Inducing decision trees via concept lattices*

Radim Belohlavek[1,3], Bernard De Baets[2], Jan Outrata[3], Vilem Vychodil[3]

[1] Dept. Systems Science and Industrial Engineering,
T. J. Watson School of Engineering and Applied Science, SUNY Binghamton
PO Box 6000, Binghamton, New York 13902–6000, USA
[2] Dept. Appl. Math., Biometrics, and Process Control, Ghent University
Coupure links 653, B-9000 Gent, Belgium
bernard.debaets@ugent.be
[3] Dept. Computer Science, Palacky University, Olomouc
Tomkova 40, CZ-779 00 Olomouc, Czech Republic
{radim.belohlavek, jan.outrata, vilem.vychodil}@upol.cz

Abstract. The paper presents a new method of decision tree induction based on formal concept analysis (FCA). The decision tree is derived using a concept lattice, i.e. a hierarchy of clusters provided by FCA. The idea behind is to look at a concept lattice as a collection of overlapping trees. The main purpose of the paper is to explore the possibility of using FCA in the problem of decision tree induction. We present our method and provide comparisons with selected methods of decision tree induction on testing datasets.

1 Introduction

Decision trees and their induction is one of the most important and thoroughly investigated methods of machine learning [4, 13, 15]. There are many existing algorithms proposed for induction of a decision tree from a collection of records described by attribute vectors. A decision tree forms a model which is then used to classify new records. In general, a decision tree is constructed in a top-down fashion, from the root node to leaves. In each node an attribute is chosen under certain criteria and this attribute is used to split the collection of records covered by the node. The nodes are split until the records have the same value of the decision attribute. The critical point of this general approach is thus the selection of the attribute upon which the records are split. The selection of the splitting attribute is the major concern of the research in the area of decision trees.

The classical methods of attribute selection, implemented in well-known algorithms ID3 and C4.5 [13, 14], are based on minimizing the entropy or information gain, i.e. the amount of information represented by the clusters of records covered by nodes created upon the selection of the attribute. In addition to that, instead of just minimizing the number of misclassified records one can minimize

* Supported by Kontakt 1–2006–33 (Bilateral Scientific Cooperation, project "Algebraic, logical and computational aspects of fuzzy relational modelling paradigms"), by grant No. 1ET101370417 of GA AV ČR, by grant No. 201/05/0079 of the Czech Science Foundation, and by institutional support, research plan MSM 6198959214.

the misclassification and test costs [9]. Completely different solutions are based on involving other methods of machine learning and data mining to the problem of selection of "splitting" attribute. For instance, in [12] the authors use adaptive techniques and computation models to aid a decision tree construction, namely adaptive finite state automata constructing a so-called adaptive decision tree. In our paper, we are going to propose an approach to decision tree induction based on formal concept analysis (FCA), which has been recently utilized in various data mining problems including machine learning via the so-called lattice-based learning techniques. For instance, in [6] authors use FCA in their IGLUE method to select only relevant symbolic (categorical) attributes and transform them to continuous numerical attributes which are better for solving a decision problem by clustering methods (k-nearest neighbor).

FCA produces two kinds of outputs from object-attribute data tables. The first one is called a concept lattice and can be seen as a hierarchically ordered collection of clusters called formal concepts. The second one consists of a non-redundant basis of particular attribute dependencies called attribute implications. A formal concept is a pair of two collections—a collection of objects, called an extent, and a collection of attributes, called an intent. This corresponds to the traditional approach to concepts provided by Port-Royal logic approach.

Formal concepts are denoted by nodes in line diagrams of concept lattices. These nodes represent objects which have common attributes. Nodes in decision trees, too, represent objects which have common attributes. However, one cannot use directly a concept lattice (without the least element) as a decision tree, just because the concept lattice is not a tree in general. See [2] and [1] for results on containment of trees in concept lattices. Moreover, FCA does not distinguish between input and decision attributes. Nevertheless, a concept lattice (without the least element) can be seen as a collection of overlapping trees. Then, a construction of a decision tree can be viewed as a selection of one of these trees. This is the approach we will be interested in in the present paper.

