On the P vs NP question: a proof on inequality

Angelo Raffaele Meo
Accademia delle Scienze di Torino

ON THE P vs NP QUESTION: A PROOF OF INEQUALITY

Summary

The analysis discussed in this paper is based on a well-known NP-complete problem which is called "satisfiability problem or SAT". From SAT a new NP-complete problem is derived, which is described by a Boolean function called "core function". In this paper it is proved that the cost of the minimal implementation of core function increases with **n** exponentially. Since the synthesis of core function is an NP-complete problem, this result is equivalent to proving that **P** and **NP** do not coincide.

1. INTRODUCTION

A brief description of the definitions and properties well known among the scientists of modern computational complexity theory which will be made reference to, 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 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, than no **NP-complete** problem has a polynomial time solution.

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

Given a Boolean expression containing only the names of a set of variables (some of which may be complemented), the operators **AND**, **OR** and **NOT**, and parentheses, is there an assignment of **TRUE** and **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:

F=(x_{11} OR x_{12} OR x_{13}) AND

(x_{21} OR x_{22} OR x_{23}) AND

…… AND …… **(1)**

(x_{t1} OR x_{t2} OR x_{t3})

where:

t is the number of clauses or triplets;

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

each variable can appear multiple times in the expression.

If the deterministic Turing machine is assumed as the computational model, with **{0,1,b}** as its set of input symbols, the input data appearing on the tape at the beginning of computation can represent the data of expression **(1)** in the following way:

b b <binary code of number of variables> <separator>b

or

$s_{11}\, n_{111}\, n_{112}\, n_{113}$…….. n_{11m}**b**

$s_{12}\, n_{121}\, n_{122}\, n_{123}$…….. n_{12m}**b**

$s_{13}\, n_{131}\, n_{132}\, n_{133}$…….. n_{13m}**b**

$s_{21}\, n_{211}\, n_{212}\, n_{113}$…….. n_{21m}**b** **(2)**

……

$s_{t3}\, n_{t31}\, n_{t32}\, n_{t33}$…….. n_{t3m}**b**

where:

b is the blank symbol;

t is the number of triplets;

$\mathbf{s_{ij}}$ denotes the sign of variable $\mathbf{x_{ij}}$

(with $\mathbf{s_{ij} = 1}$ denoting that $\mathbf{x_{ij}}$ is preceded by operator **NOT**);

$\mathbf{n_{ijk}}$ denotes the **k-th** component of the binary code **<n_{ij1} n_{ij2} … n_{ijm}>** representing the name of variable $\mathbf{x_{ij}}$;

the binary code of the number $\mathbf{n_v}$ of variables is needed in order to determine the number **m** of binary digits necessary to represent the names of variables according the rule

m = *minimum integer not less than* $\mathbf{\log_2 n_v}$

Notice that, by neglecting the bits of the binary code of the number of variables and the

bits of the separator, the number of input bits on the tape will be

$$\mathbf{t \cdot 3 \cdot (1 + \textit{minimum integer not smaller than } \log_2 (3 \cdot t))} \qquad \mathbf{(3)}$$

since the maximun value of the number of variables is **3·t.**

The properties of Turing machines processing the bit string described by **(2)** will be analyzed in this paper with reference to a family $\{C_n\}$ of Boolean circuits, where C_n has **n** binary inputs and produces the same binary output as the corresponding Turing machine.

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

2. THE CORE FUNCTION

In the case of satisfiability problem with **3** variables for clause, Boolean circuit C_n has **n (=t)** sets of inputs which the binary data described in **(2)** are applied to. (Of course, the binary code of the number of variables and the separator are not needed). The output of C_n (with **n=t**) will take the value **TRUE** when, and only when, there is an assignment of values **TRUE** and **FALSE** to variables making expression **(1) TRUE**.

In order to simplify analysis, circuit C_n will be decomposed into two processing layers as shown in **Fig. 1**, where , as usual, the number **t** of triplets plays the role of symbol **n** in the standard analysis of complexity theory.

In the following analysis, we shall use the symbol **t** when it's necessary to remember the number of triplets and **n** in the other cases.

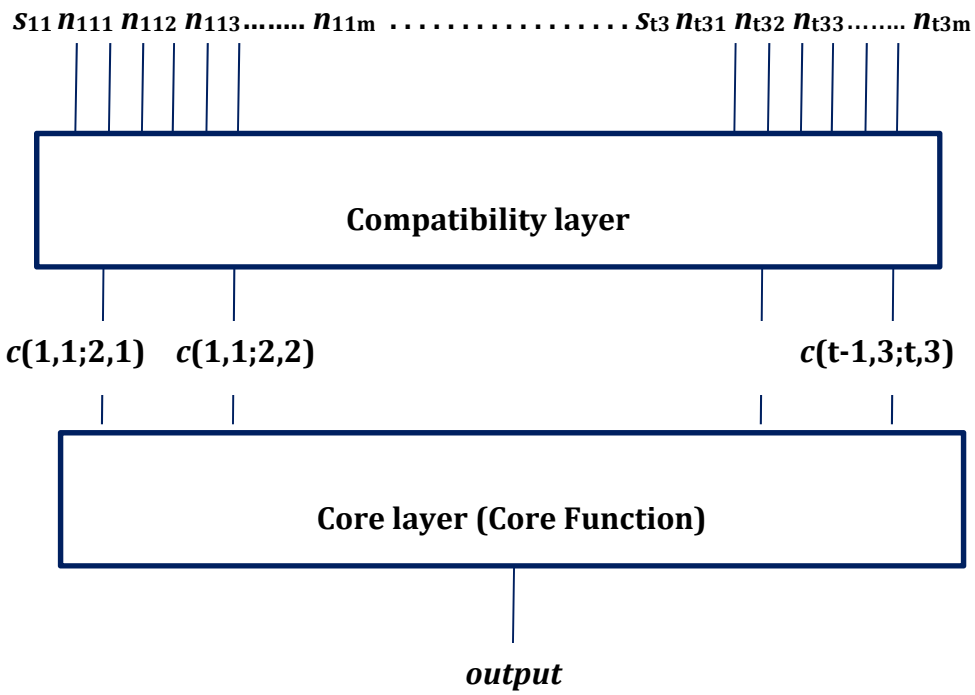


Fig. 1

Decomposition of Boolean circuit C_n into compatibility layer and core 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 $\mathbf{s_{ij}}$ of the former variable is equal to the sign $\mathbf{s_{hk}}$ of the latter,

or

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

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

The compatibility layer is composed of $\mathbf{3 \cdot t \cdot (3 \cdot t-3)/2}$ identical cells, one for each pair of variables belonging to different triplets.

As shown in **Fig. 2**, the inputs of a cell will be the sign $\mathbf{s_{ij}}$ and the name $\mathbf{<n_{ij1}\ n_{ij2}\ \ldots n_{ijm}>}$ of variable **j** of triplet **i**, and the sign $\mathbf{s_{hk}}$ and the name $\mathbf{<n_{hk1}\ n_{hk2}\ \ldots n_{hkm}>}$ 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.

$$\mathbf{c(i,j;h,k)} = \begin{cases} \mathbf{TRUE} \Leftrightarrow x_{ij} \textbf{ is compatible with } x_{hk} \\ \mathbf{FALSE} \Leftrightarrow x_{ij} \textbf{ is not compatible with } x_{hk} \end{cases}$$

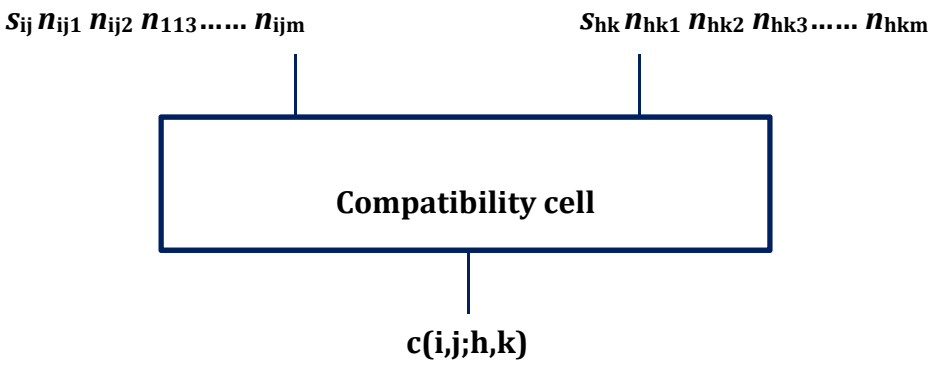


Fig. 2

Compatibility Cell

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

The core layer processes only the $\mathbf{9 \cdot t \cdot (t-1)/2}$ compatibility variables **c(i,j;h,k)** and produces the global result of computation.

As the circuit $\mathbf{C_n}$**,** also the global Boolean function implemented by $\mathbf{C_n}$ may be decomposed into two layers of functions. At the compatibility layer, the function implemented by a cell may be written as follows (by using the symbols $*$, **+**, and **!** for representing **AND, OR** and **NOT** operators, respectively):

$$
\begin{aligned}
c(i,j;h,k) = {} & s_{ij}*s_{hk} + !s_{ij}*!s_{hk} + && \text{(equal sign)} \\
& + n_{ij1}*!n_{hk1} + !n_{ij1}*n_{hk1} + \\
& + n_{ij2}*!n_{hk2} + !n_{ij2}*n_{hk2} + && \text{(at least one bit in the} \\
& \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots && \text{variables names different)} \\
& + n_{ijm}*!n_{hkm} + !n_{ijm}*n_{hkm}
\end{aligned} \tag{4}
$$

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 function can be determined by proceeding as follows.

