# A PROOF OF INEQUALITY OF THE CLASSES OF DECISION PROBLEMS P AND NP

**Angelo Raffaele Meo**

**Accademia delle Scienze di Torino**

**Politecnico di Torino**

**Abstract**

This paper is a new version of three papers presented to the Academy of Sciences of Turin in 2016 and to the Journal of Computer Science in 2020 and 2022 **(Ref(1)), (Ref (2), (Ref(3))..** According to the Journal of Computer Science, more than 4000 readers have "viewed" the two papers published by that journal and more than 1000 readers have downloaded them.

This new paper contains the answers to the questions asked by some readers and the proofs of some theorems which had been omitted for the sake of brevity. Almost all these answers and these proofs have been presented as appendixes in order to reduce the time necessary to read this paper.

The analysis discussed in this paper and in its previous versions, is based on a well-known NP-complete problem which is called "satisfiability problem" or " SAT**"**. From SAT a new NP-complete problem, called "core function", derives; this problem is described by a Boolean function of the number of the clauses of **SAT.** In this paper a proof is presented according which the number of gates of the minimal implementation of core function increases with **n** exponentially**.** Since the synthesis of core function is an NP**-**complete problem, this result can be considered as the proof of the theorem which states that the class **P** of all the decision problems which can be solved in polynomial time does not coincide with the class **NP** of the problems for which an answer can be verified in polynomial time.

## 1. DEFINITIONS

A brief description of the definitions and properties well known among the scientists of modern computational complexity theory is presented in this section.

**P** denotes the class of all the decision problems which can be solved in polynomial time.

**NP** denotes the class of all the decision problems **f** satisfying the property that the function **check(f)** analyzing a witness of the decision problem is polynomial time decidable.

“**P=NP?**”, or, in other terms, “Is **P** a proper subset of **NP**?”, is one of the most important open questions in modern computational complexity theory.

A decision problem **C** in **NP** is **NP-complete** if it is in **NP** and if every other problem **L** in **NP** is reducible to it, in the sense that there is a polynomial time algorithm which transforms instances of **L** into instances of **C** producing the same output values.

The importance of NP-completeness derives from the fact that, if we find a polynomial time algorithm for just one NP-complete problem, then we can construct polynomial time algorithms for all the problems in **NP** and, conversely, if any single NP-complete problem does not have a polynomial time algorithm, then no NP-complete problem has a polynomial time solution.

The analysis discussed in this paper will be based on the following well-known NP-complete problem which is called “satisfiability problem or **SAT”.**

Given a Boolean expression containing only the names of variables (some of which may be complemented), the operators **AND**, **OR** and **NOT**, and parentheses, is there an assignment of **TRUE** or **FALSE** values to the variables which makes the entire expression **TRUE**?

It is well known that the problem remains NP-complete also when all the expressions are written in “conjunctive normal form” with **3** variables per clause (problem **3SAT**). In this case, the analyzed expressions will be of the type:

$$\begin{aligned} &\mathbf{3SAT(t) =} \\ &(\, x_{11} \text{ OR } x_{12} \text{ OR } x_{13} \,) \text{ AND} \\ &(\, x_{21} \text{ OR } x_{22} \text{ OR } x_{23} \,) \text{ AND} \\ &\ldots\ldots\ldots\ldots \\ &(\, x_{t1} \text{ OR } x_{t2} \text{ OR } x_{t3} \,) \end{aligned} \qquad (1)$$

where:

**t** is the number of clauses or triplets;

each $\mathbf{x_{ij}}$ is a variable in complemented or uncomplemented form;

each variable may appear multiple times in that expression.

Usually, the deterministic Turin machine is assumed as the computational model. In this paper analysis will be developed with reference to a family $\mathbf{\{C_n\}}$ of Boolean circuits, where $\mathbf{C_n}$ has **n** binary inputs and it produces the same binary output as the corresponding Turing machine.

The equivalence between a deterministic Turing machine **M** processing some input **x** belonging to $\mathbf{\{0,1\}^n}$ and an **n**-input Boolean circuit $\mathbf{C_n}$ is well known. It is also known that the number of gates, or **AND, OR, NOT** operators, appearing in circuit $\mathbf{C_n}$, is polynomial in the running time of the corresponding Turing machine.

The synthesis of the state of art of question **PvsNP** can be found in **Ref(4)** and **Ref(5).**

## 2. THE CORE FUNCTION

The Booleuan circuit implementing the function described by **Eq(1)** will be called $\mathbf{C_t}$ or $\mathbf{C_n}$. Indeed, the number **t** of triplets appearing in **Eq(1)** plays the role of symbol **n** used in the standard complexity theory. In the following analysis, we shall use the symbol **t** when it is necessary to remember the number of triplets and **n** in the other cases.

In order to simplify analysis, circuit $C_n$ will be decomposed into two processing layers called "compatibility layer" and "core layer".

## Compatibility layer

A variable **j** of triplet **i** will be defined as "**compatible**" with variable **k** of triplet **h** when, and only when, either

- the sign $s_{ij}$ of the former variable is equal to the sign $s_{hk}$ of the latter variable,

or

- the name $<n_{ij1}\ n_{ij2}\ \ldots\ n_{ijm}>$ of the former variable is different from the name $<n_{hk1}\ n_{hk2}\ \ldots n_{hkm}>$ of the latter variable.

From that definition it follows that two "**not compatible**" variables have different signs and the same name; therefore, their **AND** is identically **FALSE**.

The compatibility layer is composed of **3·t·(3·t-3)/2** identical cells, one for each pair of variables belonging to different triplets.

The inputs of a cell will be the sign $s_{ij}$ and the binary code $<n_{ij1}\ n_{ij2}\ \ldots n_{ijm}>$ of the name of variable **j** of triplet **i**, and the sign $s_{hk}$ and the binary code $<n_{hk1}\ n_{hk2}\ \ldots n_{hkm}>$ of variable **k** of triplet **h**. The output of the same cell **c(i,j;h,k)** will be **TRUE** when, and only when, the two variables are compatible between themselves.

Therefore, the function implemented by a cell may be written as follows (by using the symbols $\ast$, **+**, and **!** for representing **AND, OR** and **NOT** operators, respectively):

$$\begin{aligned}
c(i,j;h,k) &= s_{ij}\ast s_{hk} + !s_{ij}\ast !s_{hk} + \\
&+ n_{ij1}\ast !n_{hk1} + !n_{ij1}\ast n_{hk1} + \\
&+ n_{ij2}\ast !n_{hk2} + !n_{ij2}\ast n_{hk2} + \\
&\ldots\ldots\ldots\ldots\ldots\ldots \\
&+ n_{ijm}\ast !n_{hkm} + !n_{ijm}\ast n_{hkm}
\end{aligned} \tag{2}$$

Variable **c(i,j;h,k)** will be called a "compatibility variable" or simply a "compatibility".

## Core layer

The Boolean function implemented by the core layer will be called the "**Core Function**" of order **t**, where **t** is the number of triplets. It will be denoted with the symbol **CF(t)** (or **CF(n)**). The core layer processes only the **9·t·(t-1)/2** compatibility variables **c(i,j;h,k)** and produces the global result of computation.The Core Function can be determined by proceeding as follows.

Consider one selection of variables appearing in **Eq(1)**, one and only one for each triplet, for all the triplets. Let

$$<1i_1>, <2i_2>, \ldots, <ti_t> \quad (3)$$

with $i_1, i_2, \ldots, i_t \in \{1, 2, 3\}$

be the indexes **<number of triplet, number of variable in the triplet>** of the selected variables. They will be called "characteristic indexes". Let $\Pi^k$ be the product of all the compatibility variables relative to the **k-th** selection (**3**):

$$\Pi^k = c(1,i_1; 2,i_2) * c(1,i_1; 3,i_3) * \ldots$$
$$\ldots * c(t-1,i_{t-1}; t, i_t) \quad (4)$$

The core function can be defined as the sum

$$\Sigma_k \Pi^k \quad (5)$$

of the products **(**4**)** relative to all the selections **(3)**.

For example, in the case of **CF(3)**, the core function can be defined as follows:

$$\begin{aligned} CF(3) = {} & c(1,1;2,1)*c(1,1;3,1)*c(2,1;3,1) + \\ & c(1,1;2,1)*c(1,1;3,2)*c(2,1;3,2) + \\ & c(1,1;2,1)*c(1,1;3,3)*c(2,1;3,3) + \\ & c(1,1;2,2)*c(1,1;3,1)*c(2,2;3,1) + \\ & \ldots\text{(other 22 products)}\ldots + \\ & c(1,3;2,3)*c(1,3;3,3)*c(2,3;3,3) \end{aligned} \quad (6)$$

It is easy to prove that there is an assignment of values **TRUE** or **FALSE** to variables appearing in **Eq(1)** which make the value of that equation equal to **TRUE** when, and only when, the core function takes the value **TRUE**. The proof of this theorem is presented in **Appendix 1.**

Notice that the processing work of a cell increases as a polynomial function **P(t)** of the number of the variables since the increment of the length of the code of the name is logarithmic. Therefore, the total processing work of the compatibility layer increases as $9 \cdot t \cdot (t-1) \cdot P(t)$ where $9 \cdot t \cdot (t-1)/2$ is the total number of the compatibility cells.

Besides, the problem solved by the core layer is clearly in **NP**, because it is easy to verify a witness solution. It follows that, since the compatibility layer polynomially reduces an NP-complete problem **(3SAT)** to the problem solved by the core layer, the core function describes a new NP-complete problem.

Some properties of core function have been discussed in **Ref(7)**

## 3. A THEOREM OF BOOLEAN MONOTONIC FUNCTIONS

Let $f(x_1,x_2, \ldots, x_h)$ be an isotonic Boolean function, that is a Boolean function which can be implemented with only **AND** and **OR** gates, applied to uncomplemented literals $x_1, x_2, \ldots, x_h$. It was believed that the minimum cost implementation of $f(x_1,x_2,\ldots,x_h)$ always contains only **OR** and **AND** gates, but A.Razborov proved that there are isotonic functions whose minimum cost implementation contains also **NOT** gates (see **Ref(6)** ).

However, there is on upper bound on the comparison of the costs of the minimum cost implementations with and without **NOT** gates. It is specified by the following theorem.

THEOREM 4.1

Let $\mathbf{I_{min}}$ be one of the minimum cost implementations of the isotonic Boolean function $\mathbf{f(x_1, x_2,\ldots,x_h)}$, the cost being defined as the total number of **AND**, **OR** or **NOT** gates. Let $\mathbf{C_{min}}$ be the cost of $\mathbf{I_{min}}$.

There exists always an implementation **J** of **f** containing only **AND** and **OR** gates (in addition, if necessary, to the **NOT** operators producing input variables $\mathbf{!x_1}$, $\mathbf{!x_2}$, ... ,$\mathbf{!x_h}$ ) such that

$$\mathbf{cost\ (J) <= 2 \cdot C_{min} + h}$$

where **h** is the number of variables.

The proof of this theorem can be found in **Appendix 2.**

This theorem will be used to simplify the analysis of Core Function circuits.

## 4. PROPERTIES OF CORE FUNCTION

It is easy to prove the following properties of core function.

PROPERTY 1

Function defined by **Eq(5)** is totally isotone.

PROPERTY 2

Any product defined by **Eq(4)** is a prime implicant of core function (that is, a product of compatibilities ("**PoC**") which implies core function and no other term of it).

PROPERTY 3

Since the different selections of each of variables defined by **Eq(3)** are **3**, the number of prime implicants of core function is equal to $\mathbf{3^t}$. Each of these prime implicants is essential (that is, it does not imply a sum of other prime implicants) and it is the product of $\mathbf{t \cdot (t-1)/2}$ compatibilities.

## 5. PRODUCTS OF COMPATIBILITIES

In the next sections, reference will be made to the following definitions.

DEFINITION OF SPURIOUS COMPATIBILITIES PAIR

A pair of compatibility variables {**c(h,k;l,m), c(p,q;r,s)**} is defined as a spurious pair if

$$\mathbf{( h = p \text{ and } k \neq q )}$$

**or** $\mathbf{( h = r \text{ and } k \neq s )}$

**or** **( l = p and m ≠ q )**

**or** **( l = r and m ≠ s )**

For example, the pair {**c(1,1;2,1), c(1,2;3,1)**} is a spurious pair since the triplet **1** is associated to two different indexes of variables (**1** and **2**).

### <u>Definition of spurious products of compatibilities</u>

A spurious product of compatibilities (spurious **PoC**) is a product of compatibility variables containing the elements of one or more than one spurious pair.

For example, the **PoC**

**c(1,1;2,1)∗c(1,2;3,1)∗c(2,1;3,1)**

is a spurious **PoC** since it contains the elements of the spurious pair

**{c(1,1;2,1), c(1,2;3,1)}**

### <u>Definition of impure products of compatibilities</u>

A **PoC** containing one or more complemented variables will be defined as an impure **PoC**. In particular a term **T** of **CF** (that is, a **PoC** implying **CF**) which contains one or more complemented variables, will be defined as an impure term of **CF**. A product of compatibilities which is neither spurious nor impure will be defined as a pure product of compatibilities.

### <u>Definition of mark</u>

Consider a pure product of compatibilities satisfying the property that all the indexes of triplet **{1,2,...,t}** appear at least once in some variable. The product of the variables of such a subset will defined as a "mark" or "pure mark" of the prime implicant of which it contains a subset of compatibilities.

For example, in the case of **CF**(**4**), the **PoC**

$$\mathbf{M = c(1,a;2,b)*c(1,a;3,c)*c(1,a;4,d)} \quad \mathbf{(7)}$$

(where the indexes of triplet are elements of the set **{1,2,3,4}** and **a, b, c, d** are elements of **{1,2,3}**)

is a mark of the prime implicant

$$\mathbf{P= c(1,a;2,b)*c(1,a;3,c)*c(1,a;4,d)*c(2,b;3,c)*c(2,b;4,d)*c(3,c;4,d)} \quad \mathbf{(8)}$$

since all the indexes of triplet appear at least once in **Eq.(7).**

### <u>Definition of spurious mark</u>

A spurious **PoC** in which all the indexes of triplet appear at least once will be called a "spurious mark". Notice that a spurious mark may be the mark of more than one prime implicant. For example, in the case of **CF(3),**

**c(1,1;2,1)∗c(1,1;3,1)∗c(1,1;2,2)**

is a spurious mark of both the prime implicants

**c(1,1;2,1)∗c(1,1;3,1)∗c(2,1;3,1)**

and

**c(1,1;2,2)∗c(1,1;3,1)∗c(2,2;3,1)**

An impure **PoC** containing a (possibly spurious) mark will be a defined as a (possibly spurious) impure mark.

DEFINITION OF EXTENDED PRIME IMPLICANT

A term **T** of core function, that is, an implicant of core function (a product of literals implying core function), contains all the uncomplemented literals of a prime implicant. Therefore, it may be defined as an "extended prime implicant" (only) to remember that it contains all the compatibilities of a prime implicant.

It may be a spurious extended prime implicant or an impure extended prime implicant or both a spurious and impure extended prime implicant.

Notice that an extended prime implicant can be viewed as a (possibly spurious or impure) mark.

DEFINITION OF REMAINDER

A PoC which is not a mark will be called a "remainder". Also a remainder may be pure (if for any triplet index there is only one index of variable in that triplet) or spurious or impure.

A pure remainder **R** may be implied by more than one prime implicant. For example, in the case of **CF(3)**, **R=c(2,1;3,1)** is a remainder which is implied by the following prime implicants

**P1 = c(1,1;2,1)∗c(1,1;3,1)∗c(2,1;3,1)**

**P2 = c(1,2;2,1)∗c(1,2;3,1)∗c(2,1;3,1)** **(9)**

**P3 = c(1,3;2,1)∗c(1,3;3,1)∗c(2,1;3,1)**

On the definitions of mark and remainder the following property is based.

