An-Najah Univ. J. Res., Vol. 14, 2000

Some Propositions, Quantified Assertions and their Uses in Computer Hardwadre Design

بعض العبارات المنطقية والجمل المسورة واستخداماتها في مجال الحاسوب

Maher A. Nabulsi

Dept. of Math & Computer Science, Collage of Science. Al-Zaytoonah Jordanian University, Amman, Jordan.

Received: (3/4/2000), Accepted: (13/3/2001)

Abstract

Mathematics and computer science are closely linked with each other, and in many respects, the functions of computer related elements might be interpreted using mathematical approaches. In this paper the strong correlation that exist between computer software and hardware elements (logic devices, digital circuits and logical processing) and discrete mathematics (propositions and quantified assertions) will be demonstrated. This will pave the way for a better understanding of the operations of basic computer circuits and lay the foundation for developing new computer related algorithms.

Keywords: Discrete, Propositions, Quantifiers, Logical Operators, and Common bus.

ملخص

يهدف هذا البحث لعرض بعض تطبيقات الرياضيات المنفصلة (العبارات والجمل المسورة) في مجـــال الحاسوب (تصميم بعض الدوائر المتكاملة وأحياناً في منطق عمل المعالج المايكروي)

ويتضمن البحث البنود التالية :

* جملة "اذا كان فان" $P \Rightarrow Q$
* المسورات الأحادية: $\forall x\, P(x)$، $\exists\, x\, P(x)$، $\exists!\, xP(x)$
* المسورات الثنائية: $\forall x\, \forall y\, P(x,y)$ ، $\exists\, x \exists\, y\, P(x,y)$ ، $\forall x \exists\, y\, P(x,y)$، $\exists\, y\, \forall x\, p(x,y)$
* توزيع المسورات $\forall x, \exists\, x$ على الروابط المنطقية AND، OR ، اذا كان فأن ($\Rightarrow$) .

وقد استخدم أسلوب التبسيط والتوضيح في إعداد هذا البحث ، وشرح نوعية مهمة من التطبيقات في مجال الحاسوب وعرض طريقة جديدة لاثبات بعض العلاقات في هذا المجال ، وهو موجـــه للمـــهتمين مـــن طلبـــة الجامعات والمختصين .

Introduction

Boolean algebra is the backbone of computer software and hardware systems and almost all computer algorithms may be interpreted using mathematical modeling. Such a phenomenon may be exploited for the purpose of developing new computer applications.

In this paper several links between mathematics and computers will be investigated, namely, the implication $(P \Rightarrow Q)$, previously discussed by other authors [1,4,5,6,7,10,11,12], the quantifiers $\forall xP(x)$, $\exists xP(x)$, $\exists !xP(x)$,two quantifiers applied to a predicate $\forall x \forall yP(x,y)$,$\exists x \exists yP(x,y)$, $\forall x \exists yP(x,y)$, $\exists y \forall xP(x,y)$ [1,2,3,10,11,12] and lastly , the distribution of quantifiers $\forall \mathbf{x}, \exists \mathbf{x}$ over logical operators AND ($\wedge$), OR ($\vee$), implies ($\Rightarrow$) [1,2,3].

The implication of P and Q

P and Q are propositional variables, each of which denotes a proposition, whose truth-value is either true or false but not both. "P implies Q $(P \Rightarrow Q)$ "is a conditional statement of the form "if-then " statement, which means:

"If P, then Q"; where "P is a sufficient condition for Q"; and "Q is a necessary condition for P". Such a statement has the following truth table [1,4,5,11,12]:

P	Q	$(P \Rightarrow Q) \Leftrightarrow (\neg P \vee Q)$
0	0	1
0	1	1
1	0	0
1	1	1

It is clear that the implication $(P \Rightarrow Q)$ is always true except when the premise P=1 and the conclusion Q=0, in the later case, the result is false.

Such an implication can be realized using computer software algorithm as shown in the following program fragment:

20 If X<=0 then go To 50
30 .....
40.....
50.....

Where, P denotes " X<=0 "; and Q denotes " go to 50".

