R U T C O R
R ESEARCH
R E P O R T

BOOLEAN SEPARATORS AND APPROXIMATE BOOLEAN CLASSIFIERS

Peter L. Hammer[a] Irina I. Lozina[b]

RRR 14-2006, JUNE 2006

RUTCOR
Rutgers Center for
Operations Research
Rutgers University
640 Bartholomew Road
Piscataway, New Jersey
08854-8003
Telephone: 732-445-3804
Telefax: 732-445-5472
Email: rrr@rutcor.rutgers.edu
http://rutcor.rutgers.edu/~rrr

[a] RUTCOR, Rutgers University, 640 Bartholomew Rd., Piscataway, NJ 08854-8003, hammer@rutcor.rutgers.edu
[b] RUTCOR, Rutgers University, 640 Bartholomew Rd., Piscataway, NJ 08854-9003, ilozina@rutcor.rutgers.edu

RUTCOR RESEARCH REPORT
RRR 14-2006, JUNE 2006

BOOLEAN SEPARATORS AND APPROXIMATE BOOLEAN CLASSIFIERS

Peter L. Hammer Irina I. Lozina

Abstract. A simple technique is proposed for associating to a binary dataset a set of synthetic variables (called *Boolean separators*) , some of which – if used either alone or in conjunction with the original variables – can enhance the accuracies of various frequently used machine learning / data mining methods. An iterative application of this technique is proposed for the generation of *approximate Boolean classifiers,* which are shown to increase the accuracy of each of the examined classification methods on each of the examined benchmark dataset.

.

Acknowledgements: We gratefully acknowledge the partial support provided by the National Science Foundation (grant number NSF-IIS-0312953) and the National Institutes of Health (grants number NIH-HL-072771-01 and number NIH-DK-067468-01). The second author also acknowledges DIMACS' partial support based on NSF grant number NSF-CCF-04-32013.

1 Introduction

A central topic of machine learning / data mining is that of analyzing binary datasets; a *binary dataset* Ω consists of a subset of n-vectors with binary $\{0,1\}$ components, each of which has an associated binary *outcome*. Clearly the set of n-vectors in Ω along with their outcomes represent a *partially defined Boolean function*. The problem consists in finding an "*extension*" of the partially defined Boolean function (i.e. a Boolean function which is defined in every binary n-vector, and which agrees in Ω with the given values) closely approximating a hidden ("*target*") function.

In this paper, the i-th components of all the vectors in Ω will be viewed as the values of a *variable* x_i; frequently variables are also called *attributes* or *features*. The n-vectors of Ω are called *observations*, while those whose outcome is 1 (respectively 0) are called *positive* (respectively *negative*) *observations*. The sets of all positive and negative observations in Ω are denoted by Ω^+ and Ω^-, respectively.

There is a rich literature in machine learning / data mining dedicated to feature selection procedures, i.e. techniques for identifying and eliminating unnecessary variables included in datasets. This is an extremely important area of research, since the number of irrelevant and/or redundant variables present in datasets appearing in real-life problems is usual very large, and their presence does not only slow down the computational aspects of data analysis, but can also introduce inaccuracies and errors.

An exactly opposite point of view has been adapted in [4], where beside the given variables, additional "artificial" variables have been introduced and added to the given ones, in order to increase the accuracy of classification. The artificial variables proposed in [4] were associated to pairs of given variables, using simple arithmetic operations; to a pair of binary variables x, y it was suggested to associate new variables of the form $a+bx+cy$, where a, b and c were real numbers, chosen in such a way that the artificial variable contributed to the separation of positive observations from negative ones. It was shown in the same paper that the introduction of artificial variables can enhance substantially the accuracy of classifications.

In this paper we shall examine the possibility and the advantages of a systematic approach for creating two types of new Boolean variables. On the one hand we shall create artificial Boolean variables (to be called *Boolean separators*), which – if added to the original variables – enhance the accuracy of the most frequently used classification methods. This goal will be achieved by associating to *every* pair of binary (0,1) variables x and y, 16 new binary variables $f_1(x,y)$, …, $f_{16}(x,y)$, where $f_i(x,y)$ (i = 1,…,16) are *all* the Boolean functions depending on two variables; a filtering procedure will be used to limit the number of variables generated in this way. On the other hand, by iterating the above procedure until no additional new variables can be created, we shall obtain some Boolean functions (to be called *Approximate Boolean Classifiers*, or *ABC*s) which depend on the original variables, and which have the property that, taken individually, each one of them can be used as a new classification system, whose accuracy will be shown to be

comparable, and usually higher, than that of most frequently used machine learning techniques. While the basic idea of using Boolean functions for producing artificial variables has already been considered in the literature (see e.g. [7], [8], and [9] for a survey), the particular approach proposed in this paper for the construction of *Approximate Boolean Classifiers* of partially defined Boolean functions will be shown to allow the efficient generation of novel, accurate classification techniques.

Given a Boolean function y depending on a subset of the Boolean variables $x_1, x_2, \ldots, x_n$ in the dataset, the *classification power of y, CP(y)*, is defined in the following way: if $\pi(y)$ denotes the number of positive observations for which the value of y is 1, and $\upsilon(y)$ denotes the number of negative observations for which the value of y is 0, then

$$CP(y) = \frac{1}{2}\left(\frac{\pi(y)}{|\Omega^+|} + \frac{\upsilon(y)}{|\Omega^-|}\right)$$

The *classification power* of the negation $\bar{y} = 1 - y$ of a variable y will be called *Co-CP(y),* clearly, *Co-CP(y)* = 1 - *CP(y)*.

After defining a simple procedure for generating Boolean separators, and an iterative algorithm for generating *ABC*s, we shall demonstrate the usefulness of these concepts by applying them to a number of publicly available datasets and to increase in this way the accuracy of several frequently used classification methods.

2 Boolean separators as artificial variables

In order to present in detail the procedure of generating Boolean separators we shall define the *negation* $\bar{x}$ of a binary $\{0,1\}$ variable x as $1 - x$, and define for any two binary variables x_i and x_j, their *disjunction* $x_i \vee x_j = x_i + x_j - x_i x_j$, their *conjunction* $x_i \,\&\, x_j$, defined as their product $x_i x_j$ (and denoted simply as $x_i x_j$), and their *sum modulo 2* as $x_i \oplus x_j = x_i + x_j - 2 x_i x_j$. Note that treating the 0,1 values of the Boolean variables as the numbers 0,1 (i.e. not as symbols) allows the definition of arithmetic operations with them, and does not lead to any confusion.

