

Learning Decision Lists over Tree Patterns and Its Application

Satoshi Kobayashi

Kohichi Hori

Setsuo Ohsuga

Research Center for Advanced Science and Technology
University of Tokyo
4-6-1, Komaba, Meguro-ku, Tokyo, 153, Japan

Abstract

This paper introduces a new concept, a *decision tree (or list) over tree patterns*, which is a natural extension of a decision tree (or decision list), for dealing with tree structured objects.

The learnability of this class is studied within the framework of the Probably Approximately Correct learning model and the Identification in the Limit model. It is found that the class k -node-DLRTP, a subclass of decision lists over regular tree patterns, is not polynomial time PAC learnable if $NP \neq RP$, but that the class k -node-DLRTP(m), a subclass of k -node-DLRTP where the number of variables in each tree pattern is bounded by m , is polynomial time PAC learnable. We also show that the class of decision lists over regular tree patterns is polynomial time inferable in the limit. Then we propose a more practical learning algorithm which employs heuristics based on an information theoretical evaluation function. This algorithm is applied to learning control strategies of rules in various domains, and its effectiveness is shown.

1 Introduction

Learning has long attracted the attention of many researchers in the artificial intelligence community. It has also been expected to be one of the methods for overcoming the knowledge acquisition bottleneck of expert systems. Learning from examples is the best studied of all proposed learning schemata, because giving examples of a concept is easier for humans than giving the precise definition of it. Most of past research reduces learning problems to search problems of target concepts in the hypothesis space with refinement operators. On the other hand, Quinlan's innovative work on learning decision trees shows another top-down approach for learning [Quinlan, 1986]. His method depends on an information theoretical evaluation function to select good attributes for decision nodes. This technique has been applied to various domains for classifying objects which have many attributes.

This paper introduces the new concept of a *decision tree over tree patterns*. This is a natural extension of a

decision tree that deals with tree structured objects.

One of the most interesting application areas of this work is inference control. We can use decision trees over tree patterns as control structures for applying appropriate rules for selected subgoals in the SLD resolution process.

In section 2, we will define the concept of decision trees over tree patterns, and in section 3, we show theoretical results on learnability of this class. The learning models used in this paper are the Probably Approximately Correct (PAC in short) learning model and the Identification in the Limit model, introduced by [Valiant, 1984] and [Gold, 1967] respectively. We prove that the class k -node-DLRTP(m) is polynomial time PAC learnable and that the class DLRTP is polynomial time inferable in the limit. In section 4, we propose a practical learning algorithm for the class DLTP. In section 5, applications of the algorithm to inference control are shown.

2 Decision Trees and Decision Lists over Tree Patterns

Tree pattern language has been applied to many research areas such as term rewriting systems, logic programming, etc. Its learnability has been well studied [Plotkin, 1970][Arimura *et al.*, 1970]. In this section, we give the definition of decision trees over tree patterns which can be used for classification problems of tree structured objects.

Let $\Sigma = \{a, b, c, \dots\}$ be a finite set of alphabets, representing *functors*, associated with a mapping *arity* from Σ to nonnegative integers, and $V = \{X, Y, Z, \dots\}$ be a countable set of alphabets disjoint from Σ whose elements are called *variables*.

Definition 1 A tree pattern on $\Sigma \cup V$ is defined recursively as follows.

- (1) t is a tree pattern if t is a 0-ary functor (i.e. $\text{arity}(t)=0$).
- (2) variables are tree patterns.
- (3) an ordered tree $f(t_1, \dots, t_n)$ is a tree pattern if f is a n -ary functor, and $t_1, \dots, t_n$ are tree patterns.

We say a tree pattern t is *regular*, if all the variables in t occur at most once.

The length of a tree pattern t is the length of the expression representing it, and is denoted by $|t|$. We

sometimes regard a tree pattern as a tree structure. The height of a tree pattern t is the height when we regard t as a tree. The depth of a specified functor or variable in a tree pattern is defined in a similar manner. For example, a tree pattern $a(b(c(X,e(f)),g(h,i(Y))),j)$ is shown in Fig.1. The length and the height of this tree pattern are 27 and 4 respectively. The depth of the functor h is 3.