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

$$<1i_1>, <2i_2>, \ldots, <ti_t> \tag{5}$$

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 Π^k be the product of all the compatibility variables relative to the **k-th** of selections **(5)**:

$$
\begin{aligned}
\Pi^k = {} & c(1,i_1; 2,i_2)*c(1,i_1; 3,i_3)*\ldots \\
& \ldots*c(t-1,i_{t-1}; t, i_t)
\end{aligned} \tag{6}
$$

The core function can be defined as the sum

$$\Sigma_k \Pi^k \tag{7}$$

of the products **(6)** relative to all the selections **(5)**.

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

$$
\begin{aligned}
& CF(3) = c(1,1;2,1)*c(1,1;3,1)*c(2,1;3,1) + \\
& c(1,1;2,1)*c(1,1;3,2)*c(2,1;3,2) + \\
& c(1,1;2,1)*c(1,1;3,3)*c(2,1;3,3) + \\
& c(1,1;2,2)*c(1,1;3,1)*c(2,2;3,1) + \\
& \ldots\text{(other 22 products)}\ldots + \\
& c(1,3;2,3)*c(1,3;3,3)*c(2,3;3,3)
\end{aligned} \tag{8}
$$

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

Notice that the processing work of the cell of **Fig. 2** 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 $\mathbf{9 \cdot t \cdot (t - 1)/2}$ 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 interesting properties of core function have been discussed in ref. (23).

3. A THEOREM OF BOOLEAN MONOTONIC FUNCTIONS

Let $\mathbf{f(x_1,x_2, ..., x_t)}$ be an isotonic Boolean function, that is a Boolean function which can be implemented with only **AND** and **OR** gates, applied to uncomplemented literals $\mathbf{x_1, x_2, ..., x_t}$. It was believed that the minimum cost implementation of $\mathbf{f(x_1,x_2,...,x_t)}$ 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. (**8**)).

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.

3.1. THEOREM

Let $\mathbf{I_{min}}$ be one of the minimum cost implementations of the isotonic Boolean function $\mathbf{f(x_1, x_2,...,x_t)}$, 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 such that

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

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

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. 3)**. 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.**

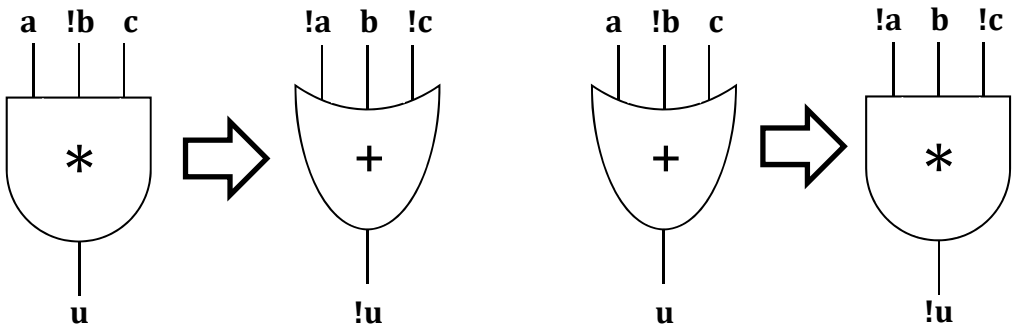


Fig. 3

The transformation of 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. 4**).

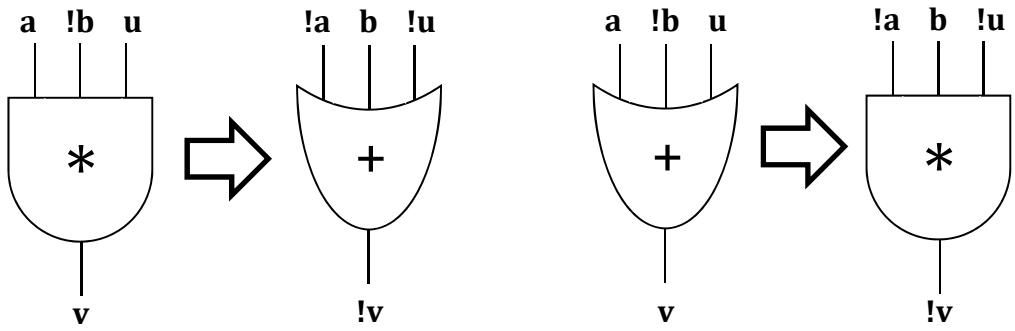


Fig. 4

The transformation of gates of level 2

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

As an example, the two level subnetwork of **Fig. 5** will be transformed into the subnetwork of Fig. **6**. 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 $\mathbf{I_{min}}$**,** 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 twice the number of gates of $\mathbf{I_{min}}$**.**

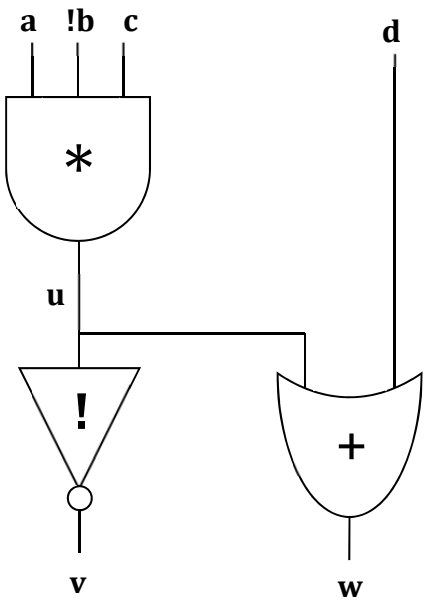


Fig. 5

A two level subnetwork

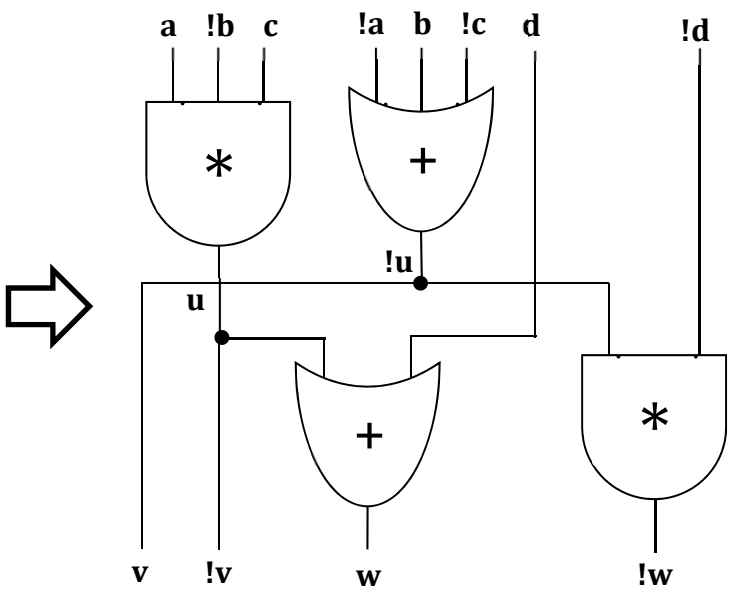


Fig. 6

The transformation of the subnetwork

At level **0**, before the gates of **Fig. 6**, **t NOT** gates might be necessary to generate the complemented input variables $!x_i$. Therefore, **t** has been added in the statement of the theorem.

This theorem will be very important in order to simplify the analysis of core function circuits.

4. PROPERTIES OF CORE FUNCTION

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

4.1. PROPERTY 1

Core function is totally isotone.

4.2. PROPERTY 2

Any product **(6)** is a prime implicant of core function (that is, a product of compatibilities ("PoC") which implies core function and no other term of it).

4.3. PROPERTY 3

Since the different selections of each of variables **(5)** are **3**, the number of prime implicants of the 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 section, reference will be made to the following definitions.

5.1. 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**

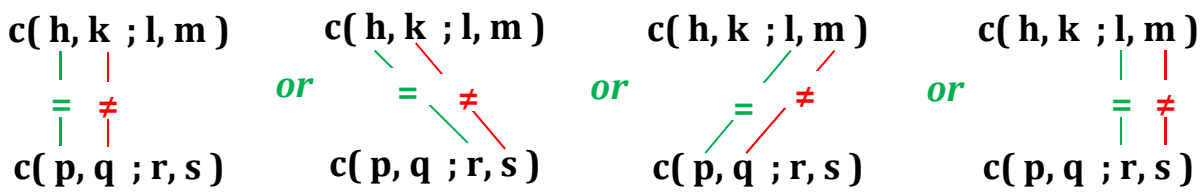
(h = p and k ≠ q)

or **(h = r and k ≠ s)**

or **(l = p and m ≠ q)**

or **(l = r and m ≠ s)**

In a graphic scheme:



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**).

5.2. DEFINITION OF SPURIOUS PRODUCTS OF COMPATIBILITIES

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)}

5.3. DEFINITION OF IMPURE PRODUCTS OF COMPATIBILITIES

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**) that contains one or more complemented variables, will be defined as an impure term.

5.4. DEFINITION OF CORE OF A POC

The product of all the uncomplemented variables of **T** will be defined as the core of **T.**

5.5. DEFINITION OF MARK

Consider a not spurious subset of compatibilities satisfying the property that each of the indexes of triplet appears at least once in some variable. The product of the variables of such a

subset will defined as a "mark" of the prime implicant of which it contains a subset of compatibilities.

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

$$\mathbf{M = c(1,a;2,b)*c(1,a;3,c)*c(1,a;4,d)} \qquad \mathbf{(9)}$$

(where **a, b, c, d** are elements of **{1,2,3}**)

is a mark of the prime implicant

$$\mathbf{P= c(1,a;2,b)*c(1,a;3,c)*c(1,a;4,d)*c(2,b;3,c)*c(2,b;4,d)*c(3,c;4,d)} \qquad \mathbf{(10)}$$

since all the indexes of triplet appear at least once in **(9)**.

5.6. DEFINITION OF SPURIUS MARK

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 the 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** whose core is a (possibly spurious) mark will be a defined as a (possibly spurious) impure mark.

5.7. DEFINITION OF EXTENDED PRIME IMPLICANTS

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.