PROPERTY 4

Let **$P_1$** and **$P_2$** be two **PoC's** such that **$P_1 * P_2$** is equal to a prime implicant **P** of core function. Either **$P_1$** or **$P_2$** is a mark of **P. Appendix 3** contains a proof of this property,

## 6. THE EXTERNAL CORE FUNCTION

Let **$I_j$** be a prime implicant of **CF(n)**. The external core function relative to **$I_j$**, **ECF(n,$I_j$)**, is defined as the sum of all the minterms of **CF(n)** which imply **$I_j$** and no other prime implicant **$I_k$** of **CF(n)** with **k≠j**. (Remember that a minterm of a Boolean function **F** is a product of all the variables of **F**, some complemented and some other uncomplemented, implying **F**).

Of course,

$$\mathbf{ECF(n,I_j) = I_j * \Pi_{k \neq j} (!I_k)} \qquad \mathbf{(10)}$$

where all the prime implicants of core function are involved and **!I$_k$** denotes the complement of **I$_k$** (i.e., **NOT I$_k$**).

The global external core function of order **n,** or **ECF(n)**, will be defined as the sum of **ECF(n, I$_j$)**'s relative to all the prime implicants **I$_j$** of **CF(n)**:

$$\mathbf{ECF(n) = \textstyle\sum_j ECF(n, I_j)} \qquad \mathbf{(11)}$$

The importance of external core function derives from the following theorems.

THEOREM 7.1

Let **T** be a term (or extended prime implicant) of **CF(n)**. It may be the product of all the compatibilities of a prime implicant **I$_j$** of **CF(n)** and other compatibilities, that is,

$\mathbf{T = I_j * X}$

where **X** is a possibly empty **PoC.** **T** can also be written as **T = T(I$_j$).**

All the minterms of **T(I$_j$)** contained in **ECF(n)** are minterms of **ECF(n,I$_j$).**

THEOREM 7.2

Let **T** be a term of **CF (n)** implying two or more than two prime implicants of **CF(n)** :**T = T (I$_j$, I$_k$).**

The number of minterms of **T(I$_j$,I$_k$)** belonging to **ECF(n)** is equal to **0**.

THEOREM 7.3

Let **T = T (I$_j$) = I$_j$∗X** be a term of **CF(n)** which is spurious for a single not complemented compatibility **X**.

If **NMT(F)** denotes the number of minterms of Boolean function **F,** the number of minterms of **I$_j$∗X** contained in **ECF(n,I$_j$ )** is

$$\mathbf{NMT(I_j * X * ECF(n,I_j)) <= (1/2) \cdot NMT(ECF(n, I_j))} \qquad \mathbf{(12)}$$

However, for large values of **n,**

$$\mathbf{NMT(I_j * X * ECF(n,I_j)) \approx (1/2) \cdot NMT(ECF(n, I_j))} \qquad \mathbf{(13)}$$

The proofs of **Eq(12)** and **Eq(13)** can be found in **Appendix 4**.

By proceeding in the same way it is possible to generalize the preceding THEOREM 7.3 as follows.

THEOREM 7.4

Let

$\mathbf{I_j * X_1 * X_2 * \ldots X_m}$

be a spurious term characterized by **m s**purious not complemented compatibilities.

The number of its minterms contained in **ECF(n, I$_j$)** is

$$\mathbf{NMT(I_j * X_1 * X_2 * \ldots * X_m * ECF(n, I_j)) <= (1/(2^m)) \cdot NMT(ECF(n, I_j)} \qquad \mathbf{(14)}$$

However, for large values of n,

$$\mathbf{NMT(I_j * X_1 * X_2 * \ldots * X_m * ECF(n, I_j)) \approx (1/(2^m)) \cdot NMT(ECF(n, I_j)} \qquad \mathbf{(15)}$$

<u>THEOREM 7.5</u>

Let **T=T (I$_j$)** be an impure term of **CF(n)** characterized by a single impure variable **(!X)** :

$\mathbf{T = I_j * (!X)}$.

For large values of n, the number of minterms of **ECF(n,I$_j$)** contained in **T** is

$$\mathbf{NMT(I_j * (!X) * ECF(n,I_j)) \approx (1/2) \cdot NMT(ECF(n, I_j))} \qquad \mathbf{(16)}$$

The proof of this theorem can be found in **Appendix 5**.

<u>THEOREM 7.6</u>

Let **T=T(I$_j$)** be an impure term of **CF(n)** characterized by **m** impure variables:

$\mathbf{T=I_j * (!X_1) * (!X_2) * \ldots (!X_m)}$

For large values of **n**, the number of minterms of **ECF(n,I$_j$)** contained in **T** is

$$\mathbf{NMT(T * ECF(n,I_j)) \approx ((1/2)^m) \cdot NMT(ECF(n,I_j))} \qquad \mathbf{(17)}$$

This theorem is an obvious extension of **Theorem 7.**5.

Notice that **NMT(ECF(n,I$_j$)) = NMT(ECF(n,I$_k$))** for any **j** and **k.** It will be called **NMT1(n)**

## 7. THE VALUE OF A NODE

Let **U** be a node of the network implementing Core Function and let **F(U)** be the Boolean function of compatibilities **c(i,j,h,k)** implemented by **U**. Since the subnetwork having **U** as its input does not contain any **NOT** gate, we can write:

$$\mathbf{CF = F(U) * x_1 + F(U) * x_2 + \ldots + y_1 + y_2 + \ldots} \qquad \mathbf{(18)}$$

where $\mathbf{x_1, x_2, \ldots, y_1, y_2, \ldots}$, are products of variables of Core Function, that is , products of compatibilities. Notice also that every **F(U)*x$_i$** and every **y$_j$** must be an extended prime implicant of core function. As we shall see in some examples, generally a single product of compatibilities is sufficient to implement core function according to the following equation:

$$\mathbf{CF = F(U) * x + y_1 + y_2 + \ldots} \qquad \mathbf{(19)}$$

where **x** is a single compatibility.

**$x_1$, $x_2$, ..., $y_1$, $y_2$, ...** ,or **x** will be called “completion code”.

More than one solution of **Eq(18)** or **EQ(19)** can produce Core Function. However, we are looking for a solution characterized by the following property: the total number of minterms of the external core functions **ECF(n,$I_j$)** of the prime implicants produced by **F(U) * $x_1$ + F(U) * $x_2$** or by **F(U) * x** takes the maximum value. By definition, this maximum value will be considered as the value **val(U)** of the node **U** or the value **val(F(U))** of the Boolean function implemented by **U**.

Besides, for the sake of simplicity, the following data will be assumed.

1. The value of a node which is a pure mark will be assumed as **NMT1(n)**. Indeed, for example, the mark **c(1,1;2,1) * c(1,1;3,1) * c(1,1;4,1),** multiplied by the remainder **c(2,1:3,1) * c(2,1;4,1) * c(3,1;4,1),** becomes a prime implicant of CF(4). It is easy to prove that the value of a node which is a sum of matks and remainders is less than, or equal to the sum of the values of of the involved marks.
2. The value of a remainder is equal to **0**. The value of a sum of remainders is equal to **0**. This choice is justified by observing that a remainder must be multiplied by a mark in order to produce a prime implicant and that mark is generated outside the circuit producing node **U**
3. Variables **x, $x_1$, $x_2$**, … must be remainders. This choice is justified by observing that they are generated outside the circuit producing node **U.**

The reasons for these definitions are discussed in **Appendix 13.**

The values of **$x_1$, $x_2$, ..., $y_1$, $y_2$, ...** ,or **x** which appear in the best solution of **Eq(18)** or **Eq(19)** will be called “optimal completion code”.

## 8, THE VALUE OF AN OR GATE

An **n** inputs **OR** gate can be implemented as a set of **(n-1)** two inputs **OR** gates. Therefore, we can restrict our attention to two inputs **OR** gates.

The value of an **OR** gate having **A** and **B** as its inputs and **U** as its output can be defined as:

**val (A OR B) = val (U) – ( val(A) + val(B) )**

It is easy to prove the following simple rules for evaluating the value of functions **F(U), F(A)** and **F(B),** under the hypothesis that these three functions are written as the sums of their prime implicants which will become the remainders and the marks of core function.

1. The value of a sum of marks is always equal to, or less than, the sum of the values of the prime implicants of core function which imply those marks.That is, **val(U)** is always less or equal to **val(A) + val(B).** The proof of this property is summerized in **Appendix 6,**
2. Theoretically, a mark might derive from the Boolean sum of two or more than two remainders. For example, the mark of **CF(4) m = c(1,1;4,1) ⋆ c(2,1;4,1) ⋆ c(3,1;4,1)** might derive from the sum of the two remainders **$r_1$ = c(1,1;4,1) ⋆ c(2,1;4,1) ⋆ !c(1,1;3,1)** and and **$r_2$ = c(3,1;4,1) * c(1,1;3,1).** Let remainders **$r_1$** and **$r_2$** be two of the inputs of the **OR** operation producing mark **m** and let **U** be the output of this **OR** operation. Since the circuit producing **CF** does not contain **NOT** circuits, the value of the circuit producing **CF** can be written as follows:

**CF = U*$x_1$+U*$x_2$+…+$y_1$+$y_2$+… = $r_1$*$x_1$+$r_2$*$x_1$+$r_1$*$x_2$+$r_2$*$x_2$+…+$y_1$+$y_2$+ …**

Since $r_1$ and $r_2$ are remainders, every $x_i$ must be a mark. Besides, either there is a $y_k$ equal to the prime implicant **I(m)** deriving from mark **m** or one of the products $r_i * x_j$ is equal to **I(m)**, and, therefore, the mark of **I(m)** is produced outside the considered **OR** operation. It follows that the production of a mark as the sum of two remainders can not be used in order to generate new prime implicants.

From these rules it is easy to prove that **val(U)** is never larger than **val(A) + val(B)** and, therefore, the value of an **OR** gate can be always considered as equal to **0**.

## 9. THE VALUE OF AN AND GATE. THE MOST POWERFUL AND GATE.

As in the case of **OR** gates, an **n** inputs **AND** gate can be implemented as a set of **(n-1)** two inputs **AND** gates. Therefore, we can restrict our attention to two inputs **AND** gates.

The value of an **AND** gate having **A** and **B** as its inputs and **U** as its output can be defined as:

$$\mathbf{val\,(A\ \ AND\ \ B)\ =\ val\,(U) - (\,val(A) + val(B)\,)}$$

Since we are interested in identifying the most powerful **AND** gate, we shall assume that both **F(A)** and **F(B)** are sums of remainders so that both **val(A)** and **val(B)** are equal to **0** and the value of the considered gate is equal to the value of output **U.**

The most powerful **AND** gate can be identified by proceeding as follows.

1. Let $\mathbf{A = (a_1 + a_2 + a_3 + \ldots)}$ and $\mathbf{B = (b_1 + b_2 + b_3 + \ldots)}$, where all the $\mathbf{a_i}$ and $\mathbf{b_j}$ are remainders.
2. Consider the product $\mathbf{a_1 * b_1}$ if it is a mark.
   If $\mathbf{a_1}$ is a remainder, at least one of the **t** indexes of triplet does not appear in the list of triplet indexes of $\mathbf{a_1}$ because, otherwise, $\mathbf{a_1}$would be a mark. Let it be **i'**.
   For the same reason, at least another triplet index does not appear in the list of triplet indexes of $\mathbf{b_1}$. Let it be **j'**.
   By example:
   $\mathbf{a_1 = c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1)}$ **(20)**
   $\mathbf{b_1 = c(1,1;4,1) * c(2,1;4,1)}$
   $\mathbf{m_1 = a_1 * b_1}$
   Triplet index **4** is missing in $\mathbf{a_1}$; triplet index **3** is missing in $\mathbf{b_1}$. In order that $\mathbf{a_1 * b_1 * x}$ is a prime implicant of **CF(4)**, **x** must be equal to **c(3,1;4,1)**.
3. **Eq(20)** is the example of two remainders whose product is a mark without spurious or impure variables. Obviously, the value of that mark is **NMT1(4)**.
   According to Theorems **7.3, 7.4, 7.5, 7.6**, a mark containing a spurious or impure compatibility has a value equal to about $\mathbf{(1/2)\cdot NMT1(n)}$ while a mark containing **m** spurious or ·impure compatibilities has a value equal to about $\mathbf{(1/2^m)\cdot NMT1(n)}$.
4. Assume that $\mathbf{a_2 * b_1}$ is equal to a new mark as
   $\mathbf{m_2 = c(1,2;2,1) * c(1,2;3,1) * c(2,1;4,1)}$ **(21)**
   and $\mathbf{m_2 * x}$ is a new prime implicant of **CF(4).**
   We can start by assuming that $\mathbf{a_2}$ is equal to **c(1,2;2,1) * c(1,2;3,1) * c(2,1;3,1)**.
   Since the optimal completion code **x** is equal to **c(3,1;4,1)** and $\mathbf{a_2}$ cannot contain all the three compatibilities involving **<1,2>**, the value of $\mathbf{b_1}$ must be corrected by adding **c(1,2;4,1)** to $\mathbf{b_1}$**:**
   $\mathbf{b_1' = b_1 * c(1,2;4,1)}$
   Therefore,
   $\mathbf{val\,(a_1 * b_1') = (1/2)\cdot NMT1\,(4)}$
   $\mathbf{val\,(a_2 * b_1') = (1/2)\cdot NMT1\,(4)}$
   No increment of the total value has been obtained by introducing a new mark.
5. In order to implement the new mark $\mathbf{m_2}$ without reducing the value of $\mathbf{m_1}$ it is necessary to

introduce a new remainder

**$b_2$ = c(1,2;4,1) * c(2,1;4,1)**

so that

**$m_2 = a_2 * b_2$**

6. The result stated in 4. can be extended as follows.
   The most powerful **AND** gate can be obtained by producing every mark as the product of a remainder **$a_i$** by a corresponding remainder **$b_j$**, and it is not useful to produce two marks **$m_1$** and **$m_2$** by means of three remainders as follows:
   **$m_1 = a_i * b_j$**
   **$m_2 = a_k * b_j$**
   This result is proved in **Appendix 7.**

7. However, the products **$a_1 * b_2$** and **$a_2 * b_1$** are not marks. Therefore, it is necessary to introduce the following corrections:

$$\begin{aligned} a_1 &= c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1) \\ b_1 &= c(1,1;4,1) * c(2,1;4,1) * c(1,1;2,1) \\ a_2 &= c(1,2;2,1) * c(1,2;3,1) * c(2,1,3,1) * !c(1,1;2,1) \qquad (22) \\ b_2 &= c(1,2;4,1) * c(2,1;4,1) * c(1,2;2,1) * !c(1,1;2,1) \end{aligned}$$

   so that:

   **val ($m_1 = a_1 * b_1$) = NMT1(4)**
   **val ($m_2 = a_2 * b_2$) = NMT1(4) · (1/2)**

   Notice that **c(1,1;2,1)** appears in both **$a_1$** and **$b_1$** and **c(1,2;2,1)** appears in both **$a_2$** and **$b_2$**.

