Computer Science Journal of Moldova, vol.**8**, no.1(22), 2000

Reduction algorithms for solving large systems of logical equations

A. Zakrevskij

Abstract

Large systems of logical equations are considered in this paper, each depending on a restricted number of variables. A method of reduction is suggested that reduces the number of roots in separate equations, which in its turn saves time spent for finding roots of the whole system. Three mechanisms of reduction are proposed, each looking for some prohibited combinations of variables in separate equations (combinations that do not satisfy the equations). The first procedure looks for constants (prohibited values of some variables, or 1-bans). The second one looks in a similar way for prohibited combinations of values on pairs of variables (2-bans) and finds all their logical consequences closing the set of discovered 2-bans. The third analyses the equations by pairs, finds r common variables for them, and checks one by one all different combinations of their values looking for prohibited ones (r-bans). The found bans are used for deleting some roots in other equations. After this new bans could be found, so the procedure of reduction has the chain nature. It greatly facilitates solving large systems of logical equations. Sometimes it is enough to find the only root of a system or prove its inconsistency.

1 Introduction

A lot of problems related to logical design, diagnostics (both in medicine and engineering), pattern recognition, information security and many others are reduced to formulating and solving large systems of logical equations. This task is rather difficult because the considered systems could contain many equations and variables, so it is impossible to solve them using direct methods which are based on scanning Boolean space of variables and checking its elements one by one.

However, as a rule the number of variables in separate equations is greatly restricted, for example not exceeding 10. This allows to represent every equation by a short Boolean vector of its roots, providing a compact description of the regarded system and efficient application of vector logical operations.

The most interesting from practical point of view is the case of systems having few roots or none at all. This situation is typical for checking the system for satisfiability (a popular task!) and solving some problems of diagnostics and recognition. The well known tree searching technique could be used in this case, especially combined with new means that powerfully reduce the area of search.

Three competing algorithms of that kind are suggested in this paper.

2 Formulation of the problem

Every system of logical equations with Boolean variables is easily reduced to the form

$$F = (\varphi_1(\mathbf{u}_1) = 1, \;\; \varphi_2(\mathbf{u}_2) = 1, \;\; \ldots, \;\; \varphi_m(\mathbf{u}_m) = 1),$$

where $\varphi_i(\mathbf{u}_i)$ are Boolean functions with arguments selected from the set $\mathbf{x} = (x_1, x_2, \ldots, x_n) : \mathbf{u}_i \subseteq \mathbf{x}, i = 1, 2, \ldots, m$. To solve system F means to find its roots – combinations of values of variables $x_1, x_2, \ldots, x_n$, which turn into 1 each function φ_i. It is necessary in some cases to obtain all the roots, sometimes it is enough to find only several roots or even arbitrary one of them, sometimes it is needed to know if there exists some root, i. e. to solve the well-known satisfiability problem.

Let us represent any Boolean function φ_i with k arguments ($k = |\mathbf{u}_i|$) from the system F by a pair of Boolean vectors: 2^k-component *vector* $\mathbf{v}_i$ *of function values* (using the conventional component ordering) and n-component *vector* $\mathbf{a}_i$ *of function arguments.* For example, if $\mathbf{x} = (a, b, c, d, e, f, g, h)$, then the pair of vectors $\mathbf{v}_3 = 01101010$ and $\mathbf{a}_3 = 00101001$ represents the function $\varphi_3(c, e, h)$ which takes value 1 on four combinations 001, 010, 100 and 110 of argument values and takes value 0 on all others.

When n is small (equals 10, for example), representation of all functions φ_i can be unified by introducing in each function missing variables from $\mathbf{x}$ (as fictive ones) and correspondingly extending vectors $\mathbf{v}_i$. After that the complete solution of the system is easily reached by the componentwise conjunction of these extended vectors: all roots of the system will be enumerated in the resulting vector $\mathbf{v}$. But this operation is practically impossible for large n – for example, the length of vector $\mathbf{v}$ will exceed 10^{18} for $n = 60$.

New methods could be useful in that case, based on reduction procedures described below.

3 Method of spreading of constants

This method was proposed in [1, 2] and can be formalized as follows.

The rules of equivalence transformation of system F are introduced below, where φ_j denotes an arbitrary function from F and x_i – some variable from $\mathbf{x}$. These rules enable to simplify F with preserving the set of its roots.