5.8. DEFINITION OF REMAINDER

A **PoC** which is neither a (possibly spurious or impure) mark nor an (extended) prime implicant will be called a "remainder". A remainder can be associated to one or more prime implicants, of which it contains a subset of compatibilities.

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

$$\mathbf{R = c(2,b;3,c)*c(2,b;4,d)*c(3,c;4,d)} \qquad \mathbf{(11)}$$

is a remainder of the prime implicant **(10)**.

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

$$\begin{aligned} \mathbf{P1} &= \mathbf{c(1,1;2,1)*c(1,1;3,1)*c(2,1;3,1)} \\ \mathbf{P2} &= \mathbf{c(1,2;2,1)*c(1,2;3,1)*c(2,1;3,1)} \\ \mathbf{P3} &= \mathbf{c(1,3;2,1)*c(1,3;3,1)*c(2,1;3,1)} \end{aligned} \qquad \mathbf{(12)}$$

On the definitions of mark and remainder the following properties are based.

5.9. PROPERTY 4

A not spurious mark **M** specifies a corresponding prime implicant **P** uniquely. Indeed, if all the indexes of triplet appear in **M**, the product **(6)** is completely defined.

We shall write

P = I(M)

to state that **P** is the prime implicant specified by **M**.

As already mentioned, a remainder **R** does not specify a corresponding prime implicant uniquely. In the example relative to **CF(3)** above described, three prime implicants correspond to **R = c(2,1;3,1)**, as shown by (**12**), since a single index of triplet is missing in that remainder. In general, if **z** triplets are not involved in **R**, there are $\mathbf{3^z}$ different ways of involving the missing triplets.

Hence the following property follows.

5.10. PROPERTY 5

A not spurious remainder **R** in which the indexes of **z** triplets are missing corresponds to $\mathbf{3^z}$ different prime implicants.

Finally, the following property can be proved. The proof is not too difficult and it is omitted for the sake of brevity.

5.11. PROPERTY 6

Let $\mathbf{P_1}$ and $\mathbf{P_2}$ be two **PoC's** such that $\mathbf{P_1*P_2}$ is equal to a prime implicant **P** of a core function. Either $\mathbf{P_1}$ or $\mathbf{P_2}$ is a mark of **P.**

6. THE EXTERNAL CORE FUNCTION

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

Of course,

$$\mathbf{ECF(n,I_j) = I_j * \Pi_{k \neq j} (!I_k)} \qquad \mathbf{(13)}$$

where $\mathbf{!I_k}$ denotes the complement of $\mathbf{I_k}$, i.e. **(NOT $\mathbf{I_k}$).**

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

$$\mathbf{ECF(n) = \sum_j ECF(n, I_j)} \qquad \mathbf{(14)}$$

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

6.1. THEOREM 1

Let **T** be a term (or extended prime implicant) of **CF(n)**. It must be the product of all the compatibilities of a prime implicant $\mathbf{I_j}$ of **CF(n)** and other compatibilities, that is,

$$\mathbf{T = I_j * X}$$

where **X** is a possibly empty **PoC,** which can also be written as $\mathbf{T = T(I_j)}$

All the minterms of $\mathbf{T(I_j)}$ contained in **ECF(n)** are minterms of $\mathbf{ECF(n,I_j)}$.

Indeed, for any $\mathbf{k \neq j}$,

$$\mathbf{T(I_j) * ECF(n,I_k) = I_j * X * I_k * \Pi_{l \neq k} (!I_l) = 0} \qquad \mathbf{(15)}$$

6.2. THEOREM 2

Let **T** be a term of **CF (n)** implying two or more prime implicants of **CF(n)** as, for example,

$$\mathbf{T = T (I_j, I_k)}$$

The number of minterms of

$\mathbf{T(I_j,I_k)}$ belonging to **ECF(n)** is equal to **0**.

Indeed,

$$\mathbf{T(I_j, I_k) * ECF(n, I_h) = 0} \qquad \mathbf{(16)}$$

for any **h**.

The preceding theorems 1 and 2 are nearly obvious. On the contrary, the following theorem 3 appears rather complex.

6.3. THEOREM 3

Let $\mathbf{T = T\ (I_j) = I_j{*}X}$ be a term of **CF (n)** which is spurious for a single compatibility **X**.

If **NMT(F)** denotes the number of minterms of Boolean function **F** , the number of minterms of $\mathbf{I_j{*}X}$ contained in $\mathbf{ECF(n,I_j)}$ is

$$\mathbf{NMT(Ij * X * ECF(n, Ij))} <= \frac{\mathbf{1}}{\mathbf{2}} \cdot \mathbf{NMT(ECF(n, Ij))} \qquad \mathbf{(17)}$$

Proof

The number of minterms of **T** contained in $\mathbf{ECF(n,I_j)}$ is equal to the number of minterms contained in

$$\mathbf{I_j{*}X{*}ECF(n,I_j) = I_j{*}X{*}\Pi_{k\neq j}(!I_k) = I_j{*}X{*}(A{*}(!X)+B) = I_j{*}X{*}B} \qquad \mathbf{(18)}$$

where **A** and **B** are two antitone functions (that is, two monotone functions which can be described with complemented variables only) containing neither **X** nor **!X**.

The number of minterms of $\mathbf{ECF\ (n,\ I_j)}$ is equal to the number of minterms contained in

$$\mathbf{I_j{*}\Pi_{k\neq j}\ (!I_k) = I_j{*}A{*}(!X) + I_j{*}B} \qquad \mathbf{(19)}$$

Besides,

$$\mathbf{NMT\ (I_j{*}B) = 2{*}NMT\ (I_j{*}X{*}B)} \qquad \mathbf{(20)}$$

since **X** appears neither in **B** nor in $\mathbf{I_j}$.

The statement of this Theorem 3 derives from the comparison of **(18), (19)** and **(20)**.

By proceeding in the same way it is possible to generalize the preceding Theorem 3 as follows.

6.4. THEOREM 4

Let

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

are **m s**purious compatibilities.

The number of its minterms contained in $\mathbf{ECF(n,\ I_j)}$ is

$$\mathbf{NMT(I_j{*}X_1{*}X_2{*}...{*}X_m{*}ECF(n,\ I_j))} <= \frac{1}{2^m}\cdot\mathbf{NMT(ECF(n,\ I_j))} \qquad \mathbf{(21)}$$

Proof

For the sake of brevity, the proof of **(21)** is restricted to the case **m=2**.

In this case we can write:

$$\mathbf{ECF(n,\ I_j) = I_j{*}A{*}(!X_1)+I_j{*}B{*}(!X_2)+I_j{*}C{*}(!X_1){*}(!X_2)+I_j{*}D} \qquad \mathbf{(22)}$$

where functions **A, B, C, D** do not contain variables $\mathbf{X_1}$ or $\mathbf{X_2}$.

Notice that

$\mathbf{X_1{*}X_2{*}ECF(n,\ I_j) = X_1{*}X_2{*}I_j{*}D}$

and **(X_1∗X_2∗D)** contains **¼** of the minterms of **D.**

From these two remarks the statement of Theorem 4 derives.

The following Theorems 5 and 6 are analogous to preceding Theorems 3 and 4, respectively.

6.5. THEOREM 5

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

T = I_j∗(!X)

The number of minterms of **ECF(n,I_j)** contained in **T** is

$$\mathbf{NMT}\left(\mathbf{Ij} * (!\,\mathbf{X}) * \mathbf{ECF}(\mathbf{n}, \mathbf{I}_j)\right) <= \left(\frac{\mathbf{1}}{\mathbf{2}} + \frac{\mathbf{K(n)}}{\mathbf{2}}\right) \cdot \mathbf{NMT}\left(\mathbf{ECF}(\mathbf{n}, \mathbf{I}_j)\right) \qquad \textbf{(23)}$$

where **K(n)** is positive and less than **1** and it is a quickly decreasing function of **n**. The proof of **Theorem 5** and the properties of function **K(n)** are discussed in **Appendix 1.**

6.6. THEOREM 6

Let **T=T(I_j)** an impure term of **CF(n)** characterized by **m** impure variables:

T=I_j∗(!X_1)∗(!X_2)∗...(!X_m)

The number of minterms of **ECF(n,I_j)** contained in **T** is

$$\mathbf{NMT}\left(\mathbf{T} * \mathbf{ECF}(\mathbf{n}, \mathbf{I}_j)\right) <= \left(\frac{\mathbf{1}}{\mathbf{2}} + \frac{\mathbf{K(n)}}{\mathbf{2}}\right)^{m} \cdot \mathbf{NMT}\left(\mathbf{ECF}(\mathbf{n}, \mathbf{I}_j)\right) \qquad \textbf{(24)}$$

Also Theorem 6 is discussed in Appendix 1.

Notice that **NMT(ECF(n,I_j)) = NMT(ECF(n,I_k))** for any **j** and **k.** It will be called **NMT1**.

7. THE REFERENCE ARCHITECTURE

Fig. 7 shows the network which will implement core function. It is characterized by a number of subnetworks each of which has the structure shown by **Fig. 8.** As an alternative, the network of **Fig. 7** might be composed by a single network of the type of **Fig. 8**.