The reminder of the paper is organized as follows. The next section contains preliminaries from decision trees and formal concept analysis. In Section 3 we present our approach of decision tree induction based on FCA. The description of the algorithm is accompanied with an illustrative example. The results of some basic comparative experiments are summarized in Section 4. Finally, Section 5 concludes and outlines several topics of future research.

2 Preliminaries

2.1 Decision trees

A decision tree can be seen as a tree representation of a finitely-valued function over finitely-valued attributes. The function is partially described by assignment of class labels to input vectors of values of input attributes. Such an assignment is usually represented by a table with rows (records) containing values of input attributes and the corresponding class labels. The main goal is to construct a decision tree which represents a function described partially by such a table and, at the same time, provides the best classification for unseen data (i.e. generalises sufficiently).

Each inner node of a corresponding decision tree is labeled by an attribute, called a decision attribute for this node, and represents a test regarding the values of the attribute. According to the result of the test, records are split into n classes which correspond to n possible outcomes of the test. In the basic setting, the outcomes are represented by the values of the splitting attribute. Leaves of the tree cover the collection of records which all have the same function value (class label). For example, the decision trees in Fig. 1 (right) both represent the function $f : A \times B \times C \to D$ depicted in Fig. 1 (left). This way, a decision tree serves as a model approximating the function partially described by the input data.

A	B	C	$f(A, B, C)$
good	yes	false	yes
good	no	false	no
bad	no	false	no
good	no	true	yes
bad	yes	true	yes

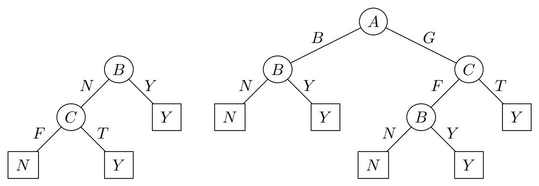

Fig. 1. Two decision trees representing example function f

A decision tree induction problem is the problem of devising a decision tree which approximates well an unknown function described partially by a relatively few records in the table. These records are usually split to two subsets called a training and testing dataset. The training dataset serves as a basis of data from which the decision tree is being induced. The testing dataset is used to evaluate the performance of the decision tree induced by the training dataset.

A vast majority of decision tree induction algorithms uses a strategy of recursive splitting of the collection of records based on selection of decision attributes. This means that the algorithms build the tree from the root to leaves, i.e. in top-down manner. The problem of local optimization is solved in every inner node. Particular algorithms differ by the method solving the optimization problem, i.e. the method of selection of the best attribute to split the records. Traditional criteria of selection of decision attributes are based on entropy, information gain [13] or statistical methods such as χ-square test [10]. The aim is to induce the smallest possible tree (in the number of nodes) which correctly decides training records. The preference of smaller trees follows directly from the Occam's Razor principle according to which the best solution from equally satisfactory ones is the simplest one.

The second problem, which is common to all machine learning methods with a teacher (methods of supervised learning), is the overfitting problem. Overfitting occurs when a model induced from training data behaves well on training data but does not behave well on testing data. A common solution to the overfitting problem used in decision trees is pruning. With pruning, some parts of the decision tree are omitted. This can either be done during the tree induction process and stop or prevent splitting nodes in some branches before reaching leaves, or after the induction of the complete tree by "post-pruning" some leaves or whole branches. The first way is accomplished by some online heuristics of classification "sufficiency" of the node. For the second way, evaluation of the ability of the tree to classify testing data is used. The simplest criterion for pruning is based on the majority of presence of one function value of records covered by the node.