Assertion 1 *If $x_i\varphi_j = 0$, the inequality (prohibition) $x_i \neq 1$ could be added to F, if $x_i'\varphi_j = 0$, then $x_i \neq 0$ could be added.*

Assertion 2 *If F contains inequality $x_i \neq 1$, then variable x_i in φ_j can be changed for 0 (for 1 in the case $x_i \neq 0$).*

Assertion 3 *If system F contains both inequalities $x_i \neq 1$ and $x_i \neq 0$, it is inconsistent (has no roots).*

The first assertion can be used for *finding constants* in the system. If the system is obviously consistent, $x_i = 0$ follows from inequality $x_i \neq 1$ and $x_i = 1$ – from $x_i \neq 0$. The probability of finding some constants in the system grows with decreasing the number of variables and roots in analyzed equations.

The second assertion could be used for *spreading of constants*. Replacing some variables by constants usually decreases the number of roots in regarded equations which, in its turn, helps to discover new constants. So, the process of constants spreading has the chain nature. As a result, the dimension of processed equations is decreasing, sometimes down to zero – when all variables of the regarded equation receive definite values. If function φ_j turns into 1, the corresponding equation is deleted from the system; if φ_j turns into 0, it becomes evident that the system is inconsistent.

Simple enough, this method turned out to be very efficient, being applied to some problems of cryptography. A special problem of cryptanalysis of the mechanical rotor encryption machine Hagelin M-209-B, which was applied in several forms by Germans during the second world war, was investigated in [3]. It was shown that its cryptanalysis can be reduced to solving a definite system of many logical equations (about five hundred) each of which contains six Boolean variables, meanwhile the general number of variables equals 131 – the set of their values constitutes the sought-for key. To solve this system a method was proposed in [3] based on using Reduced Ordered Binary Decision Diagrams (ROBDDs) [4] for representation of the regarded functions. Its computer implementation on Pentium Pro 200 showed that under some suppositions it enables to find the key in several minutes.

Application of the method of spreading of constants using vector representation of the considered Boolean functions and taking into account the specific of the regarded system of logical equations was far more efficient. It accelerates the search for the key more than in thousand times: the run-time for the problem described above usually did not exceed 0.1 second in a series of experiments using C++ and PC Pentium 100 [5].

4 Method of syllogisms

Let us regard equation $\varphi(z_1, z_2, \dots, z_k) = 1$ with function φ taking value 1 on s randomly selected inputs. When s is small, it is possible to find some constant – a prohibition on a value of some variable (a ban of rank 1, or 1-*ban*). But it is more probable to reveal a prohibition on some combination of values of two variables (2-*ban*), which determines the corresponding *implicative regularity*, or connection between these variables. For example, connection "if a, then not b" prohibits combination of values $a = 1$, $b = 1$. It could be revealed in φ if $ab\varphi = 0$. For convenience, represent this ban by product ab (having in mind equation $ab = 0$).

In a similar way, 2-bans ab', $a'b$, $a'b'$ are defined. They are interpreted easily as general affirmation and general negation category statements. By that besides three such statements of Aristotle syllogistic (ab' – all A are B, $a'b$ – all B are A, ab – none of A is B) the fourth is used: $a'b'$ – none of objects is A and is not B. Such a statement was not considered by Aristotle, inasmuch as he did not regarded empty classes [6].

The chances for revealing some bans of rank r (containing r variables) are big enough if the mathematical expectation M of the number of such bans is not less than 1: $M \geq 1$. It is calculated by formula $M = C_k^r 2^r (1 - 2^{-r})^s$ [7], reduced down to $k/2^{s-1}$ for 1-bans and to $2k(k-1)(3/4)^s$ for 2-bans. In correspondence with this formula the condition $M \geq 1$ is satisfied under some restrictions on numbers of variables (k) and especially of roots (s) in equation $\varphi(z_1, z_2, \dots, z_k) = 1$, and these restrictions are considerably weaker for 2-bans compared with 1-bans. (It is assumed that roots are randomly placed.) For practically interesting values of k (from 5 to 10) they are represented by the following table, calculated from the given formula. There are shown maximal values of s satisfying condition $M \geq 1$ for 1-bans (s^1_{max}) and 2-bans (s^2_{max}). The relative values of parameter s are given for 2-bans by the last string: $s^2_{max}\% = 100 s^2_{max}/2^k$).

k	5	6	7	8	9	10
s^1_{max}	3	3	3	4	4	4
s^2_{max}	12	14	15	16	17	18
$s^2_{max}\%$	37	22	12	6	3	1,7

It follows from the table that when k is small implicative regularities can be found even in the case of rather high percentage of roots in the regarded equations.