The result of execution of (P⇒Q) by the computer processor can be analyzed as follows:

a) **If P is false and Q is false,** that is; if the premise (X>0) is false and the execution process does not go to line 50 then the result is true , as depicted by the following truth table:

P	Q	(P⇒Q)
0	0	1

i. e. It is true not to go to line 50 if the condition P is false.

b) **If P is false and Q is true,** that is;
If the premise (X>0) is false and the execution process goes to line 50 then the result is true. This is because line 50 may be executed sequentially without the premise P, as shown in the following truth table segment:

P	Q	(P⇒Q)
0	1	1

From a) and b) it can be concluded that the premise P is a sufficient condition for Q.

c) **If P is true and Q is false,** i.e. if the premise (X<=0) is true and the execution does not go to line 50 then the result is false, as depicted in the following truth table segment:

P	Q	(P⇒Q)
1	0	0

In this case, it is not true not to go to line 50 when the premise (X<=0) is true.

d) **If P is true and Q is true,** i.e. if the premise (X<=0) is true and the execution goes to line 50 then the result is true. This, too, means that, it is true to go to line 50 if the premise P is true, as illustrated in the following truth table:

P	Q	(P⇒Q)
1	1	1

From c) and d) it can be concluded that Q is a necessary condition for P.

The Quantified Assertions, $\forall xP(x)$, $\exists xP(x)$, $\exists !xP(x)$:

These quantified assertions denote:

(a) $\forall xP(x)$

For all values of x selected from the universe of discourse U, the predicate P(x) is true [2], i.e. the set of solution is equal to the universe U.

Thus, if U={1,2,3} then $\forall xP(x) \Leftrightarrow P(1) \wedge P(2) \wedge P(3)$ [3,11] For example , If P(x): $x \geq 1$ then $\forall xP(x)$ is true.

An application of the proposition $\forall xP(x)$ in logic design can be realized as a multiple input AND-gate, as shown in Fig. (1) .

(b) $\exists xP(x)$

For at least one value of x selected from the universe, the predicate P(x) is true , [2]. i.e. the set of solution is a subset of the universe U.

Thus, If U={1,2,3,} then $\exists xP(x) \Leftrightarrow P(1) \vee P(2) \vee P(3)$ [3,11]. For example, If P(x): $x \geq 2$ then $\exists xP(x)$ is true.

An application of $\exists xP(x)$ in logic design can be realized as a multiple input OR-gate , Fig(2)

(c) $\exists! xP(x)$

There is one and only one value of X, which yields P (x) as true [1]. Thus, If U = {1,2,3}, then; $\exists!\, x\, P\,(x) \Leftrightarrow [P\,(1) \wedge \neg P(2) \wedge \neg P(3)] \vee [P(2) \wedge \neg P(1) \wedge \neg P(3)] \vee [P(3) \wedge \neg P(1) \wedge \neg P(2)]$. Transferring information, generated from one source through a common bus and at the same time, inhibiting other sources connected to the bus is a good application for $\exists! xP(x)$. The status of the bus is true if only one source transmits information. This is in agreement with the results attained from $\exists! xP(x)$ which yields a true value if only one value of x makes P(x) true. A practical example of which is a data bus system constructed from three state buffers [8]. Figure 3 shows a design for multiple lines to one line (common) bus system. It should be noted that only one line must be active at any given time, while all others must be inhibited. This can be achieved using a suitable decoder to control the respective buffers.

Two quantifiers and two level logic gates

(a) $\forall x \forall y P(x,y)$

If the universe {0,1}, then;

$\forall x \forall y P(x,y) \Leftrightarrow \forall y \forall x P(x,y) \Leftrightarrow [P(0,0) \wedge P(0,1)] \wedge [P(1,0) \wedge P(1,1)]$

It is clear that this is a two level AND-AND gates.

In Boolean algebra AND-AND $\Leftrightarrow$ one level AND, can be written as follows:

$(A.B).(C.D) \Leftrightarrow A.B.C.D$

AND-AND $\Leftrightarrow$ AND, Fig (4)

(b) $\exists x \exists y P(x,y)$

If the universe {0,1} then;

$\exists x \exists y P(x,y) \Leftrightarrow \exists y \exists x P(x,y) \Leftrightarrow [P(0,0) \vee P(0,1)] \vee [P(1,0) \vee P(1,1)]$

This may be represented by a two level OR-OR gates.the Boolean function of which can be written as

OR-OR $\Leftrightarrow$ OR

i.e. $(A+B)+(C+D) \Leftrightarrow A+B+C+D$

OR-OR $\Leftrightarrow$ OR, Fig (5)

(c) $\forall x \exists y P(x,y)$

let the universe {0,1} then ;

$\forall x \exists y P(x,y) \Leftrightarrow [P(0,0) \vee P(0,1)] \wedge [P(1,0) \vee P(1,1)]$

Such a quantifier may be represented by a two level OR-AND gate.