2.2 Formal concept analysis

In what follows, we summarize basic notions of FCA. An object-attribute data table describing which objects have which attributes can be identified with a triplet $\langle X, Y, I\rangle$ where X is a non-empty set (of objects), Y is a non-empty set (of attributes), and $I \subseteq X \times Y$ is an (object-attribute) relation. Objects and attributes correspond to table rows and columns, respectively, and $\langle x, y\rangle \in I$ indicates that object x has attribute y (table entry corresponding to row x and column y contains $\times$; if $\langle x, y\rangle \notin I$ the table entry contains blank symbol). In the terminology of FCA, a triplet $\langle X, Y, I\rangle$ is called a formal context. For each $A \subseteq X$ and $B \subseteq Y$ denote by $A^{\uparrow}$ a subset of Y and by $B^{\downarrow}$ a subset of X defined by

$$A^{\uparrow} = \{y \in Y \mid \text{for each } x \in A:\ \langle x, y\rangle \in I\},$$
$$B^{\downarrow} = \{x \in X \mid \text{for each } y \in B:\ \langle x, y\rangle \in I\}.$$

That is, $A^{\uparrow}$ is the set of all attributes from Y shared by all objects from A (and similarly for $B^{\downarrow}$). A formal concept in $\langle X, Y, I\rangle$ is a pair $\langle A, B\rangle$ of $A \subseteq X$ and $B \subseteq Y$ satisfying $A^{\uparrow} = B$ and $B^{\downarrow} = A$. That is, a formal concept consists of a set A (so-called extent) of objects which fall under the concept and a set B (so-called intent) of attributes which fall under the concept such that A is the set of all objects sharing all attributes from B and, conversely, B is the collection of all attributes from Y shared by all objects from A. Alternatively, formal concepts can be defined as maximal rectangles of $\langle X, Y, I\rangle$ which are full of $\times$'s: For $A \subseteq X$ and $B \subseteq Y$, $\langle A, B\rangle$ is a formal concept in $\langle X, Y, I\rangle$ iff $A \times B \subseteq I$ and there is no $A' \supset A$ or $B' \supset B$ such that $A' \times B \subseteq I$ or $A \times B' \subseteq I$.

A set $\mathcal{B}(X, Y, I) = \{\langle A, B\rangle \mid A^{\uparrow} = B, B^{\downarrow} = A\}$ of all formal concepts in data $\langle X, Y, I\rangle$ can be equipped with a partial order $\leq$ (modeling the subconcept-superconcept hierarchy, e.g. *dog* $\leq$ *mammal*) defined by

$$\langle A_1, B_1\rangle \leq \langle A_2, B_2\rangle \text{ iff } A_1 \subseteq A_2 \text{ (iff } B_2 \subseteq B_1\text{)}. \tag{1}$$

Note that $^{\uparrow}$ and $^{\downarrow}$ form a so-called Galois connection [5] and that $\mathcal{B}(X, Y, I)$ is in fact a set of all fixed points of $^{\uparrow}$ and $^{\downarrow}$. Under $\leq$, $\mathcal{B}(X, Y, I)$ happens to be a complete lattice, called a concept lattice of $\langle X, Y, I\rangle$, the basic structure of which is described by the so-called main theorem of concept lattices [5].

For a detailed information on formal concept analysis we refer to [3, 5] where a reader can find theoretical foundations, methods and algorithms, and applications in various areas.

3 Decision tree induction based on FCA

As mentioned above, a concept lattice without the least element can be seen as a collection of overlapping trees. The induction of a decision tree can be viewed as a selection of one of the overlapping trees. The question is: which tree do we select?

Transformation of input data Before coming to this question in detail, we need to address a particular problem concerning input data. Input data to decision tree induction contains various type of attributes, including yes/no (logical) attributes, categorical (nominal) attributes, ordinal attributes, numerical attributes, etc. On the other hand, Input data to FCA consists of yes/no attributes. Transformation of general attributes to logical attributes is known as conceptual scaling, see [5]. For the sake of simplicity, we consider input data with categorical attributes in our paper and their transformation (scaling) to logical attributes. Decision attributes (class labels) are usually categorical. Note that we need not transform the decision attributes since we do not use them for the concept lattice building step.

