Exploiting Value Hierarchies in Rule Learning*

Vojtěch Svátek

Prague University of Economics, Faculty of Informatics and Statistics, Department of Information and Knowledge Engineering, W. Churchill Sq.4, Prague, CZ-13067
e-mail: svatek@vse.cz, phone: +42-2-24095462, fax: +42-2-24225942

Abstract. We investigate the impact of attribute-value hierarchies on learning rules and performing classification with them. Hierarchies of input attribute values enable to introduce abstract terms into the premises of rules, while hierarchies of goal classes suggest the shift from one-shot classification to hierarchical refinement of the conclusion and/or step-by-step explanation. Hierarchies can be either man-made or constructed by means of hierarchical clustering; we suggest an objective function for clustering, which accounts for extrinsic dissimilarity of source attribute values. Experiments on different datasets have been conducted, using the ESOD algorithms as base learner and classifier.

One of possibilities how to the improve comprehensibility and often even accuracy of knowledge induced by learning systems is to gather static domain knowledge and to feed it to the learner together with data. A form of domain knowledge which is relatively easy to obtain as well as to incorporate into a large class of algorithms is abstraction *hierarchies* on attribute values. In section 1 we describe a way of use of hierarchical input attributes and class attributes, in learning and classification. In section 2 we treat the situation when the hierarchies are to be constructed automatically. In section 3 we show an implementation of algorithms from previous sections. In section 4 we describe experiments in three domains. Section 5 overviews related research. Finally, section 6 summarizes the main ideas and outlines prospects for future work.

1 Exploiting existing value hierarchies

We adopt the following notion of value hierarchy. Let A be an attribute and $V = V_O \cup V_A$ the domain of its values. V_O is the set of *observable* values (those present in data), while V_A is the set of *abstract* values, including the universal value *any*. As a *hierarchy* on A, we consider a directed acyclic graph on V, with one source - the value *any* - and the sinks corresponding to values from V_O; internal nodes correspond to values from V_A (see e.g. Fig. 1 in section 2).

The way how abstract values of *input* attributes are introduced into target knowledge much depends on the learning technique: usually, some form of general-to-specific search is performed. Learning with abstract values is a means

* This work has been partially sponsored by grant no.VS96008 of the Czech Ministry of Education (MŠMT).

to improve comprehensibility, via lifting the classification process at a more *abstract level*, and to make target knowledge more *concise*, as one rule with abstract values expresses a hypothesis otherwise dispersed in several rules. Furthermore, if the learner is biased towards high *coverage* to avoid overfitting, abstract (or conjoined) values are necessary for any rules to be induced from smaller datasets, as individual values have too low frequency.[2] Typical examples of value hierarchies are taxonomies of natural objects (see [Ar96] and section 4.1), geographical/administrative partitionings (see [Ar96]), and linguistic thesauri (see [Al95]).

Given a hierarchy of *classes* (i.e. on the goal attribute), a *hierarchical rulebase* can be built via providing each non-leaf node with a partial ruleset deciding among immediate successor nodes: at every non-leaf node c of the hierarchy, we

- determine the subset of objects whose class c_i is successor of c, and replace this class with c_j which is immediate successor of c and predecessor of c_i (unless c_i itself is immediate successor of c) - we obtain a dataset O_c;
- run LEARNER on O_c, which yields a ruleset R_c to be linked to node c.

This *macro-learning* algorithm is independent of the type of embedded learner. To distinguish the hierarchical rulebase from a ruleset learned without hierarchy (on the whole dataset), we will denote the latter as *flat ruleset*. The hierarchical rulebase can be used for classification, which has the form of class *refinement*; this *macro-classification* algorithm works, for an unlabelled object o, as follows.

1. Let $Node = any$.
2. Let R be the ruleset linked to node $Node$ in H_C. Let c be the class returned by CLASSIFIER for the given object, based on R. Let $Node = c$.
3. If $Node$ is a leaf-class then stop and return $Node$, else go to 2.

