Published in:
Encyclopedia of Computer Science & Technology, (A. Kent and J.G. Williams, Eds.), Marcel Dekker, Inc., New York, NY, Vol. 39, pp. 89-106, 1998.

The Knowledge Acquisition Problem in Monotone Boolean Systems[1]

Evangelos Triantaphyllou 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: trianta@lsu.edu

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

1. Introduction

Learning is the main property that characterizes an intelligent system. As similar situations will usually appear again and again in a given environment, the goal of learning is to gain experience that can be used to improve the performance of the system when similar situations occur in the future. Gaining experience often requires the system to build up an internal knowledge base that can efficiently fetch the information when needed.

The learning process can be divided into two phases: the knowledge acquisition phase and the rule generation phase. In the knowledge acquisition phase, it is relatively easy to gather a large amount of knowledge

[1]*The authors gratefully acknowledge the support from the Office of Naval Research (ONR) Grant N00014-95-1-0639.*
Corresponding Author: Evangelos Triantaphyllou

or facts. However, critical facts are usually difficult to collect or easy to neglect. Therefore, the time, manpower and other resources spent on knowledge acquisition can be very large if this process is not managed properly.

This article examines the knowledge acquisition problem for learning in a monotone boolean system. In such systems, it is assumed that all examples are represented by binary vectors in space E^n and each bit of a vector represents an attribute of the example. The attributes are assumed to be binary, i.e. to be either "True" or "False"(i.e., "0" or "1"). All examples are divided into two classes, and are thus regarded as positive and negative examples. The relation among the examples can be expressed in the form of a monotone boolean function, in which an example is regarded as a positive example when the function value for the example is 1 and as a negative example when the function value is 0. The goal of the knowledge acquisition phase in a monotone boolean system is to infer the function and thus be able to determine the class membership for all examples in the problem space.

2. Monotone Boolean Functions

To express the relation among the examples in the form of a monotone boolean function requires that the class membership of all examples be known. To determine the class membership of all examples is the same as to restore the underlying monotone boolean function, and thus this knowledge acquisition process is known as ***monotone boolean function inference***.

As discussed earlier, it could be very costly to determine the class membership of all examples in the problem space if the process is not arranged properly. For instance, to get the function value of all examples in space E^{10} could mean to test each one of the 1,024 examples in that space. Even if each test requires only 30 minutes to get the result, this process could be too slow to be acceptable in many practical situations. Furthermore, the considerable costs related with each test could be another reason that prevents the use of this kind of testing.

When the details of different kinds of tests are omitted, each example submitted for testing can be regarded as posing a question and the results come from the test can be regarded as getting an answer. Therefore, it is desirable to ask a sequence of appropriate questions, i.e. to test only a small number of examples from the problem space , so that the class membership of all examples in the space can be determined. The selection of the examples, or the question-asking strategy, is critical in reducing the number of questions.

When the relation among the examples is expressed as a general boolean function, there is no way to determine the class membership of other examples based on the classified examples (training set). Therefore every example should be examined if the relation among the examples (i.e., the inferred boolean function) is required to be 100% correct. This means that the size of the examples in the training set will always be 2^n, the same as the number of all the examples in space E^n. However, the number of questions can be reduced when the relation can be expressed as a ***monotone boolean function*** (to be discussed later). As the class membership of all examples in a monotone boolean system satisfy the monotone property, it is possible that a small number of classified examples can be used to determine the class membership of new examples. This, in turn, can significantly expedite the learning process and thus reduce the costs. Under monotonicity examples can be ordered 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 $x=<x_1, x_2, ..., x_n>$ ***precedes*** *vector $y=<y_1, y_2, ..., y_n>$ (denoted as x˜y) if and only if x_i#y_i for 1#i#n. If, at the same time, x...y then x* ***strictly precedes*** *y (denoted as x–y).*

According to this definition, the vectors in space E^2 can be ordered as follows:

$<11> \succ <01> \succ <00>$

and $<11> \succ <10> \succ <00>$.