Name	body temp.	gives birth	fourlegged	hibernates	mammal
cat	warm	yes	yes	no	yes
bat	warm	yes	no	yes	yes
salamander	cold	no	yes	yes	no
eagle	warm	no	no	no	no
guppy	cold	yes	no	no	no

Name	bt cold	bt warm	gb no	gb yes	fl no	fl yes	hb no	hb yes	mammal
cat	0	1	0	1	0	1	1	0	yes
bat	0	1	0	1	1	0	0	1	yes
salamander	1	0	1	0	0	1	0	1	no
eagle	0	1	1	0	1	0	1	0	no
guppy	1	0	0	1	1	0	1	0	no

Fig. 2. Input data table (top) and corresponding data table for FCA (bottom)

Let us present an example used throughout the presentation of our method. Consider the data table with categorical attributes depicted in Fig. 2 (top). The data table contains sample animals described by attributes *body temperature*, *gives birth*, *fourlegged*, *hibernates* and *mammal*, with the last attribute being the decision attribute (class label). The corresponding data table for FCA with logical attributes obtained from the original ones in an obvious way is depicted in Fig. 2 (bottom).

Step 1 We can now approach the first step of our method of decision tree induction—building the concept lattice. In fact, we do not build the whole lattice. Recall that smaller (lower) concepts result by adding attributes to greater (higher) concepts and, dually, greater concepts result by adding objects to lower concepts. We can thus imagine the lower neighbor concepts as refining their parent concept. In a decision tree the nodes cover some collection of records and are split until the covered records have the same value of class label. The same applies to concepts in our approach: we need not split concepts which cover objects having the same value of class label. Thus, we need an algorithm which generates a concept lattice from the greatest concept (which covers all objects) and iteratively generates lower neighbor concepts.

For this purpose, we can conveniently use the essential ideas of Lindig's NextNeighbor algorithm [8]. NextNeighbor efficiently generates formal concepts together with their subconcept-superconcept hierarchy. Our method, which is a modification of NextNeighbor, differs from the ordinary NextNeighbor in two aspects. First, as mentioned above, we do not compute lower neighbor concepts of a concept which covers objects with the same class label. Second, unlike NextNeighbor, we do not build the ordinary concept hierarchy by means of a covering relation. Instead, we are skipping some concepts in the hierarchy. That is, a lower neighbor concept c of a given concept d generated by our method, can in fact be a concept for which there exists an intermediate concept between c and d. This is accomplished by a simple modification of NextNeighbor algorithm.

NextNeighbor The NextNeighbor algorithm builds the concept lattice by iteratively generating the neighbor concepts of a concept $\langle A, B\rangle$, either top-down the lattice by adding new attributes to concept intents or bottom-up by adding new objects to concept extents. We follow the top-down approach. The algorithm is based on the fact that a concept $\langle C, D\rangle$ is a neighbor of a given concept $\langle A, B\rangle$ if D is generated by $B \cup \{y\}$, i.e. $D = (B \cup \{y\})^{\downarrow\uparrow}$, where $y \in Y - B$ is an attribute such that for all attributes $z \in D - B$ it holds that $B \cup \{z\}$ generates the same concept $\langle C, D\rangle$ [8], i.e.

$$\begin{aligned}&(Next)Neighbors\ of\ \langle A, B\rangle = \\ &\quad \{\langle C, D\rangle \mid D = (B \cup \{y\})^{\downarrow\uparrow}, y \in Y - B\ such\ that \\ &\qquad (B \cup \{z\})^{\downarrow\uparrow} = D\ for\ all\ z \in D - B\}.\end{aligned}$$

Our modification From the monotony of the (closure) operator forming a formal concept it follows that a concept $\langle C, D\rangle$ is not a neighbor of the concept $\langle A, B\rangle$ if there exists an attribute $z \in D - B$ such that $B \cup \{z\}$ generates a concept between $\langle A, B\rangle$ and $\langle C, D\rangle$. This is what our modification consists in. Namely, we mark as (different) neighbors all concepts generated by $B \cup \{y\}$ for $y \in Y - B$, even those for which there exists a concept in between, i.e.

$$\begin{aligned}&(Our)Neighbors\ of\ \langle A, B\rangle = \\ &\quad \{\langle C, D\rangle \mid D = (B \cup \{y\})^{\downarrow\uparrow}, y \in Y - B\}.\end{aligned}$$

It is easy to see that our modification does not alter the concept lattice and the overall hierarchy of concepts, cf. NextNeighbor [8].