(d) $\exists y \forall x P(x,y)$

let the universe {0,1} then ;

$\exists y \forall x P(x,y) \Leftrightarrow [P(0,0) \wedge P(1,0)] \vee [P(0,1) \wedge P(1,1)]$

It is clear that this is a two level AND-OR gate.

From (c) and (d) it can be concluded that $\forall x \exists y P(x,y) \neg\Leftrightarrow \exists y \forall x P(x,y)$. This is because OR-AND $\neg\Leftrightarrow$ AND-OR

i.e. $(A+B)(C+D) \neg\Leftrightarrow AB+CD$, [9] , Fig (6)

The analogy adopted in proving the non-equivalence of (c) and (d) is different from that found in other related publications [1,2,3,12], all of which use mathematical approaches to prove the non-equivalence, such as ;

If the universe is the set of integers I, Then

$\forall x \exists y [x+y=0] \neg\Leftrightarrow \exists y \forall x [x+y=0]$,

where the left hand side is true and the right hand side is false.

The distribution of quantifiers over the logical operators

Previous work [1,2,3] have shown that there is no unified method to prove the truth of relationships between assertions involving quantifiers and logical operators.

In the next section, the six possible cases of the distribution will be presented. One of these cases will be discussed in details (the remaining cases will be dealt with in the same maner).

(a) $\forall x[P(x)\wedge Q(x)] \Leftrightarrow [\forall xP(x)\wedge\forall xQ(x)]$

AND-AND $\Leftrightarrow$ AND-AND

$\forall$ distributes over $\wedge$.

(b) $\exists \mathbf{x[P(x)\vee Q(x)]} \Leftrightarrow [\exists xP(x)\vee\exists xQ(x)]$

OR-OR $\Leftrightarrow$ OR-OR

$\exists$ distributes over $\vee$.

(c) $\forall \mathbf{x[P(x)\vee Q(x)]} \neg\Leftrightarrow [\forall xP(x)\vee\forall xQ(x)]$

$\forall$ Does not distribute over $\vee$.

But $[\forall xP(x)\vee\forall xQ(x)]\Rightarrow\forall x[P(x)\vee Q(x)]$ is valid (always true) and the converse is not valid.

(d) $\exists \mathbf{x[P(x)\wedge Q(x)]} \neg\Leftrightarrow [\exists xP(x)\wedge\exists xQ(x)]$

$\exists$ Does not distribute over $\wedge$. However ;

$\exists x[P(x)\wedge Q(x)]\Rightarrow[\exists xP(x)\wedge\exists xQ(x)]$ is valid and the converse is not valid.

(e) $\exists \mathbf{x[P(x)\Rightarrow Q(x)]} \neg\Leftrightarrow[\exists xP(x)\Rightarrow \exists xQ(x)]$

$\exists$ Does not distribute over $\Rightarrow$

However, $[\exists xP(x) \Rightarrow \exists xQ(x)] \Rightarrow \exists x[P(x)\Rightarrow Q(x)]$ is valid and the converse is not valid.

(f) $\forall \mathbf{x[P(x)\Rightarrow Q(x)]} \neg\Leftrightarrow [\forall xP(x)\Rightarrow\forall xQ(x)]$

$\forall$ Does not distribute over $\Rightarrow$.

However, $\forall x\ [P(x) \Rightarrow Q(x)] \Rightarrow [\forall x P(x) \Rightarrow \forall x Q(x)]$ is valid and the converse is not valid.

In the next section, a new method for analysing the distributions over logical operators will be presented. Such a method can be applied to all possible cases.

For each of the above cases (a, b, ...f), a universe is taken , such as $\{0,1\}$, which is a subset of the natural numbers (set N) , the equivalence of each side is found by replacing the variables by specific values drawn from the universe, then a truth table is constructed in order to verify the relationship.

