Discussion and correspondence
The project, and the future of Computing Science courses

D. C. Brown*

Computing Laboratory, University of Kent, Canterbury

Computing Science lecturers have still to agree on exactly which elements of this rapidly expanding field are to be taught to undergraduate students. There is disagreement about the name of the subject area (ACM, 1968) and doubt about whether some of the subjects adopted by the computing fraternity really belong to Computing Science (Strachey, 1970). In fact Strachey comes to the conclusion 'that those subjects which we ought to teach are beyond us, while those which are within our capacity are on the whole irrelevant'. The ACM have chosen to divide Computing into three broad categories:

1. Information Structures and Processes.
2. Information Processing Systems.
3. Methodologies.

Obviously some subjects fall into more than one category. Examples of these would be Systems Analysis and Compiler Design. The ACM also mentions related subjects which are gradually becoming bound up with computing; for example Statistics, Linguistics, and Management.

The future

With rapid advances in knowledge, it is becoming increasingly important to produce some scientists with skills in more than one discipline. Large computing projects will require an interdisciplinary approach, and it is essential that undergraduate Computing Science courses 'provide the students with the intellectual maturity which will allow him to stay abreast of his own discipline and to interact with other disciplines' (ACM, 1968). It has already been recommended that all University students should acquire some knowledge of computing (UGC, 1970). I feel that the future of Computing Science teaching lies in an extension of this recommendation. In practice this would mean an increasing number of joint courses, with Computing Science being offered in several 'flavours'. e.g. Engineering, Business Studies, Science, Humanities, and Social Science.

Fragmentation

Computing Science has developed very little theory in comparison with other sciences, but it is generally agreed that this is because of the fact that, again by comparison, it is a young subject. Amarel, who is interested in ordering and formalising the vast amount of rather unstructured information in the field, is of the opinion that 'if the rate of generation of general principles and methods in the field will continue to lag behind the rate of production of specific computer systems and specific application packages, then there will be a tendency for computer science curriculi to be overwhelmed by a variety of fragments of detailed, specialised material—and for students of computer science to have negligible opportunity of being exposed to other disciplines, and to areas of intellectual activity where computers do not necessarily play major roles' (Amarel, 1972). This fragmentation is already evident in the all embracing reports from the ACM (1968) and the BCS (1967). One can readily see the difficulty in relating the different subjects in the field, and also the theoretical and practical aspects. Fragmentation is the very essence of our educational methods—all through school, and even afterwards, this tendency is strong, and probably most of all in the sciences. If we add to this tendency the demonstrably compartmentalised subject of Computing, what hope is there for our poor computing students?

Projects

The ACM recommend that students attain a 'reasonable level of programming competence' by 'including computer work of progressive complexity and diversity in the required courses', and that 'it is also desirable that each student participates in a "true to life" programming project' (ACM, 1968). This is one aspect of the teaching of Computing with which most people are in agreement, and it is obvious that suitable practical work is essential. It is common practice for students studying Computing Science degrees to attempt a reasonably substantial project during their final year.

A recommendation

With sufficient emphasis, and proper supervision, the final year project can be used as an initial solution to the problems outlined above. The project must be wide in scope, probably open ended, involving more than one discipline and more than one subject area within Computing Science. If possible it should be a team effort involving group discussions and decisions. Emphasis should be placed on adequate design of the software involved before the individual programs are written, careful documentation, and adequate testing.

A suitable task for such a project team would be to design and build a simplified Question Answering System (QAS) (Simmons, 1970; Brown, 1972). This is a form of Information Retrieval System which usually has the following characteristics:

1. An Input Language—as near natural as possible.
2. A Syntactic Analysis section—to determine the structure of the input statements or questions.
3. A Semantic Analysis section—to determine the meaning of the input.
4. Data Structures—the representation of information within the machine.
5. Inference Procedures—to obtain information implied by the stored information.
6. Output Language—as near natural as possible.

A project of this sort would allow members of the team to work in a subject area that interested them. It must be noted that as there are many approaches to the design of QAS it will allow students to do some individual reading, but it does, of course, mean that the project will have to be well supervised. I consider that the following topics could be brought into the project:

Computational Linguistics, Formal Grammars,
Mathematical Logic, Heuristics, Problem Solving,
Data Structures, Information Retrieval.

The project would also be a starting point for a study of the future applications of computers, such as the large scale use of

*Present address: Marconi Space & Defence Systems Ltd, Stanmore, Middlesex

Robots, Intelligent Information Retrieval Systems and Computer Aided Instruction.

Conclusion

In order to overcome problems of fragmentation, and as an initial solution to the requirement for Computing students with an interdisciplinary approach, it is suggested that more emphasis be placed on final year undergraduate projects. They should be used to widen a student's knowledge as well as to reinforce existing knowledge, and ought to involve groups of students. The author believes that, in future, Computing courses should be joint courses, biased towards one particular applications area, and concentrating on the techniques applicable in that broad area.

ACM (1968). Curriculum 68, *CACM*, Vol. 11, No. 3.

Amarel, S. (1972). A set of goals and approaches for education in Computer Science, *AFIPS conference proceedings*, Vol. 40.

British Computer Society, Education Committee (1967). Annual Education Review, *The Computer Bulletin*, Vol. 11, No. 1.

Brown, D. C. (1972). BIBLIO—A Bibliography of Question Answering Systems, Unpublished—University of Kent, Canterbury.

Simmons, R. F. (1970). Natural Language Question Answering Systems: 1969, *CACM*, Vol. 13, No. 1.

Strachey, C. (1970). Proceedings of the Symposium on Computer Science, Girton College, Cambridge; Aug. 1969, *Bulletin of the Institute of Mathematics and its Applications*, Vol. 6, No. 1.

UGC (1970). Teaching Computing in Universities, Report of a Joint Working Party (Chairman: K. Barrill) SBN-11-700165-1, University Grants Committee.

To the Editor
The Computer Journal

Sir

The paper 'A note on compiling arithmetic expressions' (Rohl and Linn, 1972) suggested that certain optimisations may be made in the compilation of arithmetic expressions by treating, for example, $a - b - c * d$ as if $-c * d - b + a$ had been written. The impression may have been given that this optimisation was applicable to compilers for all the well known high level languages, but this is not in fact so. The PL/1 specification (IBM, 1969) is particularly explicit on this point:

> 'The operators + and * are commutative, but not associative, as low-order rounding errors will depend on the order of evaluation of an expression. Thus, $A + B + C$ is not necessarily equal to $A + (B + C)$.'

(consider, for example, the case of $A = 1{\cdot}0001$, $B = -1{\cdot}0000$, $C = -4{\cdot}0000_{10} - 5$ when the expression is evaluated by floating point hardware with a mantissa of only 5 decimal digits; moreover, even in integral arithmetic, overflow could occur in one case but not in the other).

Whilst it might be argued that users should not allow such ill-conditioned expressions to arise, or that they should be required to insert brackets whenever the order of evaluation was of importance, the fact remains that implementors must abide by the official specifications of the various programming languages, much though one might regret the details of some of them. Thus it turns out that only in FORTRAN is the implementor given the necessary discretion to make this particular optimisation.

In ALGOL 60 (Naur *et al.*, 1963), the relevant sentence is 'The sequence of operations within one expression is generally from left to right . . .' This particular sentence has been interpreted in many ways but, if it is to mean anything at all, then the 'operations' referred to must be the actions carried out on behalf of the operators. Substantially the same sentence appears in the specification of ANSI COBOL.

In ALGOL 68 (van Wijngaarden *et al.*, 1969) it is clear that in $a - b - c * d$, the operations required by the second $-$ are to be performed on two operands which are the result of $a - b$ and the result of $c * d$.

There is however a simpler optimisation, which is legitimate with all the languages mentioned, in which the two operands of an operator may be interchanged if shorter code results thereby. Thus $a + c * d$ may be compiled as $c * d + a$. This relies on the commutative properties of + and *, which are not in dispute—indeed this optimisation is also applicable to $-$ and / on those machines which provide the 'reverse minus' and 'reverse divide' operations (hardware designers please note).

Yours faithfully,
C. H. Lindsey

Department of Computer Science
The University,
Manchester M13 9PL
15 December 1972

References

Rohl, J. S., and Linn, J. A. (1972). A note on compiling arithmetic expressions, *The Computer Journal*, Vol. 15, No. 1.

IBM (1969). PL/1 Language Specifications. Form Y33-6003-1.

van Wijngaarden, A. (Ed.) *et al.* (1969). Report on the Algorithmic Language ALGOL 68, *Numerische Mathematik*, Vol. 14, p. 79.

Naur, P. (Ed.) *et al.* (1963). Report on the Algorithmic Language ALGOL 60, *The Computer Journal*, Vol. 5, No. 4.

To the Editor
The Computer Journal

Sir

A note on the JK method

I should like to draw attention to a point concerning the JK method, a new method suggested by Kaiser (1972) for finding the eigensystem of a real symmetric matrix A. The method is similar to Jacobi's method; the purpose of this note is to make the connection more precise. The notation of Kaiser's paper will be used in an extended form.