However, the vectors <01> and <10> cannot be compared according to the above definition.

Based on the order of the vectors , an ***increasing monotone boolean function*** is defined as follows [Rudeanu, 1974] :

A boolean function f defined in space E^n is said to be an increasing (isotone) monotone boolean function if and only if for any vectors x, y O E^n, such that x˜y, then f(x)#f(y).

Similarly, a ***decreasing monotone boolean function*** is defined as follows [Rudeanu, 1974]:

A boolean function f defined in space E^n is said to be a decreasing (antitone) monotone boolean *function if and only if for any vectors x, y 0 E^n, such that x˜y, then f(x)$f(y).*

In a monotone boolean system, a function is either an increasing monotone boolean function or a decreasing monotone boolean function. However, as the method used to acquire the class memberships for the examples is the same for both cases, in this paper it is assumed that a hidden function is always an increasing monotone boolean function.

Monotonicity is a very strong constraint and, sometimes, cannot be easily satisfied. Fortunately, it can easily be proved that every general boolean function $q(x_1, ..., x_n)$ can be described in terms of several increasing $g_i(x_1, ..., x_n)$ and decreasing $h_i(x_1, ..., x_n)$ monotone boolean functions [Kovalerchuk, *et al.*, 1995]. That is:

$$q(x) = \bigvee_{i=1}^{m} (g_i(x) \bigwedge h_i(x)).$$

For the number $\psi(n)$ of monotone boolean functions defined on the vectors in space E^n, it is known ([Alekeseev, 1988] and[Kleitman, 1969]) that:

$$\Psi(n) = 2^{\binom{n}{\lfloor n/2 \rfloor}(1+\epsilon(n))},$$

where $0<\epsilon(n)<c(\log n)/n$, c is a constant and $\lfloor n/2 \rfloor$ is the largest integer less than or equal to $n/2$.

A boolean function can be of any form. All the forms are regarded as equivalent as long as they give the same correct true-false function values for all input boolean vectors. However, it is convenient to represent a boolean function in either the Conjunctive Normal Form (CNF) or the Disjunctive Normal Form (DNF) (see, for instance, [Blair *et al.*, 1985], [Cavalier *et al.*, 1990], [Hooker, 1988a and 1988b], [Jeroslow, 1988 and 1989], and [Williams, 1986]). Peysakh, 1987 describes an algorithm for converting any boolean expression into CNF. The CNF form can be described as follows:

$$\bigwedge_{j=1}^{k} (\bigvee_{i \in D_i} \alpha_i),$$

where a_i is either attribute A_i or its negation $\overline{A_i}$, j is the number of attribute combinations and D_j is the jth index set for the jth attribute combination. Similarly, DNF can be described as follows:

$$\bigvee_{j=1}^{k} \left(\bigwedge_{i \in D_i} \alpha_i \right).$$

3. Shannon Function and the Hansel Theorem

Suppose the relations among the binary examples in the problem space can be expressed with a monotone boolean function f. This function f can be obtained by classifying all vectors with the help of the appropriate operator A_f (also called an *oracle*) which, when fed with a vector $x=(x_1, x_2, x_3, ..., x_n)$, returns the class membership (or function value $f(x)$) of vector x. Let $A=\{F\}$ be the set of all algorithms which can be used to determine the class membership of all vectors in the space, and $\varphi(F, f)$ be the number of accesses to the operator A_f required to obtain the monotone boolean function $f \in M_n$ (where M_n is the set of all monotone boolean functions defined on n variables). Based on the above notation, 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 determine the class membership of all vectors and restore the underlying monotone boolean function is given by the following equation (also known as ***Hansel's theorem***) [Hansel, 1966]:

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

The significance of the Hansel theorem is that the total number of questions needed to infer any monotone boolean function defined by the relations among vectors in space E^n will not exceed $\varphi(n)$ if a proper question-asking strategy is applied.

