# Correspondence

*To the Editor*
*The Computer Journal*

Sir

There have now, to my knowledge, appeared two letters in your journal regarding the spelling and derivation of the word ALGORITHM (Marriot, Baldota and Kshirasagar). Both of these letters have shown a general lack of knowledge about how our language came to have a word of such central importance to our discipline.

It was during the early part of the 8th century AD that Baghdad rose to become the great intellectual centre of the Arabic world. The Caliph al-Mansur encouraged contact with the Indian subcontinent and was instrumental in having both Greek and Indian works on mathematics and astronomy translated into Arabic. This centre of learning was encouraged by al-Mansur's successors and, in the early part of the 9th century, the mathematician Mohammed ibn Musa al-Khowarizmi (Mohammed son of Moses the man from Khwarezm—Khwarezm is the area around the modern city of Khiva), who was also known as Abu Ja'far or Abu Abdallah after the Arabic practice of naming the father by the name of his son, wrote a book called 'Hisab aljabr w'almuqabala' which was translated several times during the 12th century into Latin. These translations often used a corruption of the Arabic title, or of the author's name, as a title for the book; for example the Latin corruption of the title 'Algebra et Almucabala' ultimately gave a name to the field of algebra, the most common title was 'Liber Algorismi' (the book of al-Khowarizmi), and most of the translations start, like the one made by Robert of Chester in the 12th century, with 'Dixit Algoritmi: laudes deo rectori . . .' (Algoritmi has spoken: praise be to God, *etc.*) This book was one of the major sources through which Europeans came to know the Hindu-Arabic methods of arithmetic.

The rise of learning in Europe was combined with a search for the 'lost knowledge' contained in Greek and Roman sources. The Greeks thought of 'arithmetic' as being the study of numbers (like number theory) while the practice of doing arithmetic calculations was referred to as 'logistic'. The early European scholars kept this difference and referred to number theory as 'arithmetic' while the Latin authors used various corruptions of the title 'Liber Algorismi' (algoritmi, algorismi, algorismo, algorismus, and algorithmus) to indicate the practice of doing computations. These terms were later further corrupted when they were translated into French (changing the 'al' to 'au') as 'augrisme', 'augrime', and 'argorisme'; into Spanish (by dropping the 'al') as 'guarasma' and 'guarismo'; and into English as 'algorism' and (via the French) as 'augrim'. The word 'algorithm' is simply another corruption of 'algorism'.

By the middle of the 18th century the meanings of the two words 'algorism' and 'algorithm' had changed slightly so that 'algorithm' stood for the set of rules by which calculations were performed, while 'algorism' was the term used for the actual performance of the calculation. Thus, the person wishing to multiply two numbers together would first consult a book of algorithm to find out how it was done, then actually obtain an answer by algorism.

Although the makers of the *Oxford English Dictionary* failed to find the proper quotation for 'algorithm', D. E. Smith gives an instance which shows that the word has been a proper part of the English language (in its modern sense) for over 200 years. Its previous forms in Old English (augrym, augrim) were in use by Chaucer in the middle of the 14th century.

I hope that this note will help clear up some of the confusion surrounding the history of the word algorithm. I should also add, for the sake of the pure academic, that the careful reader will find several variations on the actual Arabic title of al-Khowarizmi's book. This is because there is no known complete original Arabic copy of the work, and the title has to be deduced from the many translations which are still with us.

Yours faithfully,
M. R. Williams

Department of Computer Science
University of Calgary
Calgary
Alberta
Canada T2N 1N4
11 April 1977

**References**

Baldota, S. N., and Kshirsagar, V. K. (1977). *The Computer Journal*, Vol. 20, No. 1, pp. 95-96.

Marriot, H. A. (1974). *The Computer Journal*, Vol. 17, No. 2, p. 187.

Smith, D. E. (1925). *History of Mathematics*, Vol. 11, p. 9, Ginn and Co., Boston.