   According to the corrections introduced in **Eq(22)**, both the products **$a_1 * b_2$** and **$a_2 * b_1$** are equal to **0.**
   It is also possible to introduce two spurious compatibilities in order that both **$a_1 * b_2 * x$** and **$a_2 * b_1 * x$** become prime implicants of core function.
   For example,

$$\begin{aligned} a_1 &= c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1) \\ b_1 &= c(1,1;4,1) * c(2,1;4,1) * c(1,2,4,1) \\ a_2 &= c(1,2;2,1) * c(1,2;3,1) * c(2,1,3,1) \qquad (23) \\ b_2 &= c(1,2;4,1) * c(2,1;4,1) * c(1,1;4,1) \end{aligned}$$

   In this case,
   **val ($m_1 = a_1 * b_1$) = NMT1(4) · (1/2)**
   **val ($m_2 = a_2 * b_2$) = NMT1(4) · (1/2)**

   It follows that **Eq(22)** is better than **Eq(23).** It is easy to prove that **Eq(22)** represents the best solution to implement two marks from the viewpoint of the values..

8. The two pairs of remainders appearing in **($a_1 + a_2$) * ($b_1 + b_2$)** can produce four different marks. **Appendix 8** shows the best implementation for **CF(4)**. The value of one of these marks is equal to **(1/8) * NMT1(4)** and their total value is **(1/2) * NMT1(4).** Therefore, there is no point in continuing this line.

9. By following the same line of reasoning which has made it possible to prove that **Eq(22)** is

the best solution for implementing two marks, it is easy to prove that the best solution for implementing three marks is the following one:

**$a_1$ = c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1)**

**$b_1$ = c(1,1;4,1) *c(2,1;4,1) * c(1,1;2,1)**

**$a_2$ = c(1,2;2,1) * c(1,2;3,1) * c(2,1,3,1) * !c(1,1;2,1)**
**$b_2$ = c(1,2;4,1) * c(2,1;4,1) * c(1,2;2,1) * !c(1,1;2,1)** **(24)**

**$a_3$ = c(1,3;2,1) * c(1,3;3,1) * c(2,1;3,1) * !c(1,1;2,1) * !c(1,2;2,1)**
**$b_3$ = c(1,3;4,1) *c(2,1;4,1) * !c(1,1;2,1) * !c(1,2;2,1)**

The value of this solution is
**(1 + (1/2) + (1/4) ) · NMT1(4)**

10. **Appendix 9** shows the best solution for implementing the marks of all the nine prime implicants of **CF(4)** compatible with the conditions that the variables **<3,2>, <3,3>, <4,2>, <4,3>** do not appear in that product and the completion code **x** takes the value **c(3,1;4,1).**
The value of the gate implementing those imarks is

$$(1+(1/2) + (1/4)) \cdot (1+(1/4)+(1/16)) \cdot \mathrm{NMT1}(4) \qquad (25)$$

which is slightly less than

$$(1+(1/2) + (1/4))^2 \cdot \mathrm{NMT1}(4) \qquad (26)$$

As shown in **Appendix 9**, **Eq(25)** and **Eq(26)** can be generalized according to the following equations on the

value of the best gate implementing the marks of **$3^{(n-2)}$** prime implicants of **CF(n)**

$$\mathrm{val1}(n) = (1+1/2+1/4))^{n-3} . (1+1/4+1/16) \cdot \mathrm{NMT1}(n) \qquad (27)$$

which is slightly less than

$$\mathrm{val2}(n) = (1+(1/2) + (1/4))^{n-2} \cdot \mathrm{NMT1}(n) \qquad (28)$$

In order to prove that the solution proposed in **Appendix 9** is the most powerful one, consider three marks which are different for the value of one and only one triplet index. For example, the three marks **$m_7$ = $a_7$ * $b_7$, $m_8$ = $a_8$ * $b_8$, $m_9$ = $a_9$ * $b_9$**, which have been defined in **Appendix 9**, are different only for the values in triplet index **1**. A set as **{$m_7$, $m_8$, $m_9$}** will be called a "set of connected marks".
In this example, in order that **$a_7$ * $b_8$ = 0** and **$a_8$ * $b_7$ = 0**, both **$a_8$** and **$b_8$** must contain compatibility **!c(1,1;2,3)**. Therefore, the value of mark **$m_8$** will be multiplied by **1/2.**
In order that **$a_7$ * $b_9$ = $a_9$ * $b_7$ = $a_8$ * $b_9$ = $a_9$ * $b_8$ = 0**, both **$a_9$** and **$b_9$** must contain **!c(1,1;2,3) *!c(1,2;2,3).** Therefore, the value of mark **$m_9$** will be multiplied by **1/4.**
No other solution makes it possible to reduce the values of **$m_8$** and **$m_9$** by a smaller value.
Every mark **m** appearing in **Appendix 9** belongs to **t** sets of connected marks.
It is easy to verify on the data of **Appendix 9** that all the triplets {**$m_i$, $m_j$, $m_k$**} of connected

marrks have received the same type of corrections and only those corrections have been applied.
Therefore, we can state that the solution proposed in this paper leads to the best solution and that the maximum value of an **AND** gate of the type above specified is slightly less than **val2(n) = $(1+1/2 + 1/4)^{n.2}$ · NMT1(n).**

## 10. TO COMPLETE THE MOST POWERFUL GATE

— So far all the new marks contained only **<3,1>** and **<4,1>** of triplet **3** or **4.** in the compatibilities used in the synthesis of core ffunction**..** This condition can be removed in order to try to increase the value of the considered **AND** gate.

For example, as shown in **Appendix 10**, we can add nine new remainders $\mathbf{a_{10} \ldots a_{18}}$ to $\mathbf{a_1 \ldots a_9}$ and $\mathbf{b_{10} \ldots b_{18}}$ to $\mathbf{b_1} \ldots b_9$, where the new remainders are obtained by replacing all the appearances of <**4,1**> with <**4,2**>. Thus nine new marks and nine new prime implicants will be generated but the value of the considered gate will not be doubled. Indeed, the optimal completion code **x**, which was **c(3,1;4,1**> becomes **c(3,1;4,1) * c(3,1;4,2)** and the value of all the marks will be multiplied by **(1/2).**

In **Appendix 10** the lists **($a_1 + a_2 + \ldots + a_9$)** and **($b_1 + b_2 + \ldots + b_{9u}$)** have been updated as follows:

**val2(n) =**

**($a_1 + \ldots + a_{10} + \ldots + a_{19+} + \ldots + a_{28} + \ldots + a_{37} + \ldots + a_{46} + \ldots + a_{55} + \ldots + a_{67} + \ldots + a_{73} + \ldots$)**

**($b_1 + \ldots + b_{10} + \ldots + b_{19} + \ldots + b_{28} + \ldots + b_{37} + \ldots + b_{46} + \ldots + b_{55} + \ldots + b_{67} + \ldots + b_{73} + \ldots$)**

where

**$a_{10}$, ...,**has been obtained from **$a_1$, ...,** and **$b_{10}$, ...,** has been obtained from **$b_1$, ...,** by replacing all the appearanees of **<4,1>** with **<4,2>**,

**$a_{19}$, ...** has been obtained from **$a_1$, ...,** and **$b_{19}$, ...** has been obtimal from **$b_1$,...**

by replacing all the appearauces of **<4,1>** with **<4,3>**, and so on, involving, in the right order,

**<3,2>** and **<4,1>, <3,2>** and **<4,2>, <3,2>** and **<4,3>, <3,3>** and **<4,1>, <3,3>** and **<4,2>, <3,3>** and **<4,3>** in the tables of **$a_i$**'s and **$b_j$**'s.

Also the complation code has been updated:

**x = c(3,1;4,1) * c(3,1;4,2) * c(3,1;4,3) * c(3,2;4,1) * c(3,2;4,2) * c(3,2;4,3) * c(3,3;4,1) * c(3,3;4,2) * c(3,3;4,3)**

Thus, the **81** prime implicants of **CF(4)** have been generated , but the value of many of them becomes very small because many complemented compatibilities must be introduced in order that all the products **$a_1$ * $b_j$** become equal to **0**. Besides, the multiplication of every mark by the completion code **x** has dramatically reduced the value of the corresponding prime implicants .

A better solution can be obtained by integrating the completion codes in the tables of remainders and by multiplying the new terms produced by complemented compatibilities in such a way that all the products **$a_1$ * $b_j$** with **i<>j** become equal to **0.**

A first analysis developed in **Appendix** 10 leads to the following result:

**val2(n) =**

**($a_1 + \ldots + a_{10} + \ldots + a_{19+} + \ldots + a_{28} + \ldots + a_{37} + \ldots + a_{46} + \ldots + a_{55} + \ldots + a_{67} + \ldots + a_{73} + \ldots$)**

multiplied by **(29)**

$$(b_1 + \ldots + b_{10} + \ldots + b_{19} + \ldots + b_{28} + \ldots + b_{37} + \ldots + b_{46} + \ldots + b_{55} + \ldots + b_{67} + \ldots + b_{73} + \ldots)$$

is nearly equal to:

$$2 \cdot (1 + (1/2) + (1/4))^{n-2} \cdot \mathrm{NMT1}(n).$$

A slightly different result is proved in **Appendix 13 (Eq(11.4) )**. The question of **Eq(29)** and **Eq(11.9)** will be briefly discussed in the next **Section 11.**

## 11 . THE VALUE OF THE MOST POWERFUL GATE

In **Appendix 10** e **Appendix 11** two different starting points for determining the value of the most powerful AND gate have been suggested (**Eq(10.1)** and **Eq(11.4) ).**

This problem of the evaluation of the value of the most powerful AND gate is very complex because of the difficulty of the correct specification not only of the value of **(A*B)** but also of **val(A)** and val**(B).** Fortunately, these evaluations are not necessary to conclude our proof of the question **PvsNP** as shown by the following analysis.

The product $(a_1+a_2+\ldots+a_9) * (b_1+b_2+\ldots+b_9)$ which have been defined by **Eq(9.1)** of **APPENDIX 9** does not produce all the prime implicants of Core Function. Indeed, the prime implicants containing variables different from those appearing in the completion code **x** (in our example: **<3,2>, <3,3>, <4,2>, <4,3>)** do not appear in the list of prime implicants which have been generated.

A simple solution for producing all the prime implicants of Core Function is the following one.

First, multiply $a_1, a_2, \ldots, a_9$ by **c(3,1;4,1).**

Then we can extend the list $(a_1 + a_2 + \ldots + a_9)$ with $(a_{10} + a_{11} + \ldots + a_{18})$ and the list $(b_1 + b_2 + \ldots + b_9)$ with $(b_{10} + b_{11} + \ldots + b_{18})$ in order to obtain all the marks and all the prime implicants containing both variables **<3,1>** and **<4,2>,** in addition to the marks and the prime implicants containing variables **<3,1>** and **<4,1>** obtained by the product $(a_1 + a_2 + \ldots + a_9) * (b_1 + b_2 + \ldots + b_9)$.

The product $(a_{10} + a_{11} + \ldots + a_{18}) * (b_{10} + b_{11} + \ldots + b_{18})$ produces the marks of all the nine implicants of Core Function containing only **<3,1>** and **<4,2>** and no other variable of triplets **3** amd **4.** Similarly, a new product $(a_{19} + \ldots + a_{27}) * (b_{19} + \ldots + b_{27}) * c(3,1;4,3)$ can produce all the prime implicants of Core Function containing only **<3,1>** and **<4,3>** of the variables of triplets **3** and **4**, while the product $(a_{28} + \ldots a_{36}) * (b_{28} + \ldots + b_{36}) * c(3,2;4,1)$ can produce all the prime implicants of Core Function characterized by variables **<3,2>** and **<4,1>.**

In this way the marks of all the prime implicants of Core Function will be produced by the product

$$(a_1 + \ldots + a_{10} + \ldots + a_{19} + \ldots + a_{28} + \ldots + a_{37} + \ldots + a_{46} + \ldots + a_{55} + \ldots + a_{67} + \ldots + a_{73} + \ldots) *$$

$$(b_1 + \ldots + b_{10} + \ldots + b_{19} + \ldots + b_{28} + \ldots + b_{37} + \ldots + b_{46} + \ldots + b_{55} + \ldots + b_{67} + \ldots + b_{73} + \ldots) \qquad (30)$$

where the nine pairs of variables **<3,1>, <4,1>; <3,1>, <4,2>; <3,1>,< 4,3>; <3,2>, <4,1>;**

**<3,2>, <4,2>; <3,3>,<4,1>; <3,3>, <4,2>; <3,3>, <4,3>** are involved.

The solution described in **Appendix 10** is obtained by integrating the completion codes in the tables of remainders contained in **Eq(30)** and by multiplying the new terms produced by complemented compatibilities in order that all the products $a_1 * b_j$ with **i<>j** become equal to **0,** as follows.

For all **$i<=9$**

**$a_i' = a_i * c(3,1;4,1)$**

**$b_i' = b_i * c(3,1;4,1)$;**

for all **$i>9$** and **$<= 18$**

**$a_i' = a_i * c(3,1;4,2) * !c(3,1;4,1)$**

**$b_i' = b_i * c(3,1;4,2) * !c(3,1;4,1)$.**

for all **$i>18$** and **$<= 27$**

**$a_i' = a_i * c(3,1;4,3) * !c(3,1;4,1) * !c(3,1;4,2)$**

**$b_i' = b_i * c(3,1;4,3) * !c(3,1;4,1) * !c(3,1;4,2)$**

and so on.

The final result of this line of corrections will be the following equation:

$$\text{val}\ ((a_1 + \dots + a_{10} + \dots + a_{19} + \dots + a_{28} + \dots + a_{37} + \dots + a_{46} + \dots + a_{55} + \dots + a_{64} + \dots + a_{73} + \dots) *$$

$$(b_1 + \dots + b_{10} + \dots + b_{19} + \dots + b_{28} + \dots + b_{37} + \dots + b_{46} + \dots + b_{55} + \dots + b_{64} + \dots + b_{73} + \dots)) = \qquad (10.1)$$

$$(1+1/2 + 1/4 + 1/8 + 1/16 + 1/32 + 1/64 + 1/128 + 1/256) \cdot (1 + 1/2 + 1/4) \cdot (1 + 1/4 + 1/16) \cdot NMT1(4) \approx$$

$$2 \cdot (1 + 1/2 + 1/4) \cdot (1 + 1/4 + 1/16) \cdot NMT1(4)$$

It is easy to prove that every elementary product as **$(a_{10} + a_{11} + \dots + a_{18}) * (b_{10} + b_{11} + \dots + b_{18})$** produces a subset of prime implicants of Core Function disjoint from the other subsets of prime implicants; that is, a prime implicant produced by a subset does not appear in any other subset. Besides, the value of a subset is the optimal one for that subset of prime implicants.

The question of the comparison of **Eq(10.1)** and **Eq(11.4)**, that is, the determination of the value of the most powerful gate, is interesting and important, but it is very difficult. However, as already stated, in order to solve the problem **PvsNP** the solution of that question is not necessary.

The solution here proposed starts from **Eq(30)** with the following corrections:

For all **$i<=9$**

**$a_i' = a_i * c(3,1;4,1)$**

**$b_i' = b_i * c(3,1;4,1)$;**

for all **$i>9$** and **$<= 18$**

**$a_i' = a_i * c(3,1;4,2)$** **(31)**

**$b_i' = b_i * c(3,1;4,2)$**

for all **$i>18$** and **$<= 27$**

**$a_i' = a_i * c(3,1;4,3)$**

**$b_i' = b_i * c(3,1;4,3)$**

and so on.