[Kovalerchuk, *et al.*, 1996] proposed a method on how to classify the examples in a monotone boolean system by issuing a sequence of membership inquires to an operator or "oracle." That method is based on the concept of the ***Hansel chains*** and is optimal in the sense of the ***Hansel theorem*** and the ***Shannon function.***

4. Hansel Chains

A ***chain*** in space E^n is a sequence of binary vectors. All binary vectors in space E^n can be organized into several chains, which are called ***Hansel chains*** [Hansel, 1966]. For any two adjacent vectors x and y in a Hansel chain (where y follows x), the vector x is required to be different than vector y by only one bit so that vector x strictly precedes vector y.

The Hansel chains in space E^n can be generated recursively based on the Hansel chains in space E^{n-1}. Algorithm 1**,** as shown in Figure 4.1, is a modified version of the method proposed by [Hansel, 1966] to generate Hansel chains in space E^n.

```
Algorithm 1: Hansel Chains Generation in space $E^n$
Input:    Dimension $n$, Hansel chains of dimension $n$-1;
Output: Hansel chains of dimension $n$;

Note that the Hansel chains of space $E^{n-1}$ are assumed to be known and also
$H^{1,1}$={<0>, <1>}.

For each single chain $C$ of the Hansel Chains in space $E^{n-1}$ do the following:
Step 1: Form a new chain $C^{min}$ in space $E^n$ by attaching the element '0' to the right
        of each vector in chain $C$ ;
Step 2: Form a new chain $C^{max}$ in space $E^n$ by attaching the element '1' to the right
        of each vector in chain $C$ ;
Step 3: Move the last vector in chain $C^{max}$ to $C^{min}$ ;
Step 4: Add $C^{min}$ to the Hansel Chains of dimension $n$;
Step 5: If after removing the last vector form $C^{max}$ to $C^{min}$, $C^{max}$ is not empty, then
        add chain $C^{max}$ to the Hansel chains of dimension $n$;
The above 5 steps will be repeated until all chains in space $E^{n-1}$ have been
processed.
```

Figure 4.1 An Algorithm for the Generation of Hansel Chains in Space E^n.

For space E^1, there is only a single Hansel chain that consists of two vectors <0>, <1>. That is:

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

To form the Hansel chains for space E^2, there are 3 steps to be followed:

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

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

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

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

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

Now, the two Hansel chains in E^2 can be listed as follows:

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

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

To form the Hansel chains for space E^3, the previous 3 steps will be repeated. That is:

Step 1: Attach the element "0" to the front of each vector in $H^{2,1}$ and $H^{2,2}$ and get chains $C^{3,1\min}$ and $C^{3,2\min}$, respectively, as follows:

$C^{3,1\min}$ = {<000>, <001>, <011>},

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

Step 2: Attach the element "1" to the front of each vector in $H^{2,1}$ and $\mathrm{H}^{2,2}$ and get chains $C^{3,1\max}$ and $C^{3,2\max}$, respectively, as follows:

$C^{3,1\max}$ = {<100>, <101>, <111>},

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

Step 3: Move the last vector form $C^{3,1\max}$ and $C^{3,2\max}$ to the end of their counterpart $C^{3,1\min}$ and $C^{3,2\min}$, respectively, to form the Hansel chains in E^3 as follows:

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

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

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

Since there is only one vector in chain $C^{3,2\max}$, this chain can be deleted after the vector <110> is moved to $C^{3,2\min}$. So there are only three chains in the final set with Hansel chains, namely: $H^{3,1}$, $H^{3,2}$ and $H^{3,3}$. In general, the Hansel chains for space E^n can be generated recursively by repeating the 3 steps described above from the Hansel chains in space E^{n-1}. Table 4.1 lists the Hansel chains generated for space E^3 .

