeq (\bigwedge_{i \in \{1,\ldots,n\} \setminus \{a\}} p_i) \vee (\bigwedge_{i=1}^{k} q_i) \quad \text{and} \quad V_2 \simeq \bigwedge_{i=1}^{k} q_i.$$

From the induction hypothesis we conclude that

$$\mathsf{size}(V_1) \geq f(n-1, k) \geq (n-1) * k + \min(n-1, k);$$

since all k attributes $q_1, \ldots, q_k$ are relevant in V_2 we conclude $\mathsf{size}(V_2) \geq k$. Hence

$$\mathsf{size}(V) = \mathsf{size}(p_a(V_1, V_2)) = 1 + \mathsf{size}(V_1) + \mathsf{size}(V_2) \geq$$

$$1 + (n-1) * k + \min(n-1, k) + k = n * k + \min(n, k+1) \geq n * k + \min(n, k).$$

In the remaining case $V = q_b(V_1, V_2)$ for some $b = 1, \ldots, k$ we obtain symmetrically

$$\mathsf{size}(V) = \mathsf{size}(q_b(V_1, V_2)) = 1 + \mathsf{size}(V_1) + \mathsf{size}(V_2) \geq$$

$$1 + n + n * (k-1) + \min(n, k-1) = n * k + \min(n+1, k) \geq n * k + \min(n, k),$$

concluding the proof of Theorem 15. □

The next theorem states that the minimal size of a decision tree equivalent to some proposition can be exponential in the size of the proposition.

Theorem 16. *Let p_i, q_i be pairwise distinct attributes for $i = 1, \ldots, n$, $n \geq 1$. Then for both $P = \bigvee_{i=1}^{n}(p_i \wedge q_i)$ and $P = \bigwedge_{i=1}^{n}(p_i \vee q_i)$ the minimal size of a decision tree equivalent to P is $2^{n+1} - 2$.*

Proof. We only give the proof of the first part; for the second part a similar proof can be given. The first part immediately follows from the following claim that we prove to hold for every $n \geq 1$ by induction on n.

> **Claim:** Let p_i, q_i, r_j be pairwise distinct attributes for $i = 1, \ldots, n$, $j = 1, \ldots, k$, $k \geq 0$. Then the minimal size of a decision tree equivalent to $(\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i=1}^{n}(p_i \wedge q_i))$ is $2^{n+1} + k - 2$.

For $n = 1$ the required size $2^{n+1} + k - 2 = k + 2$ is achieved by

$$r_1(\mathsf{true}, r_2(\mathsf{true}, \ldots, r_k(\mathsf{true}, p_1(q_1(\mathsf{true}, \mathsf{false}), \mathsf{false})) \cdots)),$$

which is indeed minimal since all $k + 2$ attributes $r_1, r_2, \ldots, r_k, p_1, q_1$ are relevant.

For the induction step let T be a decision tree of minimal size equivalent to $(\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i=1}^{n}(p_i \wedge q_i))$ for $k \geq 0$ and $n \geq 2$. Then either $T = p_a(T_1, T_2)$ for some $a = 1, \ldots, n$ or $T = r_b(T_1, T_2)$ for some $b = 1, \ldots, k$; the case $T = q_a(T_1, T_2)$ may be ignored by symmetry between p_a and q_a.

First assume $T = p_a(T_1, T_2)$ for some $a = 1, \ldots, n$. By minimality we may conclude that $p_a \notin \mathsf{attr}(T_i)$ for $i = 1, 2$. Since

$$p_a \wedge (\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i=1}^{n}(p_i \wedge q_i)) \simeq p_a \wedge (q_a \vee (\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i \in \{1,\ldots,n\} \setminus \{a\}} (p_i \wedge q_i)))$$

and

$$(\neg p_a) \wedge (\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i=1}^{n}(p_i \wedge q_i)) \simeq (\neg p_a) \wedge ((\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i \in \{1,\ldots,n\} \setminus \{a\}} (p_i \wedge q_i)))$$

we conclude from Lemma 14 that

$$T_1 \simeq (q_a \vee (\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i \in \{1,\ldots,n\} \setminus \{a\}} (p_i \wedge q_i)))$$

and

$$T_2 \simeq ((\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i \in \{1,\ldots,n\} \setminus \{a\}} (p_i \wedge q_i))).$$

From the induction hypothesis and the minimality of T we conclude that

$$\mathsf{size}(T_1) = 2^n + (k + 1) - 2 \quad \text{and} \quad \mathsf{size}(T_2) = 2^n + k - 2,$$

yielding $\mathsf{size}(T) = 1 + \mathsf{size}(T_1) + \mathsf{size}(T_2) = 2^{n+1} + 2k - 2$. In case of $k = 0$ the shape $T = p_a(T_1, T_2)$ is the only possible shape for T and we are done.