In order to identify the Boolean separators associated to a dataset we shall apply the following straightforward enumerative procedure:

(i) for every pair of Boolean variables x_i, x_j we shall generate all the 16 Boolean functions $y_k(x_i, x_j)$:

$$1,\ 0,\ x_i,\ \bar{x}_i,\ x_j,\ \bar{x}_j,\ x_i \vee x_j,\ x_i x_j,\ x_i \vee \bar{x}_j,\ x_i \bar{x}_j,\ \bar{x}_i \vee x_j,\ \bar{x}_i x_j,\ \bar{x}_i \vee \bar{x}_j,\ \bar{x}_i\ \bar{x}_j,\ x_i \oplus x_j,\ \overline{x_i \oplus x_j}$$

depending of them;

(ii) calculate the *CP* of each $y_k(x_i, x_j)$ and retain only those functions whose *CP* exceeds the maximum of $CP(x_i)$ for all $i = 1,\ldots, n$.

In fact we do not actually have to calculate all the 16 functions enumerated above; indeed, it is sufficient to list only the 5 functions

$$x_i x_j,\ x_i \vee x_j,\ x_i \bar{x}_j,\ x_i \vee \bar{x}_j,\ x_i \oplus x_j.$$

This simplification is possible because the other functions are either negations of these five ($\bar{x}_i \vee \bar{x}_j$, $\bar{x}_i\,\bar{x}_j$, $\bar{x}_i \vee x_j$, $\bar{x}_i x_j$, $\overline{x_i \oplus x_j}$), or constant functions (1 or 0), or represent the original variables (x_i or x_j) or their negations ($\bar{x}_i$ or $\bar{x}_j$), and because the *CP* of the negation of a function is available if the CP of the function itself is known:

$$CP(\overline{y_k(x_i, x_j)}) = |\Omega| - CP(y_k(x_i, x_j))$$

Example. Let us consider a dataset containing 3 negative observations (A,B,C) (the "class" of these is labeled 0) and 3 positive observations (D, E, F) (the "class" of these is labeled 1), described in terms of 4 binary variables (x_1, x_2, x_3, x_4):

Obs.	x_1	x_2	x_3	x_4	class
A	0	1	0	0	0
B	1	1	1	0	0
C	0	0	0	1	0
D	1	0	1	0	1
E	1	0	0	0	1
F	0	0	1	1	1

We shall examine now the Boolean separators depending on the $\binom{4}{2}$ = 6 possible pairs of original variables. As mentioned above, for each pair of variables we shall list only 5 of the 16 Boolean functions depending on these 2 variables. For example, for the pair x_1, x_2 we shall construct the following functions:

Obs.	$x_1\,x_2$	$x_1 \vee x_2$	$x_1\,\bar{x}_2$	$x_1 \vee \bar{x}_2$	$x_1 \oplus x_2$
A	0	1	0	0	1
B	1	1	0	1	0
C	0	0	0	1	0
D	0	1	1	1	1
E	0	1	1	1	1
F	0	0	0	1	0
CP	2/6	1/2	5/6	4/6	4/6
Co-CP	4/6	1/2	1/6	2/6	2/6

In the line called *CP* we indicate the classification power of each of the 5 functions above, and in the line *Co-CP* we indicate the classification power of the complement of the function in the respective column. For example, the *CP* of the function x_1x_2 is 2/6 (since this function agrees with the outcome in the observations A and C), and the *Co-CP* of the function is 4/6 (since the

complement of this function agrees with the outcome in the observations B, D, E and F). Similar tables for all the other pairs of variables have been constructed.

Since the largest value of *CP* or *Co-CP* corresponding to the variables $x_1,\ldots, x_4$ is 4/6, we shall retain only those Boolean separators which have a *CP* or a *Co-CP* of 5/6 or higher; the retained columns are the following:

$$x_1\bar{x}_2,\ \ x_1 \vee x_3,\ x_1 \oplus x_3,\ x_2 \vee \bar{x}_3,\ x_2 \vee x_4,\ x_2\ \bar{x}_4,\ x_2 \oplus x_4.$$

Obs.	x_1	x_2	x_3	x_4	$x_1\bar{x}_2$	$x_1 \vee x_3$	$x_1 \oplus x_3$	$x_2 \vee \bar{x}_3$	$x_2 \vee x_4$	$x_2\bar{x}_4$	$x_2 \oplus x_4$
A	0	1	0	0	0	0	0	1	1	1	1
B	1	1	1	0	0	1	0	1	1	1	1
C	0	0	0	1	0	0	0	1	1	0	1
D	1	0	1	0	1	1	0	0	0	0	0
E	1	0	0	0	1	1	1	1	0	0	0
F	0	0	1	1	0	1	1	0	1	0	1
CP	4/6	1/6	4/6	1/2	5/6	5/6	5/6	1/6	1/6	1/6	1/6
Co-CP	2/6	5/6	2/6	1/2	1/6	1/6	1/6	5/6	5/6	5/6	5/6

It can be seen that the Boolean separator $x_2\bar{x}_4$ takes the same values as x_2 in each of the 6 observations, therefore it can be eliminated from the table. Similarly, both $x_2 \vee x_4$ and $x_2 \oplus x_4$ take complementary values to those of $x_1\bar{x}_2$, and therefore it is enough to retain one (say, $x_1\bar{x}_2$) of these 3 Boolean separators. The set of original variables and retained Boolean separators (to be denoted by x_5, x_6, x_7, and x_8) becomes

Obs.	x_1	x_2	x_3	x_4	$x_5 = x_1\bar{x}_2$	$x_6 = x_1 \vee x_3$	$x_7 = x_1 \oplus x_3$	$x_8 = x_2 \vee \bar{x}_3$	class
A	0	1	0	0	0	0	0	1	0
B	1	1	1	0	0	1	0	1	0
C	0	0	0	1	0	0	0	1	0
D	1	0	1	0	1	1	0	0	1
E	1	0	0	0	1	1	1	1	1
F	0	0	1	1	0	1	1	0	1
CP	4/6	1/6	4/6	1/2	5/6	5/6	5/6	1/6	
Co-CP	2/6	5/6	2/6	1/2	1/6	1/6	1/6	5/6	

3 Iterative procedure for finding Approximate Boolean Classifiers (*ABC*s)

The Boolean separators identified in the process described in section 2 can be regarded as synthetic variables associated to the dataset. As such, they can be simply added to the original data, and the process described in Section 2 for Boolean separators can now be repeated on the augmented dataset. Moreover, the resulting Boolean separators can again be added to the new

dataset, and the process can be repeated again. The process will stop when none of the *CP*s or *Co-CP*s of the Boolean separators created in a certain step exceed the highest values of *CP*s or *Co-CP*s found in all previous iterations.