Table 4.1 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

5. The Sequential Hansel Chains Strategy

An interactive learning approach based on Hansel chains was proposed by [Kovalerchuk, *et al.*, 1996] and can significantly reduce the number of inquiries needed to determine a hidden monotone boolean function in space E^n. This interactive learning approach assumes that there is no example classified initially. By systematically choosing a set of vectors from the Hansel chains and by asking about their class memberships, all other vectors in the space can be classified. The algorithm proposed is optimal in the sense of the Shannon function and the Hansel theorem. The typical process of this interactive learning is:

(1) Generate the Hansel chains in space E^n.

(2) Sort the Hansel chains in increasing order of the size (i.e., the number of vectors) of the Hansel chains.

Start form the first Hansel chain and do the following:

(3) Start form the first unclassified vector in the chain and require the class membership of that

vector.

(4) Use the class membership of this classified vector to determine the class membership of as many undetermined vectors as possible.

(5) If all the vectors in the chain are determined, then process the next Hansel chain.

Steps (3), (4), (5) will be repeated until all chains have been processed.

As in step (3), the vectors are selected sequentially in each Hansel chain, the algorithm is therefore called a ***Sequential Hansel Chains Approach***. The above steps are described in detail in Figure 5.1.

Algorithm 2. Sequential Hansel Chains Question-Asking Approach

```
Input:   Dimension $n$;
Output:  Number of questions asked to determine the class membership of all the
         vectors for space $E^n$.

Step 0: {Hansel chain generation.}
    Use algorithm 1 to generate all Hansel Chains for space $E^n$.
    The number of Hansel chains is $K$.
Step 1: {Initialization.}
    Sort the Hansel chains in increasing order of the size of the chains.
    Set the Current Chain Pointer $CurChain$=1;
    Set the number of Questions NumQuestion=0;
Step 2: {Some vectors are unclassified.}
    Select the first unclassified vector $v$ in chain $C_i$, where $i=CurChain$;
Step 3: {Inquire the class membership of the vector.};
    Class($v$) =ANSWER($v$);
    NumQuestion=NumQuestion+1;
Step 4: {Use monotonicity property to mark other vectors.};
    For(each chain $C_j$ , $j$=1, 2, 3, ..., $K$) Do
        Mark the class membership of vectors in $C_j$ that can be
        determined;
Step 5: {Check for completion condition}
    IF( there are no unclassified vectors in the current chain) THEN
        IF ($CurChain=K$) THEN
            Output the class membership of all the vectors and the
            number of questions needed;
        ELSE {There are other chains not processed}
            $CurChain=CurChain$+1;
            Goto Step 2;
    ELSE { There are unclassified vectors in the current chain}
        Goto Step 2;
```

Figure 5.1 A Sequential Hansel Chains Question-Asking Approach.

The following example is a step-by-step demonstration of how the sequential Hansel chains approach can be used to determine all positive and negative examples in space E^3 and eventually form the underlying hidden monotone boolean function.

First, the Hansel chains for space E^3 are generated by using algorithm 1, as are listed in Table 4.1. In step 1, the previous Hansel chains are sorted in descending order of their size. Table 5.1 lists the sorted Hansel chains. The current chain pointer *CurChain*=1, indicates that the algorithm will begin to process from the first chain in that sequence.

Table 5.1 Sorted Hansel Chains.

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

After the Hansel chains are generated and sorted, steps (2), (3), (4) and (5) will be repeated until all vectors in space E^3 are classified.

Iteration 1:

Step 2: As *CurChain*=1 and no vector has been classified, the vector <100> is selected for testing.

Step 3: Suppose the result of the test indicates that the class membership value of vector <100> is 0 (i.e., false).

Step 4: Based on the monotone property of the hidden boolean function, the class membership of vector <100> indicates that vector <000> also has a class membership value of 0. Therefore, the vector <000> and vector <100> can be classified as negative.

Step 5: As there is an unclassified vector in chain 1, it is necessary to start another iteration and go back to step 2.

The specifics of iteration 1 are given in Table 5.2, in which the symbol "<--" indicates the vector selected in this iteration.

Table 5.2 Vectors Classified in Iteration 1.