The reason for this modification is that we have to record as neighbors of a concept $\langle A, B\rangle$ all the concepts which are generated by the collection of attributes

B with one additional attribute. In the resulting decision tree the addition of a (logical) attribute to a concept means making a decision on the corresponding categorical attribute in the tree node corresponding to the concept. Due to lack of space we postpone a pseudocode of the algorithm of Step 1 to the full version of the paper. Part of the concept lattice built from data table in Fig. 2 (bottom), with our new neighbor relationships drawn by dashed lines, is depicted in Fig. 3.

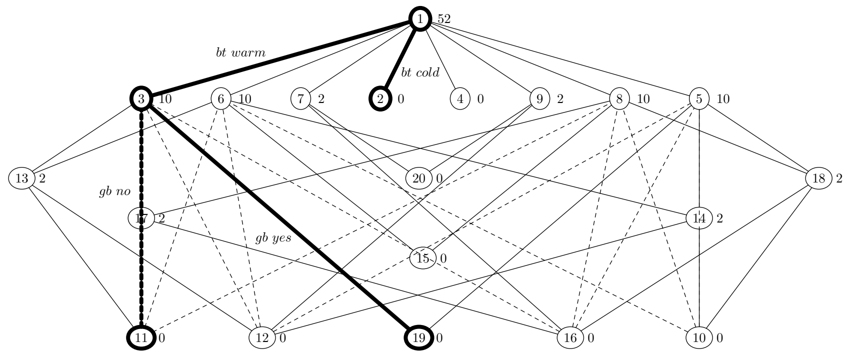

Fig. 3. Part of the concept lattice and tree of concepts (solid) of data table in Fig. 2

Step 2 The second step of our method is the selection of a tree of concepts from the part of the concept lattice built in the first step. First, we calculate for each concept $c = \langle A, B\rangle$ the number L_c of all of its lower concepts. Note that each lower concept is counted for each different attribute added to the concept c, cf. our modification of concept neighbor relation. For instance, if a concept $d = \langle C, D\rangle$ is generated from concept c by adding either attribute x or attribute y (i.e. $D = (B \cup \{x\})^{\downarrow\uparrow}$ or $D = (B \cup \{y\})^{\downarrow\uparrow}$, respectively), the concept d is counted twice and L_c is increased by two.

Next, we select a tree of concepts from the part of the concept lattice by iteratively going from the greatest concept (generated by no attributes or, equivalently, by all objects) to minimal concepts. The selection is based on the number L_c of lower concepts of the currently considered concept c (recall that L_c is not the number of lower concepts of c in common sense, cf. the computation of L_c above which is due to our modification of concept neighbor relation).

The root node of the tree is always the greatest concept. Then, for each tree node/concept c we define collections $\mathcal{N}_c^a$ of concepts, which will be candidate collections of children nodes/concepts of c in the resulting selected tree. $\mathcal{N}_c^a$ is a collection of lower neighbor concepts of c such that (a) each concept d in $\mathcal{N}_c^a$ is generated from concept c by adding a (logical) attribute transformed from the categorical attribute a (recall that logical attributes stand for values of categorical attributes) and (b) $\mathcal{N}_c^a$ contains the concept d for every logical attribute transformed from categorical attribute a. There (1) may exist, and usually exists, more than one such collection $\mathcal{N}_c^a$ of neighbor concepts of concept c, for more that one categorical attribute a, but, on the other side (2) there may exist no such collection.

(1) In this case we choose from the several collections $\mathcal{N}_c^a$ of neighbor concepts of concept c the collection containing a concept d with the minimal number L_d of its lower concepts. Furthermore, if there is more than one such neighbor concept, in different collections, we choose the collection containing the concept which covers the maximal number of objects/records. It is important to note that this point is the only non-deterministic point in our method since there still can be more than one neighbor concepts having equal minimal number of lower concepts and covering equal maximal number of objects. In that case we choose one of the collections $\mathcal{N}_c^a$ of neighbor concepts arbitrarily.