4.1 Closing the system P

Suppose, that by examining equations of the system F one by one, we have found a set P of 2-bans. Let us consider the task of *closing* it, i. e. adding to it all other 2-bans which logically follow from P (so called resolvents of P). This task is equivalent to the polysyllogistic problem. Denote the resulting *closed set of* 2*-bans* as $Cl(P)$. A method to find it is suggested below. It differs from the well-known method of resolutions [8] and its graphical version [9] by application of vector-matrix operations which speed up the logical inference.

Let X^1_t and X^0_t be the sets of all literals that enter 2-bans contained in F together with literal x_t or x'_t, correspondingly. We introduce operator Cl_t of *partial closing* of set P in regard to variable x_t, extending this set by unifying it with direct product $X^1_t \times X^0_t$ containing results of all possible resolutions by this variable.

Assertion 4 $Cl_t(P) = P \cup X^1_t \times X^0_t \subseteq Cl(P)$.

Assertion 5 $Cl(P) = Cl_1 Cl_2 \dots Cl_n(P)$.

In such a way, the set P can be closed by separate variables, one by one.

The set P can be represented by a square Boolean matrix $\mathbf{P}$ of the size $2n$ by $2n$, with rows $\mathbf{p}_{t1}, \mathbf{p}_{t0}$ and columns $\mathbf{p}^{t1}, \mathbf{p}^{t0}$ corresponding to literals $x_t, x'_t, t = 1, 2, \dots, n$. Elements of matrix $\mathbf{P}$ correspond to pairs of literals, and non-diagonal elements having value 1 represent discovered 2-bans. So, the totality of 1s in row $\mathbf{p}_{t1}$ (as well as in column $\mathbf{p}^{t1}$) indicates set X^1_t, and the totality of 1s in row $\mathbf{p}_{t0}$ (column

$\mathbf{p}^{t0}$) indicates set X_t^0. Using vector operations, we can construct the matrix $\mathbf{P}^+$, presenting the result of closing operation: $\mathbf{P}^+ = Cl(P)$.

For example, if $\mathbf{x} = (a, b, c, d)$ and 2- bans ab', ac, $a'd'$, bc' are found forming set P, then

$$\mathbf{P} = \begin{array}{ccccc} \text{aa'} & \text{bb'} & \text{cc'} & \text{dd'} & \\ 00 & 01 & 10 & 00 & \text{a} \\ 00 & 00 & 00 & 01 & \text{a'} \\ 00 & 00 & 01 & 00 & \text{b} \\ 10 & 00 & 00 & 00 & \text{b'} \\ 10 & 00 & 00 & 00 & \text{c} \\ 00 & 10 & 00 & 00 & \text{c'} \\ 00 & 00 & 00 & 00 & \text{d} \\ 01 & 00 & 00 & 00 & \text{d'} \end{array} \quad \mathbf{P}^+ = \begin{array}{ccccc} \text{aa'} & \text{bb'} & \text{cc'} & \text{dd'} & \\ \mathbf{c}0 & \mathbf{c}1 & 1\mathbf{b} & 0\mathbf{c} & \text{a} \\ 00 & 00 & 00 & 01 & \text{a'} \\ \mathbf{c}0 & 00 & 01 & 0\mathbf{c} & \text{b} \\ 10 & 00 & 00 & 0\mathbf{a} & \text{b'} \\ 10 & 00 & 00 & 0\mathbf{a} & \text{c} \\ \mathbf{b}0 & 10 & 00 & 0\mathbf{b} & \text{c'} \\ 00 & 00 & 00 & 00 & \text{d} \\ \mathbf{c}1 & \mathbf{ca} & \mathbf{ab} & 0\mathbf{c} & \text{d'} \end{array}$$

– the bans-consequences are marked in matrix $\mathbf{P}^+$ by symbols of variables by which the corresponding resolutions were executed.