*To the Editor*
*The Computer Journal*

Sir

**New graphic symbols for the quantifier logic**

This note is a proposal to introduce new graphic symbols for the existential, and the universal quantifiers, and for the least number operator of symbolic logic as a step towards the use of shapes that are visually more suggestive of the functions that they indicate. The proposed new symbols are shown in **Fig. 1**.

Quantifier symbols: $\mathbb{V}$ replacing $\exists$, and $\mathbb{\Lambda}$ replacing $\forall$ are graphically the mirror images of one another, thus suggesting that the operations called by them are also inversely related, as indeed they are through the extended De Morgan rules for the systematic replacement of the universal by the existential quantifiers, or vice versa, with the simultaneous negation of argument, and of the whole bound expression. The two symbols are visually related, by being similar in outline to the standard denotations for the basic logical functions from which they arise by generalisation, as in the original interpretation of C. S. Peirce (1839-1914) or E. Schroeder (1841-1902). Thus the existential quantifier can be regarded as an infinite disjunction ($A_1 \vee A_2 \vee A_3 \vee \ldots$), while the universal quantifier is then a continued infinite conjunction ($A_1 \wedge A_2 \wedge A_3 \ldots$). In the new graphics this 'stacking', or replication is suggested by a second slanting stroke within the 'vee', yet the symbols are still sufficiently dissimilar from both 'v' and 'w' to avoid being confused with them. Mirror reflection rather than a rotational inversion is preferable, since it results in the middle slanting stroke assuming opposite

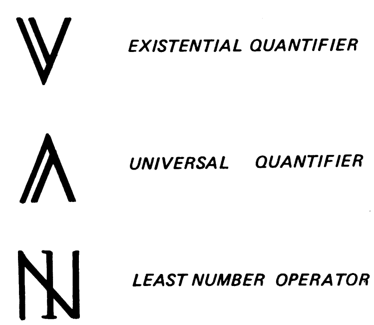


**Fig. 1 New graphic symbols for logic**

diagonal orientation (sinistral v. dextral) thus helping further in the visual discrimination.

The least number operator is defined as the smallest natural number $i$ for which a predicate depending on $i$ is true, or zero if none exists. It is the fundamental entity in the theory of recursive functions, and one of the most useful devices to produce denotations for functions over a natural domain. It is normally denoted by $\mu$, which is inconvenient in many contexts utilising Greek letters for other quantities, and introduces a lower case symbol for a functional, which is undesirable since most conventions use upper case for such entities. The symbol proposed here would be an upper case, and would be formed from the superposed letters 'l' for 'least' and 'N' for 'number', thus being mnemonic and intuitive, while at the same time still distinctly different from the rest of the alphabet, and from the proposed quantifier symbols.

All above symbols bind the variables appearing immediately at their right, and have a scope of the first parenthesised expression following in the text.

Yours faithfully,
W. A. ZAREMBA

Bechtel Inc.
San Francisco
California
20 April 1977

*To the Editor*
*The Computer Journal*

Sir

**Cost-benefit analysis: Service Bureau v. in-house processing**

Finding and comparing discounted costs in the manner suggested by Alewine and Fleck (AF) would be a proper procedure only for a corporation which pays no tax on profits.

Alewine and Fleck suggest that, having established the economic worthwhileness of the applications, the choice between service bureau or in-house processing should be made on the basis of the net present value (NPV) of the cash expended in each year of the alternatives. The purpose of finding the NPV is to take account of the diminished current value of a cash benefit which is postponed in time (or, of course, the diminished current cost of an expense which is postponed in time). NPV is usually taken as $F/(1 + i/100)^n$ where F is the future benefit (or cost), i.e. the amount which will be entered into the books of account of the corporation, $i$ is an arbitrary annual interest rate (often taken to be the rate of interest earned generally by the corporation on its capital employed in the business), and $n$ is the number of years of postponement.

Normally, the corporation as a whole will incur increased taxation on the net benefits produced by data processing projects (or effectively obtain a rebate on their losses). Since assessment, calculation and payment of tax take time, there is a lag in these tax payments or refunds. The seriousness of this consequence depends on the disparity in the timing of the returns of the alternatives being considered, the rate of tax, the period of the lag and the interest rate used for discounting.