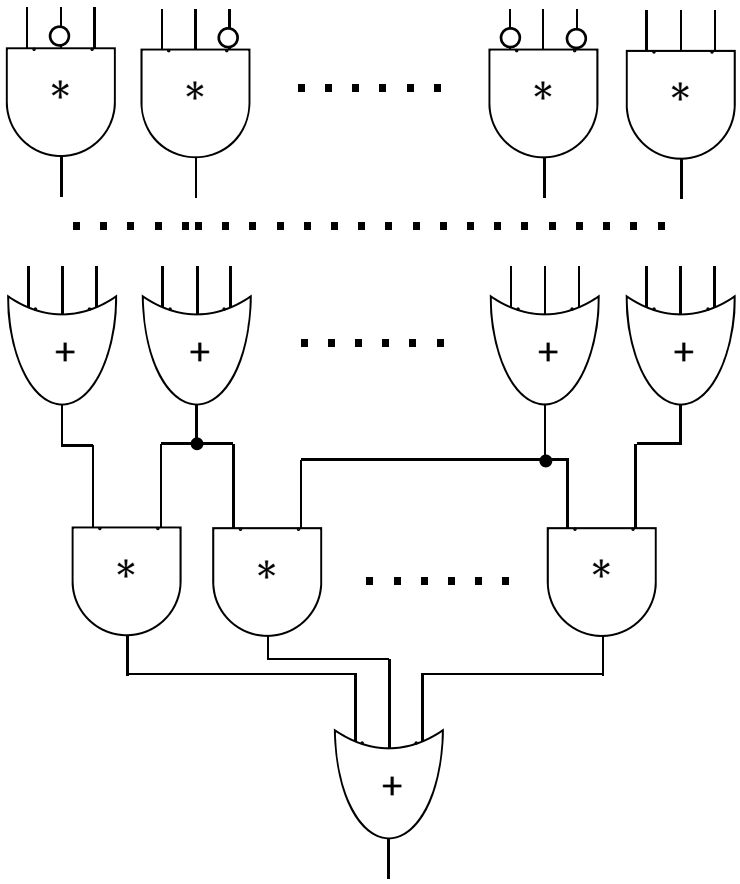


Fig. 7

The Reference Architecture

The circuit presented in **Fig. 8** will be called a "primary composite addendum (**PCA**)". Every $\mathbf{F_i}$ will be called a "primary composite addendum factor" (**PCAF**).

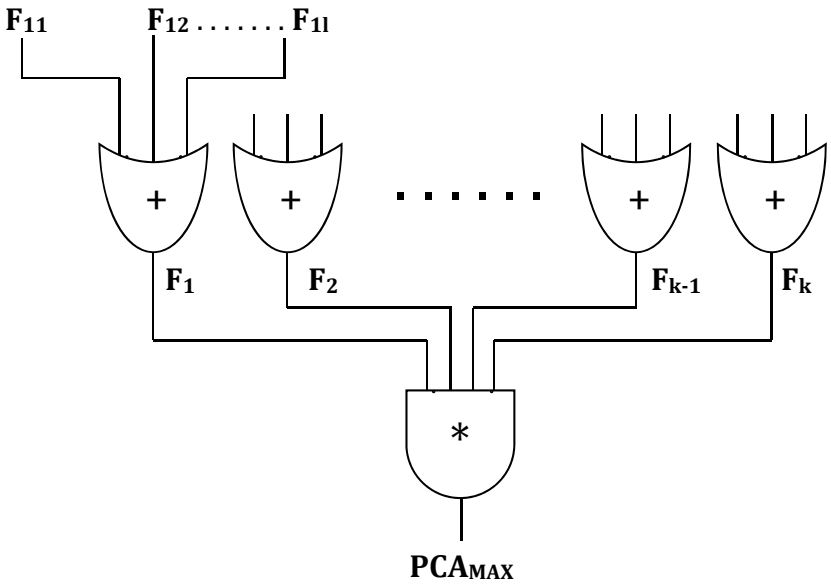


Fig. 8

The primary composite addendum

If the number of **PCA's** of the minimum cost implementation of **CF(n)** increased with **n** according to an exponential law, also the cost of this implementation would increase according to an exponential law, the cost being represented by the number of **AND** gates at the bottom of **Fig. 7**.

Therefore, the following analysis refers to the case in which the number of **PCA's** of the minimum cost implementation of **CF(n)** increases with **n** according to a polynomial law.

Besides, reference will be made to the following definitions. The merit of a (possibly, impure or spurious) prime implicant $\mathbf{P_i}$ of **CF(n)** will be defined as the number of minterms of **ECF(n)** that $\mathbf{P_i}$ covers and the merit of a **PCA** will be defined as the number of minterms of **ECF(n)** that this **PCA** covers.

We shall discuss the properties of the **PCA** which contains the maximum number of minterms of **ECF(n).** It will be called $\mathbf{PCA_{MAX}}$**.**

It is easy to prove that the number of minterms of **ECF(n)** contained in the function implemented by $\mathbf{PCA_{MAX}}$ increases with **n** as $\mathbf{3^n}$. Besides, also the number of prime implicants of **CF(n)** implemented by $\mathbf{PCA_{MAX}}$ increases with **n** as $\mathbf{3^n}$**.**

8. SYNTHESIS OF MAXIMUM MERIT PCA

Consider the decomposition of the maximum merit **PCA** into **k** factors

$\mathbf{F_1, F_2, F_3 \ldots, F_k}$

(**Fig. 8**). Consider also the partial product

$\mathbf{F_{2\text{-}k} = F_2 * F_3 * \ldots * F_k}$

where the symbols $\mathbf{F_2, F_3, .., F_k}$, above used to denote processing units have the meaning of their corresponding Boolean values, as will be done in the future when such a choice will not generate confusion.

Obviously, the value of the maximum merit **PCA**, that is, the function implemented by it, will be

$\mathbf{val(PCA_{MAX}) = F_1 * F_{2\text{-}k}}$

Let $\mathbf{P_1, P_2, \ldots, P_v}$ be the prime implicants of function $\mathbf{F_1}$ and $\mathbf{Q_1, Q_2, \ldots, Q_w}$ the prime implicants of function $\mathbf{F_{2\text{-}k}}$. Obviously, the value of the maximum merit **PCA** will be the sum of all the $\mathbf{v*w}$ products $\mathbf{P_i * Q_j}$. Some of these products will be equal to **0**; the other ones will be (possibly, impure or spurious) implicants of **CF(n).**

The number of minterms of **ECF(n)** covered by each of these implicants will be defined as its merit.

Notice that any product $\mathbf{P_i * Q_j}$ "must" be an implicant of **CF(n) (**possibly, extended with spurious or impure variables). Otherwise, the considered solution would not be a correct implementation of **CF(n).**

Fig. 9 shows the symbols which will be used in the following analysis.

An arc connecting node $\mathbf{P_i}$ with node $\mathbf{Q_j}$ denotes that the product $\mathbf{P_i * Q_j}$ is a (possibly impure or spurious) implicant of **CF(t).** For example, this is the case of arcs $\mathbf{P_1 - Q_1, P_1 - Q_2, P_2 - Q_1, P_2 - Q_2}$ in **Fig. 9.** The labels of the arcs $\mathbf{I_0, I_1, I_2, I_3, I_0'}$ (perhaps, the same as $\mathbf{I_0}$), $\mathbf{I_1'}$ are the names of the prime implicants represented by those arcs. A missing arc denotes that the corresponding product is equal to **0**; thus, for example, $\mathbf{P_1 * Q_3 = 0}$ or $\mathbf{P_4 * Q_3 = 0.}$

Notice that an arc might be labelled with the product of two or more different prime implicants, as in the case of $\mathbf{P_4 - Q_4}$ which has been labelled with the product $\mathbf{I_3 * I_4.}$ However, as already proved, the merit of the product of two or more prime implicants is equal to **0.**

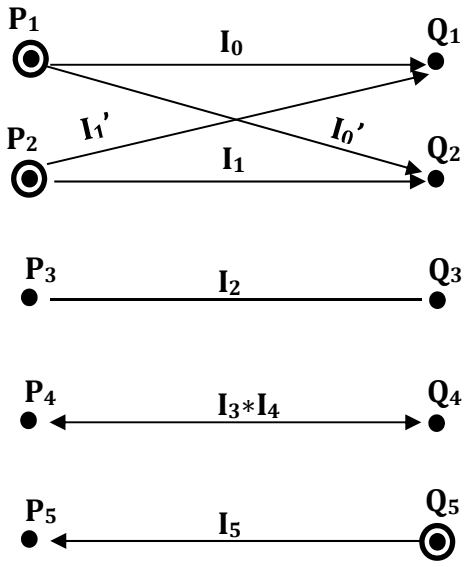


Fig. 9

The Prime Implicants produced by a PCA_{MAX}

Three different cases are worth mentioning.

Case 1.

Both P_3 and Q_3 are marks, and $I(P_3) = I(Q_3)$. Of course, in this case, $I_2 = I(P_3) = I(Q_3)$. Notice that, by virtue of Property 6 of previous Section **5**, if $P_3 * Q_3$ is not equal to **0**, at least one of these two terms is a mark of the generated prime implicant.

Case 2

P_2 is a mark and Q_2 is a remainder. Obviously, $I_1 = I(P_2)$. The considered arc is oriented from P_2 to Q_2 in order to remember that P_2 is the "origin" of the arc, that is, the mark of the corresponding prime implicant.

This is also the case of the arcs $P_1 - Q_1$, $P_1 - Q_2$, $P_2 - Q_1$.

Notice that in **Case 1** both P_3 and Q_3 might be considered as origins of the prime implicant $I_2 = P_3 * Q_3$.

Case 3

P_5 is a mark of a prime implicant $I(P_5)$ while Q_5 is a mark of a different prime implicant $I(Q_5) \neq I(P_5)$. Since the produced prime implicant I_5 coincides with $I(Q_5)$, the arc has been oriented from Q_5 which is considered as the origin of the arc.

- - - - - - - - -

Since the number of prime implicants implemented by PCA_{MAX} increases with **n** as 3^n, also the number of origins born in the decomposition of **Fig. 5** increases with **n** as 3^n.

Assume that the number of origins labeled as $\mathbf{Q_j}$ is larger than the number of origins labeled as $\mathbf{P_i}$. In this case, of course, the number of $\mathbf{Q_j}$ origins increases with **n** as $\mathbf{3^n}$. The case in which the number of $\mathbf{Q_j}$ origins is less than the number of $\mathbf{P_i}$ origins can be treated in a similar way.

We can organize the $\mathbf{Q_j}$ origins as follows.