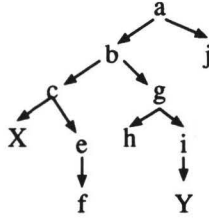


Figure 1: Tree pattern

$TP(\Sigma)$ denotes the set of all tree patterns on $\Sigma \cup V$. We define $subTP$ as a function from $TP(\Sigma) \times \{\text{nonnegative integers}\}$ to $TP(\Sigma)$ such that $subTP(t,i)$ is a tree pattern of height i obtained from t by replacing the nodes of depth i with different variables. For instance, $subTP(a(b(c(X,e(f)),g(h,i(Y))),j),2) = a(b(Z,W),j)$.

A substitution is a homomorphism θ from $TP(\Sigma)$ to $TP(\Sigma)$ such that $\theta(f) = f$ for any 0-ary functor $f \in \Sigma$. We write $p \leq' q$ if there exists a substitution θ such that $p = \theta(q)$.

Definition 2 A tree pattern p defines the following language.

$$L(p) = \{t \in TP(\Sigma) \mid t \leq' p\}$$

Let C be a set of alphabets representing classes of objects. A *decision tree over (regular) tree patterns* is a binary tree whose internal nodes, called *decision nodes*, are labeled by (regular) tree patterns and whose leaves are labeled by elements of C . The label of the node n is denoted by $label(n)$.

A decision tree over tree patterns dl defines the function $func_{dl}$ from $TP(\Sigma)$ to C . When we are given a tree pattern t , then a unique path from the root to a leaf is determined as follows. If the current node of the path is labelled by the tree pattern p , then test for $t \leq' p$. If the test succeeds, select the left child, otherwise, select the right child, as the next node of the path, respectively. The label of the leaf of the path is the value of the tree pattern t by the function $func_{dl}$. For the decision tree dl defined in Fig.2, the values of $func_{dl}(a(b,c))$ and $func_{dl}(b(a(c),e(Z)))$ are c_2 and c_3 respectively.

A *decision list over (regular) tree patterns* is a decision tree over (regular) tree patterns whose decision nodes all have leaves as left children, and is written as a sequence of tuples,

$$(tp_1, c_1), (tp_2, c_2), \dots, (tp_r, c_r), (X, c_{r+1})$$

where tp_i and cp_i ($1 \leq i \leq r$) are elements of $TP(\Sigma)$ and C respectively. X is a tree pattern composed of only one variable X itself. Each tp_i represents the label of the i th decision node from the root, and c_i represents its left child's label. This is a natural extension of decision

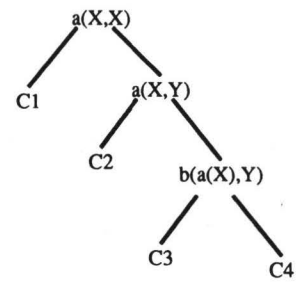


Figure 2: Decision tree on tree patterns

lists[Rivest, 1987] over tree patterns. The decision tree in Fig.2 is also a decision list and is written;

$$(a(X, X), c_1), (a(X, Y), c_2), (b(a(X), Y), c_3), (X, c_4).$$

Decision lists over tree patterns can be used as control structures to apply appropriate rules for selected subgoals in SLD resolution. For this purpose, we regard selected subgoals as tree patterns to be classified and applied rules as classes.

For example, let us consider the following rules written in Prolog.

r1 $del([X|Y], X, Z) :- del(Y, X, Z).$

r2 $del([X|Y], Z, [X|W]) :- del(Y, Z, W).$

r3 $del([], X, []).$

Although $del(X, Y, Z)$ is intended to mean that Z is the list obtained from the list X by removing all Y 's in X , this program does not work correctly. If we control the inference as shown in Fig.3, then this program will work correctly.

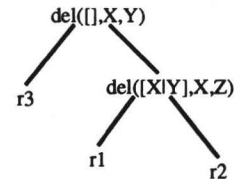


Figure 3: Decision list on tree patterns for inference control

In section 5, we show experiments of learning such control strategies as decision lists over tree patterns from correct traces of the given program.

3 Theoretical Results

We define the following classes of functions.

DLTP(DLRTP) the class of functions which can be defined by decision lists over (regular) tree patterns.

k-node-DTTP the class of functions which can be defined by decision trees over tree patterns whose number of decision nodes is bounded by k .

k-node-DLTP(k-node-DLRTP)

the subclass of DLTP(DLRTP) where the number of tuples in decision lists is bounded by $k + 1$.

the algorithm outputs DL . This construction gives us a polynomial-time ordering for k -node-DLRTP(m), since the consistency check can be done in polynomial time. This completes the proof. $\square$

This theorem is an extended result of Theorem 1 in [Arikawa *et al.*, 1992].