(2) This case means that in every potential collection $\mathcal{N}_c^a$ there is missing at least one neighbor concept generated by some added (logical) attribute transformed from categorical attribute a, i.e. $\mathcal{N}_c^a$ does not satisfy the condition (b). We solve this situation by substituting the missing concepts by (a copy of) the least concept $\langle Y^{\downarrow}, Y \rangle$ generated by all attributes (or, equivalently, no objects). The least concept is a common and always existing subconcept of all concepts in a concept lattice and usually covers no objects/records (but need not!).

Finally, an edge between concept c and each neighbor concept from the chosen collection $\mathcal{N}_c^a$ is created in the resulting selected tree. The edge is labeled by the added logical attribute. Again, we postpone a pseudocode of the algorithm of Step 2 to the full version of the paper.

To illustrate the previuos description, let us consider the example of a part of the concept lattice in Fig. 3. The concepts are denoted by a circled number and the number of lower concepts is written to the right of every concept. We select the tree of concepts as follows. The root node of the tree is the greatest concept 1. As children nodes of the root node are selected concepts 2 and 3 since they form a collection $\mathcal{N}_1^{body\ temp.}$ of all lower neighbor concepts generated by both added (logical) attributes *bt cold* and *bt warm*, respectively, transformed from the categorical attribute *body temp.*. Note that we could have chosen the collection $\mathcal{N}_1^{gives\ birth}$ instead of collection $\mathcal{N}_1^{body\ temp.}$, but since both concepts 2 from $\mathcal{N}_1^{body\ temp.}$ and 4 from $\mathcal{N}_1^{gives\ birth}$ have the equal minimal number L_2 and L_4 of lower concepts and both cover the equal maximal number of objects/records, we have chosen the collection $\mathcal{N}_1^{body\ temp.}$ arbitrarily, according to case (1) from the description. The edges of the selected tree are labeled by the corresponding logical attributes. Similarly, the children nodes of concept 3 will be concepts 11 and 19, and this is the end of the tree selection step since concepts 4, 11 and 19 have no lower neighbors. The resulting tree of concepts is depicted in Fig. 3 with solid lines.

Step 3 The last step (third one) of our method is the transformation of the tree of concepts into a decision tree. A decision tree has in its every node the chosen categorical attribute on which the decision is made and the edges from the node are labeled by the possible values of the attribute. The leaves are labeled by class label(s) of covered records. In the tree of concepts the logical attributes transformed from (and standing for the values of) the categorical attribute are in the labels of edges connecting concepts. Hence the transformation of the tree of concepts into a decision tree is simple: edges are relabeled to the values of the categorical attribute, inner concepts are labeled by the corresponding categorical attributes, and leaves are labelled by class label(s) of covered objects/records.

The last problem to solve is multiple different class label(s) of covered records in tree leaves. This can happen for several reasons, for example the presence of conflicting records in input data differing in class label(s) only (which can result for instance from class labelling mistakes or from selecting a subcollection of attributes from original larger data) or pruning the complete decision tree as a strategy to the overfitting problem. Common practice for dealing with multiple different target class label(s) is as simple as picking the major class label value(s) as the target classification of records covered by leave node and we adopt this solution. A special case are leave nodes represented by (a copy of) the least concept (which comes from the possibility (2) in Step 2), since the least concept usually covers no objects/records. These nodes are labelled by the class label(s) of their parent nodes.

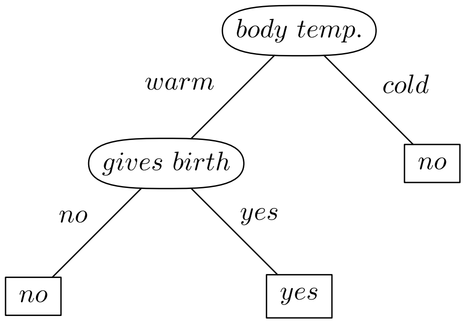

Fig. 4. The decision tree of input data in Fig. 2

The resulting decision tree of input data in Fig. 2 (top) transformed from the tree of concepts in Fig. 3 is depicted in Fig. 4.