Referring to case (f), namely;

$$\forall x[P(x) \Rightarrow Q(x)] \neg\Leftrightarrow [\forall x P(x) \Rightarrow \forall x Q(x)]$$

Then the left hand side:

$$\forall x[P(x) \Rightarrow Q(x)] \Leftrightarrow \forall x[\neg P(x) \vee Q(x)] \Leftrightarrow [\neg P(0) \vee Q(0)] \wedge [\neg P(1) \vee Q(1)]$$

And the right side;

$$[\forall x P(x) \Rightarrow \forall x Q(x)] \Leftrightarrow \neg \forall x P(x) \vee \forall x Q(x) \Leftrightarrow \exists x \neg P(x) \vee \forall x Q(x)$$
$$\Leftrightarrow [\neg P(0) \vee \neg P(1)] \vee [Q(0) \wedge Q(1)]$$

Elements P (0), P (1), Q (0), and Q (1) are considered as four variables in the proposition. These variables are used to construct the truth table shown in table 1. Where;

$A \wedge B \Leftrightarrow [\neg P(0) \vee Q(0)] \wedge [\neg P(1) \vee Q(1)]$ is the left hand side, and

$C \vee D \Leftrightarrow [\neg P(0) \vee \neg P(1)] \vee [Q(0) \wedge Q(1)]$ is the right hand side.

It is clear, from the truth table, that the two sides are not equal, but the left side implies the right side is always true. i.e. $(A \wedge B) \Rightarrow (C \vee D)$, and the converse is not valid. Such an approach may be applied to all other cases.

It should be pointed out at this stage that, other methods for investigating the above relationships, do exist, however no known method was found to be is applicable to all possible six cases (a to f)

Table (1):

P(0)	P(1)	Q(0)	Q(1)	$\neg$P(0) $\vee$Q(0)	$\neg$P(1) $\vee$Q(1)	A$\wedge$B	$\neg$P(0)$\vee$ $\neg$P(1)	Q(0)$\wedge$ Q(1)	C$\vee$D
0	0	0	0	1	1	1	1	0	1
0	0	0	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1
0	1	0	0	1	0	0	1	0	1
0	1	0	1	1	1	1	1	0	1
0	1	1	0	1	0	0	1	0	1
0	1	1	1	1	1	1	1	1	1
1	0	0	0	0	1	0	1	0	1
1	0	0	1	0	1	0	1	0	1
1	0	1	0	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	0
1	1	1	0	1	0	0	0	0	0
1	1	1	1	1	1	1	0	1	1
						⇑			⇑

Conclusions:

- The implication (P$\Rightarrow$Q) is an If-Then statement, and its truth table is clearly implemented when the processor executes the if-then statement.
- The quantifiers, $\forall$xP(x) and $\exists$xP(x) , are implemented as multiple input AND and multiple input OR logic gates respectively.
- Transferring information from one source through a common bus and inhibiting other sources from the bus is a good application for $\exists!$xP(x).

- $\forall x \forall y P(x,y) \Leftrightarrow \forall y \forall x P(x,y) \Leftrightarrow$
  AND-AND $\Leftrightarrow$ One level multiple input AND gate.
- $\exists x \exists y P(x,y) \Leftrightarrow \exists y \exists x P(x,y) \Leftrightarrow$
  OR-OR $\Leftrightarrow$ One level multiple input OR gate.
- $\forall x \exists y P(x,y) \neg\Leftrightarrow \exists y \forall x P(x,y)$, as well as OR-AND $\neg\Leftrightarrow$ AND-OR.
- A common method is presented for proving the distribution of quantifiers over the logical operators by taking a universe such as $\{0,1\}$, finding the equivalence of each side by replacing the variables by specific values form the universe. Then constructing a truth table to show the truth of the relationship.

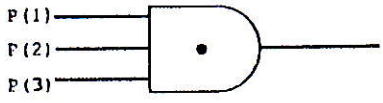

Fig.[1] : Multiple input AND-gate

$\forall x P(x) \iff P(1) \wedge P(2) \wedge \ldots$

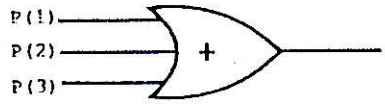

Fig.[2] : Multiple input OR-gate

$\exists x P(x) \iff P(1) \vee P(2) \vee \ldots$

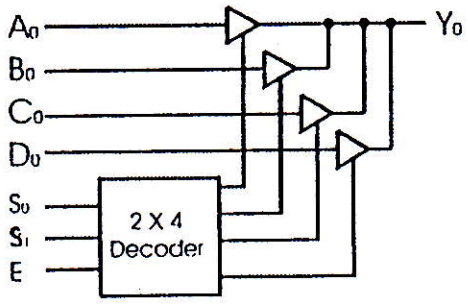

Fig.[3] : One line in the common bus constructed with three-state buffer.