3.3 Polynomial Time Inference of DLRTP in the Limit

In this subsection, we prove that DLRTP is polynomial time inferable in the limit. The notion of identification in the limit is first introduced by [Gold, 1967].

An inference machine M is an algorithm that accepts a sample as an input and outputs a conjecture. Let $\delta = s_1, s_2, \dots$ be a sample presentation. When M requests i th sample, M reads the sample s_i and adds the i th guess g_i to the sequence $g_1, g_2, \dots$ of guesses, where each g_i denotes a code of some function. We say that M on input δ converges to g if there exists a positive integer n such that $\forall i \geq n, g_i = g$.

An inference machine M is said to be *consistent* if it always outputs a conjecture g_i such that g_i is consistent with the sample set $S = \{s_1, \dots, s_i\}$. We say that M is *conservative* if it continues to produce the same guess while the guess is consistent with the sample. The class of functions F is said to be *inferable* by M , if, for any $f \in F$ and for any sample presentation δ , M on input δ converges to g such that $func_g = f$, where $func_g$ denotes the function calculated by g .

Definition 5 A class of functions F is *polynomial time inferable in the limit*, if there exists an inference machine M that infers F consistently and conservatively, and computes the guess g_i in polynomial time with respect to $\|S_i\|$, where $S_i = \{w_1, \dots, w_i\}$ and $\|S_i\|$ denotes the size of S_i as an expression.

The next algorithm M_1 is the inference machine of DLRTP which computes each guess g_i in polynomial time.

Inference Machine M_1

input a sample presentation $(t_1, c_1), (t_2, c_2), \dots$

output an infinite sequence $g_1, g_2, \dots$ of guesses.

begin

$g_0 = \text{none}; S = \phi; i = 0;$

repeat

$i = i + 1;$

$S = \text{SU}\{(t_i, c_i)\};$

if $func_{g_{i-1}}$ is not consistent with (t_i, c_i) **then**

$g_i = \text{MinDLRTP}(S);$

else $g_i = g_{i-1};$

output $g_i;$

forever

end

MinDLRTP(S)

input a sample set $S = \{(t_1, c_1), \dots, (t_n, c_n)\}$

output a decision list over tree patterns

begin

$L = \text{null list};$

while $(S \neq \phi)$ **do**

for each $(t_i, c_i) \in S$ **do**

$TP_i = \text{findMinTP}((t_i, c_i), S);$

end

$TP_j = \text{minimum height tree pattern in } \{TP_i : 1 \leq i \leq |S|\};$

add (TP_j, c_j) **at the end of** $L;$

$S = S - \{(t, c) \in S : t \in L(TP_j)\};$

end

return $L;$

end

findMinTP($(t, c), S$)

input a sample (t, c) and a sample set S

output a tree pattern

$n = \text{height of } t;$

for $i=0$ **to** $n+1$ **do**

$p = \text{subTP}(t, i);$

if $\forall (t', c') \in S (t' \in L(p) \rightarrow c' = c)$ **then break;**

end

return $p;$

end

Let $\text{maxH}(dl) = \max\{\text{height of } t_i : 1 \leq i \leq n\}$ for a decision list $dl = (tp_1, c_1), \dots, (tp_n, c_n)$, and let $\text{minMaxH}(S) = \min\{\text{maxH}(dl) : dl \in \text{DLRTP} \text{ is consistent with } S\}$. Then, the next lemma holds.

Lemma 1 $\text{MinDLRTP}(S)$ returns the decision list $dl = (tp_1, c_1), \dots, (tp_n, c_n)$ consistent with S and such that $\text{minMaxH}(S) = \text{maxH}(dl)$.

Proof

It is obvious that $\text{MinDLRTP}(S)$ terminates and outputs a decision list over tree patterns consistent with S . Therefore, $\text{minMaxH}(S) \leq \text{maxH}(dl)$ holds. It is sufficient for us to show $\text{minMaxH}(S) \geq \text{maxH}(dl)$. We prove this by induction of the number of tuples of dl , written rn .