In order to keep the size of the dataset within reasonable limits, at each step of this iterative process, after generating the new Boolean separators, we shall carry out a simplification step. We say that a variable x *dominates* a variable y if for every observation i the following implication holds: "if the value of y in i coincides with the outcome (class) of i, then the value of x in i does also coincide with the outcome (class) of i ". In the simplification step, which follows in every iteration the generation of Boolean separators, we eliminate from the augmented dataset those old and new variables which are "dominated" by another one.

We shall denote by μ the maximum of all *CP* and *Co-CP* values associated with all the variables appearing in the last dataset generated in this process, i.e. in the dataset from which no additional new Boolean separators can be generated. Those Boolean separators whose *CP* or *Co-CP* equals μ represent the *Approximate Boolean Classifiers* (*ABC*s) of the original dataset which are generated by the proposed algorithm.

It is important to note that it is not necessarily true that there exists an *ABC* whose values coincide with the correct classification of all the observations in the dataset. Our experience shows however that in every example we have studied, several *ABC*s were found which took the same values as the outcome of "almost all" observations.

Example (*continued*). Let us repeat now the procedure for generating Boolean separators, with $x_1, \ldots, x_8$ playing the role of original variables. Adding now to this table the 4 new Boolean separators, $x_2 \vee \bar{x}_6$, $x_5 \vee x_7$, $x_5 \vee \bar{x}_8$, $\bar{x}_7 x_8$ found in this iterative step and having *CP* or *Co-CP* values exceeding 5/6, and eliminating the dominated variables $x_1, \ldots, x_8$ we end up with the final table

Obs.	$f_1 = x_2 \vee \bar{x}_6$	$f_2 = x_5 \vee x_7$	$f_3 = x_5 \vee \bar{x}_8$	$f_4 = \bar{x}_7 x_8$	class
A	1	0	0	1	0
B	1	0	0	1	0
C	1	0	0	1	0
D	0	1	1	0	1
E	0	1	1	0	1
F	0	1	1	0	1
CP	0	1	1	0	
Co-CP	1	0	0	1	

In conclusion, we have found two functions (f_2 and f_3) which take exactly the same values as the class, and two additional functions (f_1 and f_4) which take exactly the same values as the complement of the class; clearly f_2 and f_3, as well as the negations of f_1 and f_4 are *ABC*s. Substituting in the expressions of these *ABC*s, the expressions of $x_5, \ldots, x_8$ as functions of the original variables $x_1, \ldots, x_4$, we find that:

$$\overline{f_1} = \bar{x}_2 (x_1 \vee x_3),$$
$$f_2 = (x_1\ \bar{x}_2) \vee (x_1 \oplus x_3),$$
$$f_3 = (x_1\ \bar{x}_2) \vee (\bar{x}_2 x_3),$$
$$\overline{f_4} = (x_1 \oplus x_3) \vee (\bar{x}_2 x_3).$$

4 Computational experiments: datasets and classification methods

We have applied the methods proposed in the previous sections for the generation of Boolean separators and *ABC*s to several datasets available on the web in the *Repository of the University of California at Irvine (http://www.ics.uci.edu/~mlearn/ MLRepository.html)*. We shall present below the datasets analyzed in these experiments, and the results of classifying these datasets with help of several frequently used classification methods.

4.1 Binarization.

In order to apply our techniques, we had first to "binarize" the data, i.e. to replace each variable taking numerical values by one or more binary variables, following the procedure in [1]. The procedure consists in associating to each numerical variable x one or more *cutpoints* c', c'', … , and associating then to each of these cutpoints a binary variable x', x'', …, defined by

$$x' = \begin{cases} 1 & if \quad x > c' \\ 0 & otherwise \end{cases}, \quad x'' = \begin{cases} 1 & if \quad x > c'' \\ 0 & otherwise \end{cases}, \quad \ldots\ldots\ldots$$

The binarization process is correct if and only if the binary (0, 1) vector representing the image of any positive observation is different from the binary image of any negative observation. It has been shown in [1] that the minimization of the number of binary variables allowing the correct binarization of a given dataset can be accomplished by solving a set-covering problem.

4.2 Datasets.

The datasets examined in this study along with their main characteristics are described in Table 1. The list of cutpoints and binarized variables used in this study are presented in Tables A, … , H in the Appendix I. Each table indicates the definition of the binary variables used in this study. For example, the first line of Table A indicates that

$$a_1 = \begin{cases} 1 & if \quad mcv > 87 \\ 0 & otherwise \end{cases}$$

Table 1. *Benchmark datasets*

Name of dataset	Abbreviation	Number of observations		Number of attributes	
		Positive	Negative	Given	Binarized
BUPA liver-disorders	**bld**	200	145	6	29
German credit	**ger**	700	300	24	57
Pima Indians Diabetes	**pid**	130	262	8	23
Cleveland heart disease	**hea**	137	160	13	17
Australian credit	**aus**	307	383	14	45
Ionosphere	**ion**	225	126	33	71
Wisconsin breast cancer	**bcw**	236	213	9	20
Congressional voting records	**vot**	124	108	16	16

Note that we have eliminated from the study the observations which include missing data, and in the case of the **bcw** dataset which contains many repetitions of some of the observations, we have retained only one copy of each observation.

4.3 Classification methods.

In the computational experiments aimed at evaluating the usefulness of Boolean separators we have used on the one hand the Logical Analysis of Data (*LAD*) methodology (see [3], [5], or the surveys [2] or [6]), and on the other hand four of the most frequently applied data-mining procedures [artificial neural networks (*Multilayer Perceptron*), linear logistic regression classifier (*Simple Logistic*), support vector machine classifier (*SMO*), decision trees (*J48*)]. The software used for *LAD* was Datascope (*http://rutcor.rutgers.edu/~salexe/LAD_kit/SETUP-LAD-DS-SE20.zip*), while for the other four methods we used the WEKA package (*http: //www.cs.waikato.ac.nz/~ml/ weka/index.html*).

4.4 Accuracy of classification.

In Table 2 we report the average classification accuracies obtained by applying the five methods mentioned above to the 8 datasets in Table 1. The averages refer to the results of 20 ten-folding experiments, using the original variables.