Let us start from an origin $\mathbf{Q_{11}}$ and a node $\mathbf{P_1}$ such that $\mathbf{Q_{11} * P_1 \neq 0}$

Collect all the origins $\mathbf{Q_{12}, Q_{13}, \ldots}$ such that $\mathbf{Q_{1j} * P_1 \neq 0}$. Of course, for all the remaining $\mathbf{Q_{ij}}$, $\mathbf{Q_{ij} * P_1 = 0}$.

Now consider a new $\mathbf{P_2}$ and a new $\mathbf{Q_{21}}$ such that $\mathbf{Q_{21} * P_2 \neq 0}$. Collect all the origins $\mathbf{Q_{22}}$, $\mathbf{Q_{23}}$,…such that $\mathbf{Q_{2j} * P_2 \neq 0}$. Repeat the same procedure till all $\mathbf{P_i}$'s and all $\mathbf{Q_{ij}}$'s have been involved.

In **Appendix 2** it is shown that the number of such subsystems cannot increase with **n** exponentially, if we accept the hypothesis that the number of minterms of **ECF(n)** contained in $\mathbf{PCA_{MAX}}$ increases with **n** as $\mathbf{(3^n) \cdot NMT1(n)}$. It follows that the prime implicants of at least one of the subsystems must cover a number of minterms of **ECF(n)** increasing with **n** as $\mathbf{(3^n) \cdot NMT1(n)}$. We can define this subsystem as "the most effective subsystem".

9. THE SYNTHESIS OF THE MOST EFFECTIVE SUBSYSTEM

Let us assume that $\mathbf{<P_1,\{Q_{1j}\}>}$ is the most effective subsystem.

Assume that $\mathbf{Q_{11}}$ is complete in all the compatibilities which characterize it with respect to the other $\mathbf{Q_{ij}}$**'s.**

For example , if index **<1,1> (** input **1** of triplet **1**) appears in $\mathbf{Q_{11}}$ and index **<1,2> (**input **2** of triplet **1)** appears in $\mathbf{Q_{12}}$, then $\mathbf{Q_{11}}$ contains all the compatibilities involving **<1,1>.** Similarly, each of the other $\mathbf{Q_{1j}}$**'s** is complete in all the indexes which characterize it with respect to the other $\mathbf{Q_{1j}}$**'s.**

Such subsystem will be defined as "complete".

A simple example of a complete subsystem relative to **CF(3)** is:

$$\begin{aligned} \mathbf{P_1} &= \mathbf{c(2,1;3,1)} \\ \mathbf{Q_{11}} &= \mathbf{c(1,1;2,1) * c(1,1;3,1)} \\ \mathbf{Q_{12}} &= \mathbf{c(1,2;2,1) * c(1,2;3,1)} \\ \mathbf{Q_{13}} &= \mathbf{c(1,3;2,1) * c(1,3;3,1)} \end{aligned} \qquad \mathbf{(25)}$$

If the considered subsystem is complete, its merit is exactly the sum of the numbers of minterms of **ECF(n)** contained in all the products $\mathbf{P1 * Q_{1j}}$'s.

Since the number of prime implicants implemented by $\mathbf{PCA_{MAX}}$ increases with **n** as $\mathbf{3^n}$, also the number of origins born in the decomposition of **Fig. 9** increases with **n** as $\mathbf{3^n}$.

Let us assume that one origin $\mathbf{Q_{ij}}$ is not complete in all the compatibilities involving an index which does not appear in other origins of the same subsystem. For example, in the case of subsystem **(25)** of **CF(3)**, assume that $\mathbf{Q_{11}}$ does not contain **c(1,1;3,1).** In that case, $\mathbf{P_1}$ should contain **c(1,1;3,1)** and the merits of $\mathbf{Q_{12}}$ and $\mathbf{Q_{13}}$ would be reduced to half of their original value (because of Theorem 3 of Section 6).

In more general terms consider the following complete system relative to **CF(6)**:

$\mathbf{P_1 = c(1,1;2,1)*c(1,1;3,1)*c(2,1;3,1)}$

$\mathbf{Q_{11} = c(1,1;4,1)*c(1,1;5,1)*c(1,1;6,1)*}$
$\mathbf{c(2,1;4,1)*c(2,1;5,1)*c(2,1;6,1)*}$
$\mathbf{c(3,1;4,1)*c(3,1;5,1)*c(3,1;6,1)*}$
$\mathbf{c(4,1;5,1)*c(4,1;6,1)*c(5,1;6,1)}$

$\mathbf{Q_{12}}$ = ...

$\mathbf{Q_{13}}$ = ...

. . . . . .

Assume that **c(1,1;4,1)** is cancelled in $\mathbf{Q_{11}}$. As a consequence, $\mathbf{P_1}$ must be multiplied by **c(1,1;4,1)** in order that $\mathbf{P_1*Q_{11}}$ is still an implicant of **CF(6)**.

This modification implies a reduction of the merits of many $\mathbf{P_1*Q_{1k}}$'**s** according to the following rules:

1) **the merits of 1/3 of them remain unchanged**
2) **the merits of 2/3 of them are multiplied by 1/2.**

Therefore, the total merit of the considered subsystem will be reduced of **(1/3)·1+(2/3)·(1/2)=2/3**

Then assume that both the compatibilities **c(1,1;4,1)** and **c(1,1;5,1)** are cancelled in some of $\mathbf{Q_{1k}}$'s and $\mathbf{P_1}$ is multiplied by **c(1,1;4,1)*c(1,1;5,1)**.

As a consequence of these modifications the merits of the $\mathbf{P_1*Q_{1k}}$'s will be changed as follows:

1) **the merits of 1/9 of them will remain unchanged;**
2) **the merits of 4/9 of them will be multiplied by 1/2;**
3) **the merits of 4/9 of them will be multiplied by 1/4.**

Therefore, the total merit of $\mathbf{P_1*\sum Q_{1k}}$ will be multiplied by $\mathbf{4/9=(2/3)^2}$.

In more general terms, it is easy to prove that if **q** compatibilities necessary to completeness are missing in $\mathbf{Q_{1k}}$ the total merit of the considered subsystem will be reduced to the extent of $\mathbf{(2/3)^q}$.

In general terms, a subsystem characterized by **m** fixed indexes in $\mathbf{P_i}$ may contain $\mathbf{3^{n-m}}$ prime implicants (or less). Therefore, if this subsystem is complete, its total merit may reach the value

$\mathbf{M = 3^{n-m}\cdot NMT1(n)}$

If **q** compatibility are missing, the merit becomes $\mathbf{(2/3)^q\cdot M}$

Very few not complete $\mathbf{Q_{ij}}$'s would reduce the merit of the considered subsystem to values not compatible with the hypothesis that the merit of the subsystem increases as $\mathbf{3^n\cdot NMT1(n)}$. Neither **q** nor **m** can be increasing functions of **n**.

10. THE SYNTHESIS OF A COMPLETE SUBSYSTEM

Assume that $\mathbf{PCA_{MAX}}$ is a complete subsystem of **CF(n)** covering a number of minterms of **ECF(n)** of the order of $\mathbf{3^n \cdot NMT1(n)}$. In this case

$$F_{2\text{-}k} = Q_1+Q_2+\ldots+Q_z$$

where **z** increase with **n** as $\mathbf{3^n}$ and each $\mathbf{Q_i}$ is a mark and it is complete in all the indexes which characterize it.

Now consider the decomposition

$$F_{2\text{-}k} = F_2 * F_{3\text{-}k}$$

where $\mathbf{F_2}$ and $\mathbf{F_{3\text{-}k}}$ can be written as sums of their prime implicants

$$F_2 = R_1+R_2+\ldots$$

$$F_{3\text{-}k} = S_1+S_2+\ldots$$

It is easy to prove that $\mathbf{F_2}$ and $\mathbf{F_{3\text{-}k}}$ must contain at least **(z-1)** marks, that is, in the best case the decomposition of $\mathbf{F_{2\text{-}k}}$ produces the reduction by one unit of the number of marks contained in $\mathbf{F_{2\text{-}k}}$.

Indeed, assume, for example, that

$$R_1 * S_1 = Q_1$$

$$R_2 * S_2 = Q_2$$

where $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{S_1}$ and $\mathbf{S_2}$ are all remainders. It is easy to verify that if $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{S_1}$ and $\mathbf{S_2}$ are all remainders, $\mathbf{R_1 * S_2 * P_i}$ and $\mathbf{R_2 * S_1 * P_i}$ are not implicants of **CF(n).**

Indeed, for example, if **CF(5)** is the considered core function and

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

$$Q_1=c(1,1;4,1)*c(1,1;5,1)*c(2,1;4,1)*c(2,1;5,1)*c(3,1;4,1)*c(3,1;5,1)*c(4,1;5,1)$$

$$Q_2=c(1,1;4,2)*c(1,1;5,1)*c(2,1;4,2)*c(2,1;5,1)*c(3,1;4,2)*c(3,1;5,1)*c(4,2;5,1)$$

$$R_1=c(1,1;4,1)*c(1,1;5,1)*c(2,1;4,1)*c(2,1;5,1)*c(4,1;5,1)$$

*(which is a remainder since **<3,1>** is missing)*

$$S_1=c(2,1;4,1)*c(2,1;5,1)*c(3,1;4,1)*c(3,1;5,1)*c(4,1;5,1) \qquad \textbf{(26)}$$

*(which is a remainder since **<1,1>** is missing)*

$$R_2=c(1,1;4,2)*c(1,1;5,1)*c(2,1;4,2)*c(2,1;5,1)*c(4,2;5,1)$$

*(which is a remainder since **<3,1>** is missing)*

$$S_2=c(2,1;4,2)*c(2,1;5,1)*c(3,1;4,2)*c(3,1;5,1)*c(4,2;5,1)$$

*(which is a remainder since **<1,1>** is missing)*

The product $\mathbf{P_i * R_1 * S_2}$ does not imply $\mathbf{P_i * Q_1}$ since **c(3,1;4,1)** is missing and it does not imply $\mathbf{P_1 * Q_2}$ since **c(1,1;4,2)** is missing, while the product $\mathbf{P_i * R_2 * S_1}$ does not imply $\mathbf{P_i * Q_1}$ since **c(1,1;4,1)** is missing and it does not imply $\mathbf{P_i * Q_2}$ since **c(3,1;4,2)** is missing.