Again, the algorithm can be coupled with any sort of classifier (compatible with the learner used for induction). Another possibility is to separate the *problem-solving* task (i.e. classification) from the *explanation* task - the former can be performed using the flat ruleset (which decides among all leaf-classes at once) while the latter using the hierarchical rulebase. The explanation has the form of path from *any* to c_{flat}, $E = (c_0 = any, c_1, c_2, \ldots, c_{n-1}, c_n = c_{flat})$, where c_{flat} is the leaf class returned by the classifier based on the flat ruleset; within each node c_i in the path, those rules are displayed which support the conclusion c_{i+1}. If the classifier returns c_{i+1} for each c_i in E, $0 \leq i < n$ (based on the ruleset linked to c_i), then we will call E *perfect-fit explanation* - this also means that the hierarchical macro-classifier would agree with the "flat" classifier. An example of such explanation can be found at Fig. 5 in section 4.2. We assume that the use of class hierarchies can improve comprehensibility, the individual refinement/explanation steps being more transparent (thanks to lower number of rules) than one-shot classification performed by the flat ruleset. If the hierarchy is relevant to the way of human problem-solving in the domain, then the explanations are likely to *reconstruct* the real behaviour of an expert.

[2] The last two statements are not contradictory but reflect a tradeoff.

2 Constructing value hierarchies

The above methods assume that the hierarchies are given in advance. If this is not the case, we can consider building hierarchies automatically, which consists in joining together distinct values of an attribute wrt. the values of other attribute/s. As we can restructure the domain of the goal attribute based on values of input attributes as well as vice versa, we will generally denote the attribute to be restructured as *target attribute* (TA) and the attribute on whose values the restructuring is based as *source attribute* (SA), to avoid ambiguity. The restructuring should maximize internal similarity and minimize mutual similarity of joined TA values in terms of SA values; those TA values should be joined which correspond to similar distributions of SA values. Unlike most existing approaches (see section 5) we relieve the assumption of equal *extrinsic dissimilarity* among different SA values. To illustrate the utility of extrinsic dissimilarity, we have elaborated this notion in more detail for source attributes with *hierarchically* structured and/or *linearly* ordered domain. Consider the set of military grades at Fig. 1, which is linearly ordered; in addition, there are (contiguous) groups of grades which share many properties, and can be viewed as abstract values in a hierarchy. A heuristic dissimilarity measure $Diss$ should comprise both factors. In our work we used the product of *hierarchical* dissimilarity

$$HDiss(x,y) = 1 - \frac{(\|e(x)\| + \|e(y)\|) \,.\, \|e(x) \cap e(y)\|}{2 \,.\, \|e(x)\| \,.\, \|e(y)\|}$$

where $e(x)$ is the set of edges in the hierarchy which belong to some path leading from *any* to x (each edge appears only once); $\|.\|$ denotes set cardinality;[3] and *linear* dissimilarity

$$LDiss(x,y) = \frac{\mid Ord(x) - Ord(y) \mid}{Total - 1}$$

where $Ord(x)$ is the ordinal number of x in the sequence and $Total$ is the total number of values; it is the relative distance of values in the ordering.
All measures are symmetric and take values from $[0,1]$. For example, for values "corporal" and "lieutenant" $HDiss = 0.\bar{6}$, $LDiss = 0.4$, and thus $Diss = 0.2\bar{6}$. If a clustering algorithm is to build a hierarchy of TA values based on the values of the source attribute "grade" it should take into account the dissimilarity among SA values, which is extrinsic to the clustering itself.

As a measure of internal *cohesion* of a dataset in terms of SA values, we use the following formula (which is actually independent of the definition of $Diss$):

$$Coh(A) = \frac{\Sigma_{i=1}^{n} \Sigma_{j=1}^{n} \|a_i\| \,.\, \|a_j\| \,.\, (1 - Diss(a_i, a_j))}{N^2} \qquad (1)$$

where n is the number of values of source attribute A, $\|a_i\|$ is the frequency of value a_i in data, and N is the total number of instances. This measure takes

[3] An advantage of this measure is its applicability to non-tree dags.

values from $[\frac{1}{n}, 1]$; with increasing N, it converges to mean dissimilarity between a pair of instances in data. To evaluate the utility of a partition of target attribute B into m values, we can further compute aggregated cohesion given B, in a way analogical to information-theoretic or probabilistic approaches:

$$Coh(B, A) = \Sigma_{k=1}^{m} P(b_k) \frac{\Sigma_{i=1}^{n} \Sigma_{j=1}^{n} \|a_i/b_k\| \,.\, \|a_j/b_k\| \,.\, (1 - Diss(a_i, a_j))}{\|b_k\|^2} \qquad (2)$$

where $\|./.\|$ stands for conditional frequency. For multiple source attributes, the overall cohesion can be calculated as the average value of (2). This typically occurs in hierarchization of the class attribute given a set of input attributes.