Table 2

	SMO	*MultilayerPerceptron*	*Simple_Logistic*	*J48*	*LAD*	*Average*
bld	50.03%	67.63%	66.24%	63.65%	69.29%	63.37%
ger	69.39%	65.50%	68.56%	66.18%	72.21%	68.37%
pid	72.31%	73.81%	72.07%	76.13%	74.60%	73.78%
hea	83.05%	78.70%	82.48%	77.16%	82.35%	80.75%
aus	86.47%	82.98%	86.66%	84.93%	85.57%	85.32%
ion	91.10%	88.78%	85.08%	88.05%	91.58%	88.92%
bcw	95.28%	94.39%	94.86%	92.78%	94.44%	94.35%
vot	97.05%	94.42%	96.49%	96.61%	97.14%	96.34%

5 Classification results for benchmark datasets using Boolean separators

Applying the algorithm described in section 2 for the generation of Boolean separators in the 8 datasets described above produces new variables which – at least in some of the cases – have remarkably high *CPs* or *Co-CPs*. For example in the case of **vot** the "best" Boolean separator is $\overline{a}_4 \vee \overline{a}_{14}$, having a *CP* of 96.19%. The list of the Boolean separators having the highest *CPs* for the 8 datasets considered is given in Table 3.

Table 3

Dataset	Boolean separator	*CP(%)*
bld	$\overline{a}_{11} \vee a_{25}$	66.32%
ger	$a_2 \vee a_{20}$	69.29%
pid	$a_3 \vee a_4$	76.44%
pid	$a_4 \vee a_{18}$	76.44%
pid	$a_4\, a_{22}$	76.40%
hea	$a_5\, a_{17}$	77.50%
hea	$\overline{a}_{11} \vee a_{17}$	77.50%
hea	$a_{16} \vee a_{17}$	77.50%
aus	$a_{27}\, a_{35}$	86.01%
ion	$a_1\, a_6$	92.13%
bcw	$a_7 \vee a_{16}$	93.19%
vot	$\overline{a}_4 \vee \overline{a}_{14}$	96.19%

It can be seen that the *CP*s of the Boolean separators are not too far from the average accuracies of the various data mining algorithms considered, being on the average about 0.7% above the average accuracies of those methods. It should also be remarked that the unusual simplicity of

the expressions of these Boolean separators makes the formulation of the corresponding approximate classification rules extremely easy.

It is natural to ask whether the combined use of the original variables and the best Boolean separators would increase the accuracy of various machine learning / data mining algorithms. We report in Table 4 the results of a series of computational experiments in which the same five data mining techniques reported in Table 2 are used for classifying the 8 benchmark problems considered above, after the addition of the best Boolean separators. The reported accuracies represent averages obtained in 20 ten-folding experiments.

Table 4

	SMO	*MultilayerPerceptron*	*Simple_Logistic*	*J48*	*LAD*	*Average*
bld	66.32%	65.86%	67.79%	64.37%	70.56%	66.98%
ger	68.93%	64.02%	68.78%	66.27%	71.87%	67.97%
pid	76.40%	72.91%	74.02%	76.67%	75.58%	75.12%
hea	82.27%	78.63%	82.50%	77.10%	83.43%	80.79%
aus	85.50%	82.87%	86.36%	84.33%	85.43%	84.90%
ion	86.39%	90.50%	88.55%	87.82%	91.06%	88.87%
bcw	95.18%	94.12%	94.82%	93.56%	94.48%	94.43%
vot	96.87%	94.32%	96.54%	96.61%	96.87%	96.24%

By comparing tables 2 and 4 one can see that the accuracy of the Pima Indians Diabetes problem increases by 1.33%, that of the BUPA liver-disorders increases by 3.61%, while all the others have very minor increases or decreases (of less than 0.5% each). On the average, the accuracies increase by about 0.5%.

6 Classification results in benchmark datasets using Approximate Boolean Classifiers

Applying the iterative algorithm described in this paper for the generation of *ABC*s in the 8 datasets described in Section 4 produces new variables, which – at least in some of the cases – have extremely simple expressions in terms of the original binary variables. For example in the case of **vot** (which has 16 original binary variables) we produce in 3 steps of the iterative process the following Boolean separators:

$$b_1 = \overline{a}_4 \vee \overline{a}_{14}$$
$$b_2 = a_{16}\, b_1$$
$$b_3 = \overline{a}_4 \vee b_2$$

Although some other *ABC*s are generated in this process, this list includes only those which had the highest *CPs*. The new variable b_3 takes the value 1 in 118 of the 124 positive observations in the dataset, and the value 0 in 106 of the 108 negative observations, its *CP* (in percentages) is

$\frac{1}{2}\left(\frac{118}{124}+\frac{106}{108}\right)*100 = 96.65\%$. Since in the next step no new variable is created having a higher *CP* or *Co-CP,* the process stops after the generation of the *ABC* b_3.

Substituting into b_2 the expression of b_1, and into b_3 the expression of b_2, we find

vot: $\overline{a}_4 \vee a_{16}\overline{a}_{14}$

It is interesting to note that this *ABC depends only on 3 of the 16 original variables*, but provides the correct classification of 224 of the 232 observations in the original dataset .

Following the same procedure of step – by – step substitutions, we can generate an *ABC* for

brw : $(a_3\, a_9)\vee (a_3\, a_{16})\vee a_7 \vee (a_9\, a_{16})\vee (a_9\, a_{18})\vee (a_{14}\, a_{16})$

and for

bld: $(\overline{a}_{11}\, a_{22}\, \overline{a}_{29})\vee (a_{21}\, a_{22}\, \overline{a}_{29})\vee (a_{22}\, a_{25}\, \overline{a}_{29})$

The *CPs* of these two *ABC*s are 96.08% and 69.33% respectively.