In case of $k > 0$ we still have to consider the case that $T = r_b(T_1, T_2)$ for some $b = 1, \ldots, k$. Since

$$r_b \wedge ((\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i=1}^{n}(p_i \wedge q_i))) \simeq r_b \wedge \mathsf{true}$$

and

$$(\neg r_b) \wedge ((\bigvee_{j=1}^{k} r_j) \vee (\bigvee_{i=1}^{n}(p_i \wedge q_i))) \simeq (\neg r_b) \wedge ((\bigvee_{j \in \{1,\ldots,k\} \setminus \{b\}} r_j) \vee (\bigvee_{i=1}^{n}(p_i \wedge q_i)))$$

we conclude from Lemma 14 that $T_1 \simeq \mathsf{true}$ and and

$$T_2 \simeq (\bigvee_{j \in \{1,\ldots,k\}\setminus\{b\}} r_j) \vee (\bigvee_{i=1}^{n} (p_i \wedge q_i)).$$

From the induction hypothesis and the minimality of T we conclude that $\mathsf{size}(T_1) = 0$ and $\mathsf{size}(T_2) = 2^{n+1} + (k-1) - 2$, yielding $\mathsf{size}(T) = 1 + \mathsf{size}(T_1) + \mathsf{size}(T_2) = 2^(n+1) + k - 2$. Since this value is smaller than the value for $\mathsf{size}(T)$ in case of $T = p_a(T_1, T_2)$, this smaller value is the real minimal value, as we had to prove, concluding the proof of Theorem 16. □

6 Conclusions and further research

In algorithmic learning often a single hypothesis can be given in various representations, hence defining a basic equivalence relation on the representation class. In this paper we studied this equivalence relation for the class of decision trees. First we proposed a simple and efficient algorithm for establishing equivalence. This result is quite subtle, for instance for BDDs as presented in [1], that only differ from decision trees in allowing sharing of subtrees, the question of establishing equivalence is NP-complete since every proposition can be represented in linear time as a BDD.

Next we investigated how to express disjunctions and conjunctions of decision trees as decision trees again. It turned out that this could not be done efficiently.

The question of establishing equivalence can be asked for other representation classes of hypotheses, for instance for so-called *ensembles*([5, 4]). For decision trees this means that some number of decision trees is given, and the resulting hypothesis is obtained by taking the majority among all boolean values yielded by each of the decision trees. We like to stress that the operations conjunction and disjunction that turned out to be expensive to express by decision trees, are simply described by ensembles: two decision trees combined with true represent the disjunction of both decision trees, and two decision trees combined with false represent the conjunction of both decision trees.

Although establishing equivalence of arbitrary ensembles of decision trees is NP-complete, it may be expected that our algorithm can be applied to give a polynomial algorithm for establishing equivalence of ensembles of decision trees of a fixed cardinality.

Acknowledgement

I like to thank Linda C. van der Gaag for many fruitful discussions and comments on earlier versions of this paper.

References

1. Bryant, R. E. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers C-35*, 8 (1986), 677–691.
2. Cockett, J. R. B. Discrete decision theory: Manipulations. *Theoretical Computer Science 54* (1987), 215–236.
3. Cockett, J. R. B., and Herrera, J. A. Decision tree reduction. *Journal of the Association for Computing Machinery 37*, 4 (1990), 815–842.
4. Dietterich, T. G. Machine-learning research: Four current directions. *AI Magazine* (Winter 1997), 97–136.
5. Hanson, L., and Salomon, P. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12* (1990), 993–1001.
6. Hyafil, L., and Rivest, R. L. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters 5*, 1 (1976), 15–17.
7. Mitchell, T. M. *Machine Learning.* McGraw-Hill, 1997.
8. Quinlan, J. R. Induction of decision trees. *Machine Learning 1* (1986), 81–106.