Published in:
Encyclopedia of Optimization, (P.M. Pardalos and C. Fludas, Eds.), pp. xxx-xxx, 1998 (in print).

Inference of Monotone Boolean Functions

EVANGELOS TRIANTAPHYLLOU, VETLE I. TORVIK and JIEPING LU

Department of Industrial and Manufacturing Systems Engineering

3128 CEBA Building

Louisiana State University

Baton Rouge, LA 70803-6409

U.S.A.

E-mail: ietrian@lsuvm.sncc.lsu.edu, vtorvik@unix1.sncc.lsu.edu,

Web URL: http://www.imse.lsu.edu/vangelis

Keywords and Phrases:

Boolean function, monotone boolean functions, isotone boolean function, antitone boolean function, classification problems, boolean function inference problem, free distributive lattice, Conjunctive Normal Form (CNF), Disjunctive Normal Form (DNF), interactive learning of boolean functions, Shannon function, Hansel theorem, pattern recognition/classification approaches, Hansel chains, optimal question-asking strategy, sequential Hansel chains question-asking strategy, binary search/Hansel chains question-asking strategy, binary search.

1. Introduction to the Boolean Function Inference Problem

The goal in a classification problem is to uncover a system that places examples into two or more mutually exclusive groups. Identifying a classification system is beneficial in several ways. First of all, examples can be organized in a meaningful way, which will make the exploration and retrieval of examples belonging to specific group(s) more efficient. The tree-like directory structure, used by personal computers in organizing files, is an example of a classification system which enables users to locate files quickly by traversing the directory paths. A classification system can make the relations between the examples easy to understand and interpret. A poor classification strategy, on the other hand, may propose arbitrary, confusing or meaningless relations. An extracted classification system can be used to classify new examples. For an incomplete or stochastic system, its structure may pose questions whose answers may generalize the system or make it more accurate.

A special type of classification problem, called the ***boolean function inference problem***, is when all the examples are represented by binary (0 or 1) attributes and each example belongs to one of two categories. Many other types of classification problems may be converted into a boolean function inference problem. For example, a multi-category classification problem may be converted into several two-category problems. In a similar fashion, example attributes can be converted into a set of binary variables.

In solving the boolean function inference problem many properties of boolean logic are directly applicable. A ***boolean function*** will assign a binary value to each boolean vector (example). See [Schneeweiss, 1989] for an overview of boolean functions. Usually, a boolean function is expressed as a conjunction of disjunctions, called the Conjunctive Normal Form (CNF), or a disjunction of conjunctions, called the Disjunctive Normal Form (DNF). CNF can be written as:

$$\bigwedge_{j=1}^{k} \left(\bigvee_{i \in \rho_j} x_i \right),$$

where x_i is either the attribute or its negation, k is the number of attribute disjunctions and ρ_j is the j-th index set for the j-th attribute disjunction. Similarly, DNF can be written as:

$$\bigvee_{j=1}^{k} (\bigwedge_{i \in \rho_j} x_i).$$

It is well known that any boolean function can be written in CNF or DNF form. Peysakh [1987] presented an algorithm for converting any boolean expression into CNF. Two functions in different forms are regarded as equivalent as long as they assign the same function values to all the boolean vectors. However, placing every example into the correct category is only one part of the task. The other part is to make the classification criteria meaningful and understandable. That is, an inferred boolean function should be as simple as possible. One part of the boolean function inference problem that has received substantial research efforts is that of simplifying the representation of boolean functions, while maintaining a general representation power.

2. Inference of Monotone Boolean Functions

When the target function can be any boolean function with n attributes, all of the 2^n examples have to be examined to reconstruct the entire function. When we have a priori knowledge about the subclass of boolean functions the target function belongs to, on the other hand, it may be possible to reconstruct it using a subset of the examples. Often one can obtain the function values on examples one by one. That is, at each inference step, an example is posed as a question to an oracle, which, in return, provides the correct function value. A function, f, can be defined by its oracle A_f which, when fed with a vector $x = <x_1, x_2, x_3, ..., x_n>$, returns its value $f(x)$. The inference of a boolean function from questions and answers is known as ***interactive learning of boolean functions***. In many cases, especially when it is either difficult or costly to query the oracle, it is desirable to pose as few questions as possible. Therefore, the choice of examples should be based on the previously classified examples.