A very large number of the terms appearing in **Eq(31)** must be corrected with complemented compatibilities in order that, for any **$i<>j$, $a_i * b_j = 0$.** As mentioned,it is very hard to identify the best choice of these complemented compatibilities and to describe the most powerful **AND** gate in the implementation of **CF(n)**. However, since the number of products **$(a_l + a_{l+1} + \dots + a_{l+8}) * (b_l + b_{l+1} + \dots + b_{l+8})$** appearing in **Eq(31)** is **9** and it has been proved that each of these products has the value shown by **Eq(27)** and **Eq(28)**, it is obvious that the value of the most powerful **AND** gate is smaller than

$$\mathbf{valmax(n) = 9 \cdot (1+1/2 + 1/4)^{n-2} \cdot NMT1(n)} \qquad (32)$$

## 12 CONCLUSIONS

Since the number of minterms of **ECF(n)** contained in **CF(n)** is equal to $\mathbf{3^n \cdot NMT1(n)}$ and the value of a gate, that is the number of new minterms produced by a gate, is less than

$$\mathbf{valmax(n) = 9 \cdot (1+(1/2) + (1/4))^{n-2} \cdot NMT1(n)}$$

the number of gates necessary to implement **CF(n)** is larger than

$$\mathbf{3^n/(9\cdot((1+1/2+1/4)^{(n-2)}))}$$

and, therefore, it increases exponentially with **n**.

Since the synthesis of core function **CF(n)** is an **NP-**complete problem, this result is equivalent to proving that **P** and **NP** do not coincide.

## APPENDIX 1

In a satisfiability equation characterized by **t** clauses, or **3SAT(t)**, we can identify $\mathbf{3^t}$ different sequences of “satisfiability equation names”, or “**sen**”, as:

$$\mathbf{<x_{1i}, x_{2j}, x_{3h}, \dots , x_{tk}>} \qquad \mathbf{(1.1)}$$

where $\mathbf{x_{1i}}$ denotes **sen i** of triplet **1,** $\mathbf{x_{2j}}$ denotes **sen j** of triplet **2 ,** and so on.

If all the sequences **(1.1)** contain one or more than one **sen** repeated in both complemented and uncomplemented form as in the following example relative to five clauses satisfiability equation **3SAT(5)**:

**Bill , !Marc, Bill , !Mary, Marc**

where both **Marc** and **!Marc** appear, then there is no assignment of values **TRUE** or **FALSE** to all the **sen** making the whole equation **TRUE**.

On the contrary, if at least one seuence **(1.1)** does not contains a **sen** repeated in the complemented and uncomplemented forms, then the satisfiability equation has a solution. For example, the following sequence

**Bill * !Marc * !Mary * Bill * !Mary**

leads to the following solution of five clauses satisfiability equation:

**Bill = TRUE**

**Marc = FALSE**

**Mary = FALSE**

If **Eq(1.1)** is one of the sequences which do not contain one or more than one **sen** repeated in complemented and uncomplemented forms, then the following product of compatibilities takes the value **1(**or **TRUE)** :

**c(1, i; 2, j) * c(1,i; 3, h) * ... * c(1, i; t,k) * ... * c(2, j; 3, h), ...** **(1,2)**

Therefore, core function takes the value **1** **(**or **TRUE)** by virtue of **Eq(5)**.

By following the same line of reasoning, it is easy to prove that if core function **CF(t)** is equal **1**, then the corresponding satisfiability equation **3SAT(t)** is satisfied.

## APPENDIX 2

In order to prove this theorem, let us divide the gates of implementation $\mathbf{I_{min}}$ of **f** into different levels and let us modify $\mathbf{I_{min}}$ as follows.

At level **1** we place the gates all inputs of which coincide with the complemented or uncomplemented input variables $\mathbf{x_i}$ or **!**$\mathbf{x_i}$ (where !$\mathbf{x_i}$ denotes the complement of variable $\mathbf{x_i}$) .

Level **2** contains the gates whose inputs coincide with input variables or outputs of level **1** gates.

In general terms, level **q** contains the gates whose inputs coincide with input variables or outputs of levels less than **q**.

We can transform $\mathbf{I_{min}}$ into **J** by deleting **NOT** gates and adding new **AND** or **OR** gates as follows.

We start from level **1**.

For any level **1** **AND** gate we add an **OR** gate whose inputs are the complements of the inputs of the considered **AND** gate **(Fig. 2.1)**. Similarly, for any level **1** **OR** gate we add an **AND** gate whose inputs are the complements of the corresponding **OR** gate.

By virtue of such operations, for any output **u** of the level **1** gates a new node will be available in the new circuit we are generating whose value will be **!u.**

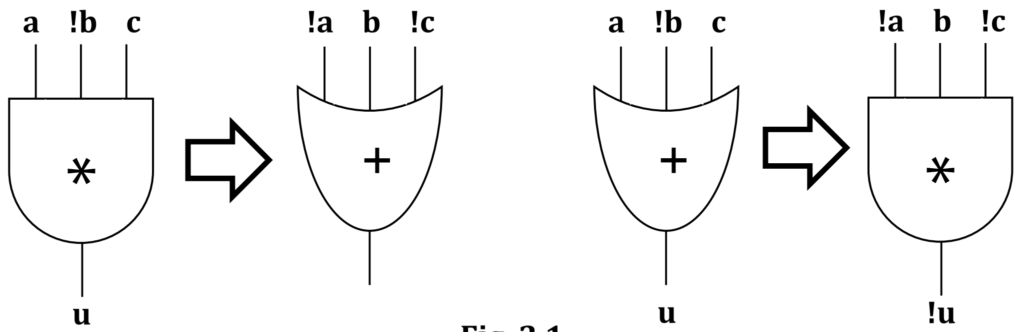

**Fig. 2.1**

**The new gates of level 1**

As a second step of processing, for any level **2** **AND** gate of implementation $\mathbf{I_{min}}$ we shall add an **OR** gate whose inputs are the complements of the inputs of the corresponding **AND** gate, in both the cases in which these inputs coincide with input variables of **f** or with outputs of level **1** gates (**Fig. 2.2**).

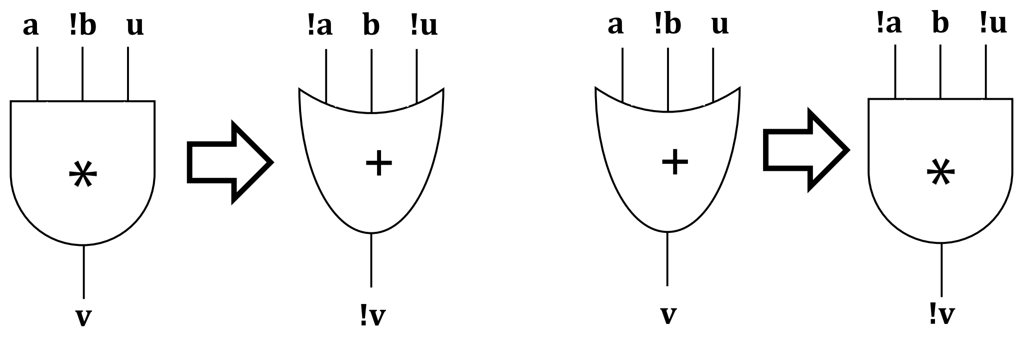

**Fig. 2.2**

**The new gates of level 2**

A similar transformation will be applied to all level **2 OR** gates.

As an example, the two level subnetwork of **2.3** will be transformed into the subnetwork of Fig. **2.4**. Notice that at the outputs of **J** not only the outputs **v** and **w** of $\mathbf{I_{min}}$ will be available, but also their complements !**v** and !**w**.

The preceding operations will be applied to all the levels of implementation of $\mathbf{I_{min}}$, in the order of increasing levels. It is apparent that, if for any input variable $\mathbf{x_i}$ also $\mathbf{!x_i}$ is available, the number of gates of **J** is less than, or equal to, twice the number of gates of $\mathbf{I_{min}}$.

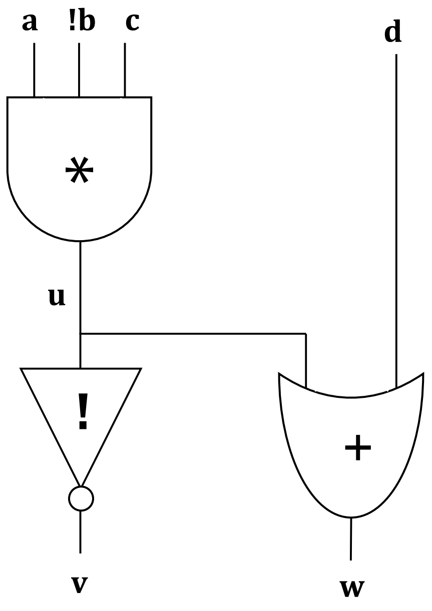

**Fig. 2.3**

**A two level subnetwork**

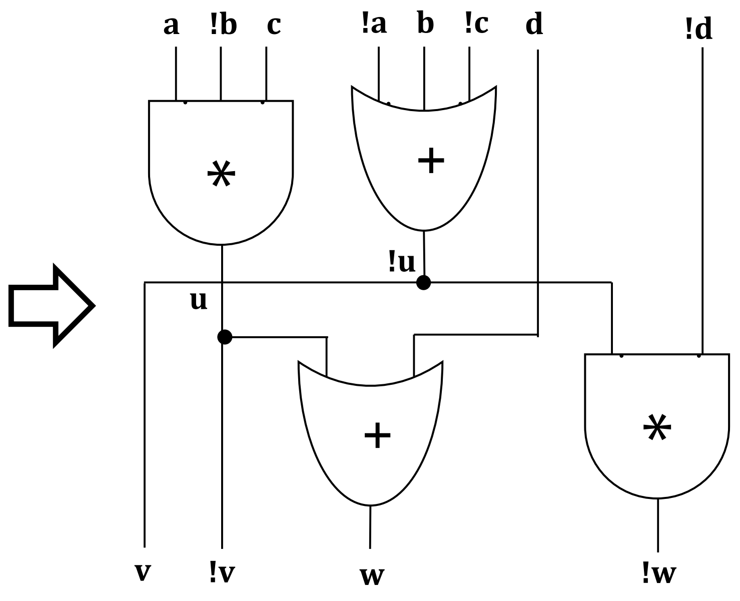

**Fig. 2.4**

**The transformation of the subnetwork of**

**F**

At level **0**, before the gates of **Fig. 2.3**, **h NOT** gates might be necessary to generate the complemented input variables **!$x_i$**. Therefore, **h** has been added in the statement of the theorem.

## APPENDIX 3

Assume that variable **U** of prime implicant **P** does not appear in **$P_1$** and variable **V** of **P** does not appear in **$P_2$**. The compatibility of **U** and **V** will be missing in the prime implicant that one wanted to implement as the product **$P_1$ * $P_2$.**

For example, let **$P_1$ = c(1,1; 4,1) * c(2,1; 4,1)** and **$P_2$ = c(1,1;2,1) * c(1,1; 3,1) * c(2,1;3,1)** be two remainders of **CF(4).** Than **<3,1>** does not appear in **$P_1$; <4,1>** does not appear in **$P_2$.** If we replace **$P_1$** with **$P_1$' = $P_1$ * c(3,1; 4,1),** which is a mark, we obtain prime implicant **$P_1$' * $P_2$.**

## APPENDIX 4

In order to prove Theorem 7.3 consider the following example relative to the external core function **ECF(4,$I_j$)** where **$I_j$** is the prime implicant defined by the mark **c(1,1;2,1) * c(3,1;4,1)** and **X = c(1,1;2,2)**.

**ECF(4, I) =**

**c(1,1;2,1) * c(1,1;3,1) * c(1,1;4,1) * c(2,1;3,1) * c(2,1;4,1) * c(3,1;4,1) ***

**( !c(1,1;2,1) + !c(1,1;3,1) + !c(1,1;4,2) + !c(2,1;3,1) + !c(2,1;4,2) + !c(3,1;4,2) ) ***

**( !c(1,1;2,1) + !c(1,1;3,1) + !c(1,1;4,3) + !c(2,1;3,1) +!c(2,1;4,3) + !c(3,1;4,3) ) ***

**( !c(1,1;2,1) +!c(1,1;3,2) + !c(1,1;4,1) + !c(2,1;3,2) + !c( 2,1;4,1) + !c(3,2;4,1) ) ***

. . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**( !c(1,1;2,2) +!c(1,1;3,1) + !c(1,1;4,1) + !c(2,2;3,1) + !c(2,2;4.1) + !c(3,1;4,1) ) ***

**( !c(1,1;2,2) + !c(1,1;3,1) + !c(1,1;4,2)+ !c(2,2;3,1) + !c(2,2;4,2) + !c(3,1;4,2) ) ***

. . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**( !c(1,1;2,3) + !c(1,1;3,1) + !c(1,1;4,1) + !c(2,3;3,1) + !c(2,3;4,1) + !c(3,1;4,1) ) ***

**( !c(1,1;2,3) + !c(1,1;3,1) + !c(1,1;4,2) + !c(2,3;3,1) + !c(2,3;4,2) + !c(3,1;4,2) ) ***

. . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**( !c(1,2;2,1) + !c(1,2;3,1) + !c(1,2;4,1) + !c(2,1;3,1) + !c(2,1;4,1) + !c(3,1;4,1) ) ***

**( !c(1,2;2,1) + !c(1,2;3,1) + !c(1,2;4,2) + !c(2,1;3,1) + !c(2,1;4,2) + !c(3,1;4,2) ) ***

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**( !c(1,2;2,2) +!c(1,2;3,1) + !c(1,2;4,1) + !c(2,2;3,1) + !c(2,2;4,1) + !c(3,1;4,1) ) ***

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**( !c(1,2;2,3) + !c(1,2;3,1) + !c(1,2;4,1) + !c(2,3;3,1) + !c(2,3;4,1) + !c(3,1;4,1) ) ***

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**( !c(1,3;2,1) + !c(1,3;3,1) + !c(1,3;4,1) + !c(2,1;3,1) + !c(2,1;4,1) + !c(3,1;4,1) ) ***

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**( !c(1,3;2,2) + !c(1,3;3,1) + !c(1,3;4,1) + !c(2,2;3,1) + !c(2,2;4,1) + !c(3,1;4,1) ) ***

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**( !c(1,3;2,3) + !c(1,3;3,1) + !c(1,3;4,1) + !c(2,3;3,1) + !c(2,3;4,1) + !c(3,1;4,1) ) ***

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

This table is characterized by **73** rows and **6** columns and produces **6·73** products of compatibilities. A minority of these products contains compatibility **!c(1,1;2,2)** while the other terms – the majority - do not contain that compatibility.

The multiplication by **c(1,1;2,2)** deletes the half of the minterms contained in the majority of terms and deletes all the minterms contained in the minority

Since the number of columns increases as $\mathbf{n^2}$ it is apparent that the ratio of the number of unchanged minterms divided by the total number of minterms decreases very quickly with the number **n** of variables.

## APPENDIX 5

In order to prove Theorem 7.5 consider function **ECF(4,I$_j$)** presented in **Appendix4** with **!X = !c(1,1;2,2),**

A minority of these products contains compatibility **!c(1,1;2,2)** while the other terms – the majority - do not contain that compatibility.

The multiplication by **!c(1,1;2,2)** deletes the half of the minterms contained in the majority of terms and leaves all the minterms contained in the minority unchangd

Since the number of columns increases as $\mathbf{n^2}$ it is apparent that the ratio of the number of unchanged minterms divided by the total number of minterms decreases very quickly with the number **n** of variables.

## APPENDIX 6

Consider the sum of two marks $\mathbf{M_1}$ and $\mathbf{M_2}$.

For the sake of simplicity, assume that $\mathbf{M_1}$ and $\mathbf{M_2}$ are characterized by a single different variable **(** $\mathbf{V_1}$ of $\mathbf{M_1}$ and $\mathbf{V_2}$ of $\mathbf{M_2}$ **)**. If all the compatibilities involving $\mathbf{V_1}$ and $\mathbf{V_2}$ appear in $\mathbf{M_1}$ and $\mathbf{M_2}$, respectively, then **val(** $\mathbf{M_1 + M_2}$ **)** can be equal to **val(** $\mathbf{M_1}$**) + val(** $\mathbf{M_2}$**)**.