The closed set $Cl(P)$ could be found also by the *increment algorithm of expansion* of P: every time when a new 2-ban p is added by a special operation $ins(p, P)$ all resolvents are included in P too. In that case after each step the set P will remain closed: $P = Cl(P)$.

Operation $ins(p, P)$ is defined as follows.

Assertion 6 *If $P = Cl(P)$, than $Cl(P \cup \{p\}) = P \cup D$, where*

$$D = (\{x\} \cup X^0) \times (\{y\} \cup Y^0), \quad \text{if } \ p = xy,$$
$$D = (\{x\} \cup X^0) \times (\{y'\} \cup Y^1), \quad \text{if } \ p = xy',$$
$$D = (\{x'\} \cup X^1) \times (\{y\} \cup Y^0), \quad \text{if } \ p = x'y,$$
$$D = (\{x'\} \cup X^1) \times (\{y'\} \cup Y^1), \quad \text{if } \ p = x'y'.$$

4.2 Finding all prime bans

Consider now the problem of finding all *prime bans* (which do not follow from one another) deduced from system P. It is known that no set of 2-bans can produce any bans of higher rank. But it can produce some 1-bans, prohibiting definite values of separate variables.

Assertion 7 *All* 1*-bans deduced from set P are represented by* 1*-elements of the main diagonal of matrix* $\mathbf{P}^+$*.*

In the regarded example 1-bans a and d' are presented in such a way.

Assertion 8 *If the pair of* 1*-bans x and x' is found for some variable x, the system F is inconsistent.*

Note that inconsistency of F follows from inconsistency of P, but not vice versa.

4.3 The algorithm of reduction by syllogisms

The suggested method deals with a set of logical equations F, empty at the beginning. It examines the equations in cyclic order, reduces the set of roots of the current equation $f_j = 1$ by considering bans enumerated in P (prohibited roots are deleted) and looks there for new 2-bans not existing in P. These bans are added to P, at the same time operation of closing P is performed. By that some variables can receive unique values - 1s appear on the main diagonal of matrix $\mathbf{P}$ (1-bans are found). The procedure comes to the end when inconsistency is revealed (0-ban is found represented by a pair of 1s on the main diagonal of $\mathbf{P}$) or when processing m equations one by one turns out to be unsuccessful – in that case we have as a result a reduced system of equations equivalent to the initial one.

5 Method of local reduction

First suggested in [10] this method has the local nature. That means that the possibility of reduction is looked for when examining various pairs of functions $\varphi_i(\mathbf{u}_i)$ and $\varphi_j(\mathbf{u}_j)$ with intersecting sets of arguments: $\mathbf{u}_{i,j} = \mathbf{u}_i \cap \mathbf{u}_j \neq \emptyset$.

Let us introduce some denotations. Consider characteristic set M_i of function $\varphi_i(\mathbf{u}_i)$ in the space of arguments from the set $\mathbf{u}_i$, and let $\mathbf{a}$ be its arbitrary element: $\mathbf{a} \in M_i$. The latter is a k-component Boolean vector, where k is the number of arguments of function $\varphi_i(\mathbf{u}_i)$: $k = |\mathbf{u}_i|$. Let $\mathbf{v}$ be an arbitrary subset from $\mathbf{u}_i$ ($\mathbf{v} \subseteq \mathbf{u}_i$) and $\mathbf{a}/\mathbf{v}$ – the *projection of element* $\mathbf{a}$ *onto* $\mathbf{v}$, i.e. the vector composed of those components of vector $\mathbf{a}$ which correspond to variables included in set $\mathbf{v}$.

The set of all different projections of elements from M_i onto $\mathbf{v}$ is named the *projection of set* M_i *onto* $\mathbf{v}$ and designated $M_i/\mathbf{v}$. Let $M_{i,j}$ be the intersection of sets $M_i/\mathbf{u}_{i,j}$ and $M_j/\mathbf{u}_{i,j}$, and $M_{i/j}$ – the set of all such elements from M_i which projections onto $\mathbf{u}_{i,j}$ belong to the set $M_{i,j}$.