Similarly, the decomposition

$\mathbf{F_{3\text{-}k}\text{-}= F_3 * F_{4\text{-}k}}$

in the best case produces the reduction by one unit of the number of marks contained in $\mathbf{F_3}$ **.**

It follows that the decomposition

$\mathbf{F_1 * F_2 * \ldots * F_k}$**.**

can generate the reduction of the number of marks by **k-1,** but it requires at least **k** gates, the **OR** gates of **Fig. 8**.

It is worth remarking that the preceding considerations hold also in the case of quasi-complete subsystems in which the absence of one or more of the few compatibilities in a $\mathbf{Q_j}$ determines the presence of the same compatibility in $\mathbf{P_i}$.

For example, in the case described by the set of equations **(26)**, if $\mathbf{P_i}$ were modified as follows:

$\mathbf{P'_i = c(1,1;2,1) * c(1,1:3,1) * c(2,1;3,1) * c(3,1;4,1) * c(3,1;4,2)}$

the double decomposition

$\mathbf{(R_1 + R_2 + G) * (S_1 + S_2 + G)}$

(where $\mathbf{R_1, R_2, S_1}$ and $\mathbf{S_2}$ are remainders) would be possible.

However, in this case, the merits of $\mathbf{P'_i * R_1 * S_1}$ and $\mathbf{P'_i * R_2 * S_2}$ (as well as the merits of many other implicants) are reduced to the half of **NMT1(n)**.

Also in this case one gate makes it possible to reduce the merit of the considered subsystem of one unit **NMT1(n)**.

The product $\mathbf{F_1 * F_2 * \ldots * F_k}$ may produce other marks in addition to those generated inherently by the product $\mathbf{F_1 * F_2 * \ldots * F_k}$. Indeed, one or more marks of **CF(n)** can be implicants of some $\mathbf{F_j}$ **.**

For example, consider function $\mathbf{F_1}$ implemented by the first of **PCAF's** represented in **Fig. 10.**

$\mathbf{F_1}$ is the output of an **OR** gate. Indeed, if it were the output of an **AND** gate, this might be merged together with the **AND** gate producing the output of the considered $\mathbf{PCA_{MAX}}$ with the reduction of the cost by one unit.

Let $\mathbf{F_{11}, F_{12}, \ldots, F_{1l}}$ be the inputs of this **OR** gate (**Fig. 10**). In its turn, node $\mathbf{F_{11}}$ contains a mark or a sum of marks as the product of functions $\mathbf{F_{111}, F_{112}, F_{113}, \ldots}$

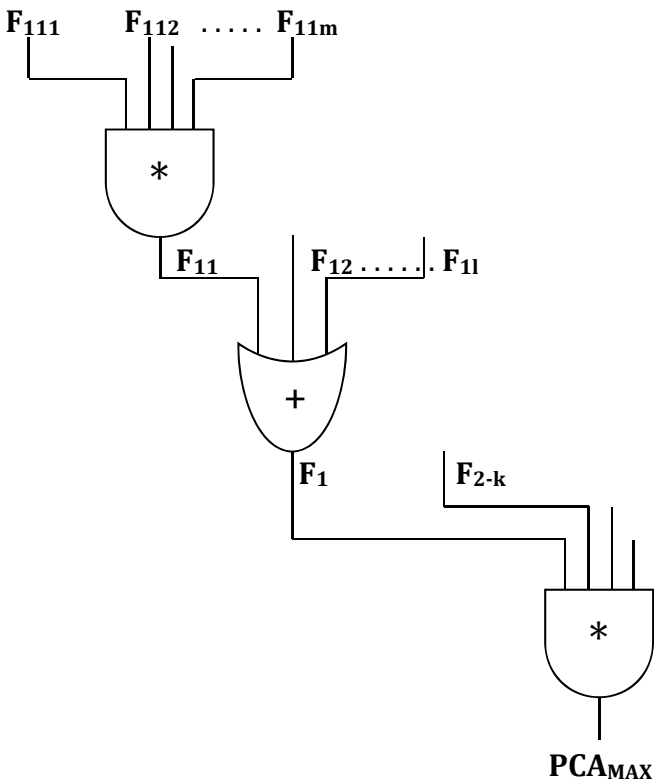


Fig. 10

The decomposition of primary composite addenda

The considerations above developed on the **AND** gates producing $\mathbf{PCA_{MAX}}$ (at the bottom of **Fig. 10**) apply also to the **AND** gate producing $\mathbf{F_{11}}$. This gate can generate **h-1** origins at the cost of **h** gates , that is, the **OR** gates producing $\mathbf{F_{111}}$, $\mathbf{F_{112}}$, $\mathbf{F_{113}}$**...**

11. CONCLUSION

In order to implement the sum of **T** prime implicants of core function, the subnetwork $\mathbf{PCA_{MAX}}$ must employ more than **T** gates. Since the number of prime implicants implemented by $\mathbf{PCA_{MAX}}$ increases with **n** as $\mathbf{3^n}$, the cost of the minimal implementation of core function **CF(n)** increases with **n** as $\mathbf{3^n}$**.** Since the synthesis of core function is an **NP-**complete problem, this result is equivalent to proving that **P** and **NP** do not coincide.

12. APPENDIX 1

12.1. PROOF OF THEOREM 5

If we consider all the prime implicants of $\mathbf{ECF(n,I_1)}$ and collect all the prime implicants which contain **(!X),** where **X** is variable not contained in $\mathbf{I_1}$, we can write:

$$\mathbf{ECF(n,I_1) = I_1 * \Pi_{k\neq 1} (!I_k) = I_1*((!X)*F+G)}$$

where **F** and **G** are two antitone functions (that is, Boolean functions which can be written as sums of products of only complemented variables) not containing variable **X**.

Let

$$\mathbf{K(n)\cdot NMT(ECF(n,I_1))}$$

be the number of the minterms of $\mathbf{ECF(n,I_1)}$ contained in $\mathbf{(!X)*F}$

and

$$\mathbf{(1-K(n))\cdot NMT(ECF(n,I_1))}$$

be the number of the minterms of $\mathbf{ECF(n,I_1)}$ contained in **G** and not contained in $\mathbf{(!X)*F}$.

Now consider an impure term

$$\mathbf{T=T(I_1) = (!X)*I_1}$$

Since the number of minterms of **T** contained in $\mathbf{(!X)*G}$ is the half of the minterms contained in **G**, the number of minterms of **T** contained in $\mathbf{ECF(n,I_1)}$ is

$$\begin{aligned}
&\mathbf{NMT((!X)*I_1*ECF(n,I_1)) = NMT(I_1*(!X)*F+I_1*(!X)*G) <}\\
&\mathbf{< (K(n) + (1/2)\cdot(1-K(n))\cdot NMT(ECF(n,I_1)) =}\\
&\mathbf{= (1/2 + K(n)/2)\cdot NMT (ECF(n,I_1))}
\end{aligned} \tag{27}$$

where sign **<** is due to the fact that **F** and **G** have common minterms.

12.2. PROOF OF THEOREM 6

Assume that the number of minterms of $\mathbf{ECF(n,I_1)}$ containing $\mathbf{(!X_1)}$ **(** or $\mathbf{(!X_2)}$ **)** is equal to $\mathbf{K(n)\cdot NMT(ECF(n,I_1))}$.

It is easy to verify that the number of minterms of $\mathbf{ECF(n,I_1)}$ containing both the variables $\mathbf{(!X_1)}$ and $\mathbf{X_2}$ is

$$\mathbf{NMT(ECF(n,I_1)*(!X_1)*X_2) <= K(n)\cdot(1-K(n))\cdot NMT(ECF(n,I_1))}$$

Similarly,

$$\mathbf{NMT(ECF(n,I_1)*(!X_2)*X_1) <= (1-K(n))\cdot K(n)\cdot NMT(ECF(n,I_1))}$$

$$\mathbf{NMT(ECF(n,I_1)*X_1*X_2) <= (1-K(n))\cdot(1-K(n))\cdot NMT(ECF(n,I_1))}$$

$$\mathbf{NMT(ECF(n,I_1)*(!X_2)*(!X_1)) <= K(n)\cdot K(n)\cdot NMT(ECF(n,I_1))}$$

It follows that the number of minterms of $\mathbf{T(I_1) = (!X_1)*(!X_2)*I_1}$ contained in $\mathbf{ECF(n,I_1)}$ is less or equal to

$$\mathbf{K(n)\cdot(1-K(n))\cdot NMT(ECF(n,I_1))/2 +}$$

$$\mathbf{(1-K(n))\cdot K(n)\cdot NMT(ECF(n,I_1))/2 +}$$

$(1-K(n))\cdot(1-K(n))\cdot NMT(ECF(n,I_1))/4 +$

$K(n)\cdot K(n)\cdot NMT(ECF(n,I_1))=$

Therefore, the number of minterms of $T(I_1) = (!X_1)*(!X_2)*I_1$ contained in $ECF(n,I_1)$ is less or equal to

$$\left(\frac{1}{2}+\frac{K(n)}{2}\right)^2\cdot NMT(ECF(n,I_1))$$

which is equivalent to eq. **(21)** for **m=2.**

The extension to any value of **m** can be easily performed by applying the same technique.