For example, if $\mathbf{M_1}$ **= c(1,1;2,1) * c(1,1;3,1) * c(1,1;4,1)** and $\mathbf{M_2}$ **= c(1,2;4,1) * c(1,2;3,1) * c(1,2;4,1)** are the two marks of **CF(4)**, by assuming the complation code **x = c(2,1;3,1) * c(2,1;4,1) * c(3,1;4,1)**, we obtain

**val(**$\mathbf{M_1}$**) = val (**$\mathbf{M_2}$**) = NMT1(4)**

**val(**$\mathbf{M_1+M_2}$**) = 2 · NMT1(4)**

This does not happen if $\mathbf{M_1}$ and $\mathbf{M_2}$ are not characterized by a single different variable.

For example, if $\mathbf{M_1}$ **= c(1,1;2,1) * c(1,1;3,1) * c(2,1;4,1)** and $\mathbf{M_2}$ **= c(1,2;2,1) * c(1,2;3,1) * c(2,1;4,1)**, by assuming **x = c(1,1;4,1) * c(1,2;4,1) * c(2,1;3,1) * c(3,1;4,1)**, we obtain

**val(**$\mathbf{M_1}$**) = val (**$\mathbf{M_2}$**) = NMT1(4)**

**val(**$\mathbf{M_1 + M_2}$**) = (1/2) * NMT1(4) + (1/2) * NMT1(4) = NMT1(4).**

Notice that **val(**$\mathbf{M_1 + M_2}$**)** may be the value of a variable introduced in a complex Boolean function or the output value of an **OR** gate having $\mathbf{M_1}$ and $\mathbf{M_2}$ as its input.

## APPENDIX 7

Consider the following example relative to **CF(4)** with **x=c(3,1;4,1)** :

$\mathbf{a_5}$ **= c(1,2;2,2) * c(1,2;3,1) * c(2,2;3,1)**

$\mathbf{b_5}$ **= c(1,2;4,1) * c(2,2;4,1) * c(1,3;4,1)**

$\mathbf{a_6}$ **= c(1,3;2,2) * c(1,3;3,1) * c(2,2;3,1)** **(7.1)**

$\mathbf{m_1 = a_5 * b_5}$

$\mathbf{m_2 = a_6 * b_5}$

Compatibility **c(1,3;4,1)** appears in $\mathbf{b_5}$ because remainder $\mathbf{a_6}$ cannot contain all the compatibilities involving **<1,3>** and this implies the reduction of the value of $\mathbf{m_1}$ by a factor equal to **1/2**.

Now consider the following solution

$\mathbf{a'_5 = a_5}$

$\mathbf{b'_5}$ **= c(1,2;4,1) * c(2,2;4,1) * c(1,2;4,1)**

$\mathbf{a'_6 = a_6}$ **(7.2)**

$\mathbf{b'_6}$ **= c(1,3;4,1) * c(2,2;4,1)**

**val(**$\mathbf{m'_1}$**) = 2 · val(**$\mathbf{m_1}$**)**

**val(**$\mathbf{m'_2}$**) = val(**$\mathbf{m_2}$**)**

It is apparent that solution **(7.2)** is better than solution **(7.1) ,**

## APPENDIX 8

Consider the product **(**$\mathbf{a_1 + a_2}$**) * (**$\mathbf{b_1 + b_2}$**)** relative to **CF(4)** where

$\mathbf{a_1 = c(1,1;2,1) * c(1,1;3,1) * c(1,1;3,2)}$

$\mathbf{a_2 = c(1,1;2,2) * c(1,1;3,2) * c(1,1;3,1)}$

$\mathbf{b_1 = c(2,1;3,1) * c(2,1;4,1) * c (3,1;4,1) * c(2,2;3,1) * c(2,2;4,1)}$

$\mathbf{b_2 = c(2,2;3,2) * c(2,2;4,1) * c(3,2;4,1) * c(2,1;3,2) * c(2,1;4,1)}$

with $\mathbf{x = c[1,1] * c[4,1]}$

The following four marks of **CF(4)** are generated:

$\mathbf{m_1 = a_1 * b_1}$ involving variables **([1,1], [2,1], [3,1], [4,1])**

$\mathbf{m_2 = a_1 * b_2}$ involving variables **([1,1], [2,1], [3,2], [4,1])**

$\mathbf{m_3 = a_2 * b_1}$ involving variables **([1,1], [2,2], [3,1], [4,1])**

$\mathbf{m_4 = a_2 * b_2}$ involving variables **([1,1], [2,2], [3,2], [4,1])**

It is easy to verifly that

$\mathbf{val(m_1) = val(m_2) = val(m_3) = val(m_4) = (1/8) \cdot NMT1(4)}$

Therefore, the total value of the considered product is $\mathbf{(1/2) \cdot NMT1(4)}$

## APPENDIX 9

Consider the following example relative to **CF(4)**.

**U = A * B =**

$\mathbf{(a_1+a_2+\ldots+a_9) * (b_1+b_2+\ldots+b_9)}$ **(9.1)**

where

$\mathbf{a_1 = c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1)}$

$\mathbf{b_1 = c(1,1;4,1) * c(2,1;4,1) * c(1,1;2,1)}$

$\mathbf{a_2 = c(1,2;2,1) * c(1,2;3,1) * c(2,1,3,1) * !c(1,1;2,1)}$

$\mathbf{b_2 = c(1,2;4,1) * c(2,1;4,1) * c(1,2;2,1) * !c(1,1;2,1)}$

$\mathbf{a_3 = c(1,3;2,1) * c(1,3;3,1) * c(2,1;3,1) * !c(1,1;2,1) * !c(1,2;2,1)}$

$\mathbf{b_3 = c(1,3;4,1) * c(2,1;4,1) * !c(1,1;2,1) * !c(1,2;2,1)}$

$\mathbf{a_4 = c(1,1;2,2) * c(1,1;3,1) * c(2,2;3,1) * !c(2,1;4,1)}$

$\mathbf{b_4 = c(1,1;4,1) * c(2,2;4,1) * c(1,1;2,2) * !c(2,1;3,1)}$

**$a_5$ = c(1,2;2,2) * c(1,2;3,1) * c(2,2;3,1) * !c(1,1;2,2) * !c(2,1;4,1)**

**$b_5$ = c(1,2;4,1) * c(2,2;4,1) * c(1,2;2,2) * !c(1,1;,2) * !c(2,1;3,1)**

**$a_6$ = c(1,3;2,2) * c(1,3;3,1) * c(2,2;3,1) * !c(1,1;2,2) * !c(1,2;2,2) * !c(2,1;4,1)**

**$b_6$ = c(1,3;4,1) * c(2,2;4,1) * !c(1,1;2,2) * !c(1,2;2,2) * !c(2,1;3,1)**

**$a_7$ = c(1,1;2,3) * c(1,1;3,1) * c(2,3;3,1) * !c(2,1;4,1) *!c(2,2;4,1)**

**$b_7$ = c(1,1;4,1) * c(2,3;4,1) * c( 1,1;2,3) * !c(2,1;3,1) *!c(2,2;3,1)**

**$a_8$ = c(1,2;2,3) * c( 1,2;3,1) *c(2,3;3,1) * !c(1,1;2,3) * !c(2,1;4,1) *!c(2,2;4,1)**

**$b_8$ = c(1,2;4,1) * c(2,3;4,1) * c(1,2;2,3) * !c(1,1;2,3) * !c(2,1;3,1) *!c(2,2;3,1)**

**$a_9$ = c(1,3;2,3) * c(1,3;3,1) * c(2,3;3,1) * !c(1,1;2,3)* !c(1,2;2,3) * !c(2,1;4,1) *!c(2,2;4,1)**

**$b_9$ = c(1,3;4,1) * c(2,3;4,1) * !c(1,1;2,3) * !c(1,2;2,3) * !c(2,1;3,1) *!c(2,2;3,1)**

The product specified by **A*B,** multiplied by the optimal completion code **x = c(3,1;4,1),** produces nine marks and nine prime implicants, whose total value is

$$\mathbf{val1(4) = (1+1/2 + 1/4) \cdot (1+1/4+1/16) \cdot NMT1(4)} \qquad \mathbf{(9.2)}$$

By extending this example to **CF(n),** we can prove that the value of an **AND** gate performing the product **A*B = ($a_1$+$a_2$+...) * ($b_1$+$b_2$+...)** (with constant **<3,1>** and **<4,1>**) is

$$\mathbf{val1(n) = (1+1/2+1/4)^{n-3} \cdot (1+1/4+1/16) \cdot NMT1(n)} \qquad \mathbf{(9.3)}$$

which is slightly less than

$$\mathbf{val2(n) = (1+1/2+1/4)^{n-2} \cdot NMT1(n)} \qquad \mathbf{(9.4)}$$

In order to prove that the solution proposed in this paper is characterized by the maximum value of the gate performing the product **A*B**, analyze in detail the preceding example **(9.1).**

First consider the product **($a_1$+$a_2$+$a_3$) * ($b_1$+$b_2$+$b_3$)**.

The product of compatibilities **$a_2$** can be obtained from **$a_1$,** and **$b_2$** can be obtained from **$b_1$**, by replacing variable **<1,1>** with variable **<1,2>.** In order that **$a_1$ * $b_2$ = 0** and **$a_2$ * $b_1$ = 0** , both **$a_2$** and **$b_2$** must contain **!c(1,1;2.1)** .

Similarly, **$a_3$** can be obtained from **$a_1$** and **$b_3$** can be obtained from **$b_1$** by replacing variable **<1,1>** with variable **<1, 3>**.In order that **$a_1$ * $b_3$ = 0, $a_3$* $b_1$ = 0, $a_2$ * $b_3$ = 0** and **$a_3$ * $b_2$ = 0,** both **$a_3$** and **$b_3$** must contain **!c(1,1;2,1) * !c(1,2;2,1).**

Then consider the product **($a_4$+$a_5$+$a_6$) * ($b_4$+$b_5$+$b_6$).** In this case the product of compatibilities **$a_5$** can be obtained from **$a_4$** and **$b_5$** can be obtained from **$b_4$** by replacing variable **<1,1>** with variable **<1,2>.** In order that **$a_4$ * $b_5$ = 0** and **$a_5$ * $b_4$ = 0,** both **$a_5$** and **$b_5$** must contain **!c(1,1;2,2).** Similarly, in order that **$a_4$ * $b_6$ = 0, $a_6$ * $b_4$ = 0, $a_5$ * $b_6$ = 0** and **$a_6$ * $b_5$ = 0,** both **$a_6$** and **$b_6$** must contain **!c(1,1;2,2) * !c(1,2;2,2).**

For similar reasons, both $\mathbf{a_8}$ and $\mathbf{b_8}$ must contain **!c(1,1;2,3)** while $\mathbf{a_9}$ and $\mathbf{b_9}$ contain **!c(1,1;2,3) * !c(1,2;2,3).**

As a second step of analysis consider the product $\mathbf{(a_1+a_4+a_7) * (b_1+b_4+b_7)}$.

The product of compatibilities $\mathbf{a_4}$ can be obtained from $\mathbf{a_1}$, and $\mathbf{b_4}$ can be obtained from $\mathbf{b_1}$, by replacing variable **<2,1>** with variable **<2,2>**. In order that $\mathbf{a_1 * b_4 = 0}$, $\mathbf{b_4}$ must contain **!c(2,1;3,1)**; in order that $\mathbf{a_4 * b_1 = 0}$, $\mathbf{a_4}$ must contain **!c(2,1;4,1).** Therefore,

$\mathbf{val(a_4 * b_4) = (1/4) \cdot NMT1(4)}$

In order that $\mathbf{a_1 * b_7 = 0}$ and $\mathbf{a_7 * b_1 = 0}$, $\mathbf{b_7}$ must contain **!c(2,1;3,1) * !c(2,2;3,1)** and $\mathbf{a_7}$ must contain **!c(2,1;4,1) *!c(2,2;4,1).**

Therefore,

$\mathbf{val(a_7*b_7) = (1/16) \cdot NMT1(4)}$

In the same way all the complemented compatibilities appearing in **Eq(9.1)** can be easily justified.

Notice that only the complemented variables absolutely necessary in order that $\mathbf{a_i * b_j = 0}$ for all **i<>j** appear in the list of values of $\mathbf{a_i}$ and $\mathbf{b_j}$. This is equivalent to proving that **Eq(9.2)**is the total value of the "best" product **A * B.** This product produces all the marks and all the prime implicants of core function **CF(4)** satisfying the condition that **<3,1>** and **<4,1>** are the only variables of triplets **3** and 4.

Now consider the following equations relative to **CF(5).**

**U = A * B =**

$\mathbf{(a_1+a_2+...+a_9+a_{10} + ... +a_{19}+ ... +a_{27}) * (b_1+b_2+...+b_9+b_{10}+ ... +b_{19}+ ... +b_{27})}$ **(9.5)**

where:

$\mathbf{a_1}$ **= c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1) * c(1,1;5,1) *c(2,1;5,1) * c(3,1;5,1)**

$\mathbf{b_1}$ **= c(1,1;4,1) *c(2,1;4,1) * c(1,1;2,1) * c(2,1;5,1) * c(4,1;5,1)**

$\mathbf{a_2}$ **= c(1,2;2,1) * c(1,2;3,1) * c(2,1,3,1) * !c(1,1;2,1) ) * c(1,2;5,1) *c(2,1;5,1) * c(3,1;5,1)**

$\mathbf{b_2}$ **= c(1,2;4,1) * c(2,1;4,1) * c(1,2;2,1) * !c(1,1;2,1) ) * c(2,1;5,1) * c(4,1;5,1)**

$\mathbf{a_3}$ **= c(1,3;2,1) * c(1,3;3,1) * c(2,1;3,1) * !c(1,1;2,1) * !c(1,2;2,1) * c(1,3;5,1) * c(2,1;5,1) * c(3,1;5,1)**

$\mathbf{b_3}$ **= c(1,3;4,1) *c(2,1;4,1) * !c(1,1;2,1) * !c(1,2;2,1) * c(2,1;5,1) * c(4,1;5,1)**

$\mathbf{a_4}$ **= c(1,1;2,2) * c(1,1;3,1) * c(2,2;3,1) * c(1,1;5,1) * c(2,2;5,1) * c(3,1;5,1) * !c(2,1;5,1)**

$\mathbf{b_4}$ **= c(1,1;4,1) * c(2,2;4,1) * c(1,1;2,2) * c(2,2;5,1) * c(4,1;5,1) * !c(2,1;5,1)**

$\mathbf{a_5}$ **= c(1,2;2,2) * c(1,2;3,1) * c(2,2;3,1) * !c(1,1;2,2) * c(1,2;5,1) *c(2,2;5,1) * c(3,1;5,1) !c(2,1;5,1)**

**$b_5$ = c(1,2;4,1) * c(2,2;4,1) * c(1,2;2,2) *!c(1,1;2,2) * c(4,1;5,1) * c(2,2;5,1) * !c(2,1;5,1)**

**$a_6$ = c(1,3;2,2) * c(1,3;3,1) * c(2,2;3,1) * !c(1,1;2,2) * !c(1,2;2,2) * c(1,3;5,1) c(2,2;5,1) * c(3,1;5,1) * !c(2,1;5,1)**

**$b_6$ = c(1,3;4,1) * c(2,2;4,1) * !c(1,1;2,2) * !c(1,2;2,2) * c(2,2;5,1) * c(4,1;5,1) * !c(2,1;5,1)**

**$a_7$ = c(1,1;2,3) * c(1,1;3,1) * c(2,3;3,1) * c(1,1;5,1) * c(2,3;5,1) * c(3,1;5,1) !c(2,1;5,1) *!c(2,2;5,1)**

**$b_7$ = c(1,1;4,1) * c(2,3;4,1) * c(1,1;2,3) * c(4,1:5,1) * !c(2,1;5,1) *!c(2,2;5,1)**

**$a_8$ = c(1,2;2,3) * c( 1,2;3,1) *c(2,3;3,1) * !c(1,1;2,3) * c(1,2;5,1) * c(2,3;5,1) * c(3,1;5,1) * !c(2,1;5,1) * !c(2,2;5,1)**