In case of $rn = 1$, the claim holds obviously. We assume that the claim holds in case of $rn = k - 1$. Let us consider the case of $rn = k$ and assume $\text{minMaxH}(S) < \text{maxH}(dl)$. By the induction hypothesis, we have that $\text{maxH}(dl') = \text{minMaxH}(S') \leq \text{minMaxH}(S) < \text{maxH}(dl)$, where $dl' = (tp_2, c_2), \dots, (tp_n, c_n)$, and $S' = S - \{(t, c) \in S : t \in L(tp_1)\}$. Let h_1 be the height of tp_1 . Then, we have $h_1 = \text{maxH}(dl)$.

There exists a decision list $dl^* = (tp_1^*, c_1^*), \dots, (tp_m^*, c_m^*)$ consistent with S such that $\text{maxH}(dl^*) = \text{minMaxH}(S)$ by the definition of minMaxH . Let h_2 be the height of tp_1^* . Then, we have $h_1 = \text{maxH}(dl) > \text{minMaxH}(S) \geq h_2$.

For tp_1^* , $\forall (t', c') \in S (t' \in L(tp_1^*) \rightarrow c' = c_1^*)$ holds. Therefore, if we select $(t, c) \in S$ such that $t \in tp_1^*$, then $\forall (t', c') \in S (t' \in L(\text{subTP}(t, h_2)) \rightarrow c' = c_1^*)$ holds, since $L(\text{subTP}(t, h_2)) \subseteq L(tp_1^*)$. This means $\text{minTP}((t, c), S)$ returns a tree pattern whose height is less than or equal to $h_2 (< h_1)$. This contradicts the fact that TP_j is the minimum height tree pattern in $\{TP_i : 1 \leq i \leq |S|\}$. So, we have $\text{maxH}(dl) \leq \text{minMaxH}(S)$ in case of $rn = k$.

This completes the proof. $\square$

Theorem 6 The inference machine M_1 infers DL RTP in polynomial time in the limit.

Proof

Let $k = \min\{\max H(dl) : dl \in \text{DL RTP}, \text{func}_{dl} = f\}$, where f is a target function, and m be the upper bound of arity. Then $f \in \text{DL-k,m-RTP}$. By the above lemma, we can conclude that M_1 converges to g such that $f = \text{func}_g$, since $|\text{DL-k,m-RTP}|$ is bounded by $((|\Sigma| + 2)^{m^k + 1})!$.

$|C|(|\Sigma| + 2)^{m^k + 1}$ (i.e. finite).

The time required to compute a conjecture for each stage is bounded by $O(p^2 \cdot q^3)$, where p is the maximum size of the given samples and q is the number of them. This is a polynomial function with respect to the sample set size. $\square$

For practical use, it is more appropriate to use M_1 equipped with some heuristics as proposed in the next section, in order to obtain a decision list of short description length.

4 A Practical Learning Algorithm

In the previous section, we give the learnability results on some subclasses of DLTP. Although polynomial time learning algorithms are obtained, the results are limited to learning of decision lists on *regular* tree patterns. In this section, we give a practical algorithm for learning decision lists on tree patterns.

The minimum description length (MDL, in short) principle says that the hypothesis of minimum description length gives good prediction for unknown data. If we are given noise free data, it is necessary for us to obtain a decision list over tree patterns of minimum length consistent with the given data. Unfortunately, Theory3 says that it is impossible to obtain such a description in polynomial time if $NP \neq P$.

In this section, we show a heuristic algorithm based on the information theoretical evaluation function. Although finding the minimum cost decision tree consistent with the given data is NP-Complete [Hyafil and Rivest, 1976], [Quinlan, 1986] shows that the entropy function of information theory can play a significant part in the top-down synthesis of small decision trees consistent with the given data. Our method is an extension of [Quinlan, 1986].

When S is a set of tree patterns, $\text{lgg}(S)$ denotes the least generalization [Plotkin, 1970] of all elements in S .