Chain Number	Index of Vectors In the Chain	Vector	Vector membership	Selected Vector in the Iteration	Answer	Other Vectors Determined
1	1	100		<--	0	
	2	101				
2	1	010				
	2	110				
3	1	000				0
	2	001				
	3	011				
	4	111				

Iteration 2:

Step 2: As *CurChain*=1 and vector <101> has not been classified, it is selected for testing.

Step 3: Suppose the result of the test indicates that the membership value of vector <101> is 1.

Step 4: Based on the monotone property of the vectors, the class membership of vector <100> determines that vector <111> will also have a membership value of 1. Therefore vectors <101> and <111> can be classified as positive.

Step 5: There is no unclassified vector in chain 1. However, as *CurChain*=1<3, which indicates that not all the vectors have been classified, it is necessary to let *CurChain*=1+1=2 so that the next iteration will begin with the vectors in Hansel chain 2. The result after iteration 2 is listed in Table 5.3.

Table 5.3 Vectors Classified in Iteration 2.

Chain Number	Index of Vectors In the Chain	Vector	Vector membership	Selected Vector in the Iteration	Answer	Other Vectors Determined
1	1	100	0			
	2	101		<--	1	
2	1	010				
	2	110				
3	1	000	0			
	2	001				
	3	011				
	4	111				1

<u>**Iteration 3:**</u>

Step 2: As *CurChain*=2 and the first vector <010> in chain 2 has not been classified, it is selected for testing.

Step 3: Suppose the result of the test determines that the membership value of vector <010> is 1.

Step 4: Based on the monotone property of the vectors, the class membership of vector <010> will determine that vectors:

<110> and < 011>

will also have a membership value of 1 (another vector, vector <111>, has already been classified by vector <101> in iteration 2). Therefore, the 3 vectors:

<010>, <110> and <011>

can be classified as positive.

Step 5: There is no unclassified vector left in chain 2. However, as *CurChain*=2<3, which indicates that not all vectors have been classified, it is necessary to increase *CurChain* to 3 so that the next iteration will start form Hansel chain 3.

The details of iteration 3 are listed in Table 5.4.

Table 5.4 Vectors Classified in Iteration 3.

Chain Number	Index of Vectors In the Chain	Vector	Vector membership	Selected Vector in the Iteration	Answer	Other Vectors Determined
1	1	100	0			
	2	101	1			
2	1	010		<--	1	
	2	110				1
3	1	000	0			
	2	001				
	3	011				1
	4	111	1			

Iteration 4:

Step 2: As *CurChain*=3 and the first (and the only) vector has not been classified is <001>. Thus, it is chosen for testing.

Step 3: Suppose the result of the test determines that the membership value of vector <001> is 1.

Step 4: As vector <010> is the only vector left unclassified, it is classified in this iteration.

Step 5: There is no unclassified vector in chain 3 and *CurChain*=3. Therefore, all vectors in E^3 have been classified.

The number of questions needed to determine the class membership of all examples is 4, the same as the number of iterations. The class membership of all examples are listed in Table 5.6.

Table 5.5 Vectors Classified in Iteration 4.

Chain Number	Index of Vectors In the Chain	Vector	Vector membership	Selected Vector in the Iteration	Answer	Other Vectors Determined
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			

Table 5.6 The Class Membership of All Vectors in the Hansel Chains.

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

The hidden function is derived from tables 5.2 to 5.5 as follows. We look at each one of the vectors which have been classified as positive by the oracle. Note that there are three such vectors, namely vectors <101>, <010>, and <001>. Then, the attributes with value "1" in these vectors indicate the attributes present in the terms when the DNF format is used. Each such vector corresponds to one DNF term. Thus, from the above vectors we get the following inferred monotone boolean function:

$$f(x) =(x_1 \wedge x_3) \vee (x_2) \vee (x_3).$$

6. The Proposed Binary Search/Hansel Chains Strategy

The major advantage of the sequential Hansel chains approach is its conceptual simplicity. However, when the sequential Hansel chains approach is applied, the unclassified vectors in the Hansel chains are tested one by one. In this situation, the vectors are selected blindly and it is possible that some less effective vectors (as explained next) will be submitted for testing first.