**$b_8$ = c(1,2;4,1) * c(2,3;4,1) * c(1,2;2,3) * !c(1,1;2,3) *c(4,1;5,1) * !c(2,1;5,1) *!c(2,2;5,1)**

**$a_9$ = c(1,3;2,3) * c(1,3;3,1) * c(2,3;3,1) * !c(1,1;2,3)* !c(1,2;2,3) * c(1,3;5,1) * c(2,3;5,1) * c(3,1;5,1) * !c(2,1;5,1) * !c(2,2;5,1)**

**$b_9$ = c(1,3;4,1) * c(2,3;4,1) * !c(1,1;2,3) * !c(1,2;2,3) * c(4,1;5,1) * !c(2,1;5,1) * !c(2,2;5,1)**

**$a_{10}$ = c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1) * c(1,1;5,2) *c(2,1;5,2) * c(3,1;5,2) * !c(4,1;5,1)**

**$b_{10}$ = c(1,1;4,1) *c(2,1;4,1) * c(1,1;2,1) * c(2,1;5,2) * c(4,1;5,2) * !c(3,1;5,1) * !c(3,1;5,1)**

**$a_{11}$ = c(1,2;2,1) * c(1,2;3,1) * c(2,1,3,1) * !c(1,1;2,1) ) * c(1,2;5,2) *c(2,1;5,2) * c(3,1;5,2) ) * !c(4,1;5,1)**

**$b_{11}$ = c(1,2;4,1) * c(2,1;4,1) * c(1,2;2,1) * !c(1,1;2,1) ) * c(2,1;5,2) * c(4,1;5,2) * !c(3,1;5,1)**

**$a_{12}$ = c(1,3;2,1) * c(1,3;3,1) * c(2,1;3,1) * !c(1,1;2,1) * !c(1,2;2,1) * c(1,3;5,2) * c(2,1;5,2) * c(3,1;5,2) ) * !c(4,1;5,1)**

**$b_{12}$ = c(1,3;4,1) *c(2,1;4,1) * !c(1,1;2,1) * !c(1,2;2,1) * c(2,1;5,2) * c(4,1;5,2) * !c(3,1;5,1)**

**$a_{13}$ = c(1,1;2,2) * c(1,1;3,1) * c(2,2;3,1) * c(1,1;5,2) * c(2,2;5,2) * c(3,1;5,2) * !c(2,1;5,2) ) * !c(4,1;5,1)**

**$b_{13}$ = c(1,1;4,1) * c(2,2;4,1) * c(1,1;2,2) * c(2,2;5,2) * c(4,1;5,2) * !c(2,1;5,2) * !c(3,1;5,1)**

**$a_{14}$ = c(1,2;2,2) * c(1,2;3,1) * c(2,2;3,1) * !c(1,1;2,2) * c(1,2;5,2) *c(2,2;5,2) * c(3,1;5,2) !c(2,1;5,2) ) * !c(4,1;5,1)**

$b_{14}$ = c(1,2;4,1) * c(2,2;4,1) * c(1,2;2,2) *!c(1,1;2,2) * c(4,1;5,2) * c(2,2;5,2)  * !c(2,1;5,2) * !c(3,1;5,1)

$a_{15}$ = c(1,3;2,2) * c(1,3;3,1) * c(2,2;3,1) * !c(1,1;2,2) * !c(1,2;2,2) * c(1,3;5,2) c(2,2;5,2) * c(3,1;5,2)  *  !c(2,1;5,2) ) * !c(4,1;5,1)

$b_{15}$ = c(1,3;4,1) * c(2,2;4,1) * !c(1,1;2,2) * !c(1,2;2,2) * c(2,2;5,2)  * c(4,1;5,2)   * !c(2,1;5,2) * !c(3,1;5,1)

$a_{16}$  = c(1,1;2,3) * c(1,1;3,1)   * c(2,3;3,1)   * c(1,1;5,2) * c(2,3;5,2) * c(3,1;5,2) !c(2,1;5,2) *!c(2,2;5,2) ) * !c(4,1;5,1)

$b_{16}$ = c(1,1;4,1) * c(2,3;4,1) *  c(1,1;2,3)  * c(4,1:5,2) * !c(2,1;5,2) *!c(2,2;5,2) * !c(3,1;5,1)

$a_{17}$ = c(1,2;2,3) * c( 1,2;3,1) *c(2,3;3,1) * !c(1,1;2,3) * c(1,2;5,2) * c(2,3;5,2) * c(3,1;5,2) * !c(2,1;5,2) * !c(2,2;5,2) ) * !c(4,1;5,1)

$b_{17}$ = c(1,2;4,1) * c(2,3;4,1) * c(1,2;2,3) * !c(1,1;2,3) *c(4,1;5,2) * !c(2,1;5,2) *!c(2,2;5,2) * !c(3,1;5,1)

$a_{18}$ = c(1,3;2,3) * c(1,3;3,1) * c(2,3;3,1) * !c(1,1;2,3)* !c(1,2;2,3) * c(1,3;5,2) * c(2,3;5,2)  * c(3,1;5,2) * !c(2,1;5,2) * !c(2,2;5,2) ) * !c(4,1;5,1)

$b_{18}$ = c(1,3;4,1) * c(2,3;4,1) * !c(1,1;2,3) * !c(1,2;2,3) * c(4,1;5,2) * !c(2,1;5,2) * !c(2,2;5,2) * !c(3,1;5,1)

$a_{19}$ =  c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1) * c(1,1;5,3) *c(2,1;5,3) * c(3,1;5,3) * !c(4,1;5,1) * !c(4,1;5,2)

$b_{19}$ =  c(1,1;4,1) *c(2,1;4,1) * c(1,1;2,1) *   c(2,1;5,3) * c(4,1;5,3) * !c(3,1;5,1) * !c(3,1;5,2)

$a_{20}$ = c(1,2;2,1) * c(1,2;3,1) * c(2,1,3,1) * !c(1,1;2,1) ) * c(1,2;5,3) *c(2,1;5,3) * c(3,1;5,3) * ! c(4,1;5,1) * !c(4,1;5,2)

$b_{20}$ = c(1,2;4,1) * c(2,1;4,1) * c(1,2;2,1) * !c(1,1;2,1) )  * c(2,1;5,3) * c(4,1;5,3) * !c(3,1;5,1) * !c(3,1;5,2)

$a_{21}$ = c(1,3;2,1) * c(1,3;3,1) * c(2,1;3,1) * !c(1,1;2,1) * !c(1,2;2,1) * c(1,3;5,3) * c(2,1;5,3) * c(3,1;5,3) * !c(4,1;5,1) * !c(4,1;5,2)

$b_{21}$ = c(1,3;4,1) *c(2,1;4,1)  * !c(1,1;2,1) * !c(1,2;2,1) * c(2,1;5,3) * c(4,1;5,3) * !c(3,1;5,1) * !c(3,1;5,2)

$a_{22}$ = c(1,1;2,2) * c(1,1;3,1) * c(2,2;3,1)  * c(1,1;5,3)  * c(2,2;5,3) * c(3,1;5,3)   * !c(2,1;5,3) * ! c(4,1;5,1) * ! c(4,1;5,2)

$b_{22}$ = c(1,1;4,1) * c(2,2;4,1) * c(1,1;2,2) * c(2,2;5,3)   * c(4,1;5,3)  * !c(2,1;5,3) * ! c(3,1;5,1) * !c(3,1;5,2)

$a_{23}$ = c(1,2;2,2) * c(1,2;3,1) * c(2,2;3,1) * !c(1,1;2,2) * c(1,2;5,3) *c(2,2;5,3) * c(3,1;5,3) !c(2,1;5,3) * ! c(4,1;5,1) * ! c(4,1;5,2)

$b_{23}$ = c(1,2;4,1) * c(2,2;4,1) * c(1,2;2,2) *!c(1,1;2,2) * c(4,1;5,3) * c(2,2;53) * !c(2,1;5,3) * ! c(3,1;5,1) * ! c(3,1;5,2)

$a_{24}$ = c(1,3;2,2) * c(1,3;3,1) * c(2,2;3,1) * !c(1,1;2,2) * !c(1,2;2,2) * c(1,3;5,3) c(2,2;5,3) * c(3,1;5,3) * !c(2,1;5,3) * ! c(4,1;5,1) * !c(4,1,5,2)

$b_{24}$ = c(1,3;4,1) * c(2,2;4,1) * !c(1,1;2,2) * !c(1,2;2,2) * c(2,2;5,3) * c(4,1;5,3) * !c(2,1;5,3) * ! c(3,1;5,1) * !c(3,1;5,2)

$a_{25}$ = c(1,1;2,3) * c(1,1;3,1) * c(2,3;3,1) * c(1,1;5,3) * c(2,3;5,3) * c(3,1;5,3) !c(2,1;5,3) *!c(2,2;5,3) * ! c(4,1;5,1) * !c(4,1;5,2)

$b_{25}$ = c(1,1;4,1) * c(2,3;4,1) * c(1,1;2,3) * c(4,1:5,3) * !c(2,1;5,3) *!c(2,2;5,3) * !c(3,1;5,1) * !c(3,1;5,2)

$a_{26}$ = c(1,2;2,3) * c( 1,2;3,1) *c(2,3;3,1) * !c(1,1;2,3) * c(1,2;5,3) * c(2,3;5,3) * c(3,1;5,3) * !c(2,1;5,3) * !c(2,2;5,3) * ! c(4,1;5,1) * !c(4,1;5,2)

$b_{26}$ = c(1,2;4,1) * c(2,3;4,1) * c(1,2;2,3) * !c(1,1;2,3) *c(4,1;5,3) * !c(2,1;5,3) *!c(2,2;5,3) * !c(3,1;5,1) * !c(3,1;5,2)

$a_{27}$ = c(1,3;2,3) * c(1,3;3,1) * c(2,3;3,1) * !c(1,1;2,3)* !c(1,2;2,3) * c(1,3;5,3) * c(2,3;5,3) * c(3,1;5,3) * !c(2,1;5,3) * !c(2,2;5,3) * !c(4,1;5,1) * !c(4,1;5,2)

$b_{27}$ = c(1,3;4,1) * c(2,3;4,1) * !c(1,1;2,3) * !c(1,2;2,3) * c(4,1;5,3) * !c(2,1;5,3) * !c(2,2;5,3) * !c(3,1;5,1) * !c(3,1;5,2)

The total value of the product **(9.5)** is

$$\text{val1}(5) = (1+(1/2) + (1/4))^2 \cdot (1+(1/4)+(1/16)) \cdot \text{NMT1}(5) \qquad (9.6)$$

The analysis of the differences between the two preceding implementations of **CF(4)** and **CF(5)** makes it easy to prove **Eq(9.3)** for **CF(n).**

## APPENDIX 10

The products **($a_1$+$a_2$+...) * ($b_1$+$b_2$+...)** which have been presented in **Appendix 9** do not produce all the prime implicants of Core Function. Indeed, the prime implicants containing variables different from those appearing in the completion code **x** (in our example: **<3,2>, <3,3>, <4,2>, <4,3>)** do not appear in the list of prime implicants which have been generated.

As a first example, we can extend the list **($a_1$ +$a_2$ + ...+$a_9$)** with **($a_{10}$ + $a_{11}$ + ... + $a_{18}$)** and the list **($b_1$ + $b_2$ +... +$b_9$)** with **($b_{10}$ + $b_{11}$ + ... +$b_{18}$)** in order to obtain all the marks and all the prime implicants containing both variables **<3,1>** and **<4,2>,** in addition to the marks and the prime implicants containing variables **<3,1>** and **<4,1>** obtained by the product **($a_1$+$a_2$+...+ $a_9$) * ($b_1$+$b_2$+...+$b_9$)** :

$a_{10}$ = c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1)

$b_{10} = c(1,1;4,2) * c(2,1;4,2) * c(1,1;2,1)$

$a_{11} = c(1,2;2,1) * c(1,2;3,1) * c(2,1,3,1) * !c(1,1;2,1)$

$b_{11} = c(1,2;4,2) * c(2,1;4,2) * c(1,2;2,1) * !c(1,1;2,1)$

$a_{12} = c(1,3;2,1) * c(1,3;3,1) * c(2,1;3,1) * !c(1,1;2,1) * !c(1,2;2,1)$

$b_{12} = c(1,3;4,2) * c(2,1;4,2) * !c(1,1;2,1) * !c(1,2;2,1)$

$a_{13} = c(1,1;2,2) * c(1,1;3,1) * c(2,2;3,1) * !c(2,1;4,2)$

$b_{13} = c(1,1;4,2) * c(2,2;4,2) * c(1,1;2,2) * !c(2,1;3,1)$

$a_{14} = c(1,2;2,2) * c(1,2;3,1) * c(2,2;3,1) * !c(1,1;2,2) * !c(2,1;4,2)$

$b_{14} = c(1,2;4,2) * c(2,2;4,2) * c(1,2;2,2) * !c(1,1;2,2) * !c(2,1;3,1)$

$a_{15} = c(1,3;2,2) * c(1,3;3,1) * c(2,2;3,1) * !c(1,1;2,2) * !c(1,2;2,2) * !c(2,1;4,2)$

$b_{15} = c(1,3;4,2) * c(2,2;4,2) * !c(1,1;2,2) * !c(1,2;2,2) * !c(2,1;3,1)$

$a_{16} = c(1,1;2,3) * c(1,1;3,1) * c(2,3;3,1) * !c(2,1;4,2) * !c(2,2;4,2)$

$b_{16} = c(1,1;4,2) * c(2,3;4,2) * c( 1,1;2,3) * !c(2,1;3,1) * !c(2,2;3,1)$

$a_{17} = c(1,2;2,3) * c( 1,2;3,1) * c(2,3;3,1) * !c(1,1;2,3) * !c(2,1;4,2) * !c(2,2;4,2)$

$b_{17} = c(1,2;4,2) * c(2,3;4,2) * c(1,2;2,3) * !c(1,1;2,3) * !c(2,1;3,1) * !c(2,2;3,1)$

$a_{18} = c(1,3;2,3) * c(1,3;3,1) * c(2,3;3,1) * !c(1,1;2,3) * !c(1,2;2,3) * !c(2,1;4,2) * !c(2,2;4,2)$

$b_{18} = c(1,3;4,2) * c(2,3;4,2) * !c(1,1;2,3) * !c(1,2;2,3) * !c(2,1;3,1) * !c(2,2;3,1)$

The product $(a_{10} + a_{11} + ... + a_{18}) * (b_{10} + b_{11} + ... + b_{18})$ produces all the nine implicants of Core Function containing only **<3,1>** and **<4,2>** and no other variable of triplets **3** amd **4.** Similarly, a new product $(a_{19} + ... + a_{27}) * (b_{19} + ... + b_{27})$ can produce all the prime implicants of Core Function containing only **<3,3>** and **<4,3>** of the variables of triplets **3** and **4**, while the product $(a_{28} + ... a_{36}) * (b_{28} + ... + b_{36})$ can produce all the prime implicants of Core Function characterized by variables **<3,2>** and **<4,1>.**

In this way all the prime implicants of Core Function will be produced by the product

$(a_1 + ... + a_{10} + ... + a_{19} + ... + a_{28} + ... + a_{37} + ... + a_{46} + ... + a_{55} + ... + a_{67} + ... + a_{73} + ...)$

$(b_1 + ... + b_{10} + ... + b_{19} + ... + b_{28} + ... + b_{37} + ... + b_{46} + ... + b_{55} + ... + b_{67} + ... + b_{73} + ...)$

where the nine pairs of variables **<3,1>, <4,1>; <3,1>, <4,2>; <3,1>,< 4,3>; <3,2>, <4,1>; <3,2>, <4,2>; <3,3>,<4,1>; <3,3>, <4,2>; <3,3>, <4,3>** are involved.