For example, if $\mathbf{u}_i = (a, b, c, d, e)$, $\mathbf{u}_j = (c, d, e, f, g, h)$, $M_i = (01101, 11010, 10011)$ and $M_j = (101110, 001101, 010010)$, then $\mathbf{u}_{i,j} = \mathbf{u}_{j,i} = (c, d, e)$, $M_{i,j} = M_{j,i} = (101, 010)$, $M_{i/j} = (01101, 11010)$ and $M_{j/i} = (101110, 010010)$.

Adhering to conventional designations, introduce the operation $M_i := M_{i/j}$ of changing M_i for $M_{i/j}$.

Assertion 9 *For any $i, j = 1, 2, \ldots, m$ operation $M_i := M_{i/j}$ is an equivalent transformation of system F, preserving the set of its roots.*

Note that application of this operation to the shown above example reduces each set M_i and M_j by one element.

Let us tell that operation $M_i := M_{i/j}$ is *applicable* to an ordered pair of functions (φ_i, φ_j) if $M_i \neq M_{i/j}$. The probability of its applicability rises with increasing of the cardinality $|\mathbf{u}_{i,j}|$ of set $\mathbf{u}_{i,j}$ and goes down when $|\mathbf{u}_{i,j}|$ decreases. For instance, it is rather high when $|M_j| < 2^s$, where $s = |\mathbf{u}_{i,j}|$.

Consider now the procedure of sequential execution of this operation on pairs where it can be applied. It could terminate with reducing some of the sets M_i down to the empty set, which will mean that system F is inconsistent, or some set of reduced functions will be found where the given operation cannot be applied to any pair. Let us call this procedure the *local reduction* of system F.

5.1 Example

Consider a system F consisting of three equations $\varphi_1(a,b,c,d) = 1$, $\varphi_2(c,d,e,f) = 1$ and $\varphi_3(e,f,g,h) = 1$ represented by two Boolean matrices: the *matrix of arguments* **A** defining the distribution of arguments over the functions, and the *matrix of functions* **F** where each function is defined by the corresponding row of its values at inputs (combinations of argument values). Note that inputs are ordered from left to right by the traditional binary code – for instance, input 1001 corresponds to the ninth component of the string (the numbering begins with zero). These inputs are presented by columns of the *matrix of constants* **C**, their components are numbered from left to right: $c_1c_2c_3c_4$. Each row of **C** presents a simplest function which coincides with one of the arguments.

For instance, function φ_1 takes value 1 at inputs 0010, 0101 and 1001 – combinations of values of arguments a, b, c, d.

$$\mathbf{F} = \begin{array}{ccccc} 0010 & 0100 & 0100 & 0000 & \varphi_1 \\ 1100 & 0000 & 1001 & 0110 & \varphi_2 \\ 0000 & 0010 & 1001 & 0010 & \varphi_3 \end{array} \quad \mathbf{A} = \begin{array}{c} \text{abcdefgh} \\ 11110000 \\ 00111100 \\ 00001111 \end{array}$$

$$\mathbf{C} = \begin{array}{ccccc} 0000 & 0000 & 1111 & 1111 & \mathbf{c}_1 \\ 0000 & 1111 & 0000 & 1111 & \mathbf{c}_2 \\ 0011 & 0011 & 0011 & 0011 & \mathbf{c}_3 \\ 0101 & 0101 & 0101 & 0101 & \mathbf{c}_4 \end{array}$$

5.2 Algorithm

Let us demonstrate the algorithm of local reduction on the given above example of system F. Regard in succession pair of functions, beginning with (φ_1, φ_2). Using operation of componentwise conjunction of corresponding rows of matrix **A**, we find for this pair common arguments c and d. Going through all combinations of values of these variables, we examine defined by them intervals in the space of arguments of function φ_1 (this space is presented by vector φ_1) and find between them intervals free of values 1 of this function. Then we delete all 1s in corresponding intervals of vector φ_2.

Vector representation of intervals and componentwise logical operations are used during this procedure. For example, considering combination 00 of values of variables c and d and executing operation of conjunction of inverses of vectors $\mathbf{c}_3$ and $\mathbf{c}_4$, we construct vector 1000 1000 1000 1000 defining the proper interval in the space of variables a, b, c, d. Its conjunction with vector φ_1 does not contain ones, therefore equation $\varphi_1 = 1$ has no roots in this interval. The corresponding interval in the space of arguments of function φ_2 is represented by vector 1111 0000 0000 0000, inasmuch as variables c and d take now left positions. All ones contained in this interval are deleted from vector φ_2, so the latter receives the value 0000 0000 1001 0110.