The *ABC*s generated for some of the other datasets are the following

aus: $(a_{13}\, a_{27}) \vee (a_{27}\, a_{32}) \vee (a_{12}\, a_{18}\, a_{27}\, a_{35})$

ion : $(a_1\, a_2\, a_6\, a_{11}\, a_{34}\, \overline{a}_5\, a_{48}) \vee (a_4\, a_8\, a_{26})$

pid: $(a_4\, a_{22}) \vee (a_{16}\, a_{18}\, a_{22}) \vee (a_{16}\, a_{22}\, a_{23}) \vee (a_3\, a_{14}\, a_{22}) \vee (a_3\, a_5)$
$\vee (a_4\, a_5) \vee (a_4\, a_{18}) \vee (a_3\, a_5\, a_{16}) \vee (a_3\, a_{16}\, a_{18}) \vee (a_{13}\, a_{18})$

hea: $(a_4\, a_{12}\, a_{16}) \vee (a_4\, a_5\, a_7\, a_{12}) \vee (a_4\, a_{17}) \vee (a_5\, a_{12}\, a_{17}) \vee (a_5\, a_{16} a_{17})$
$\vee (a_4\, a_5\, a_{16}) \vee (a_5\, a_7\, \overline{a}_{11}\, a_{12}) \vee (a_4\, \overline{a}_{11}\, a_{16}) \vee (a_5\, a_7\, \overline{a}_{11}\, a_{16})$

The *CP*s of these four *ABC*s are respectively 86.95%, 93.24%, 82.76%, 86.29%. In order to illustrate the logical structure of such formulas, we present in Appendix II an *ABC* for the dataset **ger** using the graph representation of a Boolean function typical for the VLSI literature; the *CP* of this *ABC* is 72.93%.

The construction of the above listed *ABC*s required respectively 3, 5, 4, 5, 6, 6, 5, 8 iterations for the 8 datasets considered, i.e. an average of 5.25 iterative steps.

In Table 5 we compare the average accuracies of various classification methods (as reported in Table 2) with the accuracies of the *ABC*s on the 8 benchmark datasets. It can be seen that the accuracy provided by the *ABC*s

- is ***higher for every single dataset considered*** than the ***average accuracy*** provided by the 5 classification methods, the average improvement being more than 4.1%,

- is ***higher for 6 out of the 8 datasets considered*** than the ***maximum accuracy*** provided by the 5 classification methods, the average improvement being more than 1.6%.

Table 5

Dataset	Average accuracy of 5 classification methods	Maximum accuracy of 5 classification methods	Accuracy of *ABC*	Improvement over **average** accuracy of 5 classification methods	Improvement over **maximum** accuracy of 5 classification methods
bld	63.37%	69.29%	69.33%	+5.96%	+0.04%
ger	68.37%	72.21%	72.93%	+4.56%	+0.72%
pid	73.78%	76.13%	82.76%	+8.98%	+6.63%
hea	80.75%	83.05%	86.29%	+5.54%	+3.24%
aus	85.32%	86.66%	86.95%	+1.63%	+0.29%
ion	88.92%	91.58%	93.24%	+4.32%	+1.66%
bcw	94.35%	95.28%	96.08%	+1.73%	+0.80%
vot	96.34%	97.14%	96.65%	+0.31%	-0.49%
			Average	+4.13%	+1.61%

7 Explanatory Power of *ABC*s

In numerous practical situations it is important to be able to provide an explanation of the reasons for which an observation is declared to be as positive or as negative by a classification system. For instance, if the problem is to explain to the patient (or to the insurance company) why the he/she was classified as being sick or healthy, it is convenient to be able to point to a specific condition fulfilled by the patient, and known to be characteristic for sick or for healthy people. This goal can be easily accomplished by using the terms of an *ABC*, or of its negation. Fortunately, the way in which the *ABC*s are generated makes it computationally affordable to calculate disjunctive normal forms of the *ABC*s or their negations.

Example. Let us consider again the dataset **bld.** If $a_1,\ldots, a_{29}$ are the original binary variables of the problem, if b, c, and d are Boolean separators found in iterations 1, 2, and 3, respectively, if f is an *ABC* found in the forth iteration, and if the new variables are generated by using the formulas

$$b = \bar{a}_{11} \vee a_{25}$$
$$c = a_{21} \vee b$$
$$d = \bar{a}_{29}\, c$$
$$f = a_{22}\, d,$$

then by sequentially substituting the expressions of d, c, and b, into f, we find the following DNF of the latter:

$$f = (\bar{a}_{11}\, a_{22}\, \bar{a}_{29}) \vee (a_{21}\, a_{22}\, \bar{a}_{29}) \vee (a_{22}\, a_{25}\, \bar{a}_{29}).$$

The expressions of b, c, d, and f allow us to easily calculate their complements:

$$\bar{b} = a_{11}\, \bar{a}_{25}$$
$$\bar{c} = \bar{a}_{21}\, \bar{b}$$
$$\bar{d} = a_{29} \vee \bar{c}$$
$$\bar{f} = \bar{a}_{22} \vee \bar{d}\,.$$

By sequentially substituting the expressions of $\bar{b}$, $\bar{c}$, $\bar{d}$, and $\bar{f}$, we find the latter's DNF

$$\bar{f} = \bar{a}_{22} \vee a_{29} \vee (a_{11}\, \bar{a}_{21}\, \bar{a}_{25}).$$

Using the expressions f and $\bar{f}$, we can justify the positive or negative classification of an observation. We can remark for instance, that if in a new observation the value of the variable *sgpt* is $\leq$ 19 (i.e., $a_{11} = 0$), *gammagt* is $>$ 7 (i.e., $a_{22} = 1$), and *drinks* $\leq$ 12 (i.e., $a_{29} = 0$), then the term $\bar{a}_{11}\, a_{22}\, \bar{a}_{29}$ takes the value 1, implying the positive nature of this observation. Similarly, if for example in a new observation *gammagt* is $\leq$ 7 (i.e., $a_{22} = 0$), then $\bar{f}$ takes the value 1, implying the observation's negative nature.

8 Conclusions

It has been seen that for every binary dataset we can associate a set of easy to calculate synthetic variables, called *Boolean separators*, some of which can be used either alone, or in combination with the original variables, to improve the accuracy of classification of various frequently used machine learning/ data mining techniques.

It has been also seen that by iterating (usually, only a very small number of times) the Boolean separator generating technique, we can determine *approximate Boolean classifiers*, which can provide further improvements to the accuracy of each of the examined classification techniques.

An important byproduct of the study is the fact that algebraic expressions of the *ABC*s and of their compliments are found, and can be used for a better understanding of the mathematical structure of the problem (e.g. the relative importance of variables, their monotonicities – if any,

etc.) as well as for the practical purpose of justifying the positive or negative classification of new observations.

References

[1] E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, Logical Analysis of Numerical Data, *Mathematical Programming*, 79, 163-190, 1997.

[2] E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, I. Muchnik, An Implementation of the Logical Analysis of Data, *IEEE Transactions on Knowledge and Data Engineering,* 12, No.2, 292-306, 2000.

[3] Y. Crama, P.L. Hammer, T. Ibaraki, Cause-Effect Relationships and Partially Defined Boolean Functions, *Annals of Operations Research*, 16, 299-326, 1988.