Also the complation code should be updated as follows:

**$x = c(3,1;4,1) * c(3,1;4,2) * c(3,1;4,3) * c(3,2;4,1) * c(3,2;4,2) * c(3,2;4,3) * c(3,3;4,1) * c(3,3;4,2) * c(3,3;4,3)$**

Thus, the **81** prime implicants of **CF(4)** will be generated , but the value of many of them will be very small because all the products **$a_1 * b_j$** must be equal to **0**. Besides, the multiplication of every mark by the completion code **x** will dramatically reduce the value of the corresponding prime implicants .

A better solution can be obtained by integrating the completion codes in the tables of remainders and by multiplying the new terms produced by complemented compatibilities in order that all the products **$a_1 * b_j$** with **i<>j** become equal to **0**

As a first step, assume that:

For all **i<=9**

**$a_i' = a_i * c(3,1;4,1)$**

**$b_i' = b_i * c(3,1;4,1)$;**

for all **i>9** and **<= 18**

**$a_i' = a_i * c(3,1;4,2) * !c(3,1;4,1)$**

**$b_i' = b_i * c(3,1;4,2) * !c(3,1;4,1)$.**

Similarly, if we extend the preceding example as follows:

for all **i>18** and **<= 27**

**$a_i' = a_i * c(3,1;4,3) * !c(3,1;4,1) * !c(3,1;4,2)$**

**$b_i' = b_i * c(3,1;4,3) * !c(3,1;4,1) * !c(3,1;4,2)$**

it follows that:

**$val\ ((a_1' + a_2' + \ldots + a_9' + a_{10}' + \ldots + a_{19}' + \ldots) * (b_1' + b_2' + \ldots + b_9' + b_{10}' + \ldots + b_{19}' + \ldots)) =$**

**$(1+1/2+1/4) \cdot val1\ (4) = (1 + 1/2 + 1/4) \cdot (1+1/2 +1/4) \cdot (1 + 1/4 + 1/16) \cdot NMT1(4)$**

The final result of this line of corrections will be the following equation:

**$val\ ((a_1 + \ldots + a_{10} + \ldots + a_{19} + \ldots + a_{28} + \ldots + a_{37} + \ldots + a_{46} + \ldots + a_{55} + \ldots + a_{64} + \ldots + a_{73} + \ldots) *$**

**$(b_1 + \ldots + b_{10} + \ldots + b_{19} + \ldots + b_{28} + \ldots + b_{37} + \ldots + b_{46} + \ldots + b_{55} + \ldots + b_{64} + \ldots + b_{73} + \ldots)) =$ (10.1)**

**$(1+1/2 + 1/4 + 1/8 +1/16 +1/32 +1/64 + 1/128 + 1/256) \cdot (1 + 1/2 + 1/4) \cdot (1 + 1/4 +1/16) \cdot NMT1(4) \sim$**

**$2 \cdot (1 + 1/2+ 1/4) \cdot (1+ 1/4 + 1/16) \cdot NMT1(4)$**

A slightly different result is shown in **Appendix 11.**

## APPENDIX 11

Let

**$A * B = (a_1 + a_2 + \ldots + a_{27}) * (b_1 + b_2 + \ldots + b_{27})$ (11.1)**

where:

**$a_1 = c(1,1;2,1) * c(1,1;3,1) * c(1,1;4,1) * c(1,1;5,1) * c(2,1;3,1) * c(3,1;5,1)$**

**$b_1 = c(2,1;3,1) * c(2,1;4,1) * c(3,1;4,1) * c(2,1;5,1) * c(3,1;5,1) * c(4,1;5,1)$**

$a_2$ = c(1,2;2,1) *c(1,2;3,1) *c(1,2;4,1) * c(1,2;5,1) * c(2,1;3,1) * c(3,1;5,1)

$b_2$ = $b_1$

$a_3$ = c(1,3;2,1) * c(1,3;3,1) * c(1,3;4,1) * c(1,3; 5,1) * c(2,1;3,1) * c(3,1;5,1)

$b_3$ = $b_1$

$a_4$ = c(1,1;2,2) * c(1,1;3,1) * c(1,1;41) * c(1,1;5,1) * c(2,2;3,1) !c(2,1;3,1) * c(3,1;5,1)

$b_4$ = c(2,2;3,1) *c(2,2;4,1) * c(3,1;4,1) * c(2,2;5,1) * c(3,1;5,1) * c(4,1;5,1) * !c(1,1;2,1) * !c(2,1;3,1)

$a_5$ = c(1,2;2,2) * c(1,2;3,1) * c(1,2;4,1) * c(1,2;5,1) * !c(2,1;3,1) * c(2,2;3,1) * c(3,1;5,1)

$b_5$ = $b_4$

$a_6$ = c(1,3;2,2) * c(1,3;3,1) * c(1,3;4,1) * c(1,3,5,1) * !c(2,1;3,1) * c(2,2;3,1) * c(3,1;5,1)

$b_6$ = b4

$a_7$ = c(1,1;2,3) * c(1,1;3,1) * c(1,1,4,1) * c(1,1;5,1) * c(3,1;5,1) * !c(2,1;3,1) * !c(2,2;3,1)

$b_7$ = c(2,3;3,1) * c(2,3;4,1) * c(3,1;4,1) * c(2,3;5,1) * c(3,1;5,1) * c(4,1,5,1) * !c(2,1;3,1) * !c(2,2;3,1)

$a_8$ = c(1,2;2,3) * c(1,2;3,1) * c(1,2;4,1) * c(1,2;5,1) * c(3,1;5,1) * !c(2,1;3,1) * !c(2,2;3,1)

$b_8$ = $b_7$

$a_9$ = c(1,3;2,3) * c(1,3;3,1) * c(1,3;4,1) * c(1,3;5,1) * c(3,1;5,1) * ! c(2,1;3,1) * !c(2,2;3,1)

$b_9$ = $b_7$

$a_{10}$ = c(1,1;2,1) * c(1,1;3,1) * c(1,1;4,1) * c(1,1;5,2) * c(2,1;3,1) * c(3,1;5,2) * !c(3,1;5,1)

$b_{10}$ = c(2,1;3,1) * c(2,1;4,1) * c(3,1;4,1) * c(2,1;5,2) * c(3,1;5,2) * c(4,1;5,2) * !c(3,1;5,1)

$a_{11}$ = c(1,2;2,1) *c(1,2;3,1) *c(1,2;4,1) * c(1,2;5,2) * c(2,1;3,1) * c(3,1;5,2) * !c(3,1;5,1)

$b_{11}$ = $b_{10}$

**$a_{12}$ = c(1,3;2,1) * c(1,3;3,1) * c(1,3;4,1) * c(1,3; 5,2) * c(2,1;3,1) * c(3,1;5,2) * !c(3,1;5,1)**

**$b_{12}$ = $b_{10}$**

**$a_{13}$ = c(1,1;2,2) * c(1,1;3,1) * c(1,1;4,1) * c(1,1;5,2) * c(2,2;3,1) * c(3,1;5,2) * !c(2,1;3,1) * !c(3,1;5,1) * c(3,1:5,2)**

**$b_{13}$ = c(2,2;3,1) *c(2,2;4,1) * c(3,1;4,1) * c(2,2;5,2) * c(3,1;5,2) * c(4,1;5,2) * !c(2,1;3,1) ***
**!c(3,1;5,1) * c(2,2;3,1) * c(3,1;5,2)**

**$a_{14}$ = c(1,2;2,2) * c(1,2;3,1) * c(1,2;4,1) * c(1,2;5,2) * !c(2,1;3,1) * !c(3,1;5,1) * c(3,1;5,2)**

**$b_{14}$ = $b_{13}$**

**$a_{15}$ = c(1,3;2,2) * c(1,3;3,1) * c(1,3;4,1) * c(1,3,5,2) * c(2,2;3,1) * !c(2,1;3,1) * !c(3,1;5,1) * c(3,1;5,2)**

**$b_{15}$= $b_{13}$**

**$a_{16}$ = c(1,1;2,3) * c(1,1;3,1) * c(1,1,4,1) * c(1,1;5,2) * !c(2,1;3,1) * c(3,1;5,2) !c(2,2;3,1) * !c(3,1;5,1)**

**$b_{16}$ = c(2,3;3,1) * c(2,3;4,1) * c(3,1;4,1) * c(2,3;5,2) * c(3,1;5,2) * c(4,1,5,2) * !c(2,1;3,1) * !c(2,2;3,1) * !c(3,1;5,1)**

**$a_{17}$ = c(1,2;2,3) * c(1,2;3,1) * c(1,2;4,1) * c(1,2;5,2) * c(3,1;5,2) * !c(2,1;3,1) * !c(2,2;3,1) * !c(3,1;5,1)**

**$b_{17}$ = $b_{16}$**

**$a_{18}$ = c(1,3;2,3) * c(1,3;3,1) * c(1,3;4,1) * c(1,3;5,2) * c(3,1;5,2) * ! c(2,1;3,1) * !c(2,2;3,1) * !c(3,1;5,1)**

**$b_{18}$ = $b_{16}$**

**$a_{19}$ = c(1,1;2,1) * c(1,1;3,1) * c(1,1;4,1) * c(1,1;5,3) * c(2,1;3,1) * c(3,1;5,3) * !c(3,1;5,1) * !c(3,1;5,2)**

**$b_{19}$ = c(2,1;3,1) * c(2,1;4,1) * c(3,1;4,1) * c(2,1;5,3) * c(3,1;5,3) * c(4,1;5,3) * !c(3,1;5,1) * !c(3,1;5,2)**

**$a_{20}$ = c(1,2;2,1) * c(1,2;3,1) * c(1,2;4,1) * c(1,2;5,3) * c(2,1;3,1) * c(3,1;5,3) * !c(3,1;5,1) * !c(3,1;5,2)**

**$b_{20} = b_{19}$**

**$a_{21}$ = c(1,3;2,1) * c(1,3;3,1) * c(1,3;4,1) * c(1,3; 5,3) * c(2,1;3,1) *c(3,1;5,3) * !c(3,1;5,1) * !c(3,1;5,2)**

**$b_{21} = b_{19}$**

**$a_{22}$ = c(1,1;2,2) * c(1,1;3,1) * c(1,1;4,1) * c(1,1;5,3) * c(2,2;3,1) * c(3,1;5,3) * !c(2,1;3,1) * !c(3,1;5,1) * !c(3,1;5,2)**

**$b_{22}$ = c(2,2;3,1) *c(2,2;4,1) * c(3,1;4,1) * c(2,2;5,3) * c(3,1;5,3) * c(4,1;5,3) * !c (2,1;3,1) * !c(3,1;5,1) * !c(3,1;5,2)**

**$a_{23}$ = c(1,2;2,2) * c(1,2;3,1) * c(1,2;4,1) * c(1,2;5,3) * c(2,2;3,1) * c(3,1;5,3) * !c(2,1;3,1) * !c(3,1;5,1) * !c(3,1;5,2)**

**$b_{23} = b_{22}$**

**$a_{24}$ = c(1,3;2,2) * c(1,3;3,1) * c(1,3;4,1) * c(1,3,5,3) * c(2,2;3,1) * c(3,1;5,3) * !c(2,1;3,1) * !c(3,1;5,1) * !c(3,1;5,2)**

**$b_{24} = b_{22}$**

**$a_{25}$ = c(1,1;2,3) * c(1,1;3,1) * c(1,1,4,1) * c(1,1;5,3) * c(3,1;5,3) * !c(2,1;3,1) * !c(2,2;3,1) * !c(3,1;5,1) * !c(3,1;5,2)**

**$b_{25}$ = c(2,3;3,1) * c(2,3;4,1) * c(3,1;4,1) * c(2,3;5,3) * c(3,1;5,3) * c(4,1,5,3) * !c(2,1;3,1) * !c(2,2;3,1) * !c(3,1;5,1) * !c(3,1;5,2)**

**$a_{26}$ = c(1,2;2,3) * c(1,2;3,1) * c(1,2;4,1) * c(1,2;5,3) * c (3,1;5,3) *!c(2,1;3,1) * !c(2,2;3,1) * !c(3,1;5,1) * !c(3,1;5,2)**

**$b_{26} = b_{25}$**

**$a_{27}$ = c(1,3;2,3) * c(1,3;3,1) * c(1,3;4,1) * c(1,3;5,3) * c(3,1;5,3) * !oc(2,1;3,1) * !c(2,2;3,1) * !c(3,1;5,1) * !c(3,1;5,2)**

**$b_{27} = b_{25}$**

It is easy to verify that

**val (A * B) = $3 \cdot (1 + 1/2 + 1/4)^3 \cdot$ NMT1(5)** **(11.2)**

which can be extended to **n** as follows:

**val (A * B) = $3 \cdot (1 + 1/2 + 1/4)^{n-3} \cdot$ NMT1(n)** **(11.3)**

Notice that

$\mathbf{A * B = (a_1 + a_2 + \ldots + a_{27}) * (b_1 + b_2 + \ldots + b_{27})}$

produces all the prime implicants of **CF(5)** with the exception of those which include variables **<3,2>, <3,3>, <4,2>** or **<4,3>,** exactly as the prime implicants produced by **Eq(9.5)** of **Appendix 9--**

In **Appendix 9** it has been shown that the value of the product **(A' * B')** specified by **Eq(9.6)** is equal to

$\mathbf{val\ (A' * B') = (1 + 1/2 + 1/4)^{n-3} \cdot (1 + 1/4 + 1/16) \cdot NMT1(n)}$

Since

$\mathbf{3 > (1 + 1/2 + 1/4)}$

**val (A * B)** is larger than **val (A' * B')**, but **val (A') = 0** and **val (A)** is relatively large.

However,, the value of the gate performing **(A * B)** is nearly equal to the value of the gate performing **(A' * B').**

In order to produce also the prime implicants containing variables **<3,2>, <3,3>, <4,2>, <4,3>**

first we extend the lists $\mathbf{(a_1, a_2, \ldots, a_{27})}$ and $\mathbf{(b_1, b_2, \ldots, b_{27})}$ to $\mathbf{(a_1', a_2', \ldots, a_{71}')}$ and $\mathbf{<b_1', b_2', \ldots b_{71}'>}$

defined as follows:

for every $\mathbf{i <= 27}$ : $\mathbf{a_i' = a_i * c(3,1;4,1)}$

for every $\mathbf{j > 27}$ and $\mathbf{<= (27 \cdot 2)}$ : $\mathbf{a_j' = a_{j-27} * c(3,2;4,1) * !c(3,1;4,1)}$

for every $\mathbf{k > (27 \cdot 2)}$ and $\mathbf{<= (27 \cdot 3)}$ : $\mathbf{a_k' = a_{k-54} * c(3,3;4,1) * !c(3,1;4,1) * !c(3,1;4,1)}$

The new products $\mathbf{a_i' * b_j'}$ produce also the prime implicants containing **<3,2>** and**<3,3>** and the total value of the products $\mathbf{a_i' * b_j'}$ will be incremented by $\mathbf{(1 + 1/2 + 1/4)}$.

The same procedure can be applied in order to produce also the prime implicants containing variables **<4,2>** and **<4,3>,** but**,** unfortunately, the number of complemented compatibilities needed in order that $\mathbf{a_i * b_j = 0}$ for every $\mathbf{i <> j}$ , is now larger than the ones appearing in the preceding examples. Therefore, the increment of the total value produced by the new prime implicants will be very small.