All former operations could be presented in a more compact form, by the formula

$$c'd'\varphi_1 = 0 \rightarrow \varphi_2 := 0000\ 0000\ 1001\ 0110.$$

The next operations are presented similarly:

$$cd\varphi_1 = 0 \rightarrow \varphi_2 := 0000\ 0000\ 1001\ 0000,$$

$$c'd\varphi_2 = 0 \rightarrow \varphi_1 := 0010\ 0000\ 0000\ 0000,$$

$$e'f\varphi_2 = 0 \rightarrow \varphi_3 := 0000\ 0000\ 1001\ 0010,$$

$$ef'\varphi_2 = 0 \rightarrow \varphi_3 := 0000\ 0000\ 0000\ 0010,$$

$$e'f'\varphi_3 = 0 \rightarrow \varphi_2 := 0000\ 0000\ 0001\ 0000.$$

As a result, the initial system of Boolean functions is reduced to the following one:

$$\mathbf{F} = \begin{array}{ccccl} 0\,0\,1\,0 & 0\,0\,0\,0 & 0\,0\,0\,0 & 0\,0\,0\,0 & \varphi_1 \\ 0\,0\,0\,0 & 0\,0\,0\,0 & 0\,0\,0\,1 & 0\,0\,0\,0 & \varphi_2, \\ 0\,0\,0\,0 & 0\,0\,0\,0 & 0\,0\,0\,0 & 0\,0\,1\,0 & \varphi_3 \end{array}$$

from where the unique root of the system is easily obtained: 00101110.

6 Conclusion

Solving large systems of logical equations is a hard combinatorial problem that has a lot of useful applications. To facilitate it three algorithms for reducing the number of roots in separate equations are proposed. As experiments show, these algorithms enable to solve large systems of equations containing together up to hundred variables but with restricted number of arguments in each of them. Combining the suggested algorithms with tree searching technique looks rather perspective.

References

[1] Zakrevskij A.D. *Logical equations.* Minsk: Nauka i tekhnika, 1975 (in Russian).

[2] Zakrevskij A.D. *Method of "reflected waves" for solving logical equations.* - In: Application aspects of the automata theory. Third International Workshop, v.1, Bulgarian Academy of Sciences, Varna, 1975, pp.81-84 (in Russian).

[3] Baumann M., Rohde R., Barthel R. *Cryptanalysis of the Hagelin M-209 Machine.* - 3rd International Workshop on Boolean Problems, Sept. 17-18, 1998, Freiberg (Sachsen), pp. 109-116.

[4] Bryant R.E. *Graph-based algorithms for Boolean functions manipulation.* - IEEE Transactions on Computers, v. C-35, No.8, August 1986, pp. 677-691.

[5] Zakrevskij A.D., Vasilkova I.V. *Cryptanalysis of the Hagelin machine by the method of spreading of constants.* - Proceedings of the Third International Conference on Computer-Aided Design of Discrete Devices (CAD DD'99), Minsk, November 10-12, 1999, v.1, pp.140-147.

[6] Lukasiewich J. *Aristotle syllogistic from the point of view of modern formal logic.* - Moscow, 1959 (in Russian).

[7] Zakrevskij A.D. *Logical inference in the space of multi-valued attributes.* - Computer Science Journal of Moldova, v.2, No 2, 1994, pp.169-184.

[8] Chin-Liang Chang, Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* - Academic Press, N.-Y., S.-Fr., L., 1973.

[9] Zakrevskij A.D. *To formalization of polysyllogistic.* - Logical Inference, Moscow: Nauka, 1979, pp.300-309 (in Russian).

[10] Zakrevskij A.D. *Solving systems of logical equations by the method of local reduction.* - Doklady NAN B, 1999, v. 43, No.5, pp.5-8. (in Russian).

A.D.Zakrevskij
Institute of Engineering Cybernetics,
National Academy of Sciences, Belarus,
Surganov str. 6, Minsk, 220012,
Phone: 172-842182
e-mail: zakr@newman.bas-net.by

Received February 5, 2000