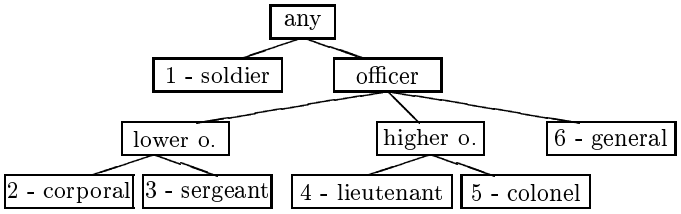

Fig. 1. Example hierarchy of military grades

Exploitation of constructed hierarchies, with each non-leaf node representing merely a union of observable values (without a semantic meaning of its own) differs from exploitation of man-made hierarchies: the former merely enable *generalization* while the latter enable *abstraction*, which has stronger impact on comprehensibility. As the hierarchization step is separated from the rule-learning step, they should be interleaved with another step - *interpretation* of constructed values (unions) by an expert, and possibly rejection of some meaningless unions.

3 Implementation

The implementation of algorithms from section 1 is based on the ESOD methods for learning weighted rules from data and for compositional classification [Be94]. ESOD's learning consists in systematic search through the space of categorial rules, in the decreasing order of coverage; only those rules are accepted which "bring new information" wrt. pseudobayesian composition of simpler rules accepted before, see Fig. 2. The motivation for using ESOD was twofold.

- the systematic search method of ESOD can be augmented to work with hierarchies of input attributes without any change to its principles;
- its compositional classification enables to quantify the plausibility of each alternative conclusion, which can be beneficial in evaluation of explanations[4].

[4] Especially in the case of multiple explanations in a non-tree hierarchy, which has not been investigated yet.

Input: Data D, goal combination C^*, combination function $\oplus$, statistical test T.

Output: KB = set of rules with weight $\in$ [0,1];

Computation:
Let KB contain only empty implication $\emptyset \Rightarrow C^*$ with relative frequency of C^* in data D as weight.
Let CAT be a list of categories (nominal attribute-value pairs) jc sorted in the descending order of coverage.
Let OPEN be a list of implications $jc \Rightarrow C^*$ for each jc from CAT, sorted in the descending order of coverage.
While OPEN is not empty repeat

- delete the first implication $Ant \Rightarrow C^*$ from OPEN;
- compute the *composed weight* w_c from the weights of all *subrules* of $Ant \Rightarrow C^*$ which are already in KB (using function $\oplus$);
- if the validity of $Ant \Rightarrow C^*$ significantly (by test T) differs from w_c then add $Ant \Rightarrow C^*$ to KB with weight w such that $w \oplus w_c$ equals to validity;
- for each jc which precedes in CAT every category from Ant and attribute j is not included in Ant, insert combination $jc \,\&\, Ant$ into OPEN according to its coverage.

Fig. 2. ESOD learning algorithm

To incorporate hierarchies on *input attributes* into the algorithm in Fig. 2, it suffices to include into CAT also abstract values (categories), and to replace the (underlined) notion of *subrule* (rule whose left-hand side contains a subset of attribute-value pairs from the original rule) with the notion of *more general* rule, namely a rule with more general combination at the left-hand side:

> A combination C is more general (or equal) than D iff for every literal $A_i = v_i$ from C, there is a literal $A_i = v_j$ from D such that either $v_i = v_j$ or there is a path leading from v_i to v_j in the hierarchy defined on attribute A_i.

The way of handling *class* hierarchies in ESOD-based *learning* is obvious, as the macro-learner from section 1 does not impose any requirements on the embedded learner; a set of weighted rules is induced for each non-leaf node of the hierarchy (hierarchical rulebase), as well as without recourse to the hierarchy (flat ruleset). The *classification* element of ESOD composes, separately for each class, the weights of rules which match the given instance, and returns the list of classes ordered according to composed weight. For testing purposes, we can assume that the class with highest weight (so-called *best-weight class*) is always chosen. This enables to integrate the ESOD classifier into the macro-classifier.

The current implementation of the *hierarchization* algorithm from section 2 is based on bottom-up hill-climbing clustering using (2) as objective function, with subsequent heuristic pruning. Continuous values are pre-discretized, the cutpoints being local maxima of (the same) objective function.