We divide the given sample set S into its subsets $S_1, \dots, S_n$, where $n = |C|$, $\forall (t, c) \in S_i (c = c_i)$ for each class $c_i \in C$, and $\bigcup_{i=1}^n S_i = S$. Let $\text{pat}(S) = \{t : (t, c) \in S\}$, $\text{lgg}'(S) = \text{lgg}(\text{pat}(S))$, $m_{i,j} = |\{t : t \leq' \text{lgg}'(S_i), (t, c) \in S_j\}|$, and $k_i = \sum_{j=1}^n m_{i,j}$, then we can define an evaluation function f_e as follows.

$$f_e(\{S_1, \dots, S_n\}) = \sum_{i=1}^n (|S_i| \cdot \sum_{j=1}^n \frac{m_{i,j}}{k_i} \cdot \log \frac{m_{i,j}}{k_i})$$

This evaluation function f_e intuitively means the complexity of the sample set $\{S_1, \dots, S_n\}$. We implement the learning algorithm of decision lists over tree patterns, called M_2 , which employs the following heuristics to obtain a smaller decision list.

heuristic 1 if there exists $i \in \{1, \dots, n\}$ such that $\forall j (i \neq j \Rightarrow \forall s \in \text{pat}(S_j) \neg (s \leq' \text{lgg}'(S_i)))$, then add $(\text{lgg}'(S_i), c_i)$ to the decision list.

heuristic 2 otherwise, find the subset S'_i of S_i for each i which minimizes $f_e(\{S_1, \dots, S_{i-1}, S'_i, S_{i+1}, \dots, S_n\})$ by the hill climbing method, select S'_j from $\{S'_1, \dots, S'_n\}$ which gives the minimum value of f_e , and add $(\text{lgg}'(S'_j), c_j)$ to the decision list.

In the next section, this algorithm is applied to the learning of control strategies for selecting appropriate rules in the SLD resolution process.

5 Applications

In this section, we apply the algorithm described in the previous section to the inference control problem in robot planning domain. The setting is shown in Fig.4. The goal is to move the box b to the place g , and open all the doors. The operators are go-to-goal, go-to-left-door, go-to-right-door, go-to-box, go-through-door, open-door, pick-up, and put-down, which are all written as Horn clauses. The total number of problem instances in this domain is 1152.

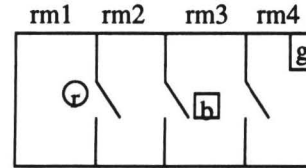


Figure 4: Robot planning domain

One of the operators is shown bellow.

put-down:

$\text{op}(\text{st}(\text{b}(\text{X1}), \text{r}(\text{X2}), \text{at}(\text{X3}), \text{have}(\text{b}), \text{on}(\text{X5}), \text{open}(\text{X6}, \text{X7}, \text{X8})), [\text{put_down}|\text{OPL}]) :-$

$\text{op}(\text{st}(\text{b}(\text{X1}), \text{r}(\text{X2}), \text{at}(\text{X3}), \text{have}(\text{nil}), \text{on}(\text{X3}), \text{open}(\text{X6}, \text{X7}, \text{X8})), \text{OPL}).$

X in $b(X)$ and Y in $r(Y)$ denote the rooms where the box and the robot exist respectively. X in $\text{on}(X)$ and Y in $\text{at}(Y)$ represent the positions of the box and the robot in the rooms. There are three positions in each room. $\text{have}(X)$ means that the robot has X . X , Y , and Z in $\text{open}(X, Y, Z)$ denote whether the 1st, 2nd and 3rd doors are open or not. In this setting, it seems rather difficult for a human to refine these rules so that all the problem instances can be solved with no backtracking. The aim of this experiment is to obtain a control strategy as a decision list which can solve almost all problem instances with no backtracking. The flow of the experiment is as follows.

for $i=1, \dots, 70$ **do**

select i problem instances randomly and obtain traces of solutions from user.

extract $\langle\langle$ selected subgoal, applied rule $\rangle\rangle$ information from the traces as samples for M_2 and run M_2 on them.

select 50 unknown problem instances randomly and test the learned control strategies on them.

end

The coverage of the learned control structure for unknown problem instances are shown in Fig.5 The number of tuples of learned decision list in case of $i=70$ is 35, and its coverage is 96 percent of the given unknown problem instances. This experiment says that M_2 can produce good (i.e. small size of) hypotheses for unknown data. In the blocks world domain, M_2 can learn a control structure for moving n-blocks from stack A to stack B with only one problem instance of moving 3-blocks. Another experiment from the circuit design domain shows similar good results of learning a general control structure from a few examples[Kobayashi, 1993].