Let us now briefly discuss the problem of overfitting. A traditional solution to overfitting problem, i.e. pruning, suggests not to include all nodes down to leaves as a part of the decision tree. One of the simplest criteria for this is picking a threshold percentage ratio of major class label value(s) in records covered by a node. Alternatively, one can use the entropy measure to decide whether the node is sufficient and need not be split. Note that several other possibilities exist. In all cases, the constraint can be applied as early as selecting the concepts to the tree, i.e. pruning can be done during decision tree induction process in our method. No pruning method is considered in this paper.

4 Comparison with other algorithms

The asymptotic time complexity of the presented algorithm is given by the (part of the) concept lattice building step since this step is the most time demanding. The concepts are computed by a modified Lindig's NextNeighbor algorithm. Since the modification does not alter the asymptotic time complexity, the overall asymptotic complexity of our method is equal to that of Lindig's NextNeighbor algorithm, namely $O(|X||Y|^2|L|)$. Here, $|X|$ is the number of input records, $|Y|$ is the number of (logical) attributes and $|L|$ is the size of the concept lattice, i.e. the number of all formal concepts.

However, for the decision tree induction problem, accuracy, i.e. the percentage of correctly and incorrectly decided records from both training and testing dataset, is more important than time complexity. We performed preliminary experiments and compared our method to reference algorithms ID3 and C4.5. We implemented our method in C language. ID3 and C4.5 were borrowed and run from the Weka[4] (Waikato Environment for Knowledge Analysis [16]), a software package which aids the development of and contains implementations of several machine learning and data mining algorithms in Java. Default Weka's parameters were used for the two algorithms and pruning was turned off where available.

Table 1. Characteristics of datasets used in experiments

Dataset	No. of attributes	No. of records	Class distribution
breast-cancer	6	138	100/38
kr-vs-kp	14	319	168/151
mushroom	10	282	187/95
vote	8	116	54/62
zoo	9	101	41/20/5/13/4/8/10

The experiments were done on selected public real-world datasets from UCI machine learning repository [7]. The selected datasets are from different areas (medicine, biology, zoology, politics and games) and all contain only categorical attributes with one class label. The datasets were cleared of records containing missing values and actually, we selected subcollections of less-valued attributes of each dataset and subcollections of records of some datasets, due to computational time of repeated executions on the same dataset. The basic characteristics of the datasets are depicted in Tab. 1. The results of averaging 10 executions of the 10-Fold Stratified Cross-validation test (which gives total of 100 executions for each algorithm over each dataset) are depicted in Tab. 2. The table shows average percentage rates of correct decisions for both training (upper item in the table cell) and testing (lower item) dataset part, for each compared algorithm and dataset. We can see that our FCA based decision tree induction method outperforms C4.5 on all datasets, with the exception of mushroom, by 2 – 4 %,

[4] Weka is a free software available at `http://www.cs.waikato.ac.nz/ml/weka/`

on both training and testing data, and gains almost identical results as ID3, again on all datasets except mushroom, on training data, but slightly outperforming ID3 on testing data by about 1 %. On mushroom dataset, which is quite sparse comparing to the other datasets, our method is little behind ID3 and C4.5 on training data, but, however, almost equal on testing data. The datasets vote and zoo are more dense than the other datasets and also contain almost no conflicting records, so it seems that the FCA based method could give better results that traditional, entropy based, methods on clear dense data. However, more experiments on additional datasets are needed to approve this conclusion.

Table 2. Percentage correct rates for datasets in Tab. 1

training % *testing %*	breast-cancer	kr-vs-kp	mushroom	vote	zoo
FCA based	88.631 79.560	84.395 74.656	96.268 96.284	97.528 90.507	98.019 96.036
ID3	88.630 75.945	84.674 74.503	97.517 96.602	97.528 89.280	98.019 95.036
C4.5	86.328 79.181	82.124 72.780	97.163 96.671	94.883 86.500	96.039 92.690

Due to lack of space only the basic experiments are presented. More comparative tests on additional datasets with additional various machine learning algorithms like Naive Bayes classification or Artificial Neural Networks trained by back propagation [11], including training and testing time measuring, are postponed to the full version of the paper. However, the first preliminary experiments show that our simple FCA based method is promising in using FCA in the decision tree induction problem.