4 Experiments

4.1 Orienteering control difficulty problem

The task was to learn rules which determine the difficulty of finding orienteering controls, based on four nominal attributes: `obj_type` (type of terrain object), `flag_pos` (position of control flag wrt. the object), `add_prop` (additional property) and `which_of` (distinction among several objects of the same type). The goal attribute was the difficulty, ranging from 1 (easy) to 3 (difficult). Value hierarchies have been constructed for each attribute; in Fig. 3 we show two fragments of the hierarchy on `obj_type`; each addresses a different way of abstraction.

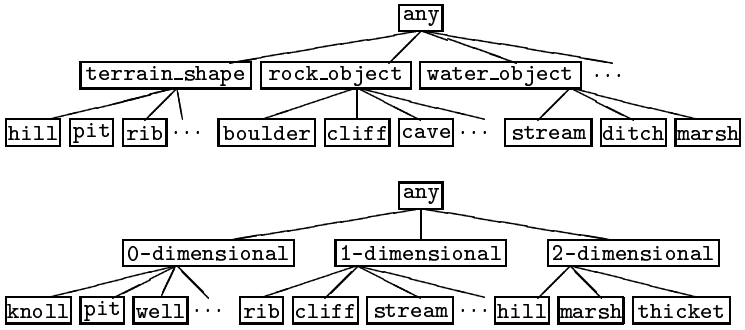

Fig. 3. Fragments of the hierarchy on `obj_type` in the orienteering domain

We ran our learner with attribute hierarchies and without them, yielding two rulebases; both have been tested on the source dataset. The rulebase learned without hierarchies contained almost only empty rules as the frequency of most observable values was too low to enable generalization. Its classification capability was equal to assigning all objects to the most frequent class overall (class 1). The only non-empty rules induced were (weight in parentheses):

```
obj_type=boulder --> class = NOT 1 (0.75)
obj_type=boulder --> class = 3 (0.78)
```

which can be interpreted as "controls at boulders are rather difficult to find". The rulebase learned with hierarchies contained more rules but the accuracy did not significantly improve. However, rules with abstract literals were meaningful and could provide valuable explanatory information:

```
obj_type=1-dim --> class = 1 (0.61)
obj_type=0-dim --> class = NOT 1 (0.61)
obj_type=in_forest AND flag_pos=not_known --> class = NOT 1 (0.59)
obj_type=rock_object --> class = NOT 1 (0.71)
obj_type=man_made --> class = 1 (0.77)
obj_type=1-dim AND flag_pos=foot_of --> class = NOT 1 (0.77)
```

The observation that e.g. controls at linear ("1-dim") or man-made objects are easy to find, unlike controls in rocky areas or at small ("0-dim") objects, comply with common knowledge of orienteering. More details on the domain and a table with results of experiments can be found in [Sv96].

4.2 Glass identification problem

The "glass" database from the UCI repository has 214 instances described by 9 continuous attributes indicating the refraction index and the content of various chemical elements. There are 7 classes grouped into a hierarchy (Fig. 4).

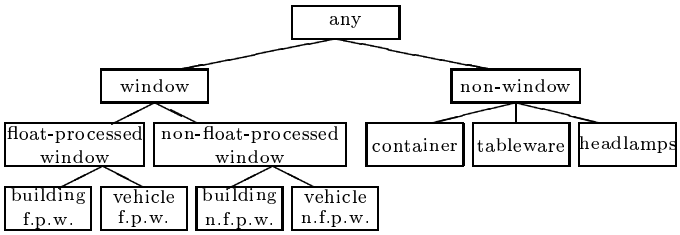

Fig. 4. Class hierarchy in the glass domain

We have first discretized and hierarchized[5] the input attributes. Then we induced both the flat ruleset (FR) and the hierarchical rulebase (HR), and tested them on source data. The results are summarized in Table 1. The average number of rules in partial rulesets of the hierarchical rulebase was significantly lower than the number of rules in the flat ruleset. Individual refinement steps within the hierarchy should thus be more transparent than the classification made merely with the flat ruleset. The hierarchical rulebase had an acceptable performance by itself; moreover, it provided a high number of *perfect-fit explanations* to the classification decisions produced by the flat ruleset. An important observation (see the last two lines in the table) was that the accuracy of classification was substantially higher for decisions with perfect-fit explanations than for the others. This seems to suggest that the lack of explanatory justification signals a lower degree of *confidence* in the conclusion.