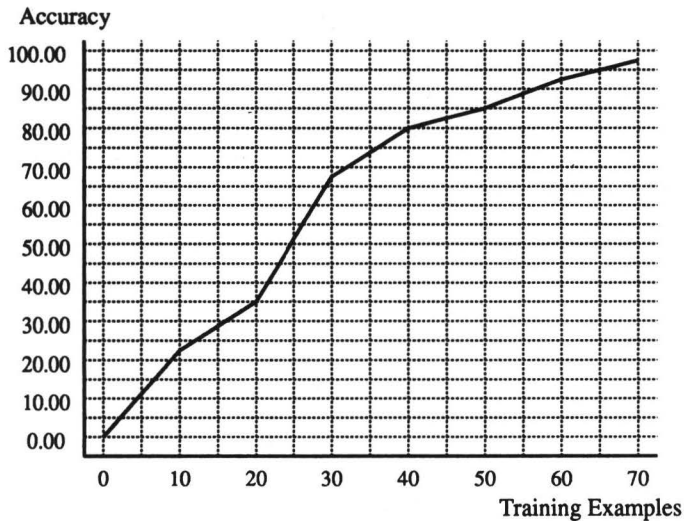


Figure 5: Coverage of learned control structure

6 Related Research

Learning decision trees is well studied. [Quinlan, 1986] proposes a method for top-down synthesis of decision trees from the given sample set based on the information theoretical evaluation function. [Quinlan and Rivest, 1989] uses MDL principle to guide the search of noise tolerant decision trees. The main difference between their works and our work is that test tree patterns are not explicitly predefined. The least generalization of samples is used for the construction of test tree patterns in the algorithm presented in section 4.

The learning problem of decision trees over regular patterns is studied in [Arikawa et al., 1992] and applied to transmembrane domains. Although our results are closely related to their work, the learnability result on the framework of identification in the limit is not studied in [Arikawa et al., 1992].

7 Conclusion

We propose the new concept of decision trees (or lists) over tree patterns. This is a natural extension of decision trees for dealing with tree structured objects. Then we show theoretical results on its learnability. Not only theoretical algorithms but the practical one is proposed

based on the information theoretical evaluation function. Its application to inference control is experimented and good results are obtained in various domains such as robot planning, blocks world, circuit design, etc.

Although the proposed heuristics work well, there seems to be many other possibilities of heuristics for learning decision lists over tree patterns. Experimental comparison must be done in future works.

Acknowledgement

We would like to thank Tatsuya Akutsu and Iannis Zannos for helpful discussions and comments.

References

- [Quinlan, 1986] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.
- [Quinlan and Rivest, 1989] J. R. Quinlan and R. L. Rivest. Inferring Decision Trees Using Minimum Description Length Principle. *Information and Computation*, 80:227–248, 1989.
- [Valiant, 1984] L. G. Valiant. A Theory of Learnable. *Comm. Assoc. Computing Machinery*, 27:1134–1142, 1984.
- [Gold, 1967] E. M. Gold. Language Identification in the Limit. *Information and Control*, 10:447–474, 1967.
- [Plotkin, 1970] G. D. Plotkin. A Note on Inductive Generalization. *Machine Intelligence*, 5:153–163, 1970.
- [Arimura et al., 1970] H. Arimura, T. Shinohara, and S. Otsuki. Polynomial Time Inference of Unions of Tree Pattern Languages. In *Proceedings of the Second Workshop on Algorithmic Learning Theory*, pages 105–114, 1991.
- [Rivest, 1987] R. L. Rivest. Learning Decision Lists. *Machine Learning*, 2:229–246, 1987.
- [Natarajan, 1989] B. K. Natarajan. On Learning Sets and Functions. *Machine Learning*, 4:67–97, 1989.
- [Kobayashi, 1993] S. Kobayashi. Explanation Based Learning: Extracting Common Structures from Multiple Training Examples *Ph.D. Th.*, Dept. of Aeronautics, Tokyo University, 1993, in Japanese.
- [Arikawa et al., 1992] S. Arikawa, S. Kuhara, S. Miyano, Y. Mukouchi, A. Shinohara, and T. Shinohara. A Machine Discovery from Amino Acid Sequences by Decision Trees over Regular Patterns. In *Proceedings of International Conference on Fifth Generation Computer Systems*, pages 618–625, 1992.
- [Hyafil and Rivest, 1976] L. Hyafil and R. L. Rivest. Constructing Optimal Binary Decision Trees is NP-Complete. *Information Processing Letters*, 5:15–17, 1976.