The bottleneck of the method could be performance, the total time of tree induction, but once one already has the (whole) concept lattice of input data, then the tree selection is very fast. This draws a possible usage and perspective of the method: decision making from already available concept lattices. The advantage of our method over other methods is the conceptual information hidden in tree nodes (note that they are in fact formal concepts). The attributes in concept intents are the attributes common to all objects/records covered by the concept/tree node, which might be usefull information for furher exploration, application and interpretation of the decision tree. This type of information is not (directly) available by other methods, for instance classical entropy based.

5 Conclusion and topics of future research

We have presented a simple novel method of decision tree induction by selection of the tree of concepts from a concept lattice. The criterion of choosing an attribute based on which the node of the tree is split is determined by the number of all lower concepts of the concept corresponding to the node. The approach interconnects areas of decision trees and formal concept analysis. We have also presented some comparison to classical decision tree algorithms, namely ID3 and C4.5, and have seen that our method compares quite well and surely deserves more attention. Topics for future research include:

– Explore the possibility to compute a smaller number of formal concepts from which the nodes of a decision tree is constructed. Or, the possibility to compute right the selected concepts only.
– The problems of overfitting in data and uncomplete data, i.e. data having missing values for some attributes in some records.
– Incremental updating of induced decision trees via incremental methods of building concept lattices.

References

1. Belohlavek R., De Baets B., Outrata J., Vychodil V.: Trees in Concept Lattices. In: Torra V., Narukawa Y.,Yoshida Y. (Eds.): Modeling Decisions for Artificial Intelligence: 4th International Conference, MDAI 2007, *Lecture Notes in Artificial Intelligence* **4617**, pp. 174–184, Springer-Verlag, Berlin/Heidelberg, 2007.
2. Belohlavek R., Sklenar V.: Formal concept analysis constrained by attribute-dependency formulas. In: B. Ganter and R. Godin (Eds.): ICFCA 2005, *Lecture Notes in Computer Science* **3403**, pp. 176–191, Springer-Verlag, Berlin/Heidelberg, 2005.
3. Carpineto C., Romano G.: *Concept Data Analysis. Theory and Applications.* J. Wiley, 2004.
4. Dunham M. H.: *Data Mining. Introductory and Advanced Topics.* Prentice Hall, Upper Saddle River, NJ, 2003.
5. Ganter B., Wille R.: *Formal Concept Analysis. Mathematical Foundations.* Springer, Berlin, 1999.
6. Mephu Nguifo E., Njiwoua P.: IGLUE: A lattice-based constructive induction system. *Intell. Data Anal.* **5**(1), pp. 73–91, 2001.
7. Newman D. J., Hettich S., Blake C. L., Merz C. J.: *UCI Repository of machine learning databases*, [`http://www.ics.uci.edu/∼mlearn/MLRepository.html`], Irvine, CA: University of California, Department of Information and Computer Science, 1998.
8. Lindig C.: Fast concept analysis. In: Stumme G.: *Working with Conceptual Structures – Contributions to ICCS 2000.* Shaker Verlag, Aachen, 2000, 152–161.
9. Ling Ch. X., Yang Q., Wang J., Zhang S.: Decision Trees with Minimal Costs. *Proc. ICML 2004.*
10. Mingers J.: Expert systems – rule induction with statistical data. *J. of the Operational Research Society.*, **38**, pp. 39–47, 1987.
11. Mitchell T.M.: *Machine Learning.* McGraw-Hill, 1997.
12. Pistori H., Neto J. J.: Decision Tree Induction using Adaptive FSA. *CLEI Electron. J.* **6**(1), 2003.
13. Quinlan J. R.: *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.
14. Quinlan J. R.: Learning decision tree classifiers. *ACM Computing Surveys*, **28**(1), 1996.
15. Tan P.-N., Steinbach M., Kumar V.: *Introduction to Data Mining.* Addison Wesley, Boston, MA, 2006.
16. Witten I. H., Frank E.: *Data Mining: Practical machine learning tools and techniques, 2nd Edition.* Morgan Kaufmann, San Francisco, 2005.