12.3. EVALUATION OF K(N)

In order to evaluate **K(n)** consider the simple case of **n=3** with

$I_1 = c(1,1;2,1)*c(1,1;3,1)*c(2,1;3,1).$

In this case

$ECF(3,I_1) = I_1*(!I_2)*(!I_3)*\ldots*(!I_{27})$

where

$!I_2 = !c(1,1;2,1)+!c(1,1;3,2)+!c(2,1;3,2)$

$!I_3 = !c(1,1;2,1)+!c(1,1;3,3)+!c(2,1;3,3)$

$!I_4 = !c(1,1;2,2)+!c(1,1;3,1)+!c(2,2:3,1)$

$!I_5 = !c(1,1;2,2)+!c(1,1;3,2)+!c(2,2;3,2)$

$!I_6 = !c(1,1;2,2)+!c(1,1;3,3)+!c(2,2;3,3)$

$!I_7 = !c(1,1;2,3)+!c(1,1;3,1)+!c(2,3;3,1)$

$!I_8 = !c(1,1;2,3)+!c(1,1;3,2)+!c(2,3;3,2)$

$!I_9 = !c(1,1;2,3)+!c(1,1;3,3)+!c(2,3;3,3)$

$!I_{10} = !c(1,2;2,1)+!c(1,2;3,1)+!c(2,1;3,1)$

.

.

.

$!I_{27} = !c(1,3;2,3)+!c(1,3;3,3)+!c(2,3;3,3)$

Consider the two following functions:

$H = !c(1,1;2,2) +$

$+!c(2,2;3,1)*!c(1,1:3,2)*!c(1,1;3,3)+$

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

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

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

$F = I_1*(!I_2)*(!I_3)*(!I_7)*(!I_8)*\ldots(!I_{27})$

from which the following equation derives:

$ECF(3,I_1) = F*!c(1,1;2,2)+F*H$

where **F** and **H** do not contain variable **c(1,1;2,2).**

Function **$ECF(3,I_1)$** is the sum of the three following functions:

$F_1 = !c(1,1;2,2)*F*(!H)$

$F_2 = !c(1,1;2,2)*F*H$

$F_3 = c(1,1;2,2)*F*H$

These functions are disjoint in the sense that **F_i** contains none of the minterms contained in **F_j** with **j <> i** . Therefore,

$NMT(ECF(3,I_1)) = NMT(F_1) + NMT(F_2) + NMT(F_3)$

Since function **F** is antitonic in all the variables contained in **H**, for any minterm of **F_1** there is a minterm of **F_2** . Therefore,

$NMT(F_1) <= NMT(F_2)$

Besides,

$NMT(F_2) = NMT(F_3)$

Therefore,

$K(3) = (NMT(F_1) + NMT(F_2)) / NMT(ECF(3,I_1)) <= (2/3)$

In more general terms

$K(n) <= (2/3)$

Consider a prime implicant **P** of **F** and assume that it contains **q** minterms of **$ECF(3,I_1)$.**

The product **F∗(!c(1,1;2,2))** produces **q/2** minterms of **$ECF(3,I_1)$**, since the minterms of **F** containing **c(1,1; 2,2)** are not minterms of **F∗!c(1,1;2,2)**.

The same prime implicant **P** of **F** produces a variable number of minterms after the multiplication by each of the five addenda of **H**. For example , the multiplication

P∗!c(2,2;3,1)∗!c(1,1;3,2)∗!c(1,1;3,3)

produces **q/8** minterms of **$ECF(3,I_1)$** if **P** contains none of the three variables

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

but it produces **q** minterms of **$ECF(3,I_1)$** if it contains all the three considered variables. Besides, the minterms generated by the product **P∗!(c(2,2;3,1))∗!(c(1,1;3,2))∗!(c(1,1;3,3))** must be added to those generated by the other addenda of **H.**

It is apparent that **K(3)** is larger than **0,5.**

Besides, it is a function which decreases when **n** increases.

Indeed, when **n** increases, the number of addenda of **H** increases.

13. APPENDIX 2

In order to understand what will be proved in this Appendix assume, for the sake of simplicity, that there is a single **PCA** in the implementation of **CF(4)** and therefore $\mathbf{PCA_{MAX}}$ **= CF(4).**

Assume also that the **PoC's** $\mathbf{P_i}$ take the following values:

$\mathbf{P_1=c(1,1;2,1)*c(1,1;3,1)*c(2,1;3,1)}$

$\mathbf{P_2=c(1,1;2,1)*c(1,1;3,2)*c(2,1;3,2)}$

$\mathbf{P_3=c(1,1;2,1)*c(1,1;3,3)*c(2,1;3,3)}$

$\mathbf{P_4=c(1,1;2,2)*c(1,1;3,1)*c(2,2;3,1)}$

.

.

.

$\mathbf{P_{27}=c(1,3;2,3)*c(1,3;3,3)*c(2,3;3,3)}$

Let the origins associated to $\mathbf{P_1}$ be the following ones:

$\mathbf{Q_{11} = c(1,1;4,1)*c(2,1;4,1)*c(3,1;4,1)}$

$\mathbf{Q_{12} = c(1,1;4,2)*c(2,1;4,2)*c(3,1;4,2)}$

$\mathbf{Q_{13} = c(1,1;4,3)*c(2,1;4,3)*c(3,1;4,3)}$

Now consider the $\mathbf{Q_{2j}}$'s associated to $\mathbf{P_2}$:

$\mathbf{Q_{21} = c(1,1;4,1)*c(2,1;4,1)*c(3,2;4,1)}$

$\mathbf{Q_{22} = c(1,1;4,2)*c(2,1;4,2)*c(3,2;4,2)}$

$\mathbf{Q_{23} = c(1,1;4,3)*c(2,1;4,3)*c(3,2;4,3)}$ **;**

the $\mathbf{Q_{3k}}$'s associated to $\mathbf{P_3}$

$\mathbf{Q_{31} = c(1,1;4,1)*c(2,1;4,1)*c(3,3;4,1)}$

$\mathbf{Q_{32} = c(1,1;4,2)*c(2,1;4,2)*c(3,3;4,2)}$

$\mathbf{Q_{33} = c(1,1;4,3)*c(2,1;4,3)*c(3,3;4,3)}$ **;**

and so on as concerns the others $\mathbf{Q_{jk}}$'s.

It is easy to verify that any product $\mathbf{P_i*Q_{ik}}$ is a prime implicant of **CF(4)** and all the prime implicants of **CF(4)** are implemented by products $\mathbf{P_i*Q_{ik}}$'s. However, no product $\mathbf{P_i*Q_{jk}}$ with $\mathbf{i \neq j}$ is an implicant of **CF(4)** and, therefore, the considered $\mathbf{\sum P_i*\sum Q_{jk}}$ is not a correct **PCA.**

For example, the product

$\mathbf{P_1*Q_{21} = c(1,1;2,1)*c(1,1;3,1)*c(2,1;3,1)*c(1,1;4,1)*c(2,1;4,1)*c(3,2;4,1)}$

is not an implicant of **CF(4)**.

Four solutions can be adopted to solve this problem.

1. To multiply $\mathbf{Q_{21}}$ by a compatibility or a product of compatibilities.
   For example:

   $\mathbf{Q'_{21}=Q_{21}*c(3,1;4,1)}$

   This choice implies the reduction of the merit of $\mathbf{P_2*Q_{21}}$ to the extent of one half.

2. To multiply $\mathbf{P_1}$ by a compatibility or a product of compatibilities.
   For example:

   $\mathbf{P'_1=P_1*c(3,1;4,1)}$

   This choice implies the reduction of the merits of $\mathbf{P_1*Q_{12}}$ and $\mathbf{P_1*Q_{13}}$ to the extent of one half.

3. To multiply $\mathbf{Q_{11}}$ by a complemented compatibility or a product of complemented compatibilities.
   For example:

   $\mathbf{Q'_{21}=Q_{21}*!c(1,1;3,1)}$

   This choice implies the reduction of the merit of $\mathbf{P_2*Q_{21}}$ to the extent of one half.

4. To multiply $\mathbf{P_1}$ by a complemented compatibility or a product of complemented compatibilities.
   For example:

   $\mathbf{P'_1=P_1*!c(3,2;4,1)}$

   This choice implies the reduction of the merits of $\mathbf{P_1*Q_{11}}$, $\mathbf{P_1*Q_{12}}$ and $\mathbf{P_1*Q_{13}}$ to the extent of one half.

Similar considerations can be applied to all the products

$\mathbf{P_i*Q_{jk}}$ with $\mathbf{i<>j}$. These considerations prove a quickly decreasing value of the whole merit of the considered **PCA**.

In order to discuss this problem in more general terms, consider the case of $\mathbf{P_i}$, $\mathbf{P_j}$ and $\mathbf{Q_{ik}}$ under the hypothesis that the numbers of variables involved in $\mathbf{P_i}$ and $\mathbf{P_j}$ are $\mathbf{m_i}$ and $\mathbf{m_j}$ respectively (with $\mathbf{m_i<=m_j}$) and that $\mathbf{n-m_j}$ is an increasing function of $\mathbf{n}$.

For the sake of simplicity, without any loss of generality, consider the following example.

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

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

$\mathbf{Q_{ik}=c(1,1;4,1)*c(1,1;5,1)*c(1,1;6,1)*c(2,1;4,1\}*c(2,1;5,1)*c(2,1;6,1)*c(3,1;4,1)*}$

$\mathbf{*c(3,1;5,1)*c(3,1;6,1)*c(4,1;5,1)*c(4,1;6,1)*c(5,1;6,1)}$

In order to transform $\mathbf{P_j*Q_{ik}}$ into an implicant of core function **CF(6)** one can adopt the above described **rule 1** consisting in adding a suitable product of compatibilities to $\mathbf{Q_{ik}}$ as follows.

$\mathbf{Q'_{ik}=Q_{ik}*c(3,2;4,1)*c(3,2;5,1)*c(3,2;6,1)}$

Such an operation implies a reduction of the merit of $\mathbf{P_i*Q_{ik}}$ to the extent of $\mathbf{1/2^{(n-m)} = 1/8}$ where $\mathbf{m= m_i=m_j=3.}$

In this example only one of variables of $\mathbf{P_i}$ (variable **3,1**) is different from a variable of $\mathbf{P_j}$ (variable **3,2**). It is easy to verify that, if the number of variables different in $\mathbf{P_i}$ and $\mathbf{P_j}$ increases, the merits of $\mathbf{P_i*Q_{ik}}$ and $\mathbf{P_j*Q_{jk}}$ more quickly decrease.