**Tables 1** and **2** show Alewine and Fleck's example reworked for a corporation that pays 50% tax on gross profits, tax payments lagging 1 year. Although the preference of the alternatives is not changed, the present value of the service bureau advantage is only $13,062, compared with Alewine and Fleck's result of $57,140. As Alewine and Fleck point out, the accurate figure is important if this cash advantage is to be weighed against other unquantified costs or benefits. **Table 3** is an exaggerated example demonstrating that taking tax into account can reverse the preference of the alternatives.

According to a survey by the National Computing Centre, 36 out of 120 respondents to a questionnaire used discounted cash flow methods in evaluating computer activities in the UK (about 20% of the surveyed computer installations responded).

Yours faithfully,
ANDREW PARKIN

School of Mathematics, Computing and Statistics
Leicester Polytechnic
PO Box 143
Leicester LE1 9BH
9 September 1976

**References**

ALEWINE, T., and FLECK, R. A. (1976). Service Bureau or in-house data processing, *The Computer Journal*, Vol. 19, No. 3.

NATIONAL COMPUTING CENTRE (1971). Economic Evaluation of Computer Based Systems.

**Table 1 Present value of service bureau processing**

| | *1975* | *1976* | *1977* | *1978* | *1979* | *1980* | *1981* | *1982* | *Total* |
|---|---|---|---|---|---|---|---|---|---|
| Costs from Alewine and Fleck table 5 | 72,720 | 99,530 | 112,130 | 126,105 | 140,805 | 157,400 | 174,110 | 192,025 | 1,074,825 |
| Taxation at 50% | —— | (36,360) | (49,765) | (56,065) | (63,052) | (70,402) | (78,700) | (87,055) | (441,399) |
| Cash outflow | 72,720 | 63,170 | 62,365 | 70,040 | 77,753 | 86,998 | 95,410 | 104,970 | 633,426 |
| NPV ($i$ = 10%) | 72,720 | 57,427 | 51,541 | 52,622 | 53,106 | 54,019 | 53,856 | 53,866 | 449,157 |

**Table 2 Present value of in-house processing**

| | *1975* | *1976* | *1977* | *1978* | *1979* | *1980* | *1981* | *1982* | *Total* |
|---|---|---|---|---|---|---|---|---|---|
| Costs from Alewine and Fleck table 5 | 141,915 | 158,795 | 149,210 | 178,780 | 142,070 | 92,260 | 96,080 | 100,910 | 1,060,020 |
| Taxation at 50% | —— | (70,957) | (79,397) | (74,605) | (89,390) | (71,035) | (46,130) | (48,040) | (479,554) |
| Cash outflow | 141,915 | 87,838 | 69,813 | 104,175 | 52,680 | 21,225 | 49,950 | 52,870 | 580,466 |
| NPV ($i$ = 10%) | 141,915 | 79,853 | 57,697 | 78,268 | 35,981 | 13,179 | 28,195 | 27,131 | 462,219 |

**Table 3 Exaggerated example, 2 year analysis**

| | $n = 0$ | $n = 1$ | *Total* |
|---|---|---|---|
| Costs of Project A | 0 | 10 | 10 |
| NPV ($i$ = 25%) | 0 | 8 | 8 |
| Costs of Project B | 10 | 0 | 10 |
| NPV ($i$ = 25%) | 10 | 0 | 10 |

'No tax' analysis favours project A.

| | $n = 0$ | $n = 1$ | *Total* |
|---|---|---|---|
| Costs of Project A | 0 | 10 | 10 |
| Tax at 50% | 0 | 0 | 0 |
| Cash outflow | 0 | 10 | 10 |
| NPV ($i$ = 25%) | 0 | 8 | 8 |
| Costs of Project B | 10 | 0 | 10 |
| Tax at 50% | 0 | (5) | (5) |
| Cash outflow | 10 | (5) | (5) |
| NPV ($i$ = 25%) | 10 | (4) | 6 |

'With tax' analysis favours project B.