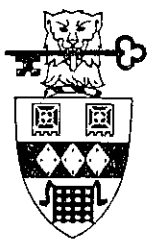
VECTOR

- Report of AGM 1987
- Survey of members
- Logic programming in Dyalog
- Education Vector
- News, reviews and APL Product Guide



The Journal of the British APL Association

A Specialist Group of the British Computer Society

Vol.4 No.1 July 1987

Contributions

All contributions to VECTOR should be sent to the Editor at the address given on the inside back cover. Letters and articles are welcomed on any topic of interest to the APL community. These do not need to be limited to APL themes nor must they be supportive of the language. Articles should be submitted in duplicate and accompanied by as much visual material as possible, including a photograph of the author. Unless otherwise specified each item will be considered for publication as a personal statement by its author, who accepts legal responsibility that its publication is not restricted by copyright. Authors are requested wherever possible to supply copy in machine-readable form ideally text files on a 5¼" IBM-PC compatible diskette. For other standards, please contact the Editor beforehand. Program listings should indicate the computer system on which they have been run. APL symbols should be displayed on a separate line and not embedded in narrative. Except where indicated, items published in VECTOR may be freely reprinted with appropriate acknowledgement.

Membership Rates 1987-88

Category	Fee p.a. £	Fee p.a. $	VECTOR copies	Passes
Nonvoting student membership	5		1	1
UK Private membership	10		1	1
Overseas private membership	18	27	1	1
Supplement for airmail (not needed for Europe)	8	12		
Corporate membership (UK)	85		10	5
Corporate membership (Overseas)	140	210		
Sustaining membership	360		neg	5

The membership year runs from 1st May to 30th April. Applications for membership should be made on the form at the end of the journal. Passes are required for entry to some Association events and for voting at Annual General Meetings. Applications for student membership will be accepted on a recommendation from a course supervisor. Overseas membership rates cover VECTOR surface postage and may be paid in £UK or $US.

Corporate membership is offered to organisations where APL is in professional use. Corporate members receive multiple copies of VECTOR and are offered group attendance of Association meetings. Partaking individuals need not be identified but a contact person should be nominated for all communications.

Sustaining membership is offered to companies trading in APL products; this is seen as a method of promoting the growth of APL interest and activity. As well as receiving public acknowledgement for their sponsorship, sustaining members receive bulk copies of VECTOR, and are offered news listings in the editorial section of the journal and opportunities to inform APL users of their products via seminars and articles.

Advertising

Advertisements in VECTOR should be submitted in typeset camera-ready A5 portrait format with a 20 mm blank border. Illustrations should be black-and-white photographs or line drawings. Rates are £250 per page. A6 and A7 sizes are offered subject to layout constraints.

Deadlines for advertisement bookings and receipt of camera-ready copy are given beneath the Quick-Reference Diary.

Advertisements should be booked with and sent to Cathy Dargue, whose address is given beneath the Index of Advertisers.

CONTENTS

H.M.W. PROGRAMMING CONSULTANTS LTD

Why not discover more about

★ **Consultancy/Support Service**

★ **APL on the IBM PC**

★ **VM/CMS Packages**

Ring Ken Jackson at

H.M.W. PROGRAMMING CONSULTANTS LTD
142 FELTHAM HILL ROAD,
ASHFORD, MIDDLESEX TW15 1HN
Telephone: Ashford 41232

Editorial

Professionalism and spreading the word

I am writing this on the eve, so to speak, of the AGM as a member of the retiring committee. Now is the time to review what we have done and to decide what we ought to do next.

The main efforts of the Association over the last year or so have been taken up on three things: Vector, "Activities" – our regular meetings at the Royal Over-Seas League – and APL 86.

Vector

Vector is now a successful and established feature of the world-wide APL scene. It would be a mistake to make a radical change in its style (as the members' responses to the recent questionnaire show). If there were a large influx of new members who needed more elementary articles on APL, there is both room in Vector to accommodate them and people to write them. The Vector team have reduced costs and increased advertising revenue. Once Vector drained almost the whole of our subscription revenue; now it is nearly self-financing and with a few more advertisements could even become profitable. We need to share out the work among more people and provide them with as much help as possible with their task. A few modems might make it easier to get more people working without excessive postal delays.

Activities

Our regular meetings were not planned far enough in advance in the early part of last year. The consequence was that our Publicity Officer was unable to publicise them except rather late and only to members. Over the year we have not obtained a single entry in any of the many listings of future events in the trade press. The current Activities team are now planning ahead and perhaps next year we shall get some publicity and this will improve attendance at meetings. When a subject, although of great interest to members, would not justify a company releasing staff to attend we should recognise this and either include it in a meeting with material of legitimate company interest or hold an evening meeting for it. Explicitly commercial presentations of APL products have not been encouraged. We are, it is true, an Association of users, but we are also an Association of Vendors and should welcome Vendor publicity and press coverage of product launches at our meetings. The Association could do a lot more for Vendors without either compromising our independence or getting accused of commercial bias.

APL 86

APL 86 is the latest and largest of a series of special events which have greatly enhanced the standing and prosperity of the Association. They have been well run (largely by volunteers) and well publicised within the APL community if not so well outside. Ideally such events should be run by people who do not have other onerous duties so as not to distract them from the essential work of the Association.

Aims

The Association has done well and served its members well over the last year, yet it has not pursued all its aims adequately. Compare what has been done with the objects of the Association as printed in Vector vol. 1 no. 4:

a To promote the use of APL
b To develop awareness and competence in APL
c To represent the interests of Association members with other bodies
d To contribute to the development of the language and definition of international standards

Practically the whole of our efforts is being spent on item b in this list. To put it another way we are spending over 90% of our efforts providing services to members and almost none on persuading non-APLers to try using APL.

Emphasis

The emphasis is right but overdone. It is right because for the Association to flourish it must provide benefits to its members attractive enough to keep them and recruit more of them. It should provide every practical kind of useful help to members to enable them to do their jobs better; a major part of this is keeping everyone up to date with news and reviews through Vector. These services must all continue and be expanded (as for example they have been by the launch of the public domain software library).

On the other hand it is overdone because our first object is to promote the use of APL and we have scarcely been doing that at all. We have left it almost entirely to the vendors.

Some initiatives in education, soon to come to fruition in a course for teachers, have been almost the only approach to anyone outside the Association to interest them in APL. We do remarkably little to help the novice, and there probably are more novices than experts in the Association. Even Vector has few articles which are designed to help a novice to improve.

Innovations

What new things could we do?

We could distribute Vector free to educational libraries to encourage interest.

We could produce some introductory leaflets to attract people to take their initial curiosity further.

We could produce an annotated booklist to help people choose the right book for their purpose.

We could provide a list of the currently available ways of providing oneself with a cheap personal APL facility, with a note of the advantages and limitations of each.

We could provide reports on the use of APL in education and sources of advice for teachers willing to try it.

We could provide a list of the names and addresses of APLers willing to help newcomers to APL in their spare time.

We could provide a range of workspaces through the public domain software library, each of which offers APL help with the teaching of a particular topic.

APL began as a notation, became a computer language and is fast becoming a microcomputer language. There may already be more people who use APL on a microcomputer than on a mainframe: if not true now it will be before long.

Soon, too, we shall be able to give away copies of I-APL to anyone who wants to try out APL on their home or school microcomputer. The I-APL mailing list (people who have written to enquire about it) is over 400 names and fast approaching 500. Note, this is before I-APL is complete and certainly before any attempt at a launch. The list will probably double or quadruple when it is launched. Over a tenth of enquirers are teachers at schools or colleges.

If the APL Association prepares to make it worth the while of these dozens of newcomers to APL to join we can double our membership or more.

That should be our aim. If APL doesn't attract new, young, keen recruits its future will indeed be bleak.

Anthony Camacho 29 May 1987

Draft APL Standard Available

A copy of the draft ISO Standard can now be yours for a mere £7.50 ($4.00 to BSI members). Write to Sales Admin (Drafts), BSI, Linford Wood, Milton Keynes, MK14 6LE. Quote document 86/67927 ISO/DIS 8485 Programming languages – APL (ISO/TC97).

Significant modification to the draft before it becomes a full International Standard is now extremely unlikely.

D. M. Ziemann

Quick-reference diary

compiled by David Preedy

Date	Venue	Event
1987		
8-11 September	Strasbourg	European Software Engineering Conference organised by AFCET, Paris
18 September	London	British APL Association meeting
16 October	London	British APL Association meeting
20 November	London	British APL Association meeting
1988		
15 January	London	British APL Association meeting
1-5 February	Sydney	APL88 – "APL – Past, Present, Future"
18 March	London	British APL Association meeting
20 May	London	British APL Association AGM & meeting
16 September	London	British APL Association meeting
21 October	London	British APL Association meeting
18 November	London	British APL Association meeting

All British APL Association meetings are to be held at the Royal Overseas League, Park Place, near Green Park tube station and start at 2pm.

Dates for future issues of VECTOR

	Vol 4 No 2	Vol 4 No 3	Vol 4 No 4
Copy date	24 Jul 87	16 Oct 87	29 Jan 88
Ad. booking	21 Aug 87	13 Nov 87	19 Feb 88
Ad. copy	28 Aug 87	20 Nov 87	26 Feb 88
Distribution	October 87	January 88	April 88

APL Training Courses

(Prices quoted are per course unless otherwise stated)

Dates shown may be subject to change. For confirmation of dates and/or further details regarding courses please contact the vendor directly *(see Vendor Addresses in the Product Guide)*.

LEVEL	COMPANY	NO. DAYS	PRICES	DATES
BEGINNERS	Cocking/Drury	3	375	15/9, 6/10, 27/10 10/11, 24/11, 8/12
	MicroAPL	1	poa	24/9, 12/11
	Uniware	5	poa	call
BEGINNERS/PC	Mercia	3	350	14/7, 8/9 +
INTERMEDIATE	Cocking/Drury	4	525	14/11
	MicroAPL	1	poa	3/9, 22/10
INTERMEDIATE/PC	Mercia	2	240	21/7, 29/9 +
	MicroAPL	1	poa	17/9, 5/11
ADVANCED	MicroAPL	1	poa	15/10, 3/12
	Uniware	5		call
SYS DESIGN	Cocking/Drury	5	595	31/8
	Mercia	3	375	11/8, 2/9 +
STATGRAPHIC	Cocking/Drury	2	poa	9/9, 14/10, 4/11
	Mercia	1	120	7/7, 24/9 +
	Uniware	2		call
EXECUSTAT	Mercia	1	95	2/7, 5/8 +
R.BASE 5000	Uniware	5		call

The following vendors run courses, details of which may be obtained directly from the vendor.

APL People
Inner Product
M.B.T.
Parallax

IBM Personal Computer APL/PC Version 2.0

6391329

IBM Personal Computer APL/PC Version 2.0, is a low cost, full function APL interpreter with a high degree of VS APL compatibility. It contains a wealth of auxiliary processors for a wide range of functions and interfaces to external devices.

- Emulates 8087 or 80287 if the co-processor is not present
- RS232 support
- IEEE-488/GPIB support
- Co-operative processing via IBM 3278/9 adapter
- Interface to IBM Macro Assembler and Professional Fortran
- APL2 GRAPHPAK compatible workspace provided
- Can run DOS functions and applications under APL
- Cover Workspaces for auxiliary processors

The interpreter, workspaces and auxiliary processors are supplied on three double-sided diskettes packaged with a comprehensive manual, quick reference card and a keyboard template. The manual includes setup, installation, tutorial and reference sections.

A separate package is supplied, containing a replacement ROM for the IBM Monochrome or Colour display adapters and a ROM puller. A program to load the APL font into the IBM Enhanced Graphics Adaptor is also included.

- Available from Authorised IBM PC Dealers.

IBM (UK) International Products Ltd
West Cross House
2 West Cross Way
Brentford
Middlesex TW8 9DY

Minutes of the AGM of the British APL Association

held on 5 June 1987

On arrival members were ticked off on a membership list and handed a voting card which was yellow (5 votes) for a corporate or sustaining member or purple (1 vote) for an individual member. Corporate or sustaining members could take five individual votes if they wished.

Chairman's survey of the year

Dick Bowman thanked Phil Goacher for making the offer to hold APL 86 in the UK; without that we could not have made the success of it that we did. APL 86 has given us a financial base which greatly increases the scope of what we can do. Some initiatives already begun are listed below.

Vector is now well established in the international APL community and is well and economically run. Its advertisements cover most of its costs. To make it an essential part of every APLer's life it needs more useful technical contributions.

We have had some difficulties with our regular meetings because the elected Activities Officer was unable to set up a forward plan. He has since resigned and a new activities team is beginning to get that under control.

The initiatives:

Surveys

1 A survey of members and their views, which will be reported in Vector.

2 A survey of the awareness and use of APL in higher education: this shows high awareness but that APL will only be used when it is seen to be relevant.

APLication

This is to be a three day special event on APL applications to be held in September 1988 at the University of Kent in Canterbury. It is not APL 88 (Northern Hemisphere).

Awards

The Committee proposes two outstanding achievement awards, which will be:

1 for recent work in APL, and
2 for spreading awareness about APL.

These awards will not be made when the Committee does not feel the best candidates are outstanding enough.

Education

A substantial contribution was made to I-APL. A subsidised course for teachers is to be held in July.

Software Library

The catalogue is growing – it made many acquisitions at APL 87 and is now beginning to generate orders. Success depends on members sending good software and telling people about the good software available in the library.

Dick Bowman ended by expressing his personal satisfaction at having been nominated to stand for the board of SIGAPL.

Secretary's report.

Anthony Camacho reported that the minute book was available for anyone to inspect and included minutes of meetings since the summary in the Turquoise Vector (3.4).

The APL book service is now installed at the University of Warwick bookshop: they take credit card orders by telephone. There will again be a booklist in Vector 4.1.

This AGM had to be postponed: the original date clashed with APL 87. A new Editor could not be elected in time to edit Vector 4.1 so the Secretary had perforce to take that on.

The Secretary had objected to paying the Royal Over-Seas League £40 for the half-day hire of an overhead projector and had found and arranged for the Association to buy the portable machine being used today.

Treasurer's report

Mel Chapman circulated a report and accounts. These showed that APL 86 had made about £50,000 profit. This year expenditure on publicity and meetings was well below budget. The £7,000 owed to us is from Vector advertisements and APL 86 delegates. The latter are to be threatened with an international blacklist.

In answer to questions he reported that:

the membership was 432 including 44 corporate and 11 sustaining members. Vector circulation is about 800.

the bank interest for the year is low because the profit for APL 86 was not paid to us until the end of the year. Interest is currently being earned at about £400 per month.

each meeting costs about £220.

Tim Perry asked what financial control there was over the donation to I-APL. Dick Bowman said the committee had approved the Technical Specification and Marketing Plan. Tim was not satisfied, saying that the contribution made in his name had been without his knowledge and against his will. He asked who ran I-APL, what was its status and what had the BAA done to ensure its contribution would not be misused. Dick Bowman emphasised that I-APL was not a BAA project and asked the Chairman of I-APL to answer.

Anthony Camacho explained that the I-APL project was controlled by a committee of five: himself, Ed Cherlin, Norman Thomson, Professor Howard Peelle and David Ziemann. I-APL had been formed into a limited company so that contributors could be sure accounts would be kept and presented at Companies' House. The Committee were also the directors of the company and had declared that I-APL would never pay a dividend but would devote all funds from donations and sales to the promotion of APL. I-APL (apart from one paid programmer) is entirely a voluntary and non-commercial organisation.

Asked about spin-off from the development of I-APL, Anthony replied that I-APL Ltd had agreed with Paul Chapman that if Paul or the project succeeded in selling the Development Environment as a commercial product, the proceeds would go half to Paul and half to I-APL Ltd.

The Chairman and Secretary then held the election of the 1987-88 committee.

The candidate list was:

Candidate	Post	Proposer	Seconder
Anthony Camacho	Chairman	R. J. Cocking	D. Preedy
Dick Bowman	* "	V. Lusmore	M. Jordan
Graham Parkhouse	*Secretary	A. J. Camacho	P. Donnelly
Cathy Dargue	Treasurer	Peter Cyriax	D. Preedy
Mel Chapman	* "	P. S. Goacher	V. Lusmore
Adrian Smith	*Journal	A. J. Camacho	D. Preedy
Norman Thomson	*Education	M. J. Kingston	A. J. Camacho
David Ziemann	*Technical	J. Barman	R. J. Cocking
Philip Goacher	*Activities	D. Parker	M. Jordan
Bernadette Leverton	Publicity	D. Eastwood	A. J. Camacho
Jill Moss	* "	V. Lusmore	Cathy Dargue
Val Lusmore	*Recruitment	M. Chapman	C. McCree
David Eastwood	*Projects	R. Nabavi	A. J. Camacho

* The asterisks mark those elected.