$\exists! x P(x)$

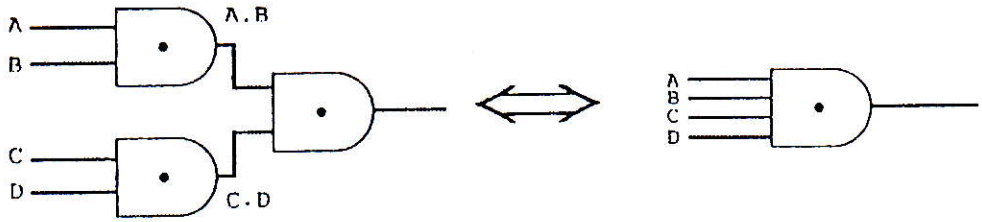

Fig.[4] : Two level AND-AND is equivalent to one level multiple input AND gate.

$\forall x \forall y\ P(x,y) \iff \forall y \forall x\ P(x,y)$

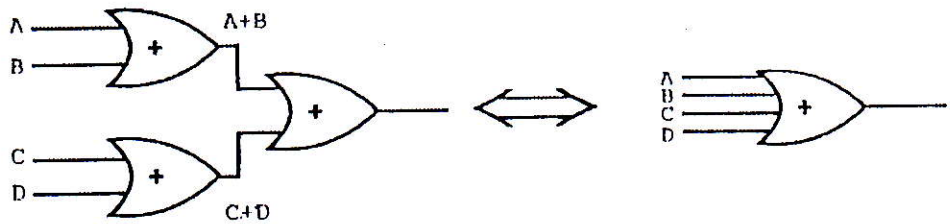

Fig.[5] : Two level OR-OR is equivalent to one level multiple input OR gate.

$\exists x \exists y\ P(x,y) \iff \exists y \exists x\ P(x,y)$

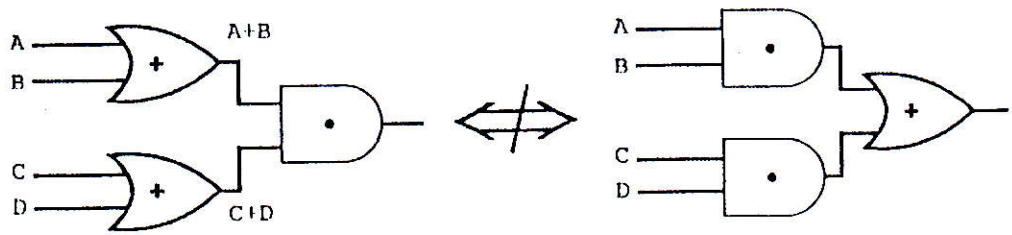

Fig.[6] : OR-AND $\not\Leftrightarrow$ AND-OR, as well as.

$\forall x \exists y\ P(x,y) \not\Leftrightarrow \exists y \forall x\ P(x,y)$

References

1. Donald F. Stanat, David F. McAllister. "Discrete Mathematics in Computer Science". Prentice-Hall (1977). 12-38.
2. Albert D. Polimeni, H. Joseph Straight. "Foundations of Discrete Mathematics". Brooks/ Cole Publishing Company (1985). 23-29.
3. Kenneth A. Ross, Charles R.B. Wright. "Discrete Mathematics". Prentice-Hall (1985). 219-235.
4. Angela B. Shiflet. "Discrete Mathematics for Computer Science". West Publishing Company (1987). 53-70.
5. Fleury, A. "The Discrete Structures Course: Making its Purpose Visible". Presented at the ACM Annual Conference, 1985.
6. Robert J. McEliece, Robert B. ASH, Carol ASH. "Introduction to Discrete Mathematics". Random House (1989). 177-190.
7. Bernard Kolman, Robert C.Busby. "Discrete Mathematics Structures for Computer Science". Prentice-Hall (1987). 405-409.
8. M. Morris Mano, "Computer System Architecture". Prentice-Hall (1993). 97-101.
9. M. Morris Mano, "Digital Design". Prentice-Hall (1991). 94-98.
10. William E. Fenton, Ed Dubinsky. "Introduction to Discrete Mathematics with ISETL". Springer (1996). 34-38,76-86.
11. H. Jerome Keisler, Joel Robbin. "Mathematical Logic and Computability". McGraw-Hill (1996). 3-7, 61-64.
12. Neville Dean. "The Essence of Discrete Mathematics". Prentice -Hall (1997). 36-43, 69-75.