In the example at Fig. 5, the classification by the flat ruleset as well as each step of the hierarchical explanation is described in terms of rules which decided about the class assignment. We can see that the explanation decomposes the process of classification: the content of Mg decides between window/non-window glass, while the content of Si indicates that the glass has been float-processed; finally, as none of the "exception" rules for class 3 is fired, it is assumed by default that the instance belongs to class 1.

4.3 Per-share revenue problem

The dataset contained 509 objects describing companies whose stock was privatized during the Czechoslovak voucher privatization, in 1992-4. There are 18 input attributes and a discretized ordered class attribute (per-share revenue) with 8 values. Among the input attributes, two were binary, two were multi-valued nominal (region and industry), and the remaining ones were discretized

[5] For simplicity, we list only the results achieved with input value hierarchies; the results with non-hierarchized values were slightly worse in terms of accuracy.

```
OBJECT:   RI=1.52101, Na=13.64, Mg=4.49, Al=1.10,
          Si=71.78, K=0.06, Ca=8.75, Ba=0.00, Fe=0.00.

CLASSIFICATION:
   1 - Building windows float processed glass
      default:      1 (0.33)
      rule:         Mg > 2.695 --> 1 (0.63)
      rule:         Si from 71.3 to 71.785 --> 1 (0.86)
      composed:     1 (0.93) - best weight by 0.34

HIERARCHICAL EXPLANATION:
   LEVEL 1: choosing between
               win - Window glass
               nonwin - Non-window glass
      default:      win (0.76)
      rule:         Mg > 2.695 --> win (0.94)
      composed:     win (0.99) - best weight by 0.98
   LEVEL 2: choosing between
               fp - Float-processed window glass
               nfp - Non-float-processed window glass
      default:      fp (0.53)
      rule:         Si from 71.3 to 71.785 --> fp (0.91)
      composed:     fp (0.94) - best weight by 0.88
   LEVEL 3: choosing between
               1 - Building windows float processed glass
               3 - Vehicle windows float processed glass
      default:      1 (0.80)
      composed:     1 (0.80) - best weight by 0.60
```

Fig. 5. Example of hierarchical explanation in the glass domain

ordered (various economic indicators). First, we have *hierarchized* all but the binary input attributes, based on the class attribute. Some well-interpretable value unions arose: for the "region" attribute, for example, the top split of the hierarchy separated the eight Czech regions from the three Slovak ones (although the dataset was collected before the political splitting of Czechoslovakia!); the next split divided the Czech regions into sets of five and three, the latter being exactly the western regions which neighbour with EC countries (Germany and/or Austria). The class attribute has been also submitted to hierarchization, which yielded a simple partition into the upper three and the lower five values.

The learning and testing results are in Table 1; "tolerant-matching" accuracy means that each object was considered as correctly classified even if the proposed class was the one immediately above/below the real class. The results are similar as for the glass domain, except that the hierarchical rulebase achieves higher accuracy that the flat ruleset. This may be attributed to the high number and thus low frequency of leaf classes, which prevents some correct rules from being accepted in flat learning, due to insufficient coverage; "focusing" rules at the *any* node, which decide between two large class unions only, are more likely to pass.

Dataset	GLASS	REVENUE	
no.of rules of FR	93	97	
avg.no.of rules per node in HR	25	38.7	
Matching method	strict		"tolerant"
acc.of FR in src.data	81.8%	35.8%	68.8%
acc.of HR in src.data	77.1%	43.0%	77.8%
perfect-fit explan.(PFE)	88.8%	58.7%	
acc.of FR / PFE	84.2%	47.5%	78.9%
acc.of FR / ¬PFE	62.5%	19.0%	54.3%

Table 1. Summary of results from the glass and revenue domains

5 Related work

Taxonomic knowledge-bases linked to input attributes have been used in the RL and KBRL [Ar96] systems to discover useful relations in raw data. The phenomen of tree-structured attributes has also been studied in the context of decision-tree learning, by Núñez [Nu91] and Almuallim et al. [Al95], with the primary aims of cutting down computation time, increasing accuracy and decreasing the size of the tree learned. Our approach seems to lay between these "discovery" and "performance" streams, as the systematic search method of ESOD is akin to discovery techniques, while it disposes an embedded performance element. Unlike [Al95], we assume that the primary input source is the data, the size of the hierarchies being moderate; the common point of both approaches is the search method, which consists of pre-computing the frequencies for all abstract values, followed with search for a kind of optimal representation.