It follows that the total value of the product $\mathbf{(a_1 + a_2 + \ldots) * (b_1 + b_2 + \ldots)}$ will be less than

$$\mathbf{3 \cdot (1 + 1/2 + 1/4)^{n-2} \cdot NMT1(n)} \qquad \mathbf{(11.4)}$$

## APPENDIX 12

As already observed, the products $\mathbf{(a_1+a_2+\ldots+a_9) * (b_1+b_2+\ldots+b_9)}$ which have been defined by **Eq(9.1)** of **APPENDIX 9** do not produce all the prime implicants of Core Function. Indeed, the prime implicants containing variables different from those appearing in the completion code **x** (in our example: **<3,2>, <3,3>, <4,2>, <4,3>)** do not appear in the list of prime implicants which have been generated.

A simple solution for producing all the prime implicants of Core Function is the following one.

First, multiply $\mathbf{a_1, a_2, \ldots, a_9}$ by **c(3,1;4,1).**

Then extend the list $\mathbf{(a_1 + a_2 + \ldots + a_9)}$ with $\mathbf{(a_{10} + a_{11} + \ldots + a_{18})}$ and the list $\mathbf{(b_1 + b_2 + \ldots + b_9)}$ with $\mathbf{(b_{10} + b_{11} + \ldots + b_{18})}$ in order to obtain all the marks and all the prime implicants containing both variables **<3,1>** and **<4,2>,** in addition to the marks and the prime implicants containing variables **<3,1>** and **<4,1>** obtained by the product **($a_1$+$a_2$+…) * ($b_1$+$b_2$+…)** :

**$a_{10}$ = c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1) * c(3,1;4,2)**

**$b_{10}$ = c(1,1;4,2) *c(2,1;4,2) * c(1,1;2,1)**

**$a_{11}$ = c(1,2;2,1) * c(1,2;3,1) * c(2,1,3,1) * !c(1,1;2,1) * c(3,1;4,2)**

**$b_{11}$ = c(1,2;4,2) * c(2,1;4,2) * c(1,2;2,1) * !c(1,1;2,1)**

**$a_{12}$ = c(1,3;2,1) * c(1,3;3,1) * c(2,1;3,1) * !c(1,1;2,1) * !c(1,2;2,1) * c(3,1;4,2)**

**$b_{12}$ = c(1,3;4,2) *c(2,1;4,2) * !c(1,1;2,1) * !c(1,2;2,1)**

**$a_{13}$ = c(1,1;2,2) * c(1,1;3,1) * c(2,2;3,1) * !c(2,1;4,2) * c(3,1;4,2)**

**$b_{13}$ = c(1,1;4,2) * c(2,2;4,2) * c(1,1;2,2) * !c(2,1;3,1)**

**$a_{14}$ = c(1,2;2,2) * c(1,2;3,1) * c(2,2;3,1) * !c(1,1;2,2) * !c(2,1;4,2) * c(3,1;4,2)**

**$b_{14}$ = c(1,2;4,2) * c(2,2;4,2) * c(1,2;2,2) *!c(1,1;2,2) * !c(2,1;3,1)**

**$a_{15}$ = c(1,3;2,2) * c(1,3;3,1) * c(2,2;3,1) * !c(1,1;2,2) * !c(1,2;2,2) * !c(3,1;4,2)**

**$b_{15}$ = c(1,3;4,2) * c(2,2;4,2) * !c(1,1;2,2) * !c(1,2;2,2) * !c(2,1;3,1)**

**$a_{16}$ = c(1,1;2,3) * c(1,1;3,1) * c(2,3;3,1) * !c(2,1;4,2) *!c(2,2;4,2) * c(3,1;4,2)**

**$b_{16}$ = c(1,1;4,2) * c(2,3;4,2) * c( 1,1;2,3) * !c(2,1;3,1) *!c(2,2;3,1)**

**$a_{17}$ = c(1,2;2,3) * c( 1,2;3,1) *c(2,3;3,1) * !c(1,1;2,3) * !c(2,1;4,2) *!c(2,2;4,2) * c(3,1;4,2)**

**$b_{17}$ = c(1,2;4,2) * c(2,3;4,2) * c(1,2;2,3) * !c(1,1;2,3) * !c(2,1;3,1) *!c(2,2;3,1)**

**$a_{18}$ = c(1,3;2,3) * c(1,3;3,1) * c(2,3;3,1) * !c(1,1;2,3)* !c(1,2;2,3) * !c(2,1;4,2) *!c(2,2;4,2) * c(3,1;4,2)**

**$b_{18}$ = c(1,3;4,2) * c(2,3;4,2) * !c(1,1;2,3) * !c(1,2;2,3) * !c(2,1;3,1) *!c(2,2;3,1)**

The product **($a_{10}$ + $a_{11}$ + … + $a_{18}$) * ($b_{10}$ + $b_{11}$ +… +$b_{18}$)** produces all the nine implicants of Core Function containing only **<3,1>** and **<4,2>** and no other variable of triplets **3** amd **4.** Similarly, a new product **($a_{19}$ +… + $a_{27}$) * ($b_{19}$ + … + $b_{27}$)** can produce all the prime implicants of Core Function containing only **<3,1>** and **<4,3>** of the variables of triplets **3** and **4**, while the product **($a_{28}$ + … $a_{36}$) * ($b_{28}$ + … +$b_{36}$)** can produce all the prime implicants of Core Function characterized by variables **<3,2>** and **<4,1>.**

In this way all the prime implicants of Core Function will be produced by the product

$$(a_1 + \dots + a_{10} + \dots + a_{19} + \dots + a_{28} + \dots + a_{37} + \dots + a_{46} + \dots + a_{55} + \dots + a_{67} + \dots + a_{73} + \dots) \quad (12.1)$$

$$(b_1 + \dots + b_{10} + \dots + b_{19} + \dots + b_{28} + \dots + b_{37} + \dots + b_{46} + \dots + b_{55} + \dots + b_{67} + \dots + b_{73} + \dots)$$

where the nine pairs of variables **<3,1>, <4,1>; <3,1>, <4,2>; <3,1>,< 4,3>; <3,2>, <4,1>; <3,2>, <4,2>; <3,3>,<4,1>; <3,3>, <4,2>; <3,3>, <4,3>** are involved.

It is easy to prove that every elementary product as $(a_{10} + a_{11} + \dots + a_{18}) * (b_{10} + b_{11} + \dots + b_{18})$ produces a subset of prime implicants of Core Function disjoint from the other subsets of prime implicants; that is, a prime implicant produced by a subset does not appear in any other subset. Besides, the value of a subset is the optimal one for that subset of prime implicants.

Obviously, as shown in **Section 10**, many complemented compatibilities will be added to the elements of **Eq(12.1)** in such a way that, for all **i<>j**, $a_i * b_j = 0$.

## APPENDIX 13

### Sums of remainders

Consider the following remainders:

$a_1 = c(1,1;2,1) * c(1,1;3,1) * c(2,1;3,1)$

$a_2 = c(1,2;2,1) * c(1,2;3,1) * c(2,1;3,1)$

If we allow the use of marks in the completion code, which will not allowed in the future conclusion , and assume the following value of the completion code

$x = c(2,1;4,1) * c(3,1;4,1) * c(1,1;4,1) * c(1,2;4,1)$

we con write:

$\text{val}\ (a_1 + a_2) = \text{NMT1}(4)$

If we add a new remainder

$a_3 = c(1,3;2,1) * c(1,3;3,1) * c(2,1;3,1)$

and apply a new complation code

$x = c(2,1;4,1) * c(3,1;4,1) * c(1,1;4,1) * c(1,2;4,1) * c(1,3;4,1)$

we obtain

$\text{val}\ (a_1 + a_2 + a_3) = (3/4) \cdot \text{NMT1}(4)$

If we assume that $a_1$ and $a_2$ are remainders of **CF(6)** and apply the following structure of completion code:

$x_1 = !c(1,2;2,1) * c(1,1;4,1) * c(2,1;4,1) * c(3,1;4,1) * c(1,1;5,1) * c(1,1;6,1) * c(2,1;5,1) * c(2,1;6,1) * c(3,1;5,1) * c(3,1;6,1) * c(4,1;5,1) * c(4,1;6,1) * c(5,1;6,1)$

$x_2 = !c(1,1;2,1) * c(1,2;4,1) * c(2,1;4,1) * c(3,1;4,1) * c(1,2;5,1) * c(1,2;6,1) * c(2,1;5,1) * c(2,1;6,1) * c(3,1;5,1) * c(3,1;6,1) * c(4,1;5,1) * c(4,1;6,1) * c(5,1;6,1)$

we obtain:

$\text{val}\ (a_1 + a_2) = \text{NMT1}(6)$

Finally, if we assume that also $a_3$ is a remainder of **CF(6)** and that the structure of completion code is as follows:

**$x_2$ = ! c(1,1;2,1) *!c(1,3;2,1) * ...**

**$x_3$ = !c(1,1;2,1) * !c(1,2;2,1) * ...**

we obtain:

**val ($a_1 + a_2 + a_3$) = (3/4) · NMT1(6)**

By proceeding as in the above examples, it is easy to prove the following theorem:

**<u>Theorem 13.1</u>**

Even accepting the presence of marks in the completion code, a sum of remainders is always less than, or equal to, **NMT1(n) .**

**<u>Sums of marks</u>**

Consider the following maks of **CF(4)**

**$m_1$ = c(1,1;2,1) * c(1,1;3,1) * c(3,1;4,1)**

**$m_2$ = c(1,2;2,1) * c(1,2;3,1) * c(3,1;4,1)**

**$m_3$ = c(1,3;2,1) * c(1,3;3,1) * c(3,1;4,1)**

Obviously,

**val ($m_1$) = val ($m_2$) = val ($m_3$) = NMT1(4)**

By assuming the completion code

**x = c(2,1;3,1) * c(2,1;4,1) *c(3,1;4,1)**

we obtain:

**val ($m_1 + m_2 + m_3$) = 3 · NMT1(4)**

Now consider the following marks of **CF(4):**.

**$m_4$ = c(1,1;2,1) * c(1,1;3,1) * c(1,1;4,1)**

**$m_5$ = c(1,2;2,1) * c(1,2;3,1) * c(1,2;4,2)**

where

**val ($m_4$) = val ($m_5$) = NMT1(4)**

by assuming the following values of completion code:

**$x_1$ = !c(1,2;2,1) * c(2,1;3,1) * c(2,1;4,1) *c(3,1;4,1)**

**$x_2$ = !c(1,1;2,1) * c(2,1;3,1) * c(2,1;4,2) *c(3,1;4,2)**

we obtain:

**val ($m_4 + m_5$) = 1 · NMT1(4)**

As shown by the preceding examples, it is easy to prove the following theorem:

**<u>Theorem 13.2</u>**

The value of a sum of **k** marks

**val ($m_1 + m_2 + ... + m_k$ )**

is olways less or equal to **k · NMT1(n)**

By summarizing, if $\mathbf{m_i}$ is a pure mark and $\mathbf{r_i}$ is a pure remainder:

$\mathbf{val\ (m_i) = NMT1(n)}$

$\mathbf{val\ (m_1 + m_2 + \ldots + m_k) = \ < k \cdot NMT1(n)}$

$\mathbf{val\ (r_i) = 0}$

$\mathbf{val\ (r_1 + r_2 + \ldots + r_k) = 0}$

<u>**Value of an AND gate**</u>

Consider an **AND** gate having nodes **A** and **B** as its inputs and node **C** as its output. Let **c** one of the prime implicants of the function implemented by node **C** and let **x** be the completion code (or let $\mathbf{x_i}$ be one of the addends of the completion code) used for evaluating **C.**

Let **I(c)** be the prime implicant of **CF(n)** defined by **c**.

If **x** ($\mathbf{x_i}$) is a mark of **CF(n)**, let **I(x)** ($\mathbf{I(x_i)}$ ) be the prime implicant of **CF(n)** defined by **x** ($\mathbf{x_i}$).

Four different chances may occur.

As a first chance, assume that **I(c) = I(x)**.

In this case, we may replace mark **x** with a remainder **x'**

As a second chance, assume that **I(c) <> I(x)** and **c * x = I(c) * I(x)** as in the following example:

**c = c(1,1;2,1) * c(1,1;3,1) *c(1,1;4,1) * c(2,1;3,1) * c(2,1;4,2) * c(3,1;4,2)**

**x = c(1,2;2,1) * c(1,2;3,1) * c(1,2;4,2) * c(2,1;3,1) * c(2,1;4,1) * c(3,1;4,1)**

In this case **val(c * x) = 0**, since the value of the product of two prime implicants is always equal to **0**, and mark **x** may be replaced with remainder **x'**, in such a way that **c * x' = I(c)**

As a third chance, assume that **I(c) <>I(x)** and **(c * x)** is equal to one of the prime implicants of **CF(n)** of which **x** is a mark, as in the following example:

**c = c(1,1;2,1) * c(1,1;3,1) *c(1,1;4,1) * c(2,1;3,1) * c(2,1;4,2) * c(3,1;4,2)**

**x = c(1,2;2,1) * c(1,2;3,1) * c(1,2;4,2) * c(3,1;4,1)**

In this case, the contribution to the value of **c** would be produced outside the analyzed **AND** gate and the value of **A°B** will not be increased.

As a fourth example, assume that **I(c) <> I(x)** and **(c * x)** is equal to one of the prime implicants of **CF(n)** of which **c** is a mark as in the following example:

**c = c(1,1;2,1) * c(1,1;3,1) *c(1,1;4,1) * c(2,1;3,1)**

**x = c(1,2;2,1) * c(1,2;3,1) * c(1,2;4,2) * c(2,1;3,1) * c(2,1;4,1) * c(3,1;4,1)**

Also in this case , mark **x** may be replaced by a remainder **x'.**

Notice that in all the four cases which have been discussed, either the value of **val(A * B)** grows or it remains constant. Besides, the assumption that the value of a sum of remainders is always equal to **0** implies a reduction of **val(A)** and **val(B)**. It follows that the value of the most powerful **AND** gate is larger than, or equal to, the value proposed in this paper, as it must be in order to obtain our conclusion.

## REFERENCES

1 A.R. Meo: "On the P versus NP question"

https://www.accademiadellescienze.it/attivita/editoria/lavori-di-soci (2016- 2020)

2 A. R. Meo: "On the P versus NP question: a new proof of inequality", Journal of Computer Science, *17*(5), 511-524. https://doi.org/10.3844/jcssp.2021.511.524

3 Meo, A. R. (2023). On a Proof of Inequality of PvsNP. *Journal of Computer Science*, *19*(1), 87-98. https://doi.org/10.3844/jcssp.2023.87.98 3. A.R. Meo: "On the P vs NP question: a proof of inequality", *arXiv*:1802.05484

4. Lance Fortnow: "The status of the P versus NP problem", Communications of the ACM, September 2009

5. "The **P** versus **NP** problem," in J. Carlson, A. Jaffe, & A. Wiles (eds.), *The Millennium Prize Problem*, pp. 88–104, Providence: American Mathematical Society

6. Razborov, "Lower bounds on the monotone complexity of some Boolean functions". *Soviet Mathematics-Doklady 31*, (1985) 485--493.

7. A.R. Meo: "Some theorems concerning the core function" in "Concurrency, Graphs and Models", Springer – Verlag Berlin Heidelberg , 2008

8. S.A.Cook: The complexity of theorem proving procedures, in: Proc. 3rd Annual ACM Symp. on Theory of Computing, (1971), pp. 151-158. ACM Press

9. K. Mulmuley and M. Sohoni. Geometric complexity theory I: An approach to the P vs. NP and related problems. SIAM Journal on Computing, 31(2):496{526, 2001.