It follows that by applying previous **rule 1** origin $\mathbf{Q'_{ik}}$ gives no valuable contribution and can be ignored.

Now consider again the above stated problem of $\mathbf{P_i}$, $\mathbf{P_j}$ and $\mathbf{Q_{ik}}$ and assume that it is solved by applying **rule 2**.

In this case

$$\mathbf{P'_j = P_j*c(3,2;4,1)*c(3,2;5,1)*c(3,2;6,1)}$$

This choice implies the reduction of the merits of all the $\mathbf{Q_{jk}}$'s, with the exception of $\mathbf{Q_{j1}}$, by different orders of magnitude. It is easy to prove the following general relation:

$$\mathbf{merit(P'_j*\sum_k Q_{jk}) = merit(P_j*\sum_k Q_{jk})*(2/3)^p}$$

where $\mathbf{p=n-m_1}$. (The proof is easy but long and it is omitted here for the sake of brevity).

It follows that the merit of $\mathbf{P_i*\sum_k Q_{ik}}$ can increase as $\mathbf{3^p{\cdot}NMT1(n)}$, but the merits of all the other $\mathbf{P_j*\sum_k Q_{jk}}$ **(with j<>i)** can increase only as $\mathbf{2^p{\cdot}NMT1(n)}$. If the merit of $\mathbf{PCA_{MAX}}$ must increase as $\mathbf{3^n{\cdot}NMT(n)}$, $\mathbf{P_i*\sum_k Q_{ik}}$ alone must increase as $\mathbf{3^n{\cdot}NMT1(n)}$. Indeed, the contributions of the other subsystems $\mathbf{P_j*\sum_k Q_{jk}}$ **(with j<>i)** are not sufficient.

The above stated properties make reference to subsystems $\mathbf{<P_i*\sum_k Q_{ik}>}$ characterized by the fact that the number $\mathbf{m_i}$ of variables involved in $\mathbf{P_i}$ does not increases as **n.** Indeed, if $\mathbf{n-m_i}$ is a constant, the merit of the considered subsystem increases as **K·NMT1(n),** where **K** is a constant and it appears to be very small with respect to the objective $\mathbf{3^n{\cdot}NMT1(n)}$. Therefore, it can be ignored unless the number of such subsystems is of the order of $\mathbf{3^n}$.

However, also if the number of such subsystems increases as $\mathbf{3^n}$, it is easy to prove, by applying the above stated properties, that the total merit of their sum is negligible.

Rule3 and **rule 4** are more effective from the point of view of the levels of merit which can be reached. However, the following property can be proved.

The number of subsystems $\mathbf{<P_i,\sum_{ik}Q_{ik}>}$ of $\mathbf{PCA_{MAX}}$ cannot increase according an exponential law.

A formal proof of the above analyzed property can be developed as follows.

Let $\mathbf{N_S}$ be the number of subsystems generated by selecting a number of $\mathbf{P_i}$'s and associated $\mathbf{Q_{ij}}$'s.

Let $\mathbf{N_C}$ be the number of correction **PoC**'s, that is, sequences of variables of **CF(n)** containing at least one complemented variable necessary in order that $\mathbf{P_i*Q_{jk}}$ is equal to **0** if $\mathbf{i \neq j}$. It is apparent that any subsystem, with a single exception, must be characterized by at least one correction **PoC** and that a subsystem must be characterized by at least one correction **PoC** different from the correction **PoC**'s of the other subsystems. Therefore,

$$\mathbf{N_C >= N_S{\cdot}(N_S - 1)/2 - 1}$$

It is also apparent that the length **L** of the longest correction **PoC** must be such that

$$\mathbf{2^L - 1 >= N_C}$$

Now assume that the number of subsystems increases according an exponential law of the type:

$$\mathbf{N_S = Y \cdot h^n}$$

From this assumption the following equations derive:

$$\mathbf{N_C = Y \cdot h^n \cdot (Y \cdot h^n - 1) /2 \approx Y^2 \cdot h^{2n}}$$

$$\mathbf{L \approx \log_2 N_C \approx \log_2 (Y^2 \cdot h^{2n}) \approx 2 \cdot \log_2 Y + 2 \cdot n \cdot \log_2 h}$$

Because of the presence of **L** complemented variables in the correction **PoC** , the number of minterms covered by a subsystem is reduced of a factor equal to about $\mathbf{1/(2^L)}$. Since **L** is less than the half of the number $\mathbf{3 \cdot n \cdot (n-1)/2}$ of variables, it is easy to prove that the number of minterms covered by the sum of all the subsystems is less than

$$\mathbf{1/2^{(L/2)} \cdot NMT(ECF(n)) \approx 1/ (Y \cdot h^n) \cdot NMT(ECF(n))}$$

Therefore, the number of minterms covered by the sum of all the subsystems increases as

$$\mathbf{(3^n/h^n) \cdot NMT(ECF(I_1,n)}$$

against the hypothesis that $\mathbf{NMT(PCA_{MAX})}$ increases as

$$\mathbf{3^n \cdot NMT(ECF(I_1,n)}$$

14. TABLE OF CONTENTS

15. REFERENCES

1. H.Scholz: Problem #1: Ein ungelostes Problem in der symbolisches Logik, J.Symbolic Logic, 17, (1952), pp.160.
2. G.Asser: Das Reprasentantenproblem im Pradikatenkalkul der ersten Stufe mit Identtat, Zietschr.f.Mathematisches Logik u. Grundlagen der Math. 1, (1955), pp.252-263.
3. 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.
4. R.M.Karp: Reducibility Among Combinatorial Problems, Complexity of Computer Computations, pp. 85-103, Plenum Press, (1972).
5. L.Levin: Universal Search Problems (in Russian), Problemy Peredachi Informatsii (= Problems of Information Transmission), 9(3), pp.265-266, (1973). A partial English translation in: B.A.Trakhtenbrot: A Survey of Russian Approaches to Perebor (Brute-force Search) Algorithms. Annals of the History of Computing 6(4):384-400, 1984.
6. T.Baker, J.Gill, and R.Solovay: Relativizations of the P =? NP question, SIAM J. of Computing, Vol.4, (1975), pp.431-442.
7. M.R.Garey, and D.S.Johnson: Computers and intractability. (1979). New York, W.H.Freeman and Co..
8. Razborov, Lower bounds on the monotone complexity of some Boolean functions. *Soviet Mathematics-Doklady 31*, (1985) 485--493.
9. Wegener, The Complexity of Boolean Functions. Wiley-Teubner, 1987
10. R.Boppana, M.Sipser: Complexity of finite functions. in: Handbook of Theoretical Computer Science, ed.J. van Leeuwen, (1990), pp.758-804.
11. S.A. Cook: Computational complexity of higher type functions. In Ichiro Sarake editor – Proceeding of the International Congress of Mathematicians – Kyoto, Japan, pages 55-60, Springer-Verlag 1991.
12. M.Sipser: The history and status of the P versus NP question, in: Proceedings of the 24th Annual ACM Symposium on Theory of Computing, (1992), pp.603-618. ACM Press
13. K.Godel: a letter to J.von Neumann, in Sipser's article listed above, or in: "Arithmetic, Proof Theory and Computational Complexity", eds. P. Clote and J. Krajicek, Oxford Press, (1993).
14. J.Krajicek: Bounded arithmetic, propositional logic, and complexity theory, Encyclopedia of Mathematics and Its Applications, Vol. 60, Cambridge University Press, (1995).
15. M.Sipser: Introduction to the theory of computation, PWS Publ.Comp.. 1997.
16. Alexander A. Razborov , Steven Rudich, Natural proofs, Journal of Computer and System Sciences, v.55 n.1, p.24-35, Aug. 1997
17. Peter W. Shor, Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM Journal on Computing, v.26 n.5, p.1484-1509, Oct. 1997
18. S.Smale: Mathematical problems for the next century, Mathematical Intelligencer, 20(2), (1998), pp.7-15.
19. P.Pudlak: The lengths of proofs, in: Handbook of Proof Theory, ed. S.R.Buss, North-Holland, (1998).
20. Ketan D. Mulmuley , Milind Sohoni, Geometric Complexity Theory I: An Approach to the P vs. NP and Related Problems, SIAM Journal on Computing, v.31 n.2, p.496-526, 2001
21. Fortnow, L. and Homer, S. A short history of computational complexity. *Bulletin of the European Association for Theoretical Computer Science 80*, (June 2003).
22. Agrawal, M., Kayal, N., and Saxena, N. PRIMEs. In *Annals of Mathematics 160*, 2 (2004)

781--793.
23. A.R. Meo: On the minimization of core function Acc. Sc. Torino – Memorie Sc. Fis. 29, (2005), 155-178,7 ff.
24. Dieter van Melkebeek, A survey of lower bounds for satisfiability and related problems, Foundations and Trends® in Theoretical Computer Science, v.2 n.3, p.197-303, January 2006
25. Subhash Khot , Guy Kindler , Elchanan Mossel , Ryan O'Donnell, Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs?, SIAM Journal on Computing, v.37 n.1, p.319-357, April 2007
26. A.R. Meo: "Some theorems concerning the core function" in "Concurrency, Graphs and Models", Springer – Verlag Berlin Heidelberg 2008.
27. Conitzer, V. and Sandholm, T. New complexity results about Nash equilibria. *Games and Economic Behavior 63*, 2 (July 2008), 621--641.
28. Sanjeev Arora, Boaz Barak, Computational Complexity: A Modern Approach, Cambridge University Press, New York, NY, 2009
29. A.R.Meo, On the minimal implementation of a monotonic Boolean function, to appear.
30. A.R.Meo, A new NP-complete problem, to appear.
31. A.R.Meo, Some properties of core functions and external core functions, to appear.