Our use of class hierarchies for generating explanations seems to be novel. However, several approaches exist which rely on knowledge elicited from expert - e.g. a KADS model of expertise [Th93] or qualitative models [Cl93] - in breaking the classification task into simpler steps or in getting a deeper insight into it. Furthermore, one of the hot issues in explanation for expert systems seems to be generation of *reconstructive* explanations in addition to trace-based ones [Wi94]; within ML, note e.g. the GEM system [VM95], where explanation arises as axis-parallel approximation of the complex representation found by an instance-based neural learner. Our separation of "shortcut" classification and step-by-step explanation has similar motivation, although the method has a different flavour.

As a predecessor to our *hierarchization* technique, we can see the value-unioning operation suggested by [Nu91], as well as other techniques exploiting internal disjunctions; it is also closely akin to *conceptual clustering* (of classes) [Fi87], [Ge89], and to *discretization* (of input attributes) [Ca91], [Le94]. Its specificity consists in the account of extrinsic-dissimilarity; among the previous clustering systems, it is only CLASSIT [Ge89] which considers extrinsic dissimilarity but only in the specific sense of continuous attribute values. Among *discretization* techniques, the one proposed by Lee and Shin [Le94] is most similar to our hill-climbing clustering in the search method (but not in the objective function).

6 Conclusions and future work

We have described a set of implemented methods for dealing with value hierarchies in learning: abstract values of input attributes improve comprehensibility of *individual rules*, while the structure of classes gives a deeper insight into the *classification process*. We also study the related task of *hierarchy building* with different extrinsic dissimilarity. Future work should involve extending the testing to non-source data and to base learners and classifiers other than ESOD. We would also like to investigate the problem of multiple explanations in non-tree dags, and to enhance the search method of hierarchization.

The author would like to thank Jiří Iványek, Radim Jiroušek and Petr Berka for their insightful comments and advice.

References

[Al95] Almuallim, H. - Akiba, Y. A. - Kaneda, S.: On Handling Tree-Structured Attributes in Decision Tree Learning. In: Proceedings of the Twelfth International Conference on Machine Learning (ML-95). Morgan Kaufmann, 12-20.

[Ar96] Aronis, J. M. - Provost, F. J. - Buchanan, B. G.: Exploiting Background Knowledge in Automated Discovery. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, 1996 (KDD-96).

[Be94] Berka, P. - Iványek, J.: Automated Knowledge Acquisition for PROSPECTOR-like Expert Systems. In: ECML'94, European Conference on Machine Learning. LNAI, Springer Verlag 1994, 339-342.

[Ca91] Catlett, J.: On changing continuous attributes into ordered discrete attributes. In: Machine Learning - EWSL'91, European Working Session on Learning, Springer Verlag 1991, 164-178.

[Cl93] Clark, P. - Matwin, S.: Using Qualitative Models to Guide Inductive Learning. In: ML'93 - 10th International Machine Learning Conference, 49-56.

[Fi87] Fisher, D. H.: Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning* 2, 139-172.

[Ge89] Gennari, J. - Langley, P. - Fisher, D. H.: Model of Incremental Concept Formation. *Artificial Intelligence Journal*, Vol.40, 11-61.

[Le94] Lee, C. - Shin, D. G.: A Context-Sensitive Discretization of Numeric Attributes for Classification Learning. In: European Conference on Artificial Intelligence, ECAI-94, 428-432.

[Nu91] Núñez, M.: The Use of Background Knowledge in Decision Tree Induction. *Machine Learning*, 6, 231-250 (1991).

[Sv96] Svátek, V.: Learning with Value Hierarchies in the ESOD framework. In: Artificial Intelligence Techniques, AIT-96, Brno 1996, 102-109.

[Th93] Thomas, J. - Laublet, P. - Ganascia, J. G.: A Machine Learning Tool Designed for a Model-Based Knowledge Acquisition Approach. In: EKAW-93, European Knowledge Acquisition Workshop, LNAI, Springer-Verlag 1993, 123-138.

[VM95] Van de Merckt, T. - Decaestecker, C.: About Breaking the Trade Off Between Accuracy and Comprehensibility in Concept Learning. In: Proceedings of the Workshop on Comprehensibility in Machine Learning, IJCAI-95, Montreal.

[Wi94] Wick, M. R.: Explanation as a Primary Task in Problem Solving. *The Knowledge Engineering Review*, Vol.9:1, 1994, 18-82.

This article was processed using the LaTeX macro package with LLNCS style