[4] Z. Csizmadia, P.L. Hammer, and B. Vizvari, Generation of Artificial Attributes for Data Analysis, *RUTCOR Research Report*, 1-2006.

[5] P.L. Hammer, Partially Defined Boolean Functions and Cause-Effect Relationships, *International Conference on Multi-Attribute Decision Making Via OR-Based Expert Systems*, University of Passau, Passau, Germany, 1986.

[6] P. L. Hammer and T. Bonates, Logical Analysis of Data: From Combinatorial Optimization to Medical Applications, *Annals of Operations Research* (forthcoming).

[7] K. Haraguchi, T. Ibaraki, and E. Boros, Classifiers based on iterative compositions of features, *Proc. 1st Intl. Conf. Knowledge Engineering and Decision Support*, 143-150, Porto, Portugal, Aug. 2004

[8] Y. Hu and D. Kibler, Generation of Attributes for Learning Algorithms, *in Proceeding of the 13th National Conference on Artificial Intelligence*, 806-811, 1996

[9] S. Markovitch and D. Rosenstein, Feature Generation Using General Constructor Functions, *Machine Learning*, Volume 49, No.1, 59-98, 2002

Appendix I

Table A *BUPA liver-disorders*

Binary variable	Original variable	Cutpoint
a_1	mcv	87
a_2	mcv	89
a_3	mcv	90
a_4	mcv	92
a_5	alkphos	51
a_6	alkphos	65
a_7	alkphos	77
a_8	alkphos	84.5
a_9	sgpt	16
a_{10}	sgpt	17
a_{11}	sgpt	19
a_{12}	sgpt	21
a_{13}	sgpt	23
a_{14}	sgpt	26
a_{15}	sgpt	39
a_{16}	sgpt	48
a_{17}	sgot	19
a_{18}	sgot	20
a_{19}	sgot	22
a_{20}	sgot	24
a_{21}	sgot	44
a_{22}	gammagt	7
a_{23}	gammagt	20
a_{24}	gammagt	29
a_{25}	gammagt	36
a_{26}	gammagt	116
a_{27}	drinks	3
a_{28}	drinks	5
a_{29}	drinks	12

Table B *German credit*

Binary variable	Original variable	Cutpoint
a_1	Attribute 1	1.5
a_2	Attribute 1	2.5
a_3	Attribute 1	3
a_4	Attribute 2	13.5

a_5	Attribute 2	15.5
a_6	Attribute 2	18
a_7	Attribute 2	21
a_8	Attribute 2	24
a_9	Attribute 2	30
a_{10}	Attribute 3	2
a_{11}	Attribute 3	2.5
a_{12}	Attribute 3	3
a_{13}	Attribute 4	14
a_{14}	Attribute 4	19
a_{15}	Attribute 4	24
a_{16}	Attribute 4	30.5
a_{17}	Attribute 4	40
a_{18}	Attribute 4	53
a_{19}	Attribute 5	1.5
a_{20}	Attribute 5	2
a_{21}	Attribute 5	2.5
a_{22}	Attribute 5	3
a_{23}	Attribute 6	2.5
a_{24}	Attribute 6	3
a_{25}	Attribute 6	3.5
a_{26}	Attribute 6	4
a_{27}	Attribute 7	2.5
a_{28}	Attribute 7	3.5
a_{29}	Attribute 8	2
a_{30}	Attribute 8	2.5
a_{31}	Attribute 8	3
a_{32}	Attribute 8	3.5
a_{33}	Attribute 9	1.5
a_{34}	Attribute 9	2
a_{35}	Attribute 9	2.5
a_{36}	Attribute 9	3
a_{37}	Attribute 10	27
a_{38}	Attribute 10	30
a_{39}	Attribute 10	33
a_{40}	Attribute 10	36
a_{41}	Attribute 10	39.5
a_{42}	Attribute 10	44.5
a_{43}	Attribute 11	2
a_{44}	Attribute 11	2.5
a_{45}	Attribute 12	1.5
a_{46}	Attribute 13	1.5

a_{47}	Attribute 14	1.5
a_{48}	Attribute 15	1.5
a_{49}	Attribute 16	0.5
a_{50}	Attribute 17	0.5
a_{51}	Attribute 18	0.5
a_{52}	Attribute 19	0.5
a_{53}	Attribute 20	0.5
a_{54}	Attribute 21	0.5
a_{55}	Attribute 22	0.5
a_{56}	Attribute 23	0.5
a_{57}	Attribute 24	0.5

Table C *Pima Indians Diabetes*

Binary variable	Original variable	Cutpoint
a_1	Number of times pregnant	1
a_2	Number of times pregnant	3.5
a_3	Number of times pregnant	7
a_4	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	127
a_5	Plasma glucose concentration a 2 hours in an oral glucose tolerance test	165
a_6	Diastolic blood pressure	67
a_7	Diastolic blood pressure	71
a_8	Diastolic blood pressure	76
a_9	Diastolic blood pressure	80
a_{10}	Triceps skin fold thickness	19.5
a_{11}	Triceps skin fold thickness	26
a_{12}	Triceps skin fold thickness	32
a_{13}	2-Hour serum insulin	69
a_{14}	2-Hour serum insulin	110
a_{15}	2-Hour serum insulin	164
a_{16}	2-Hour serum insulin	197.5
a_{17}	Body mass index	30.1
a_{18}	Body mass index	45.4
a_{19}	Diabetes pedigree function	0.347
a_{20}	Diabetes pedigree function	0.5
a_{21}	Diabetes pedigree function	0.673
a_{22}	Age (years)	23
a_{23}	Age (years)	41

Table D *Cleveland heart disease*

Binary variable	Original variable	Cutpoint
a_1	age	55.5
a_2	age	62
a_3	sex	0.5
a_4	cp	3
a_5	trestbps	109
a_6	trestbps	136
a_7	trestbps	156
a_8	chol	233
a_9	fbs	0.5
a_{10}	restecg	1
a_{11}	thalach	125.583
a_{12}	exang	0.5
a_{13}	oldpeak	0.3
a_{14}	oldpeak	0.5
a_{15}	slope	1
a_{16}	ca	0
a_{17}	thal	3