The JK method constructs a sequence of matrices $B^{(k)}$ by:

$$\begin{aligned} B^{(0)} &= A \\ B^{(k)} &= B^{(k-1)}\, R^{(k)} \\ &= AT^{(k)} \text{ where } T^{(k)} = R^{(1)}\, R^{(2)} \ldots R^{(k)}, \end{aligned} \tag{1}$$

where $R^{(k)}$ is a plane rotation matrix chosen so that a pair of columns of $B^{(k)}$ are orthogonal. $R^{(k)}$ has the form:

$$r_{pp}^{(k)} = r_{qq}^{(k)} = \cos\phi;\ r_{pq}^{(k)} = -r_{qp}^{(k)} = -\sin\phi$$
$$r_{ii}^{(k)} = 1\ (i \neq p, q);\ r_{ij}^{(k)} = 0 \text{ otherwise},$$

for some pair (p, q). The aim is that $\lim_{k\to\infty} B^{(k)} = B$, the matrix of column eigenvectors of A scaled so that the sums of squares of the components equal the corresponding eigenvalue. This contrasts with Jacobi's method in which a sequence $A^{(k)}$ is generated according to

$$\left.\begin{aligned} A^{(k)} &= R^{(k)\prime}\, A^{(k-1)}\, R^{(k)},\ A^{(0)} = A \\ &= T^{(k)\prime}\, AT^{(k)}, \text{ where } T^{(k)} = R^{(1)}R^{(2)} \ldots R^{(k)} \end{aligned}\right\} \tag{2}$$

where again $R^{(k)}$ is a plane rotation matrix, chosen in this case to annihilate an off-diagonal element of $A^{(k-1)}$. For the JK method, Kaiser suggests that the order in which the pairs of columns are made orthogonal be taken as: $(1, 2), \ldots, (1, n)$; $(2, 3), \ldots, (2, n)$; $\ldots$; $(n - 1, n)$, repeating the cycle until convergence is obtained. However, no convergence proof is given.

We shall now show that the sequence of plane rotation matrices $\{R^{(k)}\}$ generated by the JK method applied to a matrix A is identical to that generated by the row-cyclic Jacobi method applied to the matrix A^2. Suppose that the matrices $R^{(1)}, \ldots, R^{(k-1)}$ are identical for both methods, and consider iteration k. We shall suppose that, in the JK method, columns p and q of $B^{(k)}$, denoted $b_p^{(k)}, b_q^{(k)}$, are to be orthogonal; and that, in Jacobi's method, the (p, q) element of $[A^2]^{(k)}$ is to be zero. Kaiser shows that the appropriate angle of rotation for the JK method is the angle ϕ defined by

$$\tan 2\phi = \frac{2b_p^{(k-1)'}\, b_q^{(k-1)}}{b_p^{(k-1)'}\, b_p^{(k-1)} - b_q^{(k-1)'}\, b_q^{(k-1)}}\,,\ \phi \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right].$$

From (1), and in an obvious notation, we have

$$\tan 2\phi = \frac{2t_p^{(k-1)'}\, A^2 t_q^{(k-1)}}{t_p^{(k-1)'}\, A^2 t_p^{(k-1)} - t_q^{(k-1)'}\, A^2 t_q^{(k-1)}}$$

This is exactly the angle of rotation required to annihilate the (p, q) element of the matrix $T^{(k-1)'} A^2 T^{(k-1)}$, that is, $[A^2]^{(k-1)}$. Thus an induction argument shows that the matrices $\{R^{(k)}\}$ generated by the two methods are identical.

Forsythe and Henrici (1960) have proved for the row-cyclic Jacobi method that

$$\lim_{k\to\infty} R^{(1)} R^{(2)} \ldots R^{(k)} = T$$

where T is the matrix of unit-length column eigenvectors of the matrix under consideration. Thus, for the JK method,

$$\begin{aligned}\lim_{k\to\infty} B^{(k)} &= \lim_{k\to\infty} AR^{(1)} \ldots R^{(k)}\\ &= AT\\ &= B, \text{ the required matrix.}\end{aligned}$$

This therefore establishes the convergence of the JK method.

References

FORSYTHE, G. E., and HENRICI, P. (1960). The cyclic Jacobi method for computing the principal values of a complex matrix, *Trans. Amer. Math. Soc.*, Vol. 94, pp. 1-23.

KAISER, H. F. (1972). The JK method: a procedure for finding the eigenvectors and eigenvalues of a real symmetric matrix, *The Computer Journal*, Vol. 15, pp. 271-273.

Yours faithfully

K. W. BRODLIE

Department of Mathematics
The University
Dundee DD1 4HN

7 March 1973

To the Editor
The Computer Journal

Sir

I should like to bring two items to your attention.

Firstly, as regards a meaning for 'SHRDLU' (this *Journal*, Vol. 16, No. 2, p. 34). In some efforts at counting the frequency of occurrence of letters from the English alphabet in 'normal' texts, the results were, in order,

ETAOIN SHRDLU . . .

This was noted, for example, by Sir Arthur Conan Doyle, in the Adventure of the Dancing Man from *The Return of Sherlock Holmes.* David Kahn, in *The Codebreakers*, gives this sequence as

ETAONI RSHDLU.

Was this also the motivation for the linotype layout?

Secondly, another reference which is related to the article *A graphical representation of the Backus-Naur form* by Chaplin, et al. (this *Journal*, Vol. 16, No. 2, pp. 28-29) is *A syntactical chart of ALGOL* 60 by Taylor, *et al.* (*Comm. ACM*, Vol. 4, No. 9 (Sept., 1961)). The point of view in the latter is 'top-down' though, while Chaplin's is 'bottom-up'.

Yours faithfully

J. RICHARD SWENSON

Department of Computer Science
University of Toronto
Toronto 181
Canada

24 April 1973

To the Editor
The Computer Journal

Sir

Extensions to Backus Naur Motivation

I am using the Backus Naur form of notation to define the syntax of a complicated data-stream. I wish, however, to extend the B.N.F. notation by the addition of certain further conventions, and I would like to request the help of readers of the *Journal* in giving credit where credit is due for these conventions.

The first extension to the standard B.N.F. is that:

1. ⟨Item ?⟩ indicates that ⟨Item⟩ may be absent or may occur once only; e.g.

   ⟨Integer⟩ ::= ⟨Sign ?⟩ ⟨Decimal digit⟩ |
   ⟨Integer⟩⟨Decimal digit⟩

2. ⟨Item*⟩ indicates that ⟨Item⟩ is present an indefinite number of times from 1 upwards; e.g.:

   ⟨Real Number⟩ ::= ⟨Integer⟩ . ⟨Decimal digit*⟩

3. ⟨Item* ?⟩ indicates that ⟨Item⟩ is present an indefinite number of times from 0 upwards; e.g.:

   ⟨Group⟩ ::= ⟨Member* ?⟩

This extension is, I believe, due to R. A. Brooker; perhaps some-one can tell me in what it was first published, and when.

The second extension which I wish to use is that, where the number of occurrences of ⟨Item⟩ is between certain known limits, say a and b, this shall be denoted by $\langle\text{Item}\rangle\,{}^{b}_{a}$; e.g.:

$$\begin{aligned}\langle\text{Vehicle Registration Mark}\rangle ::= {} & \langle\text{letter}\rangle\,{}^{2}_{1}\,\langle\text{decimal digit}\rangle\,{}^{4}_{1}\ |\\ & \langle\text{decimal digit}\rangle\,{}^{4}_{1}\,\langle\text{letter}\rangle\,{}^{2}_{1}\ |\\ & \langle\text{decimal digit}\rangle\,{}^{3}_{1}\,\langle\text{letter}\rangle\,{}^{3}_{3}\ |\\ & \langle\text{letter}\rangle\,{}^{3}_{3}\,\langle\text{decimal digit}\rangle\,{}^{3}_{1}\,\langle\text{letter ?}\rangle\end{aligned}$$

Although this would appear to be a 'natural' or 'obvious' extension of B.N.F., I do not know of any published mention of it. If it has been published, I shall be grateful to any one who can supply me with the relevant details.

Yours faithfully,

A. C. LARMAN

Selnec Southern Bus Company
Daw Bank
Stockport SK3 0DU

6 June 1973

Errata

In the paper 'Interactive digital simulation on a small computer' (this *Journal*, Vol. 16, No. 2, pp. 118-121) by B. Gay and S. G. Payne, an error appeared in Figure 3. Line 18 of SUBROUTINE INTI should read:

2 DT = DTD/2 .

In the paper 'Lagrangian interpolation at the Chebyshev points $x_{n,\nu} \equiv \cos(\nu\pi/n)$, $\nu = 0(1)n$; some unnoted advantages' (this *Journal*, Vol. 15, No. 2, pp. 156-159) by H. E. Salzer, there are a number of errors connected with one of the references. On page 156, left-hand column, line-4, the reference to (1964) should be a reference to (1952); on page 159, in the first and second lines of the second Berman reference, '(1964) . . . *Izv, Vysš. Učebn. Zaved. Matematika*, No. 6, (43), pp. 10-14' should read '(1952) . . . *Doklady Akad. Nauk SSSR*, (N.S.), Vol. 87, pp. 167-170'; also on page 159, in the second line of the second Berman reference 'Vol. 30, Part 2, 1965, p. 632' should read 'Vol. 14, 1953, p. 542'.