One may notice that before a vector is selected for membership inquiry, a “reward” value of the vector selection can be some how evaluated. That is, one can know **at least** how many other vectors can be classified as positive or negative if this vector is classified as positive or negative, respectively. By comparing these “reward” values of all unclassified vectors, one can select a vector which, when asked, can give the maximum “reward” value. However, the computation will be very heavy if the “reward” value for each vector has to be calculated. An alternative approach is to calculate and compare the “reward” values of only the middle vector of the unclassified vectors in each Hansel chain $H^{n,i}$ (for i=1,...k, where k is the number of Hansel chains in space E^n)

and select the middle vector that appears to be the most promising.

We call this new method the ***Binary Search/Hansel Chains Strategy***. This method derives the main idea from the widely used binary search algorithm (see, for instance, [Neapolitan and Naimipour, 1995]). In summary, this binary search/ Hansel chains strategy consists of the following steps:

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

Step 2: Evaluate the "reward" value of each middle vector, i.e. the number of vectors that can be classified as positive (denoted as P) if the middle vector is positive and the number of vectors that can be classified as negative(denoted as N) if the middle vector is negative.

Step 3: Compare the (P, N) pair of all middle vectors, and then select the most promising middle vector. Next ask the membership value of that vector.

Step 4: Based on the previous answer, classify other vectors that can be determined as result of the previous answer and the monotonicity property.

Step 5: Redefine the middle vectors of each Hansel chains as necessary.

Step 6: Go back to step 2, unless all the vectors have been classified in which case exit.

The detailed description of this algorithm is shown in Figure 6.1.

```
Algorithm 3: Binary Search/ Hansel Chains Question-Asking  Approach

Input:     Dimension $n$;

Output:  Number of questions asked to determine the class of all the vectors  in space $E^n$.


Step 0:   {Initialization phase.}
        Use Algorithm 1 to form all Hansel chains in space $E^{n,}$, the number of chains is $K$.
        Let the $j$-th chain, denoted as $C_j$ (for $j$ = 1, 2, 3, ..., $K$), be comprised by the sequence  of vectors:
            $V_{j\,1}$, ..., $V_{j\,2\,,}$ ..., $V_{j\,m}$,  for $j$ = 1, 2, 3,..., $K$;
        Initialize  the upper and lower borders, $U_j$ and $L_j$, respectively, in each  chain as  follows:
            $U_j = V_{j\,1}$;   and   $L_j = V_{j\,m}$ ,      where $m$  is the total number of vectors in Hansel chain $C_j$.
Step 1:  {Some vectors are still unclassified.}
        For (each chain $C_j$ ,  $j$ = 1, 2,  3, ..., $K$) Do
            IF(Hansel chain $C_j$ has some unclassified vectors) THEN
                Get $M_j$  ($j = \lfloor (U_j - L_j)/2 \rfloor$),  the "middle " vector of the sequence  of  unclassified vectors
                                  in chain $C_j$;
                Calculate:
                    POS($M_j$ ) = Number of unclassified vectors which would be classified as
                                  positive if $M_j$ were a positive  example;
                    NEG($M_j$) = Number of unclassified vectors which would be  classified to
                                  negative if $M_j$ were a negative  example;
Step 2:  {Inquire  the value of the most "promising " unclassified example.}
        Select  the most "promising" vector $M_j$ according to a criterion;
        $M_y = M_j$ ;
        Inquire the value of $M_y$ ;
        Set  NumQuestion = NumQuestion + 1;
Step 3:   {Use  monotonicity  property to mark other unclassified vectors and update the  lower and
    upper boundary of unclassified vectors in each chain.}
        For (each chain $C_j$,  $j$ = 1, 2, 3, ..., $K$)
            Mark  the class membership of vectors in $C_j$  that can be determined;
            IF ($M_y$ is positive) THEN
                    Update $U_j$ to exclude classified vectors;
            ELSE {$M_y$  is negative}
                    Update $L_j$ to exclude classified vectors;
Step 4:   {Check for completion condition.}
        IF (no Hansel chain remains with unclassified vectors) THEN
            Output the class membership of all  vectors and the number of question needed;
        ELSE Go to Step 1;
```

Figure 6.1 The Binary Search/ Hansel Chains Question-Asking Approach.

7. An Illustrative Example

To illustrate this binary search/ Hansel chains question-asking strategy, an example is given for space E^3, in which there is a total of 8 vectors. The Hansel chains are constructed as shown in Table 4.1. The underlying monotone boolean function is the same as the one is used in section 5. At each iteration, a vector is selected as a question posed by the binary search/Hansel chains strategy.

At the beginning of iteration 1, the middle vector of each Hansel chain (as described in step 1, above) is selected and marked with the '<-- ' symbol in the table. Then, according to step (2), the "reward" value for each one of these middle vectors is calculated. For instance, if the second vector in chain 1 has a function value of 1, then there will be three other vectors (i.e., the vectors <000>, <001> and <010>) which can also be classified as positive since the inferred boolean function is assumed to be *monotone*. Therefore, the total number of vectors that can be calculated to have a function value of 1 is P= 4, which is put under the entry " 'reward' value P if the middle vector is positive" of vector <001>. Similarly, if it is calculated that the function value is 0, the "reward" value for the vector <001> will be N= 2 and hence this value is put in the entry " 'reward' value N if the middle vector is negative."

Once the "reward" values of all middle vectors have been evaluated, the most promising middle vector will be selected and its function value will be asked. Several selection criteria can be used to compare the (P, N) pairs of each middle vector and select the most promising vector. The one that is used here is to compare the smaller one of the (P, N) values (i.e., to determine min(P, N)) of each vector and select the vector whose min(P, N) is the <u>**largest**</u> among all middle vectors. If the number of such vectors is more than one, then the tie will be broken randomly. Based on this criterion, vector 2 is chosen in chain 1 and marked with the "<-- " symbol in its corresponding entry under the column "Selected middle vector with the largest Min(P,N)."

After getting the function value for vector <001>, which is assumed to be 1 in this case, this value is put in the entry "answer." Then this answer will be used to classify the vectors whose class membership can be determined by this answer and the monotonicity property. The middle vector of each Hansel chain will be updated as needed. The details of this iteration are shown in Table 7.1. Since there are still undetermined vectors, at least one more iteration is required.

Table 7.1 Details for Iteration 1.

Chain No.	Index of Vectors In the Chain	Vector	Vector membership	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							

At iteration 2, the vector <100> is chosen in a similar manner and, based on the answer, the class membership of the vectors <100> and <000> is determined. This iteration is shown in detail in Table 7.2.

Table 7.2 Details for Iteration 2.

Chain No.	Vector in the Chain	Vector	Vector membership	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 (Table 7.3), as there is no unclassified vector left in Hansel chain 1 and Hansel chain 2, the middle vectors of these two chains do not need to be considered anymore. Therefore an "X" is marked for each of the two chains in the column "middle vector in the chain." At iteration 3 the vector <010> is chosen and the remaining two vectors, <010> and <110> are determined. At this point, the class membership of all vectors has been determined and thus the question-asking process stops.

In similar manner as with the function inferred at the end of section 5, the new function is from tables 6.1, 6.2, and 6.3:

$$f(x)=(x_2) \vee (x_3).$$

It is easy to confirm that these two functions are equivalent, although the second one is much simpler.

Table 7.3 Details for Iteration 3.

Chain No.	Vector in the Chain	Vector	Vector membership	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

By using the binary search/ Hansel chains question-asking strategy, the number of questions is able to be reduced to 3 from the previous 4 needed by the sequential Hansel chains approach. Although the difference between 3 and 4 queries is not significant, in some test problems reported in [Lu and Triantaphyllou, 1997] indicate that the new strategy requires on the average 50% queries less than the existing sequential Hansel chains approach. The illustrative examples in this paper simply demonstrate the implementation of the proposed binary search/Hansel chains strategy.

The basic idea behind the binary search strategy is to select the most "promising" vector among all unclassified vectors in each iteration and submit it for testing. The selection of the most "promising " vector is based on the intuitive notion that, once the selected vector is tested and classified, there will be more vectors that can be determined based on the testing results. In the above example, when the binary search/ Hansel chains approach is used, the vectors submitted for testing are:

{ <001>, <100>, <010>}.

In the example of section 4, when the sequential Hansel chains approach is used, the vectors submitted were:

{ <100>, <101>, <001>, <010>},

in which vector <101> is not as effective as the other vectors.

8. Conclusions

This paper discussed the knowledge acquisition problem in monotone boolean systems. One of the main issues related to knowledge acquisition in such monotone boolean systems is how to reduce the number of inquiries needed to classify all vectors in the problem space.

As it has been discussed above, by using Hansel chains in the sequential question-asking strategy [Kovalerchuk, *et al.*, 1997], the number of possible questions will not exceed an upper bound as stated in the Hansel theorem. However, the performance of sequential question-asking strategy depends on the sequence of the Hansel chains and it may change dramatically when it is applied to different problems. Therefore, a new guided vector selection approach; the binary search/Hansel chains approach is proposed to address this problem. When this new method is applied, the average number of inquiries can be further reduced and the performance of the method is relatively consistent compared to that of the sequential question-asking strategy. We are currently working in this area and more recent updates can be found in our web home page with URL: http://www.imse.lsu.edu/vangelis.

References

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

[2] C. E. Blair, R. G. Jeroslow, and J, K. Lowe (1985), "Some Results and Experiments in Programing Techniques for Propositional Logic," *Computers and Operations Research*, No. 13, pp. 63-645.

[3] T. M. Cavalier, P.M. Pardalos, and A. L. Soyster (1990), "Modeling and Integer Programming Techniques Applied to Propositional Calculus", *Computers and Operations Research*, Vol. 17, No.6, pp.561-570.

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

[5] J. N. Hooker (1988a), "Generalized Resolution and Cutting Planes," *Annals of Operations Research*, Vol. 12, No.1-4, pp.217-239.

[6] J. N. Hooker (1988b), "A Method of Producing a Boolean Function Having an Arbitrarily Prescribed Prime Implicant Table," *IEEE Trans. On Computers*, Vol. 14, pp. 485-488.

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

[8] R. G. Jeroslow (1988), "*Logic-Based Decision Support*," North-Holland, Amsterdam, The Netherlands.

[9] D. Kleitman (1969), "On Dedekind's Problem: The Number of Monotone Boolean Functions," *Proc. Am. Math. Soc.*, Vol. 21, pp.677-682.

[10] V.K. Korobkov, (1965), " On Monotone Boolean Functions of Algebra Logic," *Problemy*, Nauka, Moscow, Russia (former USSR), Vol. 13, pp. 5-28 (in Russian).

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

[12] B. Kovalerchuk, E. Triantaphyllou, A. S. Deshpande and E. Vityaev (1996), " Interactive Learning Of

Monotone Boolean Functions," *Information Sciences*, Vo. 94, No. 1-4, pp.87-118.

[13] R. Neapolitan and K. Naimipour (1995), "*Foundations of Algorithms*," D. C. Heath and Company.

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

[15] S. Rudeanu (1974), "*Boolean Functions and Equations*," North-Holland, Amsterdam, The Netherlands.

[16] H. P. Williams (1986), " Linear and Integer Programming Applied to Artificial Intelligence, " *Preprint Series, University of Southampton, Faculty of Mathematical Studies*, pp.1-33, Southampton, UK.