Automatic construction of diagnostic tables

W. R. Willcox and S. P. Lapage

Central Public Health Laboratory, Colindale Avenue, London NW9 5HT

Diagnostic tables are used in the identification of biological specimens. A method is presented which generates subsets of the available tests which meet the requirements of a diagnostic table. The method always finds a set with the fewest possible tests. In each of four trial applications an alternative sequential method, due to H. G. Gyllenberg, also found a set with the fewest possible tests.

(Received August 1971)

A biological specimen is identified by comparing its properties with those appropriate to the various groups of organisms to which it may belong. The groups of organisms are referred to as *taxa* (singular, *taxon*) and it is convenient to regard the properties as the *results* of *tests* applied to the specimen. The identification process comprises two elements, *test selection* and *inference* (Gorry, 1968). Test selection is the decision as to which of the available tests should be applied to a particular specimen; inference is the analysis of the results of these tests to indicate the correct taxon. Many aids to identification have been produced (Morse, 1971), mainly printed schemes but including mechanical devices and computer programs.

A widely used scheme is the *key* (**Fig. 1**(*b*)) which combines test selection and inference elements in a decision tree. In the example, test *a* must be carried out first and depending on its result, positive or negative, another test is indicated and so on until the name of a taxon is reached. Using a key in this way the tests are carried out consecutively and the sequence of tests used will be different for objects of different taxa.

In botany and zoology the rigid test selection of a key is a disadvantage because it may be impossible to apply some tests to a given specimen because of damage or the state of development of the specimen. To allow for this, *polyclaves* of various sorts have been devised (Morse, 1971). Polyclaves are simply summaries in a suitable form of all the available data on the behaviour of the relevant taxa (**Fig. 1**(*a*)). Test selection is left completely to the user and inference is usually by sequential elimination. In the example an experienced user might note that a particular specimen shows an unusual positive result in test *e* and reference to column *e* would immediately give taxon 3 as the identification. An inexperienced user might proceed: *a* is negative eliminating 1 and 6, *b* negative eliminates 5, *d* negative eliminates 2, *e* positive eliminates 4 leaving only 3, but could still reach an identification without using test *c* which is vital in the key.

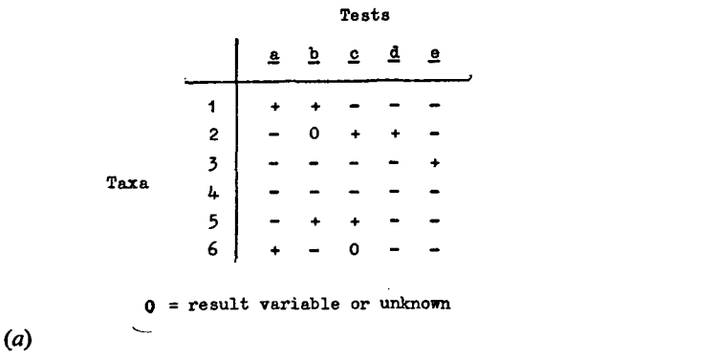
In microbiology there is often a delay of several days, or even weeks, before the result of a test is available and it is preferable to apply a number of tests, wait till all their results are known and then attempt inference. The key approach can still be used for inference but the decision tree does not now carry out test selection as all the tests in the key must be performed before entering the tree. An alternative method of inference is used in the *diagnostic table* approach. The diagnostic table (**Fig. 1**(*c*)) gives the expected results for the taxa in a subset of the available tests; identification is carried out by obtaining the results of all these tests and matching them against the results in the table. The advantage of this simultaneous inference over the sequential inference of a key is that an aberrant specimen which does not agree exactly with any of the taxa descriptions can sometimes be identified by its best match (Sneath, 1969). (Morse, 1971, takes 'diagnostic table' to mean a particular form of polyclave; the present meaning—an identification scheme using a fixed set of tests and simultaneous inference—seems more usual, at least in microbiology.)

Keys and diagnostic tables are constructed from a matrix giving the expected results of all the relevant taxa in all the available tests. An efficient diagnostic table is then constructed by finding a subset of the tests which contains the fewest possible tests, or is the cheapest if different 'costs' can be assigned to the tests, and still allows the required identification performance. The construction of a key involves the formation of a tree of tests to the same criteria except in this case the cost associated with the tree will be an average. The identification performance given by most keys and tables is to identify correctly a specimen of any of the relevant taxa provided that the specimen gives the typical pattern of results for that taxon. A more stringent requirement would be to identify correctly aberrant specimens differing in one test from the expected results.

The example (**Fig. 1**) shows two keys and a diagnostic table constructed from the same hypothetical matrix. If the tests can be carried out consecutively the first key, derived directly from the matrix, gives identification with the fewest tests, $2\frac{2}{3}$ on average. If all the necessary tests must be performed as a set the first key requires five tests but the diagnostic table only four. The key derived from the diagnostic table also requires only four tests though it is not as efficient, as a key, as the first; average length $2\frac{5}{6}$. The example involves fewer taxa and tests than most problems of practical interest but the conclusions might be expected to hold in general.

The construction of efficient keys has been considered from a theoretical point of view (Maccacaro, 1958; Möller, 1962; Gower and Barnett, 1971) and recently several computer programs have been described which construct keys automatically (Pankhurst, 1970a; Hall, 1970; Morse, 1971). Applications in various fields have been reported (Hill and Silvestri, 1962; Pankhurst, 1970b; Barnett, 1971). For microbiological applications Gower and Barnett (1971) modified their key constructing procedure so that both the efficiency of the key and the overall number of tests required were taken into account; the relative weighting of these separate factors could be varied.

Rather less work has been published on the diagnostic table problem. An efficient set of tests could obviously be found by examining all possible combinations of *t* tests, increasing *t* till a set is found which meets the performance requirements. The amount of computation required by this method becomes prohibitive for problems of practical interest (Piguet and Roberge, 1970). An alternative approach is to choose tests sequentially, each choice being made so as to optimise some function of the resulting partial table. Gyllenberg (1963) suggested as such a function the number of pairs of taxa separated by the table; two taxa were considered to be separated if they gave different results in at least one of the tests in

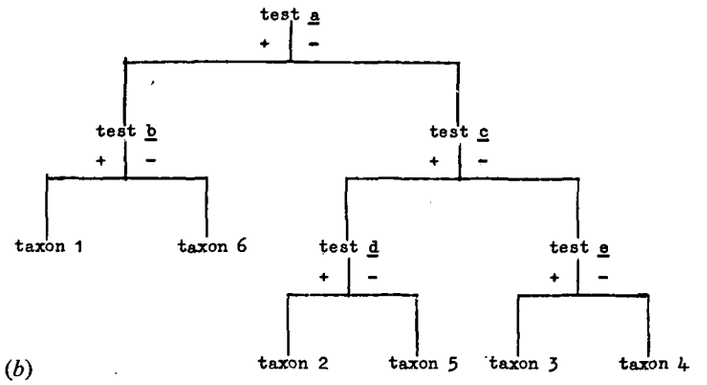


Tests

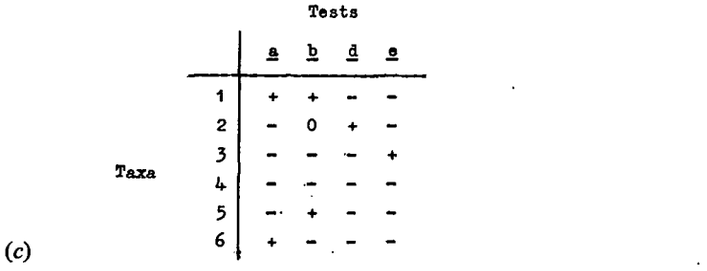
Taxa	a	b	c	d	e
1	+	+	−	−	−
2	−	0	+	+	−
3	−	−	−	−	+
4	−	−	−	−	−
5	−	+	+	−	−
6	+	−	0	−	−

0 = result variable or unknown

(*a*)



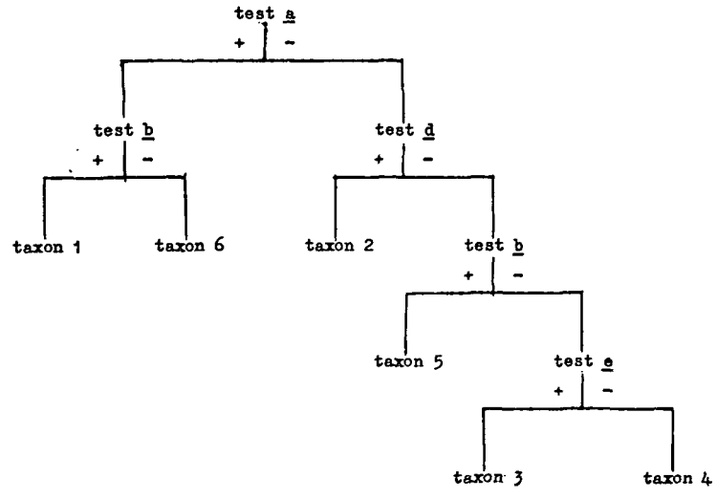
(*b*)



Tests

Taxa	a	b	d	e
1	+	+	−	−
2	−	0	+	−
3	−	−	−	+
4	−	−	−	−
5	−	+	−	−
6	+	−	−	−

(*c*)



(*d*)

Fig. 1. Two keys and a diagnostic table constructed from the same hypothetical data
(*a*) basic data (also Polyclave)
(*b*) key
(*c*) diagnostic table
(*d*) key derived from diagnostic table

the table. Rypka, Clapper, Bowen, and Babb (1967) applied this method to several sets of bacteriological data. Niemelä, Hopkins, and Quadling (1968) used a more complex function derived from information theory. As these authors point out, there is no guarantee that the set of tests found by a sequential method will contain the fewest possible tests.

A method is presented below which generates all possible sets of tests which meet the requirements of a diagnostic table. From these alternatives the most convenient can be chosen, by direct costing if cost weights are available or by subjective assessment. The results of applying the method to four sets of data are compared with the results given by a sequential method.

Method for generating diagnostic sets of tests

The basic data of the method is a matrix giving the expected result for each of the taxa in each of the tests. Initially only binary-valued tests are considered, the extension to multi-valued tests is discussed below. Entries in the matrix may be missing if the expected result is not known or if specimens of the taxon in question vary in their response to the test.

All the $(n(n-1)/2)$ possible pairs of taxa are considered and for each pair the tests which separate that pair are found. A test separates a taxon pair if the expected result of one taxon differs from that of the other; if either result is missing the test does not separate the pair. These separating tests are represented for each pair by a logical sum. For example, if tests *a*, *b* and *c* separate a given pair the representation will be $A + B + C$. The logical product of these expressions is then obtained in the form of a sum of products, giving the required sets of tests. For example, if tests *a* and *b* separate one pair of taxa and tests *a* and *c* a second pair, then $(A + B)(A + C) = AA + AC + AB + BC = A + BC$ and either test *a* alone or tests *b* and *c* together will separate both pairs of taxa. (A similar method has recently been proposed for the efficient programming of decision tables; Darwood, 1971.)

The sets of tests formed in this way meet only the most basic requirement of a diagnostic table; the only guarantee is that each taxon will differ from all others in at least one of the tests in the set. The use of more stringent requirements is considered below.

Programming the method

The method for generating alternative sets of tests involves the logical multiplication of expressions representing the separating tests for each pair of taxa. This is carried out by successively multiplying each *separating tests* expression by the current *test sets* expression formed by previous multiplications (**Fig. 2**). It is easy to show that the number of terms in the test sets expression may become very large. Consider hypothetical data which gives separating tests expressions $(A + B)$, $(C + D)$, $(E + F)$ etc. Then the product of m such expressions will have 2^m terms each representing a set of m tests. The maximum number of such expressions which might occur is half the number of tests, say $m = 25$ giving 33×10^6 terms.

For any particular data there may in fact be a large number of alternative test sets which would serve as diagnostic tables. This number may be so large that the computation becomes impossible. Although the final test sets expression is fixed by the data, the algorithm adopted should keep as small as possible the number of terms in the intermediate current test sets. This is likely to be achieved if the separating tests expressions are multiplied in reverse order of the number of tests they represent. The example (Fig. 2) shows this; forming the product in the 'natural' order generates five current sets reducing to the single final set. In general terms, multiplying first the separating tests expressions with few tests is likely to result in a relatively small number of current sets each containing a relatively large

WITHOUT ORDERING			WITH ORDERING		
Taxon pair	*Separating tests* expression	Current *test sets* expression	Taxon pair	*Separating tests* expression	Current *test sets* expression
1, 2	$A + C + D$		1, 6	B	
		$A + C + D$			B
1, 3	$A + B + E$		2, 5	D	
		$A + BC + BD + CE + DE$			BD
1, 4	$A + B$		3, 4	E	
		$A + BC + BD$			BDE
1, 5	$A + C$		4, 6	A	
		$A + BC$			$ABDE$
1, 6	B		1, 4	$A + B$	
		$AB + BC$			$ABDE$
2, 3	$C + D + E$		1, 5	$A + C$	
		$BC + ABD + ABE$			$ABDE$
2, 4	$C + D$		2, 4	$C + D$	
		$BC + ABD$			$ABDE$
2, 5	D		2, 6	$A + D$	
		$ABD + BCD$			$ABDE$
2, 6	$A + D$		3, 6	$A + E$	
		$ABD + BCD$			$ABDE$
3, 4	E		4, 5	$B + C$	
		$ABDE + BCDE$			$ABDE$
3, 5	$B + C + E$		5, 6	$A + B$	
		$ABDE + BCDE$			$ABDE$
3, 6	$A + E$		1, 2	$A + C + D$	
		$ABDE + BCDE$			$ABDE$
4, 5	$B + C$		1, 3	$A + B + E$	
		$ABDE + BCDE$			$ABDE$
4, 6	A		2, 3	$C + D + E$	
		$ABDE$			$ABDE$
5, 6	$A + B$		3, 5	$B + C + E$	
		$ABDE$			$ABDE$

Fig. 2. Example application of the method to the hypothetical data of Fig. 1 (a)

number of tests. These sets are likely to pass unchanged through further multiplications. The example shows an extreme case in which the tests *a*, *b*, *d* and *e*, necessary because each alone separates one of the taxon pairs, happen to be sufficient to separate all the other pairs. Piguet and Roberge (1970) found a similar result for a matrix of 83 taxa and 40 tests.

Each step in forming the logical product of the separating tests expressions comprises the multiplication of the separating tests expression—a sum of single elements—by the current test sets expression—a sum of products. This is carried out by enlarging each of the current test sets by including in the set each of the separating tests in turn. Any of the resulting sets which are absorbed by other sets are then removed. A term will absorb any product of itself and another term, e.g. $AB + ABC + ABDE = AB$.

For this particular form of logical multiplication two results can be derived which suggest an efficient algorithm. In the proofs of the results *A*, *B* and *C* are elements representing single tests and X and Y are products of any elements excluding *A*, *B* and *C*.

Result 1:
Any test set term which contains one of the separating tests will pass unchanged through the multiplication.

Proof:

$$(A + B + C + \ldots)(AX + \ldots) =$$
$$(AX + ABX + ACX + \ldots) + \ldots = AX + \ldots.$$

Result 2:
Only test set terms which pass unchanged and contain just one of the separating tests can absorb other sets. Only sets which have not passed unchanged can be absorbed.

Proof:
Since the multiplication can enlarge a test set by only one test, absorption can only occur in a situation such as

$$(A + \ldots)(AX + XY + \ldots) = (AX + AXY) + \ldots = AX + \ldots$$

Then set AX has passed unchanged, set XY cannot have passed unchanged or term AXY would not have arisen, and X cannot contain any of the separating tests or XY would have passed unchanged.

These results suggest the following algorithm for carrying out the logical multiplication:

1. Sort the current test sets into 'pass, cannot absorb', 'pass, can absorb' and 'must be enlarged', according to whether they contain more than one, one, or none of the separating tests.
2. Add to each of the 'must be enlarged' sets, each of the separating tests in turn. As each enlarged set is formed check it against the 'pass, can absorb' sets to see if it is absorbed. If not, add it to the current test sets in the 'cannot absorb' category.

Results

A FORTRAN program was written to generate diagnostic sets of tests from an input matrix by the method described above. A second program generated a single diagnostic set for each input matrix by a sequential choice of tests, maximising at each choice the total number of taxon pairs separated (Gyllenberg, 1963; Rypka *et al.*, 1967). Both programs were operated on the

Interact multi-access computer service of Baric Ltd.

The two programs were applied to four sets of published data (Rypka *et al.*, 1967; Niemelä *et al.*, 1968). The results obtained are summarised in **Table 1** in which the method described here is referred to as the non-sequential method. From the first data both methods produced the same single diagnostic test set. For the second data the non-sequential method generated two alternative sets of 25 tests each, the sequential method found one of these sets. The computation of the non-sequential method could not be completed for the third set of data; at the 17th taxon pair out of 28 more than 750 test sets were generated exceeding the maximum allowed by the program. The program listed the current test sets at this stage which included one set of three tests and 81 of four tests. The 3-test set and five of the 4-test sets were input to the second program which completed the sets by sequential choice of tests. Four of the 4-test sets were found to be already complete and five alternative additional tests were found to complete the 3-test set. Four tests must then be the fewest possible to separate this data and more alternative 4-test sets could probably have been found from among the 81 current when the program terminated. The sequential method found one of the 4-test sets. From the last data the non-sequential method generated 158 test sets, distributed as follows: 2 × 8-test, 17 × 9-test, 64 × 10-test, 61 × 11-test, 13 × 12-test and 1 × 13-test. The sequential method found one of the 8-test sets. Niemelä *et al.* (1968) using a different sequential method found two 8-test, three 9-test and two 10-test sets for the same data.

Considering the results as a whole, the sequential method produced in every case a diagnostic set with the fewest possible tests for the particular data. Except for the first data the non-sequential method produced several alternative sets for costing or subjective assessment. The large number of sets generated from the third data shows that the complete non-sequential analysis of even relatively small matrices may be impossible. The important factor here seems to be the large number of tests in the matrix relative to the number of taxa.

In the sequential method, if two or more tests were equally good choices at any stage the test taken by the program was that indicated by an initial ranking of the tests. If this occurred the equally good tests were listed by the program. The initial ranking of the tests could be altered interactively between successive analyses of the same matrix to follow alternative choices of tests. Trials of this procedure on some of the data showed that different test sets could be formed giving alternative diagnostic sets. The second 25-test set for the second data could not be found in this way however, and for the fourth data only two alternative sets were formed compared with 158 generated by the non-sequential method.

Extensions and alternative methods

Both the sequential and non-sequential methods described are based on the separation of taxon pairs by tests. In the description only binary-valued tests were considered and a test was taken as separating a pair of taxa if the expected result of one taxon differed from the expected result of the other. A diagnostic set of tests was considered to be complete when every pair of taxa was separated by at least one test. Possible extensions applicable to both methods are firstly allowing multi-valued tests and secondly requiring more than one test to separate each pair of taxa.

Consider a hypothetical multi-valued test with results red, green, and blue. If members of a given taxon are consistently of one colour, the expected result can be represented by a single value. If this is true for all taxa, then a red taxon, for example, will be separated by the test from green and blue taxa and the logic used to determine whether or not a binary-valued test separates a given taxon pair can be used for such multi-valued tests. But suppose a taxon some of whose members are red, some green but none blue. The test would separate such a taxon from a blue taxon but from no other. When taxa such as this occur a more complicated logic is required to recognise separations by multi-valued tests.

If a diagnostic table is constructed so that some pairs of taxa differ in only one test then a specimen aberrant in one of these tests will be incorrectly identified. Requiring at least two tests between each pair of taxa will guarantee that a specimen aberrant in just one test will be recognised as atypical. Requiring three tests will enable identification of such a specimen by matching as suggested by Sneath (1969). Lapage, Bascomb, Willcox, and Curtis (1970) selected tests as though for a diagnostic table but carried out identification by a probability calculation; they found that a reasonable identification rate was achieved when at least two tests were required to separate each taxon pair. The non-sequential method described here could allow for two tests between each taxon pair by expressing the separating tests as a sum of two-element products. For example, if tests a, b and c separated a pair the expression would be $AB + AC + BC$. The results obtained above which led to an efficient algorithm for the logical multiplication would have to be modified for the multiplication of such expressions. The sequential method is easily adapted to allow for two or any number of tests separating each pair. In practice (Lapage *et al.*, 1970) it has been found better not to maximise at each choice the total number of pairs separated by two tests, but to choose the test which separates the most of those pairs not already separated by two tests.

The sequential method used here selected tests one by one, maximising at each choice the total number of taxon pairs separated by the resulting partial diagnostic table. The choice of one test at a time is only the simplest of many possible strategies for searching for tests to optimise some function of the partial table (Gorry, 1968). Other functions have also been used (Niemelä *et al.*, 1968) and some of the functions developed for producing keys (Pankhurst, 1970a) might be applicable. The straightforward use of taxon pair separations would seem to have two advantages over other functions. Firstly the presence of unknown and variable test results does not affect the rationale of the method as it does for some of the other methods. Secondly the identification performance of the diagnostic tables

Table 1 Summary of results obtained by non-sequential (N) and sequential (S) methods

DATA			RESULTS OF METHOD N			METHOD S
Source	No. of taxa	No. of tests	Max. no. of sets	Final no. of sets	No. of tests in min. set	No. of tests in set
Rypka *et al.* (1967), Table 7	10	8	1	1	6	6
Rypka *et al.* (1967), Table 10	36	32	2	2	25	25
Rypka *et al.* (1967), Table 13	8	34	>750	*	*	4
Niemelä *et al.* (1968), Table 1	29	22	158	158	8	8

*Computation could not be completed

produced by the method can be increased by increasing the number of tests required to complete the separation of each taxon pair. In the four sets of data analysed the simple sequential method found in every case a diagnostic table with the fewest possible tests.

Acknowledgements

This work was supported by a grant from the Department of Health and Social Security. Thanks are due to Miss J. Stevens for the preparation of the manuscript.

References

BARNETT, J. A. (1971). Selection of tests for identifying yeasts, *Nature, New Biology*, Vol. 232, pp. 221-223.
DARWOOD, N. (1971). Implementing a decision table, *Computer Weekly*, No. 262, p. 6.
GORRY, G. A. (1968). Strategies for computer-aided diagnosis, *Mathematical Biosciences*, Vol. 2, pp. 293-318.
GOWER, J. C., and BARNETT, J. A. (1971). Selecting tests in diagnostic keys with unknown responses, *Nature, London*, Vol. 232, pp. 491-493.
GYLLENBERG, H. G. (1963). A general method for deriving determination schemes for random collections of microbial isolates, *Annales Academiae Scientiarum Fennicae*, A, IV, Vol. 69, pp. 1-23.
HALL, A. V. (1970). A computer-based system for forming identification keys, *Taxon*, Vol. 19, pp. 12-18.
HILL, L. R., and SILVESTRI, L. G. (1962). Quantitative methods in the systematics of Actinomycetales. III. The taxonomic significance of physiological-biochemical characters and the construction of a diagnostic key, *Giornale di Microbiologia*, Vol. 10, pp. 1-28.
LAPAGE, S. P., BASCOMB, S., WILLCOX, W. R., and CURTIS, M. A. (1970). Computer identification of bacteria. In: *Automation, Mechanization and Data Handling in Microbiology*, Society for Applied Bacteriology Technical Series No. 4, edited by A. Baillie and R. J. Gilbert, London: Academic Press, pp. 1-22.
MACCACARO, G. A. (1958). La misura della informazione contenuta nei criteri di classificazione, *Annali di Microbiologia ed Enzimologia*, Vol. 8, pp. 231-239.
MÖLLER, F. (1962). Quantitative methods in the systematics of Actinomycetales. IV. The theory and application of a probabilistic identification key, *Giornale de Microbiologia*, Vol. 10, pp. 29-47.
MORSE, L. E. (1971). Specimen identification and key construction with time-sharing computers, *Taxon*, Vol. 20, pp. 269-282.
NIEMELÄ, S. I., HOPKINS, J. W., and QUADLING, C. (1968). Selecting an economical binary test battery for a set of microbial cultures, *Canadian Journal of Microbiology*, Vol. 14, pp. 271-279.
PANKHURST, R. J. (1970a). A computer program for generating diagnostic keys, *The Computer Journal*, Vol. 13, pp. 145-151.
PANKHURST, R. J. (1970b). Key generation by computer, *Nature, London*, Vol. 227, pp. 1269-1270.
PIGUET, J. D., and ROBERGE, P. (1970). Problèmes posés par le diagnostic automatique des bâtonnets gram-négatifs, *Canadian Journal of Public Health*, Vol. 61, pp. 329-335.
RYPKA, E. W., CLAPPER, W. E., BOWEN, I. G., and BABB, R. (1967). A model for the identification of bacteria, *Journal of General Microbiology*, Vol. 46, pp. 407-424.
SNEATH, P. H. A. (1969). Computers in bacteriology, *Journal of Clinical Pathology*, Vol. 22, suppl. 3, pp. 87-92.

Correspondence

To the Editor
The Computer Journal

Sir,

I must take issue with Mr. Finn (*The Computer Journal*, Vol. 15, No. 1, p. 12, 1972) over his suggested extension to the DO loop in FORTRAN IV. One of the few redeeming features of the language is that it can deal with DO loop controlled variables in a reasonably efficient way, while the example he quotes of a complicated ALGOL 60 for statement shows how unpleasantly messy that construction can become. It also shows one of its major weaknesses; to give a simple example, assuming i is not modified inside the controlled statement, how many times will the controlled statement in the following be executed:

for $i := 1$ **step** 1 **until** 10, $i + 1$ **while** $i < 20$ **do** . . .?

Does $i + 1$ mean 'add one to the last value of i for which the controlled statement is executed' (in which case the answer is 20) or 'add one to the last value of i tested' (in which case the answer is 19, the case $i = 11$ being omitted)?

It is significant that ALGOL 68 is much more restrictive than ALGOL 60 over the matter of for loops. If FORTRAN is to be improved (other than by the most effective means, i.e. total annihilation), an ALGOL 68 kind of extension would be preferable than one based on ALGOL 60, for example

DO $i = m$ TO n BY k WHILE l

where i stands for any INTEGER variable m, n and k for any arithmetic expressions delivering an INTEGER value, and l for any LOGICAL (Boolean) expression. If l is the logical constant .TRUE. the 'WHILE l' may be omitted; if k is the integer constant 1 'BY k' may be omitted; and (in order to preserve the 'upward compatibility' so beloved of Fortranites and which has frustrated so many improvements in the past) 'TO' and 'BY' may be replaced by commas.

I agree with Mr. Finn that k should be allowed to deliver a negative increment; but allowing REAL controlled variables, which his suggestion of REAL increments also presumably implies, would do no more than encourage bad programming practice among a group of programmers already too exposed to bad influences.

Yours faithfully,

B. L. MEEK

Computer Unit
Queen Elizabeth College
Campden Hill Road
London W8
20 March 1972