Table E *Australian credit*

Binary variable	Original variable	Cutpoint
a_1	A1	0.5
a_2	A2	23.375
a_3	A2	27.035
a_4	A2	29.71
a_5	A2	33.96
a_6	A2	39.915
a_7	A3	1.395
a_8	A3	2.665
a_9	A3	4.0425
a_{10}	A3	5.895
a_{11}	A3	7.9375
a_{12}	A4	1.5
a_{13}	A5	4.5
a_{14}	A5	6
a_{15}	A5	7.5
a_{16}	A5	8.5
a_{17}	A5	10
a_{18}	A6	2.5

a_{19}	A6	4.5
a_{20}	A6	5.5
a_{21}	A6	6
a_{22}	A7	0.375
a_{23}	A7	0.875
a_{24}	A7	1.395
a_{25}	A7	2.3125
a_{26}	A7	3.875
a_{27}	A8	0.5
a_{28}	A9	0.5
a_{29}	A10	0.5
a_{30}	A10	1.5
a_{31}	A10	2.5
a_{32}	A10	4
a_{33}	A10	5.5
a_{34}	A11	0.5
a_{35}	A12	1.5
a_{36}	A13	87
a_{37}	A13	130
a_{38}	A13	170.5
a_{39}	A13	220
a_{40}	A13	280
a_{41}	A14	10
a_{42}	A14	76
a_{43}	A14	223
a_{44}	A14	518.5
a_{45}	A14	1401

Table F *Ionosphere*

Binary variable	Original variable	Cutpoint
a_1	Attribute 1	0.5
a_2	Attribute 2	0.26223
a_3	Attribute 2	0.614535
a_4	Attribute 3	-0.46032
a_5	Attribute 3	0.62245
a_6	Attribute 4	0.09844
a_7	Attribute 4	0.824325
a_8	Attribute 5	-0.44056
a_9	Attribute 6	0.149065
a_{10}	Attribute 6	0.516665
a_{11}	Attribute 7	-0.79194

a_{12}	Attribute 8	-0.779705
a_{13}	Attribute 8	-0.028715
a_{14}	Attribute 8	0.32418
a_{15}	Attribute 9	-0.45817
a_{16}	Attribute 9	0.25767
a_{17}	Attribute 9	0.62414
a_{18}	Attribute 10	-0.81263
a_{19}	Attribute 10	0.28425
a_{20}	Attribute 10	0.63587
a_{21}	Attribute 11	-0.813615
a_{22}	Attribute 11	-0.451775
a_{23}	Attribute 12	-0.82287
a_{24}	Attribute 12	-0.46797
a_{25}	Attribute 12	0.269625
a_{26}	Attribute 13	-0.411275
a_{27}	Attribute 14	-0.81353
a_{28}	Attribute 14	-0.036675
a_{29}	Attribute 14	0.3254
a_{30}	Attribute 15	-0.455625
a_{31}	Attribute 16	-0.82353
a_{32}	Attribute 16	0.277815
a_{33}	Attribute 16	0.640465
a_{34}	Attribute 17	-0.81818
a_{35}	Attribute 18	-0.8139
a_{36}	Attribute 18	-0.07269
a_{37}	Attribute 18	0.639865
a_{38}	Attribute 19	-0.81772
a_{39}	Attribute 19	-0.459005
a_{40}	Attribute 19	-0.1047
a_{41}	Attribute 19	0.652285
a_{42}	Attribute 20	-0.822895
a_{43}	Attribute 20	-0.10837
a_{44}	Attribute 20	0.245355
a_{45}	Attribute 21	-0.8181
a_{46}	Attribute 22	-0.818885
a_{47}	Attribute 22	-0.46174
a_{48}	Attribute 23	-0.823075
a_{49}	Attribute 23	-0.470615
a_{50}	Attribute 23	-0.118475
a_{51}	Attribute 24	-0.82329
a_{52}	Attribute 25	-0.82396
a_{53}	Attribute 25	-0.1123

a_{54}	Attribute 25	0.614975
a_{55}	Attribute 26	-0.816185
a_{56}	Attribute 27	-0.813725
a_{57}	Attribute 27	-0.076245
a_{58}	Attribute 28	-0.82436
a_{59}	Attribute 28	-0.463945
a_{60}	Attribute 28	-0.1021
a_{61}	Attribute 29	-0.82214
a_{62}	Attribute 29	-0.10603
a_{63}	Attribute 29	0.61558
a_{64}	Attribute 30	-0.774455
a_{65}	Attribute 30	-0.361305
a_{66}	Attribute 30	0.00406
a_{67}	Attribute 30	0.357855
a_{68}	Attribute 31	-0.822685
a_{69}	Attribute 31	0.622015
a_{70}	Attribute 32	-0.82067
a_{71}	Attribute 33	-0.81478

Table G *Wisconsin breast cancer*

Binary variable	Original variable	Cutpoint
a_1	Clump Thickness	3
a_2	Clump Thickness	4
a_3	Clump Thickness	6
a_4	Clump Thickness	7
a_5	Clump Thickness	8
a_6	Uniformity of Cell Size	2
a_7	Uniformity of Cell Size	3
a_8	Uniformity of Cell Size	4
a_9	Uniformity of Cell Shape	2
a_{10}	Uniformity of Cell Shape	6
a_{11}	Marginal Adhesion	1
a_{12}	Marginal Adhesion	2
a_{13}	Marginal Adhesion	5
a_{14}	Single Epithelial Cell Size	4.25
a_{15}	Bare Nuclei	2
a_{16}	Bare Nuclei	4
a_{17}	Bland Chromatin	4
a_{18}	Normal Nucleoli	3
a_{19}	Normal Nucleoli	9
a_{20}	Mitoses	3

Table H *Congressional voting records*

Binary variable	Original variable
a_1	handicapped-infants
a_2	water-project-cost-sharing
a_3	adoption-of-the-budget-resolution
a_4	physician-fee-freeze
a_5	el-salvador-aid
a_6	religious-groups-in-schools
a_7	anti-satellite-test-ban
a_8	aid-to-nicaraguan-contras
a_9	mx-missile
a_{10}	immigration
a_{11}	synfuels-corporation-cutback
a_{12}	education-spending
a_{13}	superfund-right-to-sue
a_{14}	crime
a_{15}	duty-free-exports
a_{16}	export-administration-act-south-africa

Appendix II

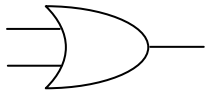
In order to apply the iterative algorithm for generating an *ABC* for **ger** we shall use the binarized attributes shown in Table B of Appendix I. One of the *ABC*s of this problem was generated in 8 iterations, and uses 8 (of the 52) original binarized attributes, corresponding to 5 (of the 24) original numerical attributes. We recall the definitions of those 8 binarized attributes which actually played an active role in the definition of this *ABC*:

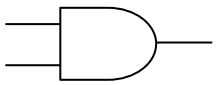
$a_1 = 1$ iff $attr.1 > 1.5$
$a_2 = 1$ iff $attr.1 > 2.5$
$a_3 = 1$ iff $attr.1 > 3$
$a_4 = 1$ iff $attr.2 > 13.5$
$a_{20} = 1$ iff $attr.5 > 2$
$a_{33} = 1$ iff $attr.9 > 1.5$
$a_{48} = 1$ iff $attr.15 > 1.5$
$a_{52} = 1$ iff $attr.19 > 0.5$

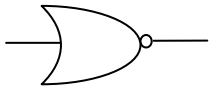
The *ABC* for **ger** was found in the 8th iteration. The Boolean separators used for the generation of this *ABC* were the following:

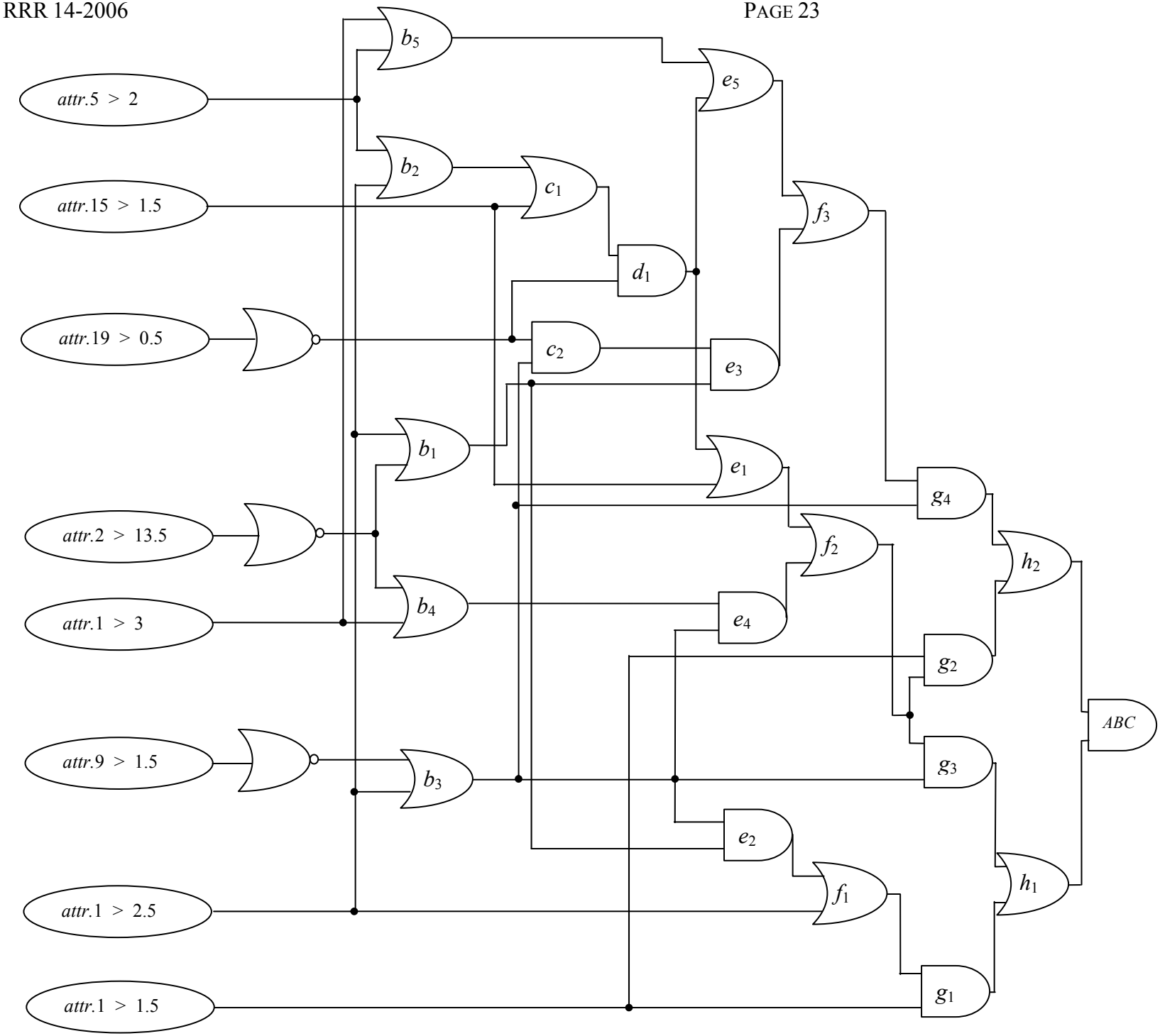
Iteration	Boolean separators				
1	$b_1 = a_2 \vee \overline{a}_4$	$b_2 = a_2 \vee a_{20}$	$b_3 = a_2 \vee \overline{a}_{33}$	$b_4 = a_3 \vee \overline{a}_4$	$b_5 = a_3 \vee a_{20}$
2		$c_1 = a_{48} \vee b_2$		$c_2 = \overline{a}_{52}\, b_3$	
3			$d_1 = \overline{a}_{52}\, c_1$		
4	$e_1 = a_{48} \vee d_1$	$e_2 = b_1\, b_3$	$e_3 = b_1\, c_2$	$e_4 = b_3\, b_4$	$e_5 = b_5 \vee d_1$
5	$f_1 = a_2 \vee e_2$		$f_2 = e_1 \vee e_4$		$f_3 = e_3 \vee e_5$
6	$g_1 = a_1\, f_1$	$g_2 = a_1\, f_2$		$g_3 = b_3\, f_2$	$g_4 = b_3\, f_3$
7		$h_1 = g_1 \vee g_3$		$h_2 = g_2 \vee g_4$	
8			$ABC = h_1\, h_2$		

We represent in Figure 1 the structure of the *ABC* for **ger** with the help of the above sequence of functions, using the typical conventions of computer engineering; each vertex in the graph of Figure 1 representing either one of the original (binarized) variables, or a Boolean operation on one or two input variables (with signals coming from the left side of the vertex), whose output signal is shown on the right side of the vertex. The Boolean operations are respectively represented as


Disjunction


Conjunction


Negation


attr.5 > 2
attr.15 > 1.5
attr.19 > 0.5
attr.2 > 13.5
attr.1 > 3
attr.9 > 1.5
attr.1 > 2.5
attr.1 > 1.5
b5
b2
c1
d1
c2
e5
e3
f3
b1
e1
g4
f2
h2
b4
e4
g2
ABC
g3
b3
e2
f1
h1
g1