Monotone boolean functions is a subset of boolean functions that have been extensively studied not only because of their wide range of applications (see [Eiter and Gottlob, 1995], [Ibaraki and Bioch, 1995], [Fredman and Khachiyan, 1996] and [Shmulevich, 1997]) but also their intuitive interpretation. Each attribute's contribution to a monotone function is either non-negative or non-positive (not both).

Furthermore, if all of the attributes have non-negative (or non-positive) effects on the function value then the underlying monotone boolean function is referred to as ***isotone*** (respectively ***antitone***). Any isotone function can be expressed in DNF and CNF without using negated attributes. In combinatorial mathematics, the set of isotone boolean functions is often represented by the free distributive lattice (FDL). To formally define monotone boolean function, consider ordering the binary vectors as follows [Rudeanu, 1974]:

Let E^n denote the set of all binary vectors of length n; let x and y be two such vectors. Then, the vector $x = \langle x_1, x_2, ..., x_n \rangle$ ***precedes*** *vector $y = \langle y_1, y_2, ..., y_n \rangle$ (denoted as $x \preceq y$) if and only if $x_i \le y_i$ for $1 \le i \le n$. If, at the same time $x \neq y$, then x* ***strictly precedes*** *y (denoted as $x \prec y$).*

According to this definition, the order of vectors in E^2 can be listed as follows:

$$\langle 11 \rangle \succ \langle 01 \rangle \succ \langle 00 \rangle$$
and
$$\langle 11 \rangle \succ \langle 10 \rangle \succ \langle 00 \rangle.$$

Note that the vectors <01> and <10> are in a sense incomparable.

Based on the order of the boolean vectors, a ***non-decreasing monotone (isotone) boolean function*** can be defined as follows [Rudeanu, 1974]:

A boolean function f is said to be an non-decreasing monotone boolean function if and only if for any vectors $x, y \in E^n$, such that $x \prec y$, then $f(x) \prec f(y)$.

A ***non-increasing monotone (antitone) boolean function*** can be defined in a similar fashion. As the method used to infer an antitone boolean function is the same as that of a isotone boolean function, we will restrict our attention to the isotone boolean functions.

When analyzing a subclass of boolean functions, it is always informative to determine its size. This may give some indications of how general the functions are and how hard it is to infer them. The number

of isotone boolean functions, $\Psi(n)$, defined on E^n is sometimes referred to as the n-th Dedekind number after [Dedekind, 1897] who computed it for $n = 4$. Since then it has been computed for up to E^8.

$\Psi(1) = 3,\ \Psi(2) = 6,\ \Psi(3) = 20,$

$\Psi(4) = 168$	by [Dedekind, 1897]
$\Psi(5) = 7{,}581$	by [Church, 1940]
$\Psi(6) = 7{,}828{,}354$	by [Ward, 1946]
$\Psi(7) = 2{,}414{,}682{,}040{,}998$	by [Church, 1965]
$\Psi(8) = 56{,}130{,}437{,}228{,}687{,}557{,}907{,}788$	by [Wiedeman, 1991]

Wiedeman's [1991]'s algorithm employed a Cray-2 processor for 200 hours to compute the value for $n = 8$. This gives a flavor of the complexity of computing the exact number of isotone boolean functions. The computational infeasibility for larger values of n provides the motivation for approximations and bounds. The best known bounds on $\Psi(n)$ are due to [Kleitman, 1969] and [Kleitman and Markowsky, 1975]:

$$2^{\binom{n}{\lfloor n/2 \rfloor}(1+c_1 2^{-n/2})} \leq \Psi(n) \leq 2^{\binom{n}{\lfloor n/2 \rfloor}(1+c_2 \log(n)/n)},$$

where c_1 and c_1 are constants and $\lfloor n/2 \rfloor$ is the integer part of $n/2$.

These bounds, which are improvements over the first bounds obtained by [Hansel, 1966], are also based on the Hansel chains described in section 4. Even though these bounds can lead to good approximations for $\Psi(n)$,when n is large, the best known asymptotic is due an extensive analysis by [Korshunov, 1981]:

$$\Psi(n) \sim \begin{cases} 2^{\binom{n}{n/2}} e^{\binom{n}{n/2-1}\left(\frac{1}{2^{n/2}} + \frac{n^2}{2^{n+5}} - \frac{n}{2^{n+4}}\right)}, & \text{for even } n. \\ 2^{\binom{n}{n/2-1/2}+1} e^{\binom{n}{n/2-3/2}\left(\frac{1}{2^{(n+3)/2}} - \frac{n^2}{2^{n+6}} - \frac{n}{2^{n+3}}\right) + \binom{n}{n/2-1/2}\left(\frac{1}{2^{(n+1)/2}} + \frac{n^2}{2^{n+4}}\right)}, & \text{for odd } n. \end{cases}$$

[Shmulevich, 1997] achieved a similar but slightly inferior asymptotic for even n in a simpler and more elegant manner, which lead him to some interesting distributional conjectures regarding isotone boolean functions.

Even though the number of isotone boolean functions is large, it is a small fraction of the number of general boolean functions, 2^{2^n}. This is the first hint towards the feasibility of efficiently inferring monotone boolean functions. Intuitively, one would conjecture that the generality of this class was sacrificed. That is true, however, a general boolean function consists of a set of areas where it is monotone. In fact, [Kovalerchuk, *et al.*, 1995] showed that any boolean function $q(x_1, ..., x_n)$ can be represented by several non-decreasing $g_i(x)$ and non-increasing $h_j(x)$ monotone boolean functions in the following manner:

$$q(x) = \bigvee_i (g_i(x) \bigwedge_j h_j(x)).$$

As a result, one may be able to solve the general boolean function inference problem by considering several monotone boolean function inference problems. Intuitively, the closer the target function is to a monotone boolean function, the fewer monotone function are needed to represent it and more successful this approach might be. [Kovalerchuk, *et al.*, 1995] formulated the problem of joint restoration of two nested monotone boolean functions f_1 and f_2. Their approach allows one to further decrease the dialogue with an expert (oracle) and restore a complex function of the form $f_1 \& \neg f_2$, which is not necessarily monotone.

3. The Shannon Function and the Hansel Theorem

The complexity of inferring isotone boolean functions was mentioned in section 2, when realizing that the number of isotone boolean functions is a small fraction of the total number of general boolean functions. In defining the most common complexity measure for the boolean function inference problem, consider the following notation. Let M_n denote the set of all monotone boolean functions, and $A = \{F\}$ be the set of all algorithms which infer $f \in M_n$, and $\varphi(F, f)$ be the number of questions to the oracle A_f required to infer f. The ***Shannon function*** $\varphi(n)$ can be introduced as follows [Korobkov, 1965]:

$$\varphi(n) = \min_{F \in A} (\max_{f \in M_n} \varphi(F, f)).$$

An upper bound on the number of questions needed to restore a monotone boolean function is given by the following equation (known as ***The Hansel Theorem***) [Hansel, 1966]:

$$\varphi(n)=\binom{n}{\lfloor n/2\rfloor}+\binom{n}{\lfloor n/2\rfloor+1}.$$

That is, if a proper question asking strategy is applied, the total number of questions needed to infer any monotone boolean functions should not exceed $\varphi(n)$. The Hansel Theorem can be viewed as the worst case scenario analysis. Recall, from section 2, that all of the 2^n questions are necessary to restore a general boolean function. [Gainainov 1984] proposed three other criteria for evaluating the efficiency of algorithms used to infer isotone boolean functions. One of them is the average case scenario and the two others consider two different ways of normalizing the Shannon function by the size of the target function.

4. Hansel Chains

For a monotone boolean function, all the vectors can be placed in chains (sequences of vectors) according to monotonicity. The Hansel chains is a particular set of chains that can be formed using a dimensionally recursive algorithm [Hansel, 1966]. It starts with the single Hansel chain in E^1:

$$H^1=\{<0>, <1>\}.$$

To form the Hansel chains in E^2, 3 steps are required as follows:

Step 1: Attach the element "0" to the front of each vector in H^1 and get chain $C^{2\min}$.

$$C^{2\min}=\{<00>,<01>\}.$$

Step 2: Attach the element "1" to the front of each vector in H^1 and get chain $C^{2\max}$.

$$C^{2\max=}\{<10>,<11>\}.$$

Step 3: Move the last vector in $C^{2\max}$, i.e. vector <11>, to the end of $C^{2\min}$.

$$H^{2,1}=\{<00>, <01>, <11>\};$$

$$H^{2,2}=\{<10>\}.$$

To form the Hansel chains in E^3, the above 3 steps are repeated:

Step 1: $C^{3,1min}$={<000>, <001>, <011>};

$C^{3,2min}$={<010>}.

Step 2: $C^{3,1max}$={<100>, <101>, <111>};

$C^{3,2max}$={<110>}.

Step 3: $H^{3,1}$ ={<000>, <001>, <011>, <111>};

$H^{3,2}$ ={<100>, <101>};

$H^{3,3}$ ={<010>, <110>}.

Note that since there is only one vector in the $C^{3,2max}$ chain, it can be deleted after the vector <110> is moved to $C^{3,2min}$. This leaves the three chains listed in Table 1. In general, the Hansel chains in E^n can be generated recursively from the Hansel chains in E^{n-1} by following the 3 steps described above.

Table 1. Hansel chains for E^3.

Chain Number	Vector In-Chain Index	Vector
1	1	000
	2	001
	3	011
	4	111
2	1	100
	2	101
3	1	010
	2	110

A nice property of the Hansel chains is that all the vectors in a particular chain are arranged in increasing order. That is, if the vectors V_j and V_k are in the same chain then $V_j < V_k$ (i.e. V_j strictly precedes $V_{k)}$ when $j < k$). Therefore, if the underlying boolean function is isotone, then one can classify vectors within a chain easily. For example, if a vector V_j is negative (i.e., $f(V_j)$=0), then all the vectors preceding V_j in the same chain are also negative (i.e., $f(V_k) = 0$ for any $k < j$). Similarly, if a vector V_j is positive, then all the vectors succeeding V_j in the same chain are also positive. The monotone ordering of the vectors in Hansel chains motivates the composition of an efficient question-asking strategy discussed in the next section.

5. Devising a Smart Question-Asking Strategy

The most straight forward question-asking strategy, which uses Hansel chains, sequentially moves from chain to chain. Within each chain one may also sequentially select vectors to pose as questions. After an answer is given, the vectors (in other chains also) that are classified as a result of monotonicity are eliminated from further questioning. Once all the vectors have been eliminated, the underlying function is revealed. The maximum number of questions for this method, called the ***Sequential Hansel Chains Question-Asking Strategy***, will not exceed the upper limit $\varphi(n)$, given in the Hansel Theorem.

Although the sequential question-asking strategy is easy to implement and effective in reducing the total number of questions, there is still room for improvements. [Sokolov, 1982] introduced an algorithm that sequentially moves between the Hansel chains in decreasing size and performs a middle vector search of each chain. His algorithm does not require the generation (and storage) of all the Hansel chains since at each iteration it only requires a single chain. This advantage is obtained at the cost of asking more questions than needed.

In an entirely different approach, [Gainanov, 1984] presented a heuristic that has been used in numerous algorithms for inferring a monotone boolean function, such as in [Boros, *et. al.,* 1997] and in [Makino and Ibaraki, 1997]. This heuristic takes as input an unclassified vector and finds a border vector (maximal false or minimal true) by sequentially questioning neighboring vectors. The problem with most of the inference algorithms based on this heuristic is that they do not keep track of the vectors classified, only the resulting border vectors. Note that for an execution of this heuristic, all of the vectors questioned are not necessarily covered by the resulting border vector, implying that valuable information may be lost. In fact, several border vectors may be unveiled during a single execution of this heuristic is stored. Many of these methods are designed to solve large problems where it might be inefficient or even infeasible to store all of the information gained within the execution of the heuristic. However, these methods are not efficient (not even for small size problems), in terms of the number of queries they require.

One may look at each vector as carrying a "reward" value in terms of the number of other vectors

that will be classified concurrently. This reward value is a random variable that takes on one of two (one if the two values are the same) values depending on whether the vector is a positive or a negative example of the target function. The expected reward is somewhere between these two possible values. If one wishes to maximize the expected number of classified vectors at each step, the probabilities associated with each of these two values need to be computed in addition to the actual values. Finding the exact probabilities is hard, while finding the reward values is relatively simple for a small set of examples.

This is one of the underlying ideas for the new inference algorithm termed the ***Binary Search/Hansel Chains Question-Asking Strategy***. This method draws its motivation, for calculating and comparing the "reward" values for the middle vectors in each Hansel chain, from the widely used binary search algorithm (see, for instance, [Neapolitan and Naimipour, 1996]). Within a given chain, a binary search will dramatically reduce the number of questions (to the order of log_2 while the sequential search is linear). Once the "reward" values of all the middle vectors have been found, the most promising one will be posed as a question to the oracle. Because each vector has two values, selecting the most promising vector is subjective and several different evaluative criteria can be used.

The Binary Search/ Hansel Chains strategy can be divided into the following steps:

Step 1: Select the middle vector of the unclassified vectors in each Hansel chain.

Step 2: Calculate the reward values for each middle vector. That is, calculate the number of vectors that can be classified as positive (denoted as P) if it is positive and negative (denoted as N) if it is negative.

Step 3: Select the most promising middle vector, based on the (P, N) pairs of the middle vectors, and ask the oracle for its membership value.

Step 4: Based on the answer in Step 3, eliminate all the vectors that can be classified as a result of the previous answer and the property of monotonicity.

Step 5: Redefine the middle vectors in each chain as necessary.

Step 6: Unless all the vectors have been classified, go back to step 2.

The inference of a monotone boolean function on E^3 by using the Binary Search/ Hansel Chains strategy is illustrated below. The specifics of iteration 1, described below, are also shown in Table 2. At the beginning of iteration 1, the middle vectors in each Hansel chain (as described in Step 1) are selected and marked with the '<-- ' symbol in Table 2. Then, according to Step 2, the reward value for each one of these middle vectors is calculated. For instance, if <001>(the second vector in chain 1) has a function value of 1, then the three vectors <000>, <001> and <010> are also classified as positive. That is, the value of P for vector <001> equals 4. Similarly, <000> will be classified as 0 if <001> is classified as 0 and thus its reward value N equals 2.

Once the "reward" values of all the middle vectors have been evaluated, the most promising middle vector will be selected based on their (P, N) pairs. Here we choose the vector whose min(P, N) value is the largest among the ones of all middle vectors. If there is a tie, it will be broken randomly. Based on this evaluative criterion, vector 2 is chosen in chain 1 and is marked with "<-- " in the column "Selected middle vector with the largest Min(P, N)" After receiving the function value of 1 for vector <001>, its value is placed in the "answer" column. This answer is used to eliminate all of the vectors succeeding <001>. The middle vector in the remaining chains are updated as needed. At least one more iteration is required, as there still are unclassified vectors.

Table 2. Iteration 1.

Chain Number	Index of Vectors In the Chain	Vector	Vector Classified	Middle Vector in the Chain	Reward *P* if the Vector is Positive	Reward *N* if the Vector is Negative	Selected Middle Vector with the Largest Min(*P*,*N*)	Answer	Other Vectors Determined
1	1	000							
	2	001		<--	4	2	<--	1	
	3	011							1
	4	111							1
2	1	100		<--	4	2			
	2	101							1
3	1	010		<--	4	2			
	2	110							

Table 3. Iteration 2. The vector <100> is chosen and based on the answer, the class membership of the vectors <100> and <000> is determined.

Chain Number	Vector In-Chain Index	Vector	Vector Classified	Middle Vector in the Chain	Reward *P* if the Vector is Positive	Reward *N* if the Vector is Negative	Selected Middle Vector with the Largest Min(*P*,*N*)	Answer	Other Vectors Determined
1	1	000		<--	4	1			0
	2	001	1						
	3	011	1						
	4	111	1						
2	1	100		<--	2	2	<--	0	
	2	101	1						
3	1	010		<--	2	2			
	2	110							

At iteration 3 (as shown in Table 4), no unclassified vectors are left in chains 1 and 2, and the middle of these chains need not be considered anymore. Therefore, an "X" is placed in the column called "middle vector in the chain." At iteration 3 the vector <010> is chosen and the function value of the remaining two vectors <010> and <110> are determined. At this point all the vectors have been classified and the question-asking process stops.

Table 4. Iteration 3.

Chain Number	Vector In-Chain Index	Vector	Vector Classified	Middle Vector in the Chain	Reward *P* if the Vector is Positive	Reward *N* if the Vector is Negative	Selected Middle Vector with the Largest Min(*P*,*N*)	Answer	Other Vectors Determined
1	1	000	0						
	2	001	1	X					
	3	011	1						
	4	111	1						
2	1	100	0	X					
	2	101	1						
3	1	010		<--	2	1	<--	1	
	2	110							1

The algorithm posed a total of 3 questions in order to classify all the examples. The final classifications

listed in Table 5 correspond to the monotone boolean function $(x_1 \wedge x_3) \vee x_2 \vee x_3$.

Table 5. The Resulting Class Memberships.

Chain Number	Vector In-Chain Index	Vector	Function Value
1	1	100	0
	2	101	1
2	1	010	1
	2	110	1
3	1	000	0
	2	001	1
	3	011	1
	4	111	1

6. Conclusions

This paper described some approaches and some of the latest developments in the problem of inferring monotone boolean functions. As it has been described here, by using Hansel chains in the sequential question-asking strategy, the number of questions will not exceed the upper bound stated in the Hansel theorem. However, by combining the binary search strategy with the notion of Hansel chains, the number of questions asked can be further reduced. At present, the binary search/Hansel chains strategy is only applied to Hansel chains with a dimension of less than 10. However, it is expected that this method can be applied to infer monotone boolean functions of larger dimensions with slight modifications.

Acknowledgement

The authors are thankful for the partial support by the U.S. Navy (Office of Naval Research) grants N00014-95-1-0639 and N0001-97-1-0632 in carrying out this research.

References

[1] Alekseev, D. V. B. (1988): 'Monotone Boolean Functions', Encyclopedia of Mathematics, Kluwer Academic Publishers, Norwell, MA, Vol. 6, pp. 306-307.

[2] Bioch, J. C. and Ibaraki, T. (1995): 'Complexity of Identifixation and Dualization of Positive Boolean Functions', Information and Computation, Vol. 123, pp. 50-63.

[3] Boros, E., Hammer, P. L., Ibaraki, T, and Kawakami, K. (1997): 'Polynomial-Time Recognition of 2-Monotonic Positive Boolean Functions Given by an Oracle ', SIAM Journal on Computation, Vol. 26, No. 1, pp. 93-109.

[4] Church, R. (1940): 'Numerical Analysis of Certain Free Distributive Structures', Duke Math. J., Vol. 9, pp. 732-734.

[5] Church, R. (1965): 'Enumeration by Rank of the Elements of the Free Distributive Lattice with 7 Generators', American Mathematical Society Notices, Vol. 12, pp. 724.

[6] Dedekind, R. (1897): 'Ueber Zerlegungen von Zahlen durch ihre grössten gemeinsamen Teiler', Festchrift Hoch. Braunschweig u. ges. Werke, II, pp. 103-148.

[7] Eiter, T. and Gottlob, G. (1995): 'Identifying the Minimal Transversals of a Hybergraph and Related Problems ', SIAM Journal on Computation, Vol. 24, No. 6, pp. 1278-1304.

[8] Fredman, M. L and Khachiyan, L. (1996): '*On the Complexity of Dualization of Monotone Disjunctive Normal Forms*', Journal of Algorithms, Vol. 21, pp. 618-628.

[9] Gainanov, D. N. (1984): 'On One Criterion of the Optimality of an Algorithm for Evaluating Monotonic Boolean Functions', U.S.S.R. Comput. Maths. Math. Phys., Vol. 24, No. 4, pp. 176- 181.

[10] Gorbunov, Y. and Kovalerchuk, B. (1982): 'An Interactive Method of Monotone Boolean Function Restoration', J. Acad. Sci. USSR Eng., Vol.2, pp.3-6 (in Russian).

[11] Hansel, G. (1966): 'Sur le Nombre des Fonctions Boolenes Monotones den Variables', C.R. Acad. Sci. Paris, Vol.262, No.20, pp.1088-1090.

[12] Kleitman, D. (1969): 'On Dedekind's Problem: The number of Monotone Boolean Functions', Proceedings of the American Mathematical Society, Vol. 21, No. 3, pp. 677-682.

[13] Kleitman, D. and Markowsky, G. (1975): 'On Dedekind's Problem: The Number of Isotone Boolean Functions. II', Trans. Am. Math. Soc., Vol. 213, pp.373-390.

[14] Kovalerchuk, B., Triantaphyllou, E. and Vityaev, E. (1995): 'Monotone Boolean Functions Learning Techniques Integrated with User Interaction', Proceedings of the Workshop "Learning from Examples vs. Programming by Demonstrations", Tahoe City, CA, U.S.A., pp. 41-48.

[15] Kovalerchuk, B., Triantaphyllou, E., Deshpande, A. S. and Vityaev, E. (1996): 'Interactive Learning Of Monotone Boolean Functions', Information Sciences, Vol. 94, No. 1-4, pp.87-118.

[16] Korobkov, V. K. (1965): 'On Monotone Functions of the Algebra of Logic', Problemy Kibernetiki, Vol. 13, pp. 5-28 (in Russian).

[17] Makino, K. and Ibaraki, T. (1997): 'The Maximum Latency and Identification of Positive Boolean Functions', SIAM Journal on Computation, Vol. 26, No. 5, pp 1363-1383.

[18] Neapolitan, R. and Naimipour, K. (1996): 'Foundations of Algorithms', D. C. Heath and Company, Lexington, MA, U.S.A,.

[19] Peysakh, J. (1987): 'A Fast Algorithm to Convert Boolean Expressions into CNF', IBM Computer Science RC 12913 (#57971), Watson, NY, U.S.A.

[20] Rudeanu, S. (1974): 'Boolean Functions and Equations', North-Holland, Amsterdam, The Netherlands.

[21] Schneeweiss, W. G. (1989): 'Boolean Functions : with Engineering Applications and Computer Applications', Springer-Verlag, Berlin, Germany.

[22] Schneeweiss, W. G. (1996): 'A Necessary and Sufficient Criterion for the Monotonicity of Boolean Functions with Deterministic and Stochastic Applications', IEEE Transactions on Computers, Vol. 45, No. 11, pp. 1300-1302.

[23] Shmulevich, I. (1997): 'Properties and Applications of Monotone Boolean Functions and Stack Filters', Unpublished Ph.D. Thesis, Dept. of Electrical Engineering, Purdue University. This Ph.D. Thesis is downloadable from the web in PDF format (The URL is: http://shay.ecn.purdue.edu/~shmulevi/).

[24] Sokolov, N.A. (1982): 'On the Optimal Evaluation of Monotonic Boolean Functions', U.S.S.R. Comput. Maths. Math. Phys., Vol. 22, No. 2, pp. 207-220.

[25] Torvik, V. I., and Triantaphyllou, E. (1998): 'The Binary Search / Hansel Chains Approach for Interactive Inference of Monotone Boolean Functions ', Working Paper, Dept. of Industrial and Manuf. Syst. Engineering, 3128 CEBA Building, Louisiana State Univ., Baton Rouge, LA, 70803-6409, U.S.A.

[26] Triantaphyllou, E., and Lu, J. (1998), 'The Knowledge Acquisition Problem in Monotone Boolean Systems,' Encyclopedia of Computer Science & Technology, (A. Kent and J.G. Williams, Eds.), Marcel Dekker, Inc., New York, NY, in print.

[26] Ward, M. (1946): 'Note on the Order of Free Distributive Lattices', Abstract 135, Bulletin of the American Mathematical Society, Vol. 52, pp. 423.

[27] Wiedemann, D. (1991): 'A Computation of the Eight Dedekind Number', Order, Vol. 8, pp. 5-6.