There was a second vote for the post of Chairman as the first vote was a dead heat and the 86-87 chairman declined to make a casting vote. The result was:

Anthony Camacho 6 corporate and 31 individual votes =61
Dick Bowman 9 corporate and 22 individual votes =67

Mel Chapman and Jill Moss won the other two contested elections easily.

The requirement that each candidate submit a brief statement (to be read out by Chairman or Secretary) outlining suitability and intentions should they be elected was not honoured by all candidates. Dick Bowman and Mel Chapman each made a little speech and David Eastwood also spoke briefly in favour of Bernadette Leverton.

The AGM ended at approximately 3.30 pm.
Anthony Camacho, retiring Hon Sec 7 June 1987

Call For Papers And Announcement Of The Second South African APL Symposium

Theme

"Building your future with APL"

23rd and 24th November 1987

CSIR Conference Centre, Pretoria

Organised by the APL Users Group in collaboration with the National Research Institution for Mathematical Sciences assisted by the Symposium Secretariat, CSIR.

Objectives:

- **Promote the use of APL in South Africa**
- **Make people aware of the future prospects of the language and the latest international trends.**
- **Demonstrate the uses of APL in practice.**
- **Promote APL as a scientific and educational environment.**
- **Establish a formal APL Users Group.**

Dr. Kenneth Iverson will present the keynote address.

Papers should be submitted as an extended abstract by June 15 and as camera ready copy by September 10.

All correspondence to:

The Symposium Secretariat S.451
CSIR PO Box 395
0001 Pretoria
South Africa

[From Frederick Macaskill of INFO*PLUS, PO Box 91030 Johannesburg]

News from Sustaining Members

Compiled by Cathy Dargue

Cocking and Drury Ltd

Cocking and Drury reported a high level of interest in the APL*PLUS Compiler, following the Whitbread sale and a recent customer mailshot. As a result, they are training consultants in the art of APL*PLUS compiler Technology. The first compiler consultant is David Ziemann, and others are expected to join him shortly.

The company is also pleased to announce a new second-generation APL interpreter for the micro-VAX and VAX range of DEC machines, running under the VMS operating system. The new product is based on APL*PLUS UNX, but boasts full integration into the VMS environment.

On the personal computing side, Cocking and Drury have announced support for APL* PLUS PC and Statgraphics for IBM's Personal System/2 range of machines, and the Compaq 386.

In order to encourage the spread of APL in the academic community, Cocking and Drury are renegotiating their APL*PLUS PC Site Licence policy with STSC. A special offer to academic sites is expected before the end of the year.

The London office is on the lookout for new larger premises, a project which is due to be completed by the end of 1987. As well as to service continuing company expansion, the move will provide the greatly improved public course facilities required to meet fast growing demand at all levels.

A recent addition to Cocking & Drury's ranks is Dave Phillips, who has joined as a Senior Consultant. Dave will be involved in the support of STSC's mainframe products. Hiss MVS expertise will be invaluable in providing effective maintenance for the company's ever growing mainframe client base.

Peter Cyriax Systems

Our interest in SQL and Databases has led us to develop links with DBAWG, the BCS Group working in this area. The principal topic at the moment is the forthcoming ISO Standard for SQL2 – but that is nearing completion, so there is increasing interest in more powerful (and friendly!) database languages supporting a wider range of applications. Our own Command Language, thanks mainly to APL, already has those sorts of features, so it seemed natural to become involved.

This should enable us to make our system conform to future high-level standards; and possibly help to get some APLish generality and flexibility into them.

Dyadic

As part of its new expansion program, Dyadic is pleased to announce the appointment of Andy Cooke as Sales Manager, and Andy Shiers as Customer Support Analyst. Andy Cooke joins Dyadic from Altos Computers where he was responsible for OEM and VAR business. Andy's primary role will be to lead the marketing and sales activities for Dyadic's expanding APL systems business based on the IBM 6150. Andy Shiers is an experienced Dyalog APL user, and a very welcome addition to Dyadic's customer support team.

Dyadic has announced plans to develop implementations of Dyalog APL for the Apple Macintosh II and the Compaq 386. These new versions should be available before the end of the summer. Dyadic also intends to develop a version of Dyalog APL for the IBM Personal System/2 under OS/2.

Dyadic enjoyed a busy and interesting week at APL87 in Dallas. During the conference, Dyadic announced a firm commitment to compatibility with APL2, and demonstrated the recent addition of Selective Specification to Dyalog APL. Selective Specification lets you use a variety of expressions on the left of the assignment arrow, and gives you faster and more concise ways of writing APL statements.

At APL87, Dyadic also announced a powerful Nested Array Editor for Dyalog APL. This new editor will allow the user to browse and edit any array, containing both character and numeric data, and arbitrarily complex in structure. Data can be changed by normal editing techniques, or by assigning the result of an APL expression to a selected portion of the array. The editor is designed to be the basis of a powerful multi-dimensional spreadsheet package. In Dallas, John Scholes demonstrated the extremely fast screen-handling facilities of the prototype version. The first release of the Nested Array Editor is scheduled or the end of the year.

IP Sharp Associates

IPSA and Reuters have recently reached agreement whereby Reuters will acquire all the outstanding shares in IPSA. Reuters principal interest in IPSA is in its base technologies of interactive networks and databases. Reuters is also clearly interested in a number of financial application products.

Reuters intention is that IPSA will continue under its present management as a separate company and will continue the development and support of all of its existing product lines. The expectation is that current developments will be speeded up especially concerning IPSANET and the number and type of public databases.

IPSA's joint venture agreement with SWIFT has now been finalised and the new company, Georisk, was incorporated in Belgium. The first product of this joint venture is STREAM, a comprehensive risk management and exposure system for international banks.

At APL87 IPSA demonstrated a new version of SHARP APL running under UNIX, called SHARP APL/UX. This APL is an implementation of Ken Iverson's 'APL Dictionary' and includes many extensions to APL which make the syntax of the language more complete and more coherent than in earlier implementations.

SHARP APL/UX resolves a major source of discussion in the APL world by extending SHARP APL to permit heterogeneous arrays (as in APL2) without giving up the distinction between enclosed and disclosed representations of a scalar. SHARP APL/UX provides upwards compatibility with SHARP APL/370 and SHARP APL/PCX; conforms to the ISO APL standard; nd combines the benefits of a number of previous APL systems, including SHARP APL, APL*PLUS and APL2.

IPSA's new APL for the IBM PC is now available. It is called APL-LAB/PC and is fully compatible with STSC's APL*PLUS PC system Release 6.3 and also includes IPSA,s QPACK software implementing the SHARP APL Package data type. A Package is a special data type consisting of a set of named APL objects. QPACK is written in assembler and allows functions and data to be paged in and out from file much more rapidly than other techniques. QPACK is also available as an add on module to existing APL*PLUS PC systems.

Mercia Software

It seems that every issue of VECTOR brings news of more APL systems – and this issue is no different.

All the excitement now is about the new APL*PLUS interpreter for the Apple MACINTOSH. APL*PLUS/MAC is a full featured APL system, integrated with all the MACINTOSH user interface facilities – windows, icons, mouse, pop-up mens. It also has the advantage that workspace and object size is only limited by how much RAM you have in your machine, a good PLUS point for users up against the 640K (or 952K with an ALL card) limit of PC-DOS. At the time of writing, Apl for the MAC is priced at £310, so it is also cheaper than the IBM PC system.

LOGOL was formally released in April. Designed as a decision support system to plan and control the flow of packaged consumer goods and service parts through a multi-echelon distribution network, LOGOL is, we believe, the largest application system to have been developed in APL*PLUS/PC. With prices starting at £10,000 – it has to be the most expensive!

We have found that not only is the product quite unique, but that there is also a large market looking for such a system. As a consequence, many large UK organisations have shown an interest, and we hope to have our first installation in July.

MicroAPL Ltd

MicroAPL's highly-successful implementations of APL.68000 on the new generation of small 68000-based micros – the Apple Macintosh, Commodore Amiga and Atari ST – now offer FREE run-time licences to developers. This removes the last obstacle in using APL to write commercial applications for small machines, and is particularly attractive to software developers because of the ease of porting applications between the three different machines. MicroAPL released a new version of the Atari ST product at Dallas in April, and will be issuing a new Macintosh version at about the time this edition of VECTOR appears.

The MicroAPL AURORA multi-processor supermicro – the system of choice for demanding APL applications – has been reduced in price, and has been supplemented by a new faster, more expandable model, the Aurora II, at the top end of the range.

Also new in the UK is a version of APL.68000 for the IBM PC/XT/AT and compatibles, running in a special plug-in card with either a 68000 or 68020 processor. Unlike other such cards, APL runs under DOS in this environment, so there is no need to learn special operating system commands, partition the disc or lose the advantage of the wealth of PC-DOS software. This new APL starts at £995 *including the plug-in card* with 1 Megabyte of memory, and workspaces of nearly 1MB and 4MB are attainable. The product is aimed at users whose applications are too large to run under APL*PLUS/PC, or who need the extra speed of a 68000 or 68020-based system.

APL People Ltd

The APL Group has not created or bought any new companies this quarter, much to the relief of the staff (and undoubtedly of the readers of Vector!) who are just about used to the current position. There were flutterings and rumours back in February of an exciting acquisition, but these have receded. As a result of our dealings we have recently persuaded Diana Mackay, ex-support manager of Sheffield Micro (a casualty of the PPL debacle), to join us and open our Sheffield office. From here, she will concentrate on servicing Midlands and Northern clients, and is responsible for the accounting software packages we sell to our Engineering customers.

APL Tran-Plan are still sorting out car-park design for the second largest group of architects in the world on their largest project, Canary Wharf in Londons Dockland. Current sections involve numerous trips to Chicago (?) and complex systems to organise the placement and ordering of all the road signs. I'm not sure when we finally get to drive our cars into the lovingly crafted spaces and pay through the nose for the privilege but watch this space.

APL People are now placing people internationally (three are off to New York in August) and have over 40 current vacancies on their books in the UK. There is an encouraging upward trend, with six software houses selling products written in APL joining the four firms of APL consultants as some of our more constant customers. Following on Roy Sykes' keynote speech at APL86, we are excited at this development because it looks as if APL is finally gaining its rightful place behind the scenes in the software industry.

Report on APL87

by J Sullivan

This is not going to be a blow-by-blow account of what happened at every session of APL87, but rather a personal view of what I found most interesting at the conference. This is not to suggest that anything I have left out was boring or uninteresting, so don't assume that because I haven't mentioned your favourite topic or session that I thought it so.

The tutorials on Sunday were a mixture of the serious and the not-so-serious. On the serious side the talks on Data Driven Control Methodology and APL techniques for Building Expert Systems in the PC Environment were most interesting and informative: the techniques used are not restricted to the fields mentioned by the authors, for example it is always better to have the data drive a program rather than to have the program drive the data.

Two humorous presentations were given by Gitte Christensen (How to manage APLers – conclusion: you can't) and Phil Smith who demonstrated some of the research he is currently carrying out. Phil Smith's talk was expanded later on in the week in another plenary session which gave much light relief amidst all the serious work that usually goes on in an APL conference.

Dr Jim Brown gave a paper on the design principles of APL2 which described some of the extensions in APL2 and discussed the reasons why they were adopted. In the paper he also explained why the result of enclosing a scalar in APL2 is identical to the original scalar. On a similar theme there were papers by Phil Benkard (Replicate Each, anyone? and Implications of APL2 Grammar) which discussed the replicate operator and APL syntax in a highly technical and exhaustive manner.

As a user of APL2 I was most interested to hear what the people from IBM had to say. The most important session from this point of view was the IBM Product Forum on the 3090 Vector Facility. As everybody is aware, APL is the ideal language for processing array operations 'in parallel', and the Vector Facility is the bit of hardware bolted on to the computer that performs operations in parallel. (Although I say 'in parallel' here, IBM will tell you that it's not true parallel but pipelined; that's near enough parallel for me.) The implications of this are that a whole new set of Assembler instructions has had to be written and programmers will have to amend their programs to take advantage of them. A new Fortran compiler has had to be produced and the users will have to amend their programs to take advantage of it and a new version of APL2 is being developed and programmers will have to do nothing in order to take advantage of the Vector Facility. We had the chance to test the Vector Facility on the IBM stand, where they had two terminals linked to a local computer centre for our use.

Workers at IBM Yorktown Heights are developing an APL translator which is designed to translate APL2 code into another high-level language for compiling and link-editing and calling from within your APL workspace via associated processor 11 using the ⎕NA facility introduced with Release 2 of APL2. Currently FORTRAN has been chosen as the high-level language, because it can be called through ⎕NA and because it lends itself to

Vectorization. There were a number of problems to be sorted out in developing the translator, particularly concerning 'where do you draw the line?' when it comes to compiling code that can only be interpreted at run-time? The answer is you don't compile everything, and you certainly don't compile bad code (does anybody actually write bad APL code these days?). For those of you who want to read more about the Yorktown translator there is a very informative article in the November 1986 copy of the IBM Journal of Research and Development.

I was also interested in the 'other' APL that is being developed along parallel lines to IBM APL2, i.e. Sharp APL. Sharp APL is being developed towards Ken Iverson's 'Dictionary of APL', which was the basis for Dr Iverson's talk at the opening plenary session in which he traced the history of APL notation. Sharp APL was also discussed in I.P.Sharp's product forum. In this the history of I.P.Sharp and their parent Reuter's was outlined, and then some figures were given on the number of Sharp APL sites throughout the world. Sharp has two significant new development efforts under way: Sharp APL/VM, and Sharp APL/UX (also known as SAX), which was on display on their stand in the exhibition. I don't intend to go into any details except to say that it is very close to 'Dictionary APL'. Further details may be had from I.P.Sharp themselves.

On the social side, there was a reception at the Dallas Museum of Art on the Monday evening. This was very enjoyable as in addition to the food and drink we were able to wander around and enjoy the exhibits. The museum has a large collection of modern art from all over the world (I've never seen so many original Picassos in one place before) together with examples of early American colonial furniture and South American native artifacts.

To sum up the conference in a few words is impossible. Between 8.30 a.m. and 5.30 p.m. it was all very hard work, and between 5.30 p.m. and bedtime it was very hard play. All in all though it was exceedingly enjoyable.

Overheard at APL 87:

"What *is* Southfork?"

"The first 80% of the code takes the first 80% of the development time; the remaining 20% of the code takes the remaining 80% of the development time". [Leslie Goldsmith]

– While waiting for the output from a function: "Here's an example of an empty array". [Alan Graham]

"That's why they invented cars – to get around Dallas." [Sephen Jaffe]

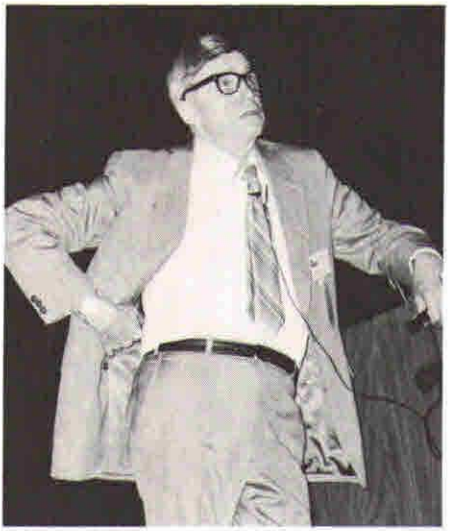
"What do you call code that hasn't been optimised? Pessimised." [Graham Driscoll]

"Plan to do a prototype – you will anyway." [Alan Graham]

"This machine is not infinitely fast; it is only infinite over 2." [Jim Brown]

See you all at Sydney?

Some of David Ziemann's photographs from Dallas



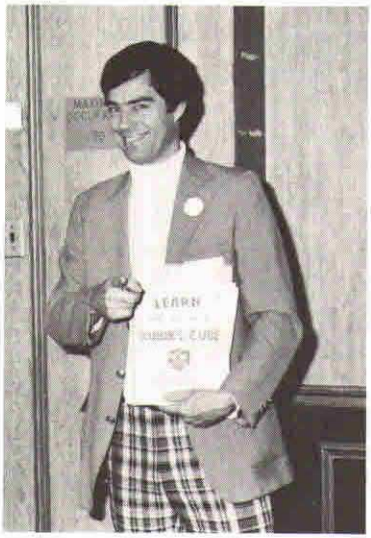
Ken Iverson

2=Carl Moore, 3=Bob Bernecky, 4=Jim Brown and 5=Cory Skutt – apologies to 1 and 6

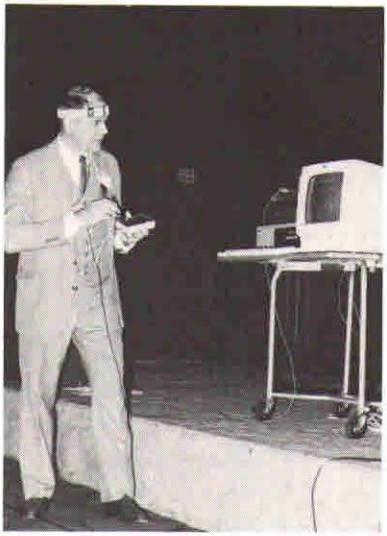
Ed Cherling helped by John Carpenter
"Lets put APL top of the heap"

Gitte Christensen

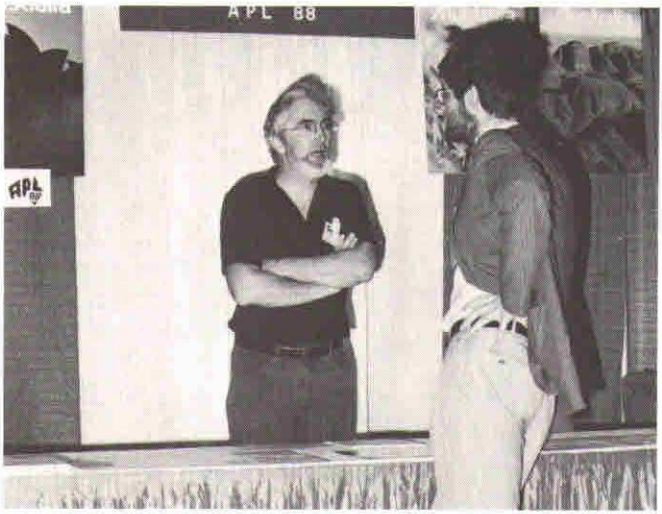
Bob Bernecky, E.E. McDonnell, Bob Smith and Ken Iverson at the ISO meeting before APL 87



Prof. Howard Peelle



Phil Smith demonstrating computer control by neural link



Neville Holmes and a booking for APL 88 from Paul Chapman

Skip Cave and Jim Brown at one of the I-APL benefit concerts they organised

John Myrna congratulating E.E. McDonnell on winning the SIGAPL outstanding achievement award

MicroAPL announces ...

The 4 Mb Workspace for your PC

If you wanted to use your PC for APL applications but didn't dare, then buy:

MultiAPL

- **APL.68000 for the PC!**
- **running under PC-DOS**

It's FAST!!

Sample configurations:

68000/1MB	£ 995
68000/1MB/Maths processor	£1295
68020/1MB/Maths processor	£2995
68020/4MB/Maths processor	£5495

Prices include:

Plug in card
APL.68000
Keyboard stickers
Manual
STSC emulation software

Contact:

MICRO APL

MicroAPL Ltd.
Unit 1F, Tideway Industrial Estate
LONDON, SW8 5BP
Telephone: 01 622 0395

Access/Barclaycard Orders accepted

The Education Vector

by Norman Thomson

By the time this column appears the course for teachers in Winchester will have taken place and I hope more enthusiasts will be helping to fuel the APL fires. When teachers comment that one of the things which is required to further the progress of APL in education is a newsletter or journal to communicate APL matters, it is satisfying to be able to tell them that Vector can supply the need and is now in its fourth year of successful publication. Furthermore issue 4.3 (two numbers ahead) is to concentrate on education.

The readers of this magazine fall broadly into three categories: those who use APL as a routine tool in their workplace, those whose business it is to market APL and related products, and those in whom APL stirs up an urge to communicate whatever it is of beauty, ingenuity or delight that brought about their first glorious APL rapture. At the risk of using overtly emotive words these might be styled the APL artisans, salesmen and idealists respectively. Or to make a musical analogy, the categories correspond to professional musicians, impresarios and connoisseurs (dilletanti?) – and this may perhaps improve the emotive balance! Of course these categories are not mutually exclusive, and readers will no doubt be able to identify people in their acquaintance who fall into at least two of them. St. Paul made the point that the soundness of an organisation is related to the balance of attributes and skills of its members, and this is no less true of the British APL Association than of the early church.

Now the purpose of this discussion is to make the point that so far it is the first two categories of membership which have made the running in our Association, largely because the cost of APL systems has ruled out the possibility of "idealist APL" gaining more than a slender foothold. With the advent of I-APL however, this barrier has been broken, and the riches which have excited and stimulated people fortunate enough to have easy access to APL are now available to a much larger potential audience. How will this audience react to the bounty? One measure will be the volume of contributions to these pages, and the hope is that the Education Vector will become not just a few lines of comment, but a valuable means of exchanging hints, experiences, ideas, algorithms and so on.

To finish on a practical and specific note, Professor Alan Hawkes reports from Swansea that the APL workspaces which are used in the Statistical Computing course now contain help files for the course-specific functions. Typing **help 'histogram'** produces suitable student assistance on how to use the histogram function. Then, when it is time to exercise statistical understanding, self-marking functions are called which as well as generating a degree of randomness in the questions, leave a record in a file. The teacher can thus assess which parts of the course caused most trouble and are worth trying to improve or extend (and which could be shortened without causing too much extra difficulty). It isn't that this couldn't be done in any other computing language – but can you think of any in which it would be a realistic possibility?

APL CONSULTANTS

LONDON & READING

Account Managers	(6 years+)	to	**25K**
Senior Consultants	(4-6 years)	to	**21K**
Consultants	(2-4 years)	to	**17K**
Junior Consultants	(1-2 years)	to	**13K**

Are your APL skills and potential being recognised and rewarded?

Cocking & Drury consultants have been implementing successful decision support applications for 10 years, with clients who appreciate the productivity benefits of APL.

In our professional team you will experience a range of APL environments - APL*Plus, VSAPL, APL2 and Unix, on both mainframes and micros. You will also be developing systems which, increasingly, need to interface with non-APL Information Centre products.

Of course as the leading APL consultancy, in a rapidly expanding market, we offer a rewarding career with first class benefits - profit sharing, free health insurance, and a non-contributory pension.

For further details call Ralph Wilson on 0734 588835

COCKING & DRURY LTD.

155 Friar Street, Reading, RG1 1HE

Caption Competition – usual rules
Entries by 30 August please.

(A)

(B)

(C)

Book Review
APL et GDDM; travail en plein ecran
B Legrand

reviewed by Adrian Smith

Introduction

The first thing to say about this book is that it is in French!! The second thing is that it really doesn't matter; the cartoons and the APL are readable in any language, and the French bits would cause few problems to even the most rusty of O-level scholars. Consequently I shall forget the language barrier entirely, and concentrate on the content and presentation of the material.

Basically, it takes you through GDDM-style screen handling 'from the bottom up'. For some reason, IBM make it terribly hard for you to do this yourself. It is not that the AP126 interface is intrinsically hard to use; rather that the relevant documentation is split over several different manuals, many of which (e.g. the 'bible' of the GDDM Programmer's Guide) ignore APL altogether. As I remember it, it took me several days to track down and cross-reference the necessary documentation, before I could even begin to follow the trail which M. Legrand has opened up with his book.

Under the most appropriate heading 'Mise en Garde', the author compares the action of the IBM routine SETCURSOR with his own 'hand-built' function:

" . . . . voici la liste de fonctions qui sont indispensables au positionnement du curseur par cette technique:

SETCURSOR ASFCUR BOX
ERRSTOP GDDNT0 GDDM
GDDMANALYZE MESSAGE
QREPLACE XBLANKS

Comment lire les brochures IBM.

Bien entendu, certaines de ces fonctions ne sont appelés qu'en cas d'incident, mais ce sont quand même plus de 75 instructions qui sont ainsi nécessaires au fonctionnement de SETCURSOR.

A titre de comparison, notre fonction ECURS n'a que deux instructions, et la totalité de notre jeu d'utilitaires compte 100 instructions seulement.

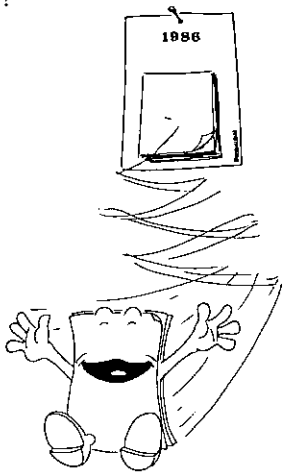
Toutes les autres instructions representent un surcout inutile"

I love that final phrase "un surcout inutile". What's more I really don't believe you could translate it, and I rather hope no one is daft enough to try!!

Who should read this Book?

To be fair, you probably do need at least a smattering of French. However having seen the mess that the translator made of M. Legrand's previous book I would stick firmly by my view that no translation is infinitely superior to a bad one.

Basically, if you want to write clean efficient full-screen applications, you should take a deep breath,)DROP 1 FSDESIGN and invest a few pounds of your company's money in this book. It won't by any means take you all the way, but it will at least show you where to start, and the cartoons will keep you amused along the way:



Libération d'une page.

The topics covered begin with basic GDDM concepts, working from the page, up through formats, and then to fields within each format. Simple functions are used throughout to illustrate the concepts, and M. Legrand is very thorough in his treatment of such detail as the position of field attributes. He also points out most of the pitfalls, such as the so-called 'numeric input' field type; this rejects the APL minus, but accepts any old rubbish like '2 . 3 . . 4' as valid. He rightly rejects it as useless.

Reservations

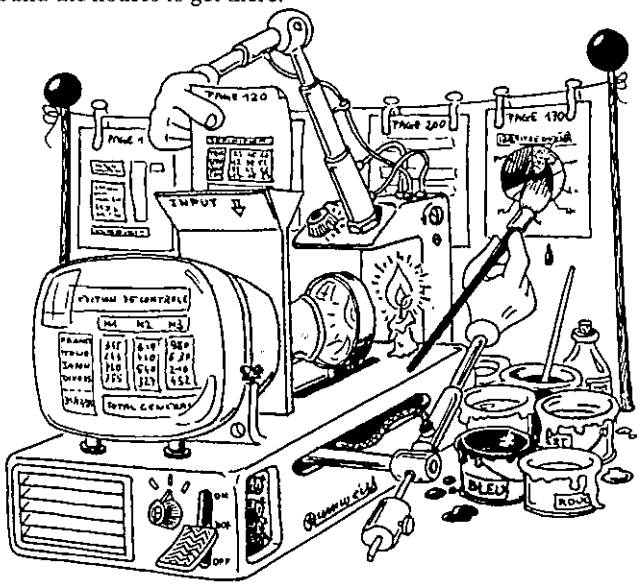
So far so good, but there have to be some reservations. My main worry is that a little knowledge is a dangerous thing. The functions in this book will get you going with a good basic understanding of GDDM; they are not the basis of a fully fledged application development system. On the other hand the IBM supplied routines are an absolute disaster area and almost anything would be an improvement. M. Legrand suggests that you use the IBM routines for screen design (where efficiency doesn't matter) and progressively replace the screen-handling routines with your own code.

You can start with the basics (such as his example of setting the cursor) and the savings in CPU should be more than enough to spur you on. It sounds reasonable enough, but if you have more than a few dozen workspaces to look after you have set yourself some nasty upgrade problems. You may also find yourself at the end of the book with a half-baked system and a nasty question mark hanging over your maintenance commitment.

Finally, there is no mention at all of graphics. Most IBM sites will surely have PGF running alongside GDDM, and the interface from APL is again straightforward, and fits in well with M. Legrand's approach. I seem to remember quoting my PIECHART function in a previous Vector; I think it was 4 lines, of which the first two were comments!

Summary

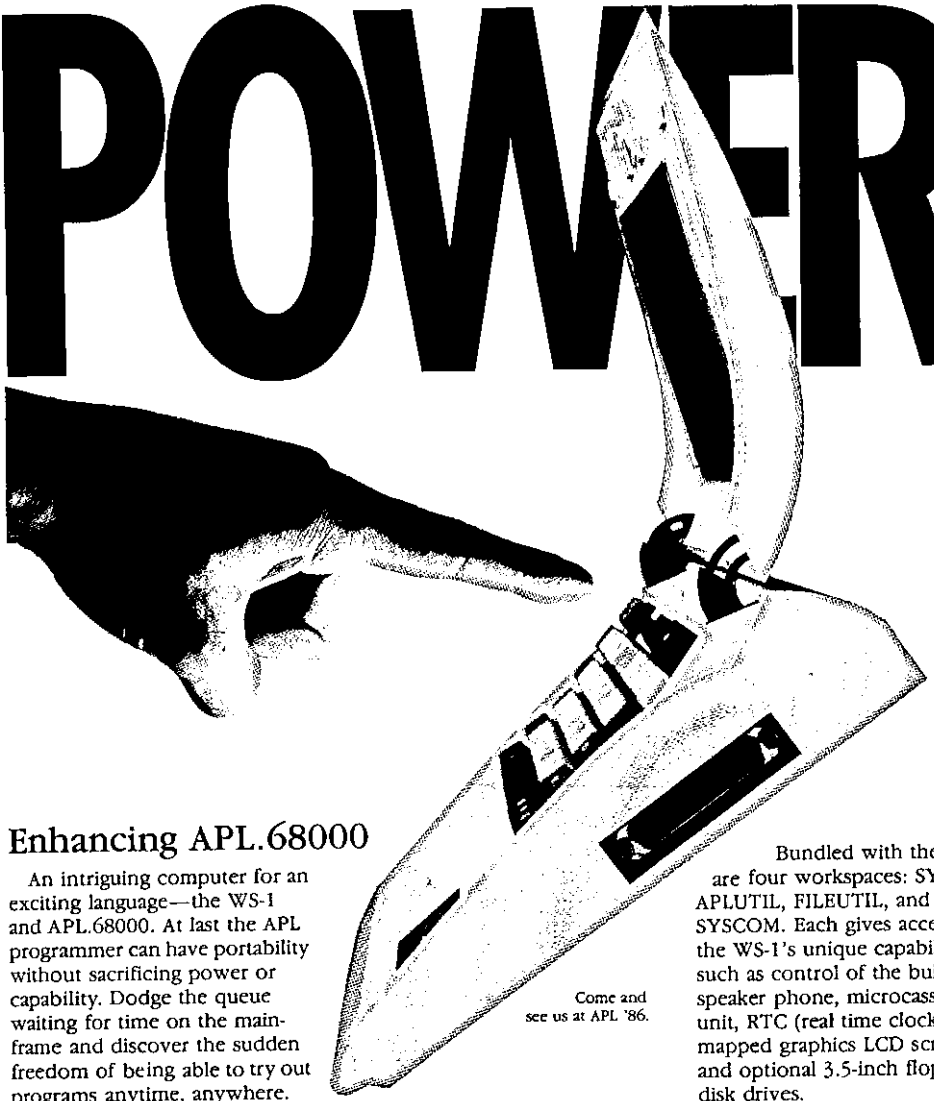
For some time, I have been puzzled by the 'accepted fact' that GDDM is not an efficient way of handling the screen from APL applications. I had never even looked at the IBM routines, beyond the most cursory glance at the code. M. Legrand has cleared things up considerably; it is not APL-GDDM that is inefficient – it is the way the IBM routines go three times round the houses to get there!



COMMENT FONCTIONNE GDDM

If you are serious about using APL in 'real' systems, you need someone in your organisation who will get right down to the nitty-gritty of GDDM at the lowest level. M. Legrand's book is the start that person needs; if he needs to go further he is (as indeed I was) on his own. Never mind; 'APL et GDDM' will have more than paid for itself by then.

POWER



Enhancing APL.68000

An intriguing computer for an exciting language—the WS-1 and APL.68000. At last the APL programmer can have portability without sacrificing power or capability. Dodge the queue waiting for time on the mainframe and discover the sudden freedom of being able to try out programs anytime, anywhere.

The APL.68000 interpreter is implemented in 86KB of ROM, running under a multi-user, multi-tasking operating system called BIG. DOS. Speed is the essence of APL programming, and now the WS-1 makes development even faster.

Come and see us at APL '86.

APL.68000 on the WS-1 has attractive enhancements such as a powerful component file system, QUAD. FMT function for alpha report formatting, QUAD. CC function for full-screen control, and extended error trapping facilities.

Bundled with the WS-1 are four workspaces: SYSFNS, APLUTIL, FILEUTIL, and SYSCOM. Each gives access to the WS-1's unique capabilities such as control of the built-in speaker phone, microcassette unit, RTC (real time clock), bit-mapped graphics LCD screen, and optional 3.5-inch floppy disk drives.

Compress these capabilities into a sleek footprint measuring less than 13 inches by 11 inches, and you have the ultimate definition of power.

ampère

FOR DISTRIBUTORSHIP INFORMATION AND PRODUCT DETAILS PLEASE CONTACT:

ampère INCORPORATED Asahi Bldg., 7-5-20 Nishi-Shinjuku, Shinjuku-ku, Tokyo, Japan. *Phone:* 03-365-0825. *Telefax:* 03-365-0999. *Telex:* J33101 AMPERE. IP Sharp Mail Box Code AMP (Group Code APLWS).

Hardware Review: The XT-286, ForteGraph, and the IBM Enhanced Keyboard

by Adrian Smith

Background

Some time ago, I hatched a dream of a 'one-per-desk' APL workstation; something which would allow me to do mainframe APL (with full 32 line graphics) as well as APL*PLUS/PC (in colour at a decent resolution). The right hardware for the job has gradually become available, in the Enhanced colour display, and the FORTEGRAPH (alias DCA) piggyback for IRMA. A brief trial of this in our dear old twin floppy PC convinced me, so off went the equipment request, and some days ago a heap of junk duly appeared on my desk.

XT-286: First Impressions

The first thing to note is that this machine isn't terribly PC compatible. It won't run Borland's Superkey, and things like the Turbo Pascal installation lock up (both these are fine on the AT and a wide variety of other kit, such as the TI-3100). It does however, run APL with no problems!

The second thing is that it is rather fast. OK the hard disk is from the XT, not the AT, but we floppy hacks find anything impressive so no quibbles there. Anyway, APL doesn't use the disks a great deal once you have loaded up. Basically it benchmarks a shade faster than a standard AT.

The Enhanced Keyboard

I really don't think I am going to like this. In fact I am on the lookout for some muggins with an old-style AT keyboard who would like an 'upgrade'! It has a dedicated cursor pad, and 12 function keys along the top of the keyboard. <Control> has moved down by the spacebar, and the backslash key is back down between <Z> and <shift> (or is it??). The <*> has sloped off to the extreme right, and there is one extra key (top left) with the back-quote on it. Sounds acceptable enough, and for a touch-typing secretary it probably is . . . however:

1. Most current software assumes it is easy to get at <shift-FN> and at <Ctrl-xxx>. These become two handed 'hunt and peck'.
2. MS Word treats the backslash as a dead key. This makes it impossible to access files on sub-directories!

3. In APL the mappings are pretty well all over the place. Again the backslash is dead, but at least you can redefine things to get it back on that new key top left. However there is worse . . . by default the numeric pad moves the cursor (fair enough) but the 'dedicated' cursor keys produce numbers!! At one point I managed to poke Num Lock to get everything giving numbers . . . it varies depending on APL/Text/Union. I'm afraid I havn't tested all the possible permutations.

In short, from an APL development point of view, this keyboard is a lemon. You should definitely avoid it until a reasonable base of software (APL included) explicitly supports it. In other words vote with your feet in favour of something which runs the current software base acceptably, and don't accept the word of your hardware people when they tell you that the new layout is the best thing since grip-top socks.

ForteGraph

You will probably have to look for this under the DCA 'IrmaGraph' label; mine says IRMA in the documentation and Forte on the board! It is quite expensive, and does require the enhanced display, but it really does do the job. What you get is an IBM 3179-G look-alike (32-line graphics done locally) with 3279-like keyboard mapping. As a mainframe terminal it is very fast (actually rather faster than the 3179-G) and the resolution is acceptable (although not quite as good as the real thing).

I have been using it as my day-to-day mainframe workstation for a couple of weeks now, and can report only the following list of minor problems:

- it has the occasional lapse when attached to 3174 control units. I don't pretend to understand the details, but every so often it locked up when you left your finger on <delete>. The problem has since gone away.
- the supplied keyboard mapping is from the US 3279 to the US PC. This causes endless hours of fun, as the APL symbols follow their non-APL associates (e.g. 'GoTo' stays with the curly bracket and ends up to the left of the '1' on the top row!!)
- one or two characters occasionally acquire random embellishments, such as extra tails or overbars. A power-off reboot usually puts things right.
- APL gets to the keyboard before IRMA, so you can't use <Shift-shift> to toggle from PC APL to VS APL and back. I have given up running IRMA resident for this reason.

The keyboard mapping is only a minor nuisance, as you can use a supplied utility to put pretty well any key anywhere. The only limitation is that you are stuck with the APL-to-nonAPL associations; for example I duplicated <shift-K> on the lower-case PC quote. This is very nice when you have APL on, but rather a pain when it is off! This apart, some of the nice things you can do are:

- Pf-1 up to Pf-10 can be put on F1 to F10. Unfortunately you can't do Pf-11 and 12, as Forte refuses to hear of F11 and F12 on the enhanced keyboard! I put them on <shift-F1/F2>.
- You can set PgUp and PgDn to double Pf-7/8, and End to double Pf-3. TSO hacks will see the point!
- My AP-126 editor uses Pf-9 for 'End and save'; I doubled this on <Ctrl-E> so I can use my APL*PLUS/PC reflexes on the mainframe.

I have also moved the square brackets to the PC keys where they are, leaving the shifted curly ones for the left and right arrows. I flirted briefly with '()' on shift '90', but my reflexes got the better of me, and they are back where they were!

Of course once you are in APL*PLUS/PC you can put anything anywhere, so whatever you settle on for your mainframe layout you can quickly match on the PC. If you want to save yourself the trouble of getting the basics right, send me a formatted floppy, and you can have copies of the 'GRAPH.DAT' for Forte and INI0 for APL*PLUS/PC.

Summary

The XT-286 is a bit of a wolf in sheep's clothing; once having disposed of the enhanced keyboard I quite got to like it. ForteGraph has done all I expected of it, and apart from some very minor niggles it provides a fully functional APL workstation for mainframe or PC use. The all-up cost (including software) was a shade under £5,000.

Hardware Review
Hercules-Plus Monochrome Graphics Card

by Adrian Smith

Introduction

The Hercules Graphics card has been the defacto standard in monochrome PC graphics for as long as anyone can remember. It exploits the excellent resolution of the monochrome PC display and allows applications to access individual pixels, rather than just the entire 14 x 9 character cell.

Of course these days you can get cloned versions of practically everything, and 'Hercules compatible' cards are well under £100 in the computing magazines. Perhaps it is with this in mind that the Hercules Plus card was developed; it offers everything that the old card had, with one very significant addition – RAMFONT. Just what is a RAMFONT, and why is it significant? I wasn't quite sure, until I'd plugged in the beast and turned the power back on: things cleared up quite quickly after that!

Installation

This was straightforward in the extreme. The card was well-packaged, and the instructions very clear. Basically if you just had a single monochrome screen, and were running the printer off the mono adaptor, you could simply swap the cards. If you wanted to install the proprietory print driver to get graphics screen copies, you needed an extra line in the AUTOEXEC.BAT, but that was all there is to it.

The software comes on two (unprotected) disks, and has built-in drivers for Lotus-123, MS WORD, and Framework. Again, you can copy the relevant bits to your working disks. If you take the same attitude to PC software as I do, you will treat the instructions as a refuge of last resort; your first action will be to try something like 'RAMFONT MEDIEVAL.FNT' just to see what happens.

The effect is rapid and startling; type DIR and you get a disk directory in the most gorgeous medieval script you could imagine. Type APL and your workspaces take on the aspect of the Book of Kells . . . except of course for all those damned symbols!! My three-year old was quite upset when Rho turned into a square-root, and Iota into a passable imitation of a gallows. As you might expect, the Hercules corporation had catered for practically every taste in the book, except APL. Bother.

Getting to Know RAMFONT

Of course the ultimate cop-out would have been to plug in the APL ROM; faced with the prospect of re-inventing the entire APL symbol set from scratch, I very nearly gave in and did just that. However, one could make a start somewhere, and there isn't much to an Iota after all! A quick dip in the manual laid bare the secrets of FONTMAN, and a few trials in DOS suggested SANSERIF as the best place to start, so off we went.

FONTMAN has a lot in common with the GDDM Image Symbol Editor, but is rather easier to use. Basically you can page through the characters with '+' and '–', and use F5 and F6 to turn pixels on and off on a 14 by 8 grid. There are various obscure commands for shifting things up and down, and copying symbols from one character to another. The syntax of these makes EDLIN look sensible; fortunately you don't need them very often.

A nice feature is the option of replacing the 'Help' menu with a display of the entire ASCII set (as redefined) life-size. I found this essential while I was getting some feel for the appearance of the characters I was designing; the alternative of backing off and peering at the screen through half-closed eyes is much less satisfactory!

Iota and Times were easy meat; Rho turned out to be a bit of a pig, and kept me going for close on half-an-hour. I began to suspect that the manual had not been understating the truth when it claimed "Designing legible and attractive characters from a matrix of rectangular dots is a skill that will take time and patience to master." And so to bed.

The following morning, I was back at my post, hacking out APL on an AT fitted with the Enhanced Graphics Adaptor. Of course I was aware that this too had a soft character set, and the thought began to form that APLCHAR.COM must somewhere define the characters. Why not drag it into APL and do a bit of reverse engineering? Sure enough it turns out that a bit of base-2 encoding on a 256 by 14 reshape of the 44 drop of said file yields a 256 by 14 by 8 bit array defining the entire character set, APL symbols included.

These are not quite appropriate for Hercules use, as you really need a 14 by 9 grid for the mono adaptor. However they are very much better than nothing, and with a couple of hours' work I can now claim to have a very decent looking Sanserif APL font which is a pleasure to use. By the time you read this, I should have lodged a copy with the Software Exchange; I might also send a copy to Hercules, in the hope that they might include it in future releases.

Where Next?

Microsoft Word has long been a favourite piece of software; however even on a fast PC/AT it is very slow when scrolling text in graphics mode. So much so that I never use the WYSIWYG option, preferring to run in standard text mode at a reasonable speed. Now I can get the italic bits in italic, the bold bits in bold, and the APL bits in APL, all on the screen at the same time, and all at the same speed as standard text! In fact my plain old PC is about 50% faster than our office AT, which is gratifying, to say the least.

From an APL point of view, there are lots of potential applications where a soft character set could be a real bonus. For some time, I have been working on a factory information system, which uses 'plant mimics' to help in the display of data. By re-defining bits of ⎕AV as valves, taps, and so on, you can do the whole thing in text mode, and consequently get it on the screen in a tiny fraction of the time it would take in graphics. Animation suddenly becomes a real possibility; so far my efforts have been restricted to the use of the cursor keys to control a 4-character mouse (actually 8, for left and right movement), which can be manoeuvred around a simple maze in search of a variety of cheeses!

Summary

Leaving aside the unfortunate omission of an APL font, it is hard to fault anything about this card. It even seems to speed up ordinary screen handling a tiny bit; I'm sure that Iota-9999 goes past faster than it did before. RAMFONT is a major step forward, in that it finally frees us from chip-swopping, and lets us run APL and (say) Word on the same machine without: (a) funny APL symbols, or (b) paragraph markers which come out as ù's. We can even stop subjecting DOS to 5-pointed stars!!

To sum up . . . . if you have a monochrome PC and £200 to spend (or that you can con your employer into spending) the Hercules Plus card represents excellent value for money.

Prize Draw Results

by Christine McCree

Everyone who replied with a completed questionnaire and supplied their name and address was entered into a prize draw.

The winner of the first prize was:
J. Ranta of the Confederation of Finnish Industries.

The winner of the second prize was:
G. W. Wood of the London Life Association Ltd.

The winner of the third prize was:
D. Williams of Imperial Tobacco Ltd.

If you won a prize and have not received it yet please contact the Treasurer – there may be some difficulty which you can help to solve.

APL Product Guide

Compiled by Cathy Dargue

VECTOR's exclusive APL Product Guide aims to provide readers with useful information about sources of APL hardware, software and services. We welcome any comments readers may have on its usefulness and any suggestions for improvements.

We do depend on the alacrity of suppliers to keep us informed about their products so that we can update the Guide for each issue of VECTOR. Any suppliers who are not included in the Guide should contact me to get their free entry – see address below.

We reserve the right to edit material supplied for reasons of space or to ensure a fair market coverage.

The listings are not restricted to UK companies and international suppliers are welcome to take advantage of these pages. Where no UK distributor has yet been appointed, the vendor should indicate whether this is imminent or whether approaches for representation by existing companies are welcomed.

For convenience to readers, the product list has been divided into the following groups:

- ★ Complete APL Systems (Hardware & Software)
- ★ APL Timesharing Services
- ★ APL Interpreters
- ★ APL Visual Display Units
- ★ APL character set printers
- ★ APL-based packages
- ★ APL Consultancy
- ★ APL Training Courses
- ★ Other services
- ★ Vendor addresses

Every effort has been made to avoid errors in these listings but no responsibility can be taken by the working group for mistakes or omissions.

Note: 'poa' indicates 'price on application'

All contributions to the APL Product Guide should be sent to:

Cathy Dargue
60 Downhall Ley
Buntingford
Herts SG9 9TL
☎0707-325161 ext 2418

COMPLETE APL SYSTEMS

COMPANY	PRODUCT	PRICES £	DETAILS
Analogic	The APL Machine	$60,000+	AP500 array processor, 4 Mb data memory, 80 Mb disk drive.
Dyadic	Dyalog APL Coprocessor	3,500+	32-bit coprocessor board for IBM PC. NS32000 cpu with FPP, up to 4Mb RAM, 16Mb virtual memory. Software includes Unix V.2, Dyalog APL, graphics support, DOS interface. Provides multi-user Unix/DOS environment.
	IBM 6150	15,000+	Multi-user Dyalog APL system with Fast 32-bit RISC processor, FPP, up to 8Mb RAM, 210Mb Disk, 16 users. Interface to SQL, graphics and APL support for standard IBM peripherals.
	Altos 3068	25,000+	Multi-user Dyalog APL system with MC68020 cpu & MC68881 FPP. Also features a LAN which supports IBM PCs as Dyalog APL terminals.
	Sun 3	15,000+	Multi-user Dyalog APL systems which can be configured as a network of workstations and or a traditional time-sharing cpu. With its 25MHZ 68020 cpu, the Sun 3/200 is the fastest APL microcomputer on the market.
Gen. Software	Myriade	poa	TI computer + APL & APL operating system
Inner Product	IBM PC	2,000 –6,000	IBM PCs supplied for turnkey applications
M.B.T.	MBT Series 10	poa	UNIX/68010 based multi-user APL system
	TORCH	poa	68000/Z80 multiprocessor
MetaTechnics	—	poa	Details on application – IBM PC compatible
MicroAPL	Aurora	20,000+	Multi-user APL computer using 68020 CPU. Std. configuration 2Mb RAM, 16 RS232 ports, 68 Mb hard disc, 720K diskette
	SPECTRUM	7,000+	Expandable multi-user APL computer using Motorola 68000. Std. configuration 1 Mb RAM, 12/36 Mb disc, 12 ports.
	Atari 1040ST	799 –999	1 Mb Mono/Colour System, includes 1 Mb disc drive & mains transformer built into Console.

APL TIMESHARING SERVICES

COMPANY	PRODUCT	PRICES £	DETAILS
Boeing	Mainstream APL	poa	Enhanced IBM VS APL (CMS)
Mercia	APL*PLUS	poa	STSC's Mainframe Service – MAILBOX etc.
I.P. Sharp	SHARP APL	poa	International Network application systems and public databases.
Uniware	APL*PLUS	call	STSC's mainframe service

APL INTERPRETERS

COMPANY	PRODUCT	PRICES £	DETAILS
APL Software	Dyalog APL	1,000-8,000	See Dyadic Systems entry
Cocking/Drury	APL*PLUS/PC Rel 6	475	STSC's full featured APL for IBM PC, PC/AT and compatibles
	Upgrade 5 to 6	120	Extension from rel 5 which incorporates 64K object support.
	Upgrade 2,3,4 to 6	220	Extension upgrades to release 6.
	Run-time	poa	Closed version of APL*PLUS/PC which prevents user exposure to APL.
	APL*PLUS UNX	poa	STSC's 2nd generation APL for IBM PC/AT, DEC, AT&T and other Unix computers.
	APL*PLUS VMS	poa	integrated 2nd generation APL for DEC m/cs under VMS
Dyadic	Dyalog APL	795 -10,000	2nd gen. APL for UNIX systems, e.g. IBM 6150, Sun, Vax, NCR, HP9000, AT&T, Altos, Apollo, Whitechapel, Sperry, etc.
Gen. Software	APL*MYRIADE	poa	Runs on Texas Instruments TI990 range.
IBM UK Product Sales	IBM PC APL	poa	Event-handling & APs for full-screen I/O disks, diskettes, asynch. comms.
Inner Product	VIZ::APL	250 -350	8-bit Zilog Z-80 CP/M
	APL*PLUS/PC	600	See under Cocking & Drury
M.B.T.	Dyalog APL	poa	See Dyadic Systems entry
	MBTAPL	poa	Enhanced Dyalog APL for MBT hardware.
	VIZ::APL	poa	Customized for TORCH hardware
Mercia	APL*PLUS/PC Rel 6	450	STSC's full-feature APL for IBM PC, and compatibles. No 64K object size limit.
	Upgrades 2,3 & 4-6	225	
	Upgrades 5 to 6	130	
	APL*PLUS/UNIX	poa	Interpreter for UNIX systems: WICAT, CADMUS, CALLAN, FORTUNE 32:16, HP, 9000/500, OLIVETTI 3B2, SUN etc.
	APL*PLUS/MAC	poa	APL for the Macintosh. Big workspaces, big objects, mouse, icons etc
MetaTechnics	APL*PLUS Rel 6	475	Discount on quantity.
MicroAPL	APL.68000	1,000+	Full implementation with component files, error trapping etc. for SPECTRUM, HP300, SUN, NCR etc.
	QL/APL (keyword)	87	Full keyword APL for QL with many extra features.
	QL/APL (APL chars)	87	VSAPL compatible APL for QL with many extra features.
	APL.68000 for Apple Macintosh	257	Full APL interpreters with support for windows, mouse, graphics etc.
	APL.68000 for Commodore Amiga	200	
	APL.68000 for Atari ST	170	
	APL.68000 for IBM-PC	995	
	APL*PLUS/PC - REL 6	450	
Portable	PortAPL	$195	IBM PC Software
		$275	Mackintosh
		$2,995	DEC VAX
I.P. Sharp	Sharp APL/PCX	2,575	For IBM XT/AT
		1,000+	For IBM mainframes
	Sharp APL/PC	325	For IBM PC or PC/XT

Uniware	APL*PLUS/PC	495	STSC's full feature APL for
	Release 6	call	IBM PC/XT/AT, Compaq, Olivetti
	Release 5 update	call	Extension upgrade from release 5
	Release 4 update	call	Extension upgrade from release 4
	Release 3 update	call	Extension upgrade from release 3
	Run-Time	call	Closed version of APL*PLUS/PC which prevents user exposure to APL
	APL*PLUS/UNX	call	STSC's full feature APL for UNIX based computers.
	PortaAPL	280	PORTABLE SOFTWARE's APL for APPLE MACINTOSH.
		2,995	PORTABLE SOFTWARE's APL for the DEC VAX.

APL VISUAL DISPLAY UNITS

COMPANY	PRODUCT	PRICES £	DETAILS
Dyadic	Lynwood j300	1,560	Monochrome ANSI 3.64 APL vdu, 15-inch high quality screen, Tek graphics, local macro keys.
	Lynwood j500	2,295	Colour ANSI 3.64 APL vdu, 15-inch high quality screen, Tek graphics, local macro keys.
	IBM 3163	791	Low-cost Monochrome APL vdu. Supports downloaded Dyalog APL font.
	IBM 3164	1,093	Low-cost Colour APL vdu. Supports downloaded Dyalog APL font.
Famell	Tandberg TDV 2221	995	Ergonomic design APL terminal, 50-19200 baud, 15" anti-reflex screen, low profile keyboard
	Tandberg TDV 2271	1,195	Combined APL/ANSI ergonomic terminal as above.
Gen. Software	Mellordata Elite 3045A	400	Second-hand
M.B.T.	various		Contact MBT for details
Meta Technics	IBM EGA compatible	299	Emulates EGA & Hercules, Half Card
MicroAPL	Insight VDT-1	795	Inexpensive APL VDU
	Insight GDT-1	1,450	With monochrome graphics
	Concept 201	1,295	APL VDU with 8 page memory
	Concept 201G	1,650	Graphics VDU
Shandell	HDS2010	1,215	ANSI3.64 DEC VT52/100/220 compatible. 15" tilt/swivel screen, low profile keyboard 8 page memory, windows, viewports, 80/132 columns, full overstrike, 2 or 3 comms, ports, 55 PF keys, NVM storage.
	HDS2010G/GX	1495+	As above plus Tektronix 4014, Retrographics VT640/D0640 and Visual 500 compatible. 1024 x 390 or 1024 x 780 resolution.
Tektronix	4114B	13,500+	19" D.V.S.T.:Graphics: 3120 x 4096 displayable; Intelligent: up to 800K memory; APL keyboard (option 4E)
	4125	21,550+	19" 2D colour graphics; Workstation (1280 x 1024);Intelligent: up to 800K memory; APL keyboard (mod AP)
	4128	26,822+	As 4125 plus 3D wireframe

APL PRINTERS

COMPANY	PRODUCT	PRICES £	DETAILS
Datatrade	Datasouth DS180+	1,295	180 cps matrix printer with 4K buffer, 9 x 7 dot matrix and APL option.
	Datasouth DS220	1,695	Letter quality; graphics capability, APL option (both available with IBM Twinex or Coax interface).
Dyadic	IBM 4201 Proprinter	poa	100, 200, 40(nlq) cps, matrix printer, with graphics. Supports downloaded Dyalog APL font.
	Toshiba P351	poa	24 pin high-quality matrix printer 100 cps letter quality, 192 cps draft.
Inner Product	Epson FX80	500	Soft char. set, 160 cps, 80 column
	Anadex 9620	1,150	200 cps., 132 col., tractor feed
	Siemens PT88	620	180 cps., 80 col., silent
	TGC Starwriter	1,180	40 cps., letter quality
M.B.T.	Facit 4565	poa	40 cps letter-quality
	Facit 4510/11/12	poa	Matrix printers
MetaTechnics	Quen-data	295	Low-cost APL Daisy-wheel printer
MicroAPL	Datasouth DS180+	1,295	See Datatrade entry
	Philips GP300	1,924	Matrix printer with letter & draft quality and APL.
	Qume Letterpro20	549	APL/ASCII Daisy-wheel printer

APL PACKAGES

COMPANY	PRODUCT	PRICES £	DETAILS
APL ◇ 385	FSM 385 PC:	50	Screen development
	DRAW 385		Screen design
	DB 385 Mainframe:	125	Relational W.S.
	GEN 385		Utilities
APL Software Ltd	*Mainframe*		
	AFM/AP	11,035	Interprocess Software for VM/CMS & MVS/TSO.
	– Keyed Access	2,650	Component File Management System (VSAPL/APL2)
	– Interactive Link	1,325	
	– Mail Exchange	2,650	
	CALL/AP	4,030	Non-APL program execution (VSAPL/APL2)
	APLPRINT	2,205	Output to high speed line printer or 328x devices (VSAPL/APL2)
	ENHANCED FORMAT	2,205	Extends Format operator to full "Quad-FMT" status (VSAPL/APL2)
	ISP	750	Input and Output Stack Processors for manipulating terminal I/O
	OSP	2,205	with facilities for Error Trapping (VSAPL)
	DISPLAY CAPTURE	poa	Allows terminal output to be collected and held for retrieval by an APL function (APL2)
	UCF	poa	User Communication Facility for data transfer between users (APL2)
	RDS	poa	Relation Data Base System
	PANEL	poa	Fullscreen management system
	PFS	poa	Program File System – APL Systems development aid
	IPLS	poa	Project Management System
	REGGPAK	poa	Regression Analysis Package
	Microcomputer		
	POWERTOOLS	295	Assembler written replacement function for commonly used CPU-consuming APL functions, includes a Forms Processor.
	Microcomputer		
	REGGPAK	poa	Regression Analysis Package
	RDS	990	Relational Database System

Beta-plan	BETA-FONT	poa	Multiple font PC character generator. Dealers required for non-Scandinavian countries.
Boeing	IMPETUS	poa	Hierarchical Planning System
Butel	Merlin	5,000	Mainframe APL spreadsheet runs under VM/CMS, TSO, VSPC
	Merlin/PC	poa	Version for APL*PLUS/PC
Cocking/Drury	*For VSAPL*		
	STSC's SHAREFILE & enhancements to VSAPL	30,000	Component files, quad- functions & nested arrays for IBM VSAPL under VM/CMS & MVS/TSO
	SHAREFILE only	15,000	
	ENHANCEMENTS only	17,000	
	COMPILER	30,000	First APL compiler. Available with APL*PLUS enhancements and Sharefile under VM/CMS & MVS/TSO
	FILEPRINT	8,000	Print APL component files
	FILESORT	8,000	Sort APL component files
	FILECONVERT	8,000	Convert non-APL files to APL files
	FILEMANAGER	8,000	Extends APL primitives to database management
	TOOLS + UTILITIES	8,000	APL Software development tools
	DATAPORT	poa	Powerful Information Centre spreadsheet incorporating data exchange between APL and FOCUS, IFPS, SAS, APL/DI, ADRSII, LOTUS123, VISICALC, MULTIPLAN, DIF files
	For APL2		
	SHAREFILE/AP	15,000	STSC's sharefile for APL2
	FMT	poa	Full featured FMT for APL2
	WSDOC	5,000	
	FILEMANAGER	8,000	
	Microcomputer		
	STATGRAPHICS Rel 2	595	Powerful Statistics and graphics on IBM PC's, PC/AT's and compatibles
	Release 2 update	165	Update from release 1 to release 2
	APL*PLUS PC Tools		
	VOL 1	325	Incl. 327x IRMA support, RAM disk, full screen data entry, menu input, report generation, games.
	VOL 2	125	Incl.file documentor, screen editor, exception handler.
	APL*PLUS PC Fin & Stat. Library	350	Financial & statistical routines
	SPREADSHEET MANAGER	195	APL-based spreadsheet for APL*PLUS/PC. Cell arithmetic; transfers to ASCII, LOTUS
	APL Debugger	95	Debugging tool for APL*PLUS PC
	UNITAB	250	Spreadsheet for APL*PLUS PC
E&S	PROTOPAK consisting of:		Packages for prototyping management information systems -- PC & mainframe
	RMS Modules		Relational databases.
	AMS	250+	Multi-dimensional arrays
	RAMS		Combined RMS & AMS.
	BMS		Dynamic financial modelling & forecasting
	FMS		Full-screen handler for APL*PLUS/PC. (AP 124-based)
	CMS		Communications package.
	SOS	poa	Scheduled ordering and stock control.
FASTCODE	FASTCODE, MONITOR	$239	Assembler written monitor for APL applications in APL*PLUS/PC
Gen. Software	PROPS	500+	Spreadsheet system for Product and/or Project Planning.

H.M.W.	INPUT	poa	Matrix manipulation package for data entry & report generation
	PRINTPAK	poa	Block printing for VM/CMS
	VIEWPAK	poa	AP124 Protocol emulator for IBM/PC
Holtech	CASH	3,500	Accounting package & hotel management system on MicroAPL
		–10,000	SPECTRUM & SAGE CPUs.
Inner Product	Viewcom	150	Control Viewdata from APL
	APL/dBASE II	150	Interface APL with dBase II
	APL/dBASE III	150	Interface APL with dBASE III
	APL/LOTUS	150	Interface APL with Lotus
	APL/WORDSTAR	150	Interface APL with Wordstar
	APL/MULTIPLAN	150	Interface APL with spreadsheet
	CEMAS	3,500	EEC monetary and agrimonetary analysis.
M.B.T.	RHOMBUS	poa	Integrated Office System
	HASLEMERE	poa	Hotel Accounting System
Mercia	STATGRAPHICS 2·1	585	Integrated stat. graphic system for PCs.
	Upgrade to Release 2·1	175	
	EXEC*U*STAT	395	Easy to use Statistics for management.
	APL*PLUS tools		
	VOL 1	225	IBM PC Utilities:IRMA3270 comms, full screen, RAM Disk report generator
	VOL 2	125	File documentation, screen editing. Exception handling.
	FINANCIAL AND STATISTICAL LIB.	325	Financial and Statistical analysis
	APL Spreadsheet Manager	195	APL spreadsheet – links to popular spreadsheet software.
	EXECUCALC	4,000	Mainframe Spreadsheet with VisiCalc and Lotus 1-2-3 functionality requires VSAPL under TSO or VM.
	EXECUPLOT	3,200	Mainframe Graphics display system with VisiPlot functionality requires VSAPL under TSO or VM and GDDM.
	MICROSPAN	250	Comprehensive APL tutor
	LOGOL	poa	Logistics management system for PC, Forecasting, Inventory Control, Scheduling, Distribution, etc
MetaTechnics	MetaScreen	99	Full-screen handler for APL*PLUS/PC, based on VSAPL AP124
	MetaPack	495	Comprehensive utilities package for APL*PLUS/PC. Includes MetaScreen, MetaWS, Browse, Toolbox, Numeric Editor.
	APL-IEEE488	99	Controls IEEE488/GPIB Bus from APL*PLUS/PC.
	PLOT/PC	99	2D & 3D Graphics package. Includes interactive diagram Editors.
	Browse	99	Scrolling of DOS files, large APL variables.
	ADAPTA DLS	poa	Production & purchasing scheduling for process manufacturing.
	ADAPTA MSP	poa	Job-shop loading & scheduling for multi-stage production.
MicroAPL	MicroTASK	250	Product development aids
	MicroFILE	250	File utilities and database
	MicroPLOT	250	Graphics for HP plotters etc
	MicroLINK	250	General device communications
	MicroEDIT	250	Full screen APL editor
	MicroFORM	250	Full screen forms design
	MicroSPAN	250	Comprehensive APL tutor
	MicroGRID	poa	Ethernet & other networking
	APLCALC	400	APL spreadsheet system
	MicroPLOT/PC	250	For APL*PLUS/PC product
	MicroSPAN/PC	250	For APL*PLUS/PC product
	PC TOOLS Vol 1	295	
	STATGRAPHICS Ref 1	495	
	STATGRAPHICS Ref 2	535	

Parallax	ExecuCalc	$5,000	Mainframe-based electronic spreadsheet for VM/CMS & MVS/TSO with links to micro products.
	ExecuPlot	$5,000	Mainframe-based colour graphics with micro links.
I.P. Sharp	ACT	poa	Actuarial system
	APS	poa	Financial Modelling
	BOXJENKINS	poa	Forecasting technique
	CONSOL	poa	Financial Consolidation
	COURSE	poa	APL Instruction
	EASY	poa	Econometric Modelling
	FASTNET	poa	Project Management
	GLOBAL LIMITS	poa	Exposure management for banks
	MABRA	poa	Record maintenance/reporting
	MAGIC	poa	Time series analysis/reporting
	MAGICSTORE	poa	N-dimensional database system
	MAILBOX	poa	Electronic Mail
	MICROCOM	poa	Mainframe to micro link
	SAGA	poa	General graphics, most devices
	SIFT	poa	Forecasting system
	SNAP	poa	Project management
	SUPERPLOT	poa	Business graphics
	VIEWPOINT	poa	4GL – Info centre product
	XTABS	poa	Survey Analysis
Sugar Mill	Stat 1	$129.95	Statistical toolbox, menu driven
Uniware	*Mainframe*		
	STSC's ENHANCEMENTS	10,715	Quad-functions & nested arrays for IBM VSAPL under VM/CMS and MVS/TSO
	STSC's SHAREFILE	10,715	Component files for IBM VSAPL under VM/CMS and MVS/TSO and for IBM APL2
	PROGRAMMER TOOLS & UTILITIES	5,715	
	FILEPRINT	5,715	
	FILESORT	5,715	
	FILECONVERT	5,715	
	FILEMANAGER (EMMA)	5,715	STSC's database package.
	APL*PLUS COMPILER	21,430	First APL compiler. Complements APL*PLUS enhancements and Sharefile under VM/CMS and MVS/TSO.
	EXECUCALC	3,995	Mainframe spreadsheet compatible with VISICALC and part of LOTUS 1-2-3 under VSAPL (VM or TSO).
	Microcomputer		
	STATGRAPHICS	725	Statistics & Graphics for PCs.
	STATGRAPHICS FCA	140	An add-on module to STATGRAPHICS: Factorial Correspondence Analysis.
	APL*PLUS/PC TOOLS		
	VOL1	325	Incl. 327 x IRMA support, RAM disk, full screen data entry, menu input, report generation, games.
	VOL2	125	Incl. File documentor, screen editor, exception handling.
	SPREADSHEET MNGR	250	APL spreadsheet with built-in ASCII, LOTUS and SYMPHONY interfaces.
	APL*PLUS/PC FIN. & STAT. LIBRARY	350	Collection of financial and statistical utilities.
	POCKET APL	140	Smaller version of APL*PLUS/PC.
	UNIASM	275	Collection of assembler routines for APL*PLUS/PC users.
	UNITAB[tm]	240	APL*PLUS/PC spreadsheet-like data entry and validation system.

The APL DEBUGGER™	105	First released APL*PLUS/PC debugger.
OVERLAYS	250	Fast assembler routines to handle overlays in APL*PLUS/PC.
R:BRIDGE	380	Interface between APL*PLUS/PC & R:BASE 5000.
DMA	380	A version of EMMA (APL database manager) for APL*PLUS/PC users.
APL2C	295	Interface between APL*PLUS/PC and DATALIGHT C language.
ADAPTA/DLS	33,333	Production & purchasing scheduling for process manufacturing.

APL CONSULTANCY

(prices quoted are per day unless otherwise marked)

COMPANY	PRODUCT	PRICES £	DETAILS
APL Consultancy	Consultancy	poa	Project management, financial applications, relational databases. Difficult problems solved. Management consultancy. Links to non-APL systems. From consultant level to managing consultant. Documentation a speciality.
APL Software Technology	Consultancy	poa	Technical & business systems, micros, networking & communications a speciality.
Boeing	Consultancy	poa	
Camacho	Consultancy	poa	Specialising in programming & manual writing.
Chapman	Consultancy	150-300	24-hour programmer: APL, C, assembler, graphics; PC, mini, mainframe, network.
Cocking/Drury	Consultancy	120-150 140-200 185-300 275-400	Junior consultant Consultant Senior consultant Managing consultant
Delphi	Consultancy	poa	Specialising in management reporting systems and APL on microcomputers.
Dyadic	Consultancy	poa	APL system design, consultancy, programming & training for Dyalog APL, VSAPL, APL*PLUS, IPSA APL etc.
E&S	Consultancy	150 –250	System prototyping: all types of information system.
FASTCODE	Consultancy	poa	Specialise in improving performance of APL applications on micros & mainframes.
Gen. Software	Consultancy	100+	
H.M.W.	Consultancy	100-250	System design consultancy, programming.
Inner Product	Consultancy	200	On-site micro-mainframe APL, PC/DOS & Assembler
Lloyd Savage	Consultancy	poa	Decision support, particularly specialising in Sales & Marketing systems.
M.B.T.	Consultancy	poa	
Mercia	Consultancy	poa	APL*PLUS & VSAPL consultancy.
MetaTechnics	Consultancy	poa	Management Information & Production. Engineering APL – C/Assembler custom programming
MicroAPL	Consultancy	poa	Technical & applications consultancy.

M.T.I.	Consultancy	poa	Specialise in Maintenance and development of existing APL systems
Parallax	Consultancy	$750	Introductory APL, APL for End-user & Advanced Topics in APL
QB On-Line	Consultancy	200	Specialising in Banking, Financial & Planning Systems.
Rochester Group	Consultancy	poa	Specialise in MIS using Sharp APL
I.P. Sharp	Consultancy	poa	Consultancy & support service world-wide.
Peter Cyriax Systems	Consultancy	100-150 120-200 160-300	Junior Consultant Consultant Senior Consultant
Uniware	Consultancy	call	Junior to managing consultancy in APL.

OTHER PRODUCTS

COMPANY	PRODUCT	PRICES £	DETAILS
APL People	Employment Agency	poa	Permanent employees placed at all levels. Contractors supplied for short/long-term projects, supervised.
Mercia	ALLCARD	495+	Memory management unit, allowing 952K under DOS – extra 312K APL*PLUS/PC workspace.
	MULTI-APL	195+	Multi-task/Multi-user/Network APL*PLUS/PC with file locking, etc.
I.P. Sharp	Productivity Tools	poa	Utilities for systems, operations, administration & analysts; auxiliary processors, comms software, international network.
	Databases	poa	Financial, aviation, energy and socioeconomic.

VENDOR ADDRESSES

COMPANY	CONTACT	ADDRESS & TELEPHONE No.
Analogic Corporation	Denise Favorat	8 Centennial Drive, Centennial Industrial Park, Peabody, Mass. U.S.A. 01961 ☎ 617-246-0300
APL 385	Adrian Smith	Brook House, Gilling East, York. ☎ 04393-385
APL Consulting	Jill Moss	17 Barton Street, Bath, Avon BA1 1HQ ☎ 0225 62602
APL People	Valerie Lusmore	17 Barton Street, Bath, Avon. ☎ 0225-62602
APL Software Ltd	Philip Goacher	27 Downs Way, Epsom, Surrey KT18 5LU ☎ 03727-21282 17 Barton Street, Bath, Avon BA1 1HQ ☎ 0225-62602
APL Software Technology (UK) Limited	John Hagger Per Hultin	14 Rosewood Avenue, Alveston, Bristol BS12 2PP ☎ 0454 415737 46 Vicarage Road, South Benfleet, Essex SS7 1PB ☎ 03745 50501
Beta-plan APS	Kim Andreasen	Stengrade 75, DK-3000 Helsingor, Denmark. ☎ 45 2 21 48 48
Boeing Computer	Suzanne Hunt	P.O. Box 747, 364 Euston Road, London NW1 3BQ
Butel Technology Ltd.	Mike Munro	Butel House, 3 Great West Rd., London W4 5QJ ☎ 01-995-1433
Anthony Camacho		2 Blenheim Road, St. Albans, Herts AL1 4NR. ☎ St. Albans 60130
Paul Chapman		18, Trevelyan Road, London, SW17 9LN ☎ 01-767 4254
Cocking & Drury Ltd.	Romilly Cocking	16 Berkeley Street, London W1X 5AE. ☎ 01-493 6172 155 Friar Street Reading RG1 1HE. ☎ 0734-588835
Datatrade Ltd.	Tony Checksfield	38 Billing Road, Northampton, NN1 5DQ. ☎ 0604-22289
Delphi Consultation Ltd.	David Crossley	Church Green House, Stanford-in-the-Vale, Oxon SN7 8LQ. ☎ 03677-384
Dyadic Systems Ltd.	Peter Donnelly	Park House, The High Street, Alton, Hampshire. ☎ 0420-87024

E & S Associates	Frank Evans	19 Homesdale Road, Orpington, Kent BR5 1JS. ☎ 0689-24741
Farnell International Instruments Ltd.	R. Fairbairn or Roger Attard	Jubilee House, Sandbeck Way, Wetherby, W. Yorks. ☎ 0937-61961 Davenport House, Bowers Way, Harpenden, Herts. ☎ 05827-69071
FASTCODE	Andrew Dickey	P.O. Box 281, Croton-on-Hudson, New York 10520, U.S.A. ☎(914) 271-3200
General Software Ltd.	M.E. Martin	22 Russell Road, Northolt, Middx. UB5 4QS. ☎ 01-864 9537
H.M.W. Programming Consultants Ltd.	Ken Jackson	142 Feltham Hill Rd, Ashford, Middx. TW15 1HN. ☎ 07842-41232
IBM UK Ltd	Chris Sell	PO Box 32, Alencon Link, Basingstoke, Hants. RG21 1EJ. ☎ 0256-56144
Inner Product Ltd.	Dominic Murphy	Eagle House, 73 Clapham Common Southside, London SW4 9DG. ☎ 01-673 3354
Lloyd Savage Ltd	Philip Johnson	Cambridge House, Oxford Road, Uxbridge. Middx, UB8 2UD. ☎ 0895-59826
Mercia Software Ltd.	Gareth Brentnall Barrie Webster	Aston Science Park, Love Lane, Birmingham B7 4BJ. ☎ 021-359 5096
MetaTechnics Systems Ltd	John Stenbridge David Toop	Unit 216, 62 Tritton Road, London, SE21 8DE. ☎ 01-670 7959
MicroAPL Ltd.	Bernadette Leverton	Unit 1F, Tideway Industrial Estate, Kirtling St, London. SW8 5BP ☎ 01-622 0395
Mine of Information	Richard Ross-Langley	PO Box 1000, St. Albans, Herts AL3 6NE. ☎ 0727 52801
Modern Business Technology Ltd. (MBT)	Michael Branson	P.O. Box 87, Guildford, Surrey GU4 8BB ☎ 04868-23956
M.T.I.	Ray Cannon	7 Pine Wood, Sunbury-on-Thames, Middx. TW16 6SH ☎ 09327 80848
Parallax Systems Inc.	Kevin Weaver	60 West 9th Street, New York, New York 10011, U.S.A. ☎ 212-475-4001
Peter Cyriax Systems	Peter Cyriax	213 Goldhurst Terrace, London NW6 3ER ☎ 01-624 7013 (Answerphone) 0860-377963 (Mobile)
Portable Software	Richard Smith	60 Aberdeen Ave, Cambridge, Mass. U.S.A. 02138. ☎ 617-547-2918
QB On-Line Systems	Philip Bulmer	5 Surrey House, Portsmouth Rd Camberley, Surrey, GU15 1LB. ☎ 0276-20789
The Rochester Group	Robert Pullman	164 Pinnacle Rd., Rochester NY 14620 ☎716-461-3169
Shandell Systems Ltd.	Maurice Shanahan	12 High Street, Chalfont St. Giles, Bucks HP8 4QA. ☎ 02407-2027
I.P. Sharp Associates Ltd.	David Weatherby	10 Dean Farrar Street, London SW1. ☎ 01-222 7033
Sugar Mill Software Corp.	Lawrence H. Nitz	1180 Kika Place, Kailua, Hawaii 96734 ☎ (808) 261-7536
Tektronix UK Ltd.	Paul Morgan	Fourth Avenue, Globe Park, Marlow, Bucks SL7 1YD. ☎ 06284-6000
Uniware	Eric Lescasse	15 Rue Erlanger – 75016 Paris – France ☎ (1) 45-27-20-61 Telex: 648348F UNIWARE

APL BOOKLIST

(In author order)

Title	Price £
Sharp APL Reference Manual, P Berry	10.50
Star Map, P Berry & J Thorstetsen	6.00
APL and Insight, P Berry and G Bartoli	4.50
APL 86 Tutorials, A Camacho (members £9.30)	12.00
A Source Book in APL, A Falkoff & K Iverson	10.00
FinnAPL Idiom Library	11.20
Application Systems in APL, Gibson Levine Metzger	30.00
APL:An Interactive Approach, Gilman & Rose	26.50
Solutions to Algebra, J Iverson	3.00
A Dictionary of APL, K Iverson	2.50
A Concise Dictionary of APL, K Iverson	2.00
Algebra: an Algorithmic Treatment, K Iverson	22.50
APL in Exposition, K Iverson	3.00
Applied Mathematics for Programmers, K Iverson	8.00
Elementary Analysis, K Iverson	8.00
Introduction to APL for Scientists & Engineers, K Iverson	3.00
Introducing APL to Teachers, K Iverson	3.00
Mathematics and Programming, K Iverson	8.00
APL Toolkit (CIPS APL SIG), R Levine	4.50
Reliable Software Through Composite Design, Myers	15.00
APL:An Introduction, H Peelle	POA
APL in Practice, Rose/STSC	40.00
Sharp APL Users Meeting Procs 1982 Vol 2	8.50
APL:Design Handbook for Commercial Systems, A. Smith	13.10
Resistive Circuit Theory, R Spence	20.00
Whizzbangs Volume II, R Sykes	14.50
Whizzbangs Volume I, R Sykes	14.50
An APL Notebook, Barrie Wetherill (when available)	1.90
APL Idiom List (Yale University)	2.00
APL 86 Conference Procs, D Ziemann (members £13.30)	17.30
APL Business Technology '83 Proceedings	11.20
APL Quote-quad the Early Years	32.00
APL Trivia Cards (per set)	4.50
Special APL 86 Wallet Offer (see Vector 3.2 page 92)	22.50

Please order direct from

WARWICK UNIVERSITY BOOKSHOP

Arts Centre, University of Warwick, Coventry CV4 7AL, Tel 0203-523388

Access, American Express and Barclaycard accepted. Order by telephone.

Postage at cost on credit card orders.

RECENT MEETINGS

This section of VECTOR is intended to document the seminars delivered at recent meetings of the Association, particularly for the benefit of those members based away from London who often find it hard to find the time to attend. It also covers other selected events which are likely to be of interest to the wider APL community.

We are dependent on the willingness of speakers to provide us with a written version of their talk, and we would remind them that "a picture's worth a thousand words". Copies of slides and transparencies will enhance their articles.

The Activities Officer (see inside back cover) will respond enthusiastically to offers from individuals who wish to contribute seminars and supporting papers.

Introductory Notes

by Adrian Smith

Three meetings are covered in this section of VECTOR. The first is really nothing to do with APL, but seeing as I was there I thought that I might as well write it up! It was the Yorkshire and Humberside OR Group's special national event on successful applications of Expert Systems.

The second is a bona-fide BAA meeting, on the subject of Quality Control in APL applications. The talks were by Chris Campen, of the other BAA (the British Airports Authority), and by Linda Kindred of Wellcome. A summary of Chris's paper is followed by my own notes on Linda's talk, and Anthony Camacho's on the panel discussion which followed.

Finally we have some notes on the March meeting on Graphics, together with Dave Preedy's paper on 'Graphics in the Boardroom' (See General Papers). This covered a lot of fascinating ground (Bach Canons in APL on Amigas!) with enough technical hitches to keep all the speakers on their toes!!

Successful Expert Systems
Royal Station Hotel, York
February 17th 1987

Introduction

This was a most stimulating and enjoyable day, in spite of the fact that APL wasn't mentioned once! As you will see, it gives us APLer's a good deal of hope, in spite of that sad omission. In particular it was a notable feature of the day that the industrialists was singing the praises of 'shells' and 'environments' whereas the academics were pouring cold water, and sounding some very cogent warnings.

My overall impression was that it is fine to go out and buy Crystal, XI-Plus, or whatever as a way of getting started. Before long you will either hit the limits of what the shell designer thought of, or will end up with 90% of your rules essentially fudging round the constraints imposed. In the long term, what you really need is a plug-in 'inference engine' which will fit naturally into your existing software, be it database, spreadsheet, or APL model.

Enough of personal prejudice . . . here is what the speakers actually said!

Expert Systems in British Gas

Tony Haws

Tony began by summarizing the current position in British Gas. He felt that the key areas to watch were: strategy for implementation; effective knowledge acquisition; interfaces to existing systems; the choice of delivery vehicle. On the first point, he noted that all but one of his 6 examples were drawn from a single enthusiastic expert (3 of these had since joined the ES group). They were now technically and financially successful, but had yet to achieve 'commercial' success in the sense of influencing the core of the company operations.

He now described six successful systems in detail:

- ENRICH. In the entire NW region, there were but 2 experts on noisy central heating systems!! Sometimes it took 6 months for a customer to get attention, and the experts were usually reluctant to be called out to such 'trivial' matters. The system started off on ICL Advisor at local depots. At least this cut it down to 2-3 visits by the fitter. Now Nth Thames are re-doing it on Crystal on hand-held micros for on-site use.
- Herbicide Advisor. This time only one expert, who got pregnant! Now a national system for dealing with nasties in the ponds round gasometers.
- The Stretford Process (coal gasification). All these are sold abroad . . . they often go wrong and the phone calls cost the users lots of money! Now sold with the expert diagnosis as part of the package. (ESP Advisor on IBM compatibles.)
- CORPS. Corrosion prediction in drill wells and pipes. Does a day's work in half an hour (See VECTOR 1.3 for a paper on PATTIE . . . . Ed)
- QUAD. ICL have a 4GL called Querymaster; casual users find it hard to frame queries, so Quad helps them.
- Physical Risk Analysis. Developed as part of contingency planning to identify the risks to computers (and subsequently to more general installations) from fire, flood, terrorist attack etc. None of the dozen or so experts could co-ordinate or structure their expertise; the ES approach got them together.

In all about 20 similar systems are under way, all of the same general type. Two classes in particular look hopeful:

- 'Real' ES, involving high levels of skill in a small and well bounded domains. ENRICH is the archetype.
- Simple, but more general, such as the risk analysis system. The savings multiply up rapidly when you apply the same system in many places.

In both types, the design is evolutionary, and will help the expert himself to generate more knowledge.

How to deliver? Software is the key!! Shells are really for beginners, the experts graduate to 'environments'; typically purpose built Lisp machines costing $15,000 upwards. Of course you can develop on one machine (compilation really hammers the system) and implement on another.

Knowledge acquisition . . . in many of the examples, the expert wrote the system himself (often in his spare time). This is not a long-term option, and apart from very basic Market Research techniques there is little experience here. Some pitfalls to watch for: rare events which the expert forgets; things which 'everyone knows'; missing links in the logic.

Selling the ideas in the organisation . . . and convincing others to spend the money. Next week they are holding a seminar for 80-odd senior managers with demos of many of the above. With luck it should give an entry into the commercial heart of the business where the real money is.

Experience of Expert Systems in ICI

Brian Hobson

A total of over 40 systems are in progress or in use, ranging through: technical sales; engineering design; diagnosis; business planning; operator guidance. They are all written using 'ISI Savoir' in which ICI have a 50% stake. After some early failures, they have achieved real commercial success from 1984 onwards. Some typical systems were:

- Wheat Counsellor. This dates back to 1984, and has changed little since. It uses ICI's private viewdata network to access current data on wheat prices, local weather, soil conditions etc. The farmer is then quizzed about his 'field' and is informed of the disease risks and a fungicide is suggested (not always ICI's!!) Some indication of cost-benefit is also given. It was initially successful, but has rather stagnated as farmers have moved to Amstrads. It is also limited to wheat (some 1200 rules); barley, oats and so on must be included if it is to attain 'critical mass'.
- SYSLAG (lagging on pipes). Here the expert spent his last two years before retirement learning the shell and getting the knowledge into the system. Because there is a lot of maths involved, over 50% of the code is actually in Pascal exits, rather than Savoir itself. The inference is all done using Bayesian probabilities; in many cases the maths is dubious (e.g. where the options are highly correlated) but with suitable fudging of prior probabilities it appears to give the right answers! It is used 15 – 20 times per month, took 6 – 9 months to develop (mostly free), and runs in 150K on an IBM PC.
- Coater Plant Diagnosis. Latex coating on plastic flying by at 60mph! One expert, who fancied Saturday mornings in bed, supported by the plant management, who were aware of the drop in productivity whenever 'Walter' wasn't around. Why not write a handbook? The ES is better for training, forces the operators to explore more options, and keeps a log for the next shift to see. This one took 3 man-months to do, and has definitely increased plant efficiency.
- Operator guidance. This schedules paint manufacture in a plant which needs running 'gently' and with some thought. 30 tons of 'White with a hint of Black' could be a problem! Here the knowledge acquisition was a problem, as the various experts shouted at one another across the conference table. At least the ES is consistent! It is attached to ICI's 'Auditor' plant monitoring package to get its data; eventually they hope to feed back directly into the plant control systems.

In summary 'Small is Beautiful' in all the really successful cases. Wheat counsellor was an early breakthrough, but has stagnated as it cannot achieve wide enough coverage of a large domain. Of the 42 systems in progress, 5 have paid off, 10 are deemed to be successes, 5 are dead ducks, and the rest must wait and see.

Criteria for Successful Expert Systems

Bob O'Keefe (University of Kent)

This was a talk with a difference! Bob very briefly outlined a system for screening graduate applicants in a major accountancy firm. Not surprisingly the details were confidential . . .

the criteria for screening 2,000 applicants for 200 jobs are not likely to become public knowledge. Instead Bob concentrated on the reasons for success, and gave us his (deliberately controversial) list of the most common reasons for failure.

First the reasons for success:

- A willing and enthusiastic 'knowledge Czar' with a very definite 'I am right' attitude. His considerable input of time was worth more than the cost of consultancy, hardware and software put together.
- Flexible software. ESP Advisor lets you out into Prolog when the going gets tough. The system could not have been done quickly and cost-effectively if they had been 'locked in' to a shell. They needed 'a programming tool, not an imposed paradigm'. Typical symptoms of the latter are systems with 3,000 rules; 2,900 of these are probably getting round the shell!
- A simple approach to uncertainty. Bayesian inference really is an absolute nightmare! They used simple 'what's A worth against B' comparisons to establish a set of weights; these were then simply combined to give an overall score.
- Validation. This took more effort (60% of the total time) than knowledge acquisition and building. Random batches of 20 – 30 applicants were run through the system; when these all checked OK, they looked for the real extreme cases and pushed it to the edge of past experience.
- User interface. It takes a lot of care to get the questions in a sensible order! Users are bothered by some 'Y/N', some 'Enter number (1 – 3)' and so on; everything was set up as a menu option to avoid this inconsistency.
- The needs of the user came first. It is quicker and simpler to use the system than to do it by hand. It guides them through the task – maybe this should have been put in a handbook 20 years ago; these days it is easier to put it in an ES.
- the impact on the organisation were built in. The system gives a better and more consistent recruitment policy, and the reponsibility can be taken lower down the structure.

Some elements which have lead to failure in other systems:

- the 'consultative mode'. The system asks all the questions, and does all the reasoning. You sit there and say 'Y', or '0.35' and so on. Attempts to use ES in financial planning have been a disaster for this reason ... here the use of 'critique mode' works much better. The expert has a first go at a plan and the system criticises it. Of course many shells embed 'consult mode' in the shell structure!
- Imposed methods of handling uncertainty. We just don't know how to do this, and should stop pretending that Bayes and Fuzzy logic are any sort of a solution.
- Shells. If you have a hammer, you see every problem as a nail. If you have a Backward-chaining Bayesian Inference Engine . . . . . In the US they are moving back to ES environments (e.g. CitiBank have spent $10M on Lisp machines) rather than using shells.

- Poor validation. 'It never got implemented, but it's a useful training tool'. This translates as 'we couldn't check its answers'. If you haven't got documented case studies, or an independent expert, you are in big trouble.

In general, ES are moving fast towards being 'bottom line' benefits, rather than just nice toys. Either someone is making better decisions, or lots more people are making good decisions. This is all well and good, but has anyone a clue as to how we should measure the results quantitatively? The DSS people have been trying for years; will the ES fraternity do any better?!

Quality Control in Application Development 20th February 1987 at the Royal Over-Seas League

Introduction

This was one of the most interesting and valuable meetings I can remember for a long time. The idea of getting a 'view from the outside' was a brave one, and (in my view at least) came off very well indeed.

Chris Campen gave us his view, from the position of Information Systems Policy Manager at the British Airports Authority. Linda Kindred followed up with the perspective of someone whose job it is to maintain (rather than create) systems in a wide range of languages.

A Formal Approach to Implementation

Chris Campen (British Airports Authority)

Summarised by Anthony Camacho

Chris defined the quality of an information system as fitness for its purpose. To design a fit system large jobs are divided into many small tasks. Quality control includes standards and reviews of work in stages. Each small task must have well defined objectives and targets. There should be regular monitoring. Work must be checked by people controlled by different managers. The users should check that specifications are met when each milestone is passed. Anyone affected by the work or with an interest in it should be allowed to participate.

The planning is thus the key and includes everything; the membership of the teams, the allocation of other resources, the standards, the frequency of various checks and reviews, the fallback and safety procedures to protect the business.

Quality control cannot be imposed because to succeed the whole team has to believe in it. When imposed everyone goes through the motions and the quality control work adds to the load without giving a commensurate benefit.

The big problem is how to change the attitudes of the staff so that they welcome such things as walkthroughs by independent assessors. The right attitude will work with inferior standards whereas no standards can save a project when the staff have the wrong attitude.

QC in the Systems Development Cycle

Linda Kindred (OR/Appl Support at Wellcome Foundation)

The strategy at Wellcome has been to separate 'maintenance and firefighting' out from systems development, and to keep a body of expert programmers and analysts in the 'applications support' department. Around 18 people are responsible for some 80 systems, comprising 2,500 programs. Of these 300-400 are APL based.

Linda has the absolute right to refuse to any system which fails her standards of quality control; in the past this has meant bouncing systems at the implementation stage, but the emphasis is now on imposing strict standards of quality throught the development cycle.

Just what is the development cycle? In the beginning there is the Budget. This used to be based on last year's usage; it is now based on agreed business requirements for the coming year. Priorities are thrashed out with the users as necessary. From then on . . .

- all work is project based. The project manager is the user. Project management techniques (networking etc) are provided by MS where required.
- one or two levels of working party are set up, again chaired by the users. MS and Internal Audit would normally sit on these, and they would meet monthly to review the network. Each 'activity' would always have an 'owner'.
- there would be 'terms of reference' to outline the scope, risks and limitations of the project.
- costs are approved stage by stage. There is no attempt to quantify total project costs at the beginning; the staged approval gives you the chance of killing projects after (say) an outline design phase.
- no fixed implementation date is set. Instead everyone must sign off the system (including Production Support who could force a re-write on the grounds of resource usage).
- the Post-implementation review (say 6 months later). What went wrong (and why)? What went well (and why)?

Applications support were traditionally only involved at the final stages. However in recent years there role has broadened (especially on major projects):

- they are represented on the working parties, with the particular role of checking the interfaces.
- they provide an inspection service to the analysts, programmers and project leader.
- they are the 'custodian' of the standards. These are all held on line on PROFS, and are used to log the benefits of past experience as well as being a rule-book for development.
- they have the QA responsibility. Clearly they are committed to this, because it is very much in the interests of the maintenance programmers. Call-outs used to average 2 per week; now 1 per month is typical.

- if you have quality from the beginning it is free. Systems are no longer bounced at implementation.

So what are their standards, and where do the benefits come from?

- Applications Support won't take on any system which 'ABENDS' 3 to 5 times per month, or takes a day's maintenance per month.
- the split between development and maintenance (at the budgeting stage) makes it quite clear to management just where the costs are coming from. It makes it much easier for the development phase to hit budget!
- changes and fixes are kept in a system log. In the long term this is used as evidence for re-writes.

The basic long-term aim is to ensure stability in the supported systems, and to free resource for urgent changes rather than basic maintenance.

Questions from the floor . . . .

What is an Abend in an APL system? When the workspace goes down and it takes you half a day to re-create the problem!

What about systems on PCs? This has hidden itself under the carpet, but managers are beginning to realise what they have got themselves into.

Who would be a typical 'user' on the working party? Definitely the senior guy, often a divisional manager.

What motivates the project teams to follow the QA standards? They will finish on time, within cost. You always get those who don't; their annual review reflects it, and they don't get the plums next time around! (Some have even been known to get 'special projects'.)

Is there a development team for APL? Lots of it was done in OR; this has been cleaned up and taken over. Some small APL developments have been picked up recently as confidence in APL maintainability has begun to increase.

Panel Discussion

Dave Parker (in the chair), Les Hollingbery, Adrian Smith, Maurice Jordan, Linda Kindred

DP In the Electricity Council they use Delta to standardize conventional program development, with an in-house front-end. This leads to a common style of code across all applications, and he has followed this with his APL methods. Basically this means standards for function, variable and label names, and a base of common utilities.

AS At the 1985 OR conference, there was an excellent talk from Nissan-UK, where it was emphasised that there are no quality control inspectors in Japan! Quality is built in, not inspected in! At Rowntrees we think we now have a APL

community who have been trained from the beginning in 'the right way' of doing things; no-one would ever think of using shared variables, or using files other than through cover functions.

Change control: for big systems there is a formal procedure for scheduling changes (e.g. to ensure 2 related changes go in separately). Each workspace has a text variable called RELEASE where programmers leave a log of what was changed and why.

For mainframe APL it doesn't matter all that much if it stops . . . usually no-one else is hurt, and you can often fix it 'on the fly'. PCs are a different matter, and delivered systems must be much more robust.

LH Les supports Information Centres at Rank Xerox. APL systems were very much ad hoc, but have now settled down and stabilised. They are looking at LOGOS from IPSA as a way of getting control over development and change. Xerox use a 'total quality' approach where everyone is there own inspector. He recommended Myers book for general advice, and a 'belt and braces' approach for critical systems (i.e. self checking software).

MJ Quality certainly matters on things like the reservation system (a 3-hour down time could lose the MD his job!); in APL they encourage the use of tools and common utilities, but not all systems conform. They suffered from an 'explosion' into APL (0 – 30 APLers overnight) and the undisciplined approach this led to. MJ is personally responsible for some 2,500 functions; few of these give him trouble, and he is trying to understand what the differences are.

Qn Do you use passwords and locked functions to protect code?

AS In the old days)SAVE was our file system. This is still OK for personal systems, but now we try to separate the code (typically on project libraries) from the data (on files on the user's own ID). We try to avoid knowing other people's passwords for routine maintenance!

DP They do much the same; he owns the workspace and the user owns the data.

General Beware the input stack!! At least with APL2 you can turn it off; it is not a safe substitute for a file system.

Qn How do APL systems compare on reliability?

LK/MJ Historically, APL has caused more trouble

LH When an APL program goes down, I have a lot less paperwork!

AS You sometimes discover a fault years after the system was implemented (e.g. a missing quote in an error message). However the real problems are caused when the system gives wrong answers, but doesn't crash!

MJ A big APL system went into parallel running after 6 man-years; most of the discrepancies were faults in the original system (unnoticed for years).

Lusmore A random-number generator was once left in a sales commission routine for 6 months before anyone spotted it!

Qn Does this conflict with the idea of fast development?

MJ Things that should have taken a day's thought and 2 day's work often took weeks when rushed into!

AS Sit on your hands for half an hour! Don't let the user (whoever he is) rush you into designing at the keyboard! Let the computer find the SYNTAX ERRORS by all means; it is up to you to find the DESIGN ERRORS at your desk.

LH There is always the danger that APL will encourage you to tackle things which are beyond you.

Qn APL may do things fast, but what kind of documentation should you do?

DP Even if you document the functions, it is hard to document the methodology. Where possible get the user to do it for you.

AS Most of the 'user guide' should be on the screen. Similarly the system documentation should be in the WS; e.g. an active data-dictionary which logs the types and descriptions of variables as well as driving the editing functions.

Camacho First write the manual; second write empty (commented) functions; third fill in the code.

MJ It is surprisingly easy to read other people's code. Good APL is really just the specification expressed in a mathematical notation.

Cyriax Often it is hard to comment adequately in English; quite often you slip into APL, particularly when describing function arguments.

Lusmore Good comments are hard to do; often they look alright until you try to use them!

AS APL doesn't let you do good layout. Compared to Pascal or PL/1 it is a typographic disaster area! You can only comment on separate lines; you can't indent; you can't leave blank lines or spaces in the middle of lines. Try reading someone else's Pascal if all these visual cues are taken out!

MJ Perhaps the most valuable documentation is a collection of sample inputs and outputs for functions.

Camacho Use meaningful names, not daft standards like 'L1:' for labels.

AS Often the choice of local names tells me who last changed the function.

Announcement

Linda closed the meeting by telling us about a group called the QA Forum which meets quarterly to discuss issues related to QC in systems development.

It is an independent body, and costs £300 pa, with a fee of £85 per meeting attended. The members are able to get specific topics on to the agenda, and workshops are held to help with particular problem areas. The collected papers from each meeting are circulated to all members, who are generally people involved in day to day QA work.

Graphics

20th March 1987 at the Royal Over-Seas League

Introduction

Moral the First: never under any circumstances rely on Barco projectors for your presentation! Failures of technology apart, this was a very successful day. My notes on Pete Donnelly's talk are followed by some of my impressions on the APL.68000 GDDM processor (demo by David Eastwood) and WIMPS on the Amiga (demo by Richard Nabavi). Finally, some all-too-brief impressions of a fascinating series of videos by Keith Waters on the animation of facial expressions.

CGI Graphics

Pete Donnelly (Dyadic Systems)

No-one could disagree about the need for a graphics standard; up to a couple of years ago GKS (UK/Germany) ruled unchallenged; 'Computer Graphics Interface' is the newer American version (shades of NIH!).

Its main advantage is that it will handle bit-mapped displays, and that (being American) it is rapidly swamping the opposition. Dyalog are building a CGI interface for workstations such as the Sun, Apollo, RT-6150 and so on. This will handle pretty well any device, and does all the usual things:

- polylines, markers and filled areas.
- text (cursor, graphics and high-quality word processing) mixed in a single display.
- multiple pages, and block copy of pixels (handy for windows!).

Pete was just nicely getting into his stride, showing us how to run more than one APL session from the same terminal, when disaster struck the Video projector, and he hastily swopped out in favour of . . . . .

GDDM / AP126 on APL.68000

David Eastwood (MicroAPL Ltd)

It was now a case of splitting into manageable groups, and forming a series of loose scrums around the available screens! The quality of my notes degenerates badly at this point . . . sorry!

GDDM-68000 is a straight emulation of GDDM 3.0 and AP126. It is written in 68000 Assembler, and looked very responsive on MicroAPL's Aurora. It takes around 50K off your free memory (not a lot, considering what it does), and as far as I could see it really does look just like GDDM. M. Legrand would be proud of them (see reviews!).

WIMPS and Bach Canons on the Amiga

Richard Nabavi (MicroAPL Ltd)

Oh what fun this was!! There are two streams at the low end of the market:

1. PC look-alikes ◊ 286 ◊ 386

2. 68000-based (Mac, Atari, Amiga) with possible PC DOS windows if you want Lotus.

These latter come with all sorts of fancy features, generally known as WIMPS (Windows, Icons, Mice ...); how do you implement APL to take advantage of WIMPS ...

- to use mice and windows to help the programmer?
- to make it easy to develop WIMPish applications?

The Amiga implementation is a fully multi-tasking APL which can generate and use windows very easily. Pull-down menus and mice are handled as a natural part of application design. You can cut and paste from the built-in terminal emulator into your local APL session, or from that session into the function editor.

With 8Mbyte of memory to go at, you can set background tasks running and still get very acceptable response at your primary APL session. Functions are provided to handle the (virtually HiFi) 4-channel sound and speech synthesis (SAY PHONETIC 'Hello, I am an Amiga'). The high point of a fascinating demo was a demonstration of the Bach Crab Canon (a theme which harmonizes with itself when reversed); surely Bach anticipated the existence of the 'reverse' function by some 200 years when he wrote this useful piece of test data!

3D Facial Animation: The Expressive Capability of Computers

Keith Waters (Middlesex Polytechnic)

I really can't do justice to this one; you just have to see the videos to get a proper grasp of what Keith was driving at. The particular problem he was attacking was the difficulty of achieving realistic 'lip-synch' in animations such as the ITV series 'Spitting Image'.

Some of the important characteristics of the face relevant to animation are:

- 14 bones and 100 muscles.
- the positioning and spacing of 20 key elements.
- 'micro expressions' lasting 1/40th of a second or less, which are missed by 24 frames per sec animation.
- the more realistic the face, the more critical your viewers!

The classic approach to the problem is to take 3D photographs of a face (on which a polygonal pattern has been drawn), and digitize these with a wire-frame diagram The problem with this is the enormous amount of data needed for the 'in betweening' between snapshots; the computer run times for even very brief animations were quite massive.

Keith had started down this track, but was now exploring a quite different approach: using a dynamic 'muscle model' to build expressions analytically rather than from a vast database of stored positions The animations done with this were quite fascinating, by tensing just two muscles in the cheeks (and computing the effects on the affected facial surface) he generated a remarkably convincing smile. Because no interpolation was needed (the step size could be as fine as he liked) the smile genuinely 'flickered' in just the way a real smile can.

In short I really think Keith is on to something. Whether it is relevant to APL systems this side of the millenium is another question; I suspect it is, in the kind of 'right brain' way that Phil Smith was showing us at APL-86. Thank God the equipment worked for this one!!

Info Center/1

an IBM licensed program that helps business professionals perform their daily tasks quickly and productively

Info Center/1 provides an integrated, multifunction information center environment compatible with predecessor products such as ADRS II and APLDI II. A full-screen interface, with prompts and extensive help facility, provides easy access to the following powerful general business functions, as well as providing the full power of APL:

Query System

The Query System provides a simple, effective way to interactively access, analyze, manipulate, and report information stored in files of up to several hundred megabytes.

Reporting System

Provides an organization with a single, comprehensive system for generating and maintaining reports. Standard calculations can be defined and stored for future use. Calculations can be made with predefined functions and with APL.

Data Entry and Validation

This tool allows information center personnel to tailor panels for users to display, update, and enter data in column format.

Financial Planning System

The Financial Planning System provides a set of 60 modeling routines that work with the Reporting System and address periodic data. Some examples are:

- Financial analyses and plans
- Statistical analyses and projections
- What if analyses and modeling
- Project evaluations and risk analyses.

Business Graphics

The Business Graphics facility is a particularly powerful yet flexible tool for interactively producing the following types of charts: line graphs, surface charts, histograms, pie charts, scatter plots, bar charts, stacked bar charts.

Technical Data

Info Center/1 is an IBM Licensed Program, Program Number 5668-897.
The program runs under CMS and TSO together with the following IBM programs or their equivalents: APL2 or VS APL, Application Prototype Environment, GDDM (Graphical Data Display Manager).
Some examples of terminals supported are: IBM 3277, 3279, 3270 PC/G and GX.

IBM

Graphics in the Boardroom
(or "You can take a horse to water . . .")

by David Preedy
Development Director, Metapraxis Ltd.

Summary

For many years it has been an accepted fact of I.T. that directors are computer-illiterate and that any attempt to provide computer facilities for them is doomed to failure.

This talk looks at some of the reasons why this may have been true in the past and examines some of the issues underlying one approach which *is* finding widespread acceptance.

1. Introduction.

This paper examines some of the underlying issues that have been found to be critical in the implementation of information systems to support Board directors of large organisations. Such systems are now becoming used more commonly and are normally described as Corporate Control Systems. This paper describes many of the issues addressed in the development of *resolve* – one of the trend-setters in this field.

2. History of computer systems for directors.

Until the last few years, there was hardly a company that provided any form of Information Technology for its Board directors – apart from the telephone and a pocket calculator. It was accepted that mainframe systems should be introduced to automate clerical tasks such as sales ordering and working out the payroll. Then some audacious and frightfully bright young things started using time-sharing bureaux to build "models" and incur large bills. More recently all the secretaries threw away their type-writers and were converted to word-processing; sales of "Tippex" tumbled and the company had to buy another shredder. Now the finance department is full of analysts, hot from Business School, beavering away over spreadsheets named after fast cars.

The investment in "high-technology" had by-passed the Board – the people who surely ought to have most to gain from successful applications. The shop-floor has been revolutionised by robots and CAM systems, the ranks of clerks have been replaced by on-line transaction systems, even the good old telephone has gone digital and now lets you do all those useful things like pick up the nearest handset to answer a call. But the Boardroom and the Directors' offices were left unchanged.

To be fair, there were a few exceptions – pioneering companies whose Chief Executive had insisted that his systems people should build him a system – but they numbered no more than about a dozen, several in the U.S.A. and a handful in the U.K. Notable examples were at B.P. and at Imperial Group, the latter system being described in VECTOR Vol 1, No 2.

Moreover the number of active systems was actually declining. The problem was that they tended to be based intrinsically around the requirements of one individual – the keen Chief Executive. No other senior directors had the commitment to give the development team a significant briefing on their own needs or on their experiences of using the system. As soon as the Chief Executive left, the product was left without a champion and the system fell into dis-repair. Sooner or later its demise was inevitable.

3. Reasons for lack of interest.

Several factors combined to cause the distinct inactivity in Director-level information systems. The following are some of the more significant:

Cost

Those systems that were developed in the late-Seventies or early-Eighties were notably expensive. Typical costs ranged from one to ten million pounds to support one or a small handful of users. Shared hardware led to unacceptably variable response times so that entire systems had to be dedicated, or other users unceremoniously excluded from access during Executive meetings – even though the system might not be used eventually that day.

Remoteness of D.P.

Typically the Data Processing department did not (and still does not) have close links with the Board. Its line of reporting is frequently via the Finance Director who is the one Director most likely to understand and be satisfied with the status quo in terms of routine reporting – if only because it is under his control. To a large extent, D.P. has only itself to blame; its past forays into management reporting tended to provide untidy line-printer output in block capitals on the wrong-sized paper with sprocket holes down the side and covered in strange green lines. The D.P. objective tended to be to computerise the existing system, rather than to try to make more effective use of the information and provide reports as a by-product. It is surely not coincidental that most Chief Executive systems were produced by O.R. (rather than mainstream D.P.) departments (and their track-record in contact at Board level has generally been rather poor).

Inflexibility of D.P. approach

The conventional D.P. development approach turns out to be remarkably unsuitable for these systems. Directors do not have the time to spend briefing Systems Analysts on their "needs", even if they know what those needs are. Rather they are looking for an evolutionary approach, driven by a team who can respond rapidly to feedback and who have the business acumen to initiate relevant new developments. Above all, they are impatient for results; after all the commercial environment won't wait while the detailed system specifications are drawn up!

Lack of suitable development staff

Following on from the previous point, few firms have staff with the suitable blend of technical experience and financial expertise needed to develop a successful system with minimal direction from above.

Techno-phobia

The typical director may well be in his fifties with no personal experience of computing. Even his children will have left school before the advent of the home computer. His experience of technology is probably limited to the telephone, the bank's cash machine and the infra-red controller for his video-recorder. Not only are the directors likely to be the least computer-experienced user group; they are the one group who can vote with their feet. If they decide that a system is unacceptable, there's no boss to tell them they have to use it.

4. Critical factors

As with any system, there are a wide range of issues which have to be considered in developing a Corporate Control System. The following are some that have been found to be pre-requisites for successful implementations.

Relevance

It goes without saying that the system must address the key issues facing the Board and that the facilities offered must reflect the styles of use of Directors. The problem facing the developer is how to identify these from the necessarily limited access to the small number of users. It turns out that an in-house team faces an almost insuperable task and most organisations decide to involve one of the small number of external consultants who have direct experience of working at this level.

Ease of use

The entire system has to be designed to provide ease of use. It is unrealistic to expect a director to have the time or the inclination to read through a weighty reference manual. Considerable research has been carried out into many behavioural issues affecting use of systems; even the presence of a full keyboard can introduce associations with secretarial functions which may cause problems. These ergonomic factors are exacerbated by the importance of using the system during meetings (without technical support) and the prestige value of using the system to show journalists and financial analysts how well the company is managed. The Chairman's office is not only his work-place; it is also a show-piece of the entire organisation.

Other issues affect ease-of-use; in particular directors want to be offered choices that are related to their business concerns rather than technical issues. It is important whether they look at a short-term or a long-term trend; they don't have the time to specify what colour they want each line displayed in, nor generally whether they want a line-plot or a histogram. A good Corporate Control System should make such decisions for them in a consistent manner.

Availability

However good a screen-based system may be, paper-based output is also essential to form the basis of one-off as well as routine reports. This is frequently the starting-point, but more ambitious users rapidly install colour graphics terminals in directors' offices for individual analysis of corporate performance. The next move is often to introduce large monitors for group discussions, or, better still, to provide video-projection facilities in the Boardroom. Now that portable microcomputers are available, many directors like to be able to take their corporate database around with them, either on visits to subsidiaries or for work at home.

Support

One of the critical factors in successful Corporate Control implementations has been the quality of user support offered and the skills required of the support team. Ideally support should be centred on a "system manager" who has overall responsibility for the system. Normally this individual is chosen for his financial rather than his technical expertise, although he obviously needs occasional recourse to computer experts. The pivotal nature of this role arises from the fact that the directors want to have a single point of reference for any queries they may have and these queries may relate to hardware problems, software issues, discrepancies or inconsistencies in the data provided or the accounting treatment of some of the figures, or genuine difficulties in the business being examined.

Adaptability

One of the key considerations is the retention of adaptability. Corporate Control applications are necessarily evolutionary; the limiting factor on system development tends to be the directors' requirements on the system, rather than technical obstacles or mere inertia. This has been a major consideration in the choice of APL as the development language for several Corporate Control Systems.

5. Directors' Information Project Stages

The evolutionary nature and the expectations of the user-base have led to the refinement of the recommended stages during the creation of a Directors' Information System.

The Design Phase aims to create a prototype system, providing the initial stab at the full corporate structure, but concentrating data coverage on a chosen part of the organisation. The aim is to give directors a working, though deliberately limited, tool so that they can evaluate the efficacy of adopting this new approach at an early stage. Technical issues are addressed to ensure that they do not present insurmountable obstacles to subsequent full implementation. The prototype is normally used to produce special report packs focussing on issues of genuine interest and to support executive committee meetings.

The next step is to proceed with the Implementation Phase. This extends the data coverage to the full corporate structure and integrates the Corporate Control System with any data sources so that the monthly updating process becomes a routine operation. There is also extensive training both for directors and for the system manager.

The third stage is the Review Phase. After a few months experience of use, it is possible to evaluate feedback on how the system is being used in practice. The monthly Board report may need to be redesigned to reflect the changing management style brought about by using the system. Experience frequently suggests revisions to the information covered by the system, and there may have been changes to the underlying corporate structure. In any case the style of use will vary according to the time in the company's financial calendar – whether the main theme is reviewing next year's budgets, evaluating the strategic plan or concentrating on the routine monthly reporting.

6. Design issues.

In terms of the detailed system design, there are a host of factors that need to be incorporated to ensure acceptance.

Access to data

The Corporate Control System rapidly becomes the master system for the provision of corporate management accounting and reporting information. It is essential that it becomes an integral part of the reporting process with electronic links wherever possible to allow automatic updating of data. Attention must be paid to ensure that data are timely and accurate.

Consistency

It is essential that the data made available to directors in their office is wholly consistent with the figures presented in reports. It normally becomes desirable to produce as many reports as possible directly from the Corporate Control System.

Corporate structure

The system must reflect the management's own view of their organisation. This structure has to form an integral part of the system and provides one of the key methods of data selection. The structure is, of course, fluid, and the system has to provide an easy means to revise it, allowing for restructuring as well as handling the acquisition and disposal of subsidiaries.

Timeliness

The updating of the system has to be fully integrated with the routine management reporting process. System management has to be as closely regulated and time-tabled as the accounts department so that the update to a director's system coincides with his receipt of the Board report pack.

Graphics

Above all, directors are looking to understand the messages underlying their corporate information, not just to know the numbers. A major role of the Corporate Control System is communication, with the Finance Director, say, explaining to his colleagues the likely impact of problems in a part of the Group. The case for graphics has been made elsewhere, and it suffices to say that effective graphics form the core of any successful Corporate Control System.

Design

Graphics on their own are insufficient to ensure a successful implementation. Considerations of ease-of-use (discussed above) lead to the "chart-book" approach, where a range of carefully-designed chart formats are provided with an integral link to the underlying corporate database.

The issue of graphics design is a vast and complex area touching on issues of creativity and "feel" as well as objective factors. Many of the design criteria were described in a previous article (see reference 1) and readers are recommended to follow up Ehrenberg's article (reference 2) and the book by Tufte (reference 3). In this paper I propose to mention only those factors that have proved to be most important.

Firstly the charts should be relevant; they must convey the important messages underlying the data. Many a time I have heard a director look at a chart and say "Well, you can't argue with that!" Because the principles of effective corporate control are independent of the organisation, it has been found empirically that a reasonably standard range of chart formats can apply. It is perfectly reasonable, and in many cases advantageous, to use the same format to present sales turnover figures as is used for sales volume, or cash flow. Most important is the immediate and automatic access to the correct data.

Secondly the information must be accessible. It must be physically accessible in the sense that it is available in the Director's office and for group discussion at committee and Board meetings, and on a portable system for use at home or on business visits. It must also be psychologically accessible; anybody at the meeting should feel comfortable using the system to support their point of view in discussion. This requires more than the traditional view of "user-friendliness"; equally important is a sense of confidence that the underlying data are correct and that the form of presentation will be relevant. Similarly, the language used should be that of business not technical jargon, whether of computing or statistical origin.

Thirdly the system must be consistent. Apart from data consistency which goes without saying, the forms of presentation must be consistent. Graphics already bring benefits in terms of rapid assimilation of key messages; these benefits become even greater if the user does not have to learn how to interpret each chart. Hence the attraction of the "chart-book" approach, allowing the user to select his combination of chart format and data series. Moreover there should be consistency between chart formats – consistency in choices of colours and line-types and in the general style of presentation. For instance in RESOLVE the colour green is reserved for Budget data; wherever a director sees a green bar in a histogram, or a green line, he doesn't need to refer to the chart legend to find out that it refers to Budgets.

Finally the presentations should be attractive. This may sound a trivial issue, but frequently these types of system are shown to outsiders, either financial analysts or journalists. And, anyway, the directors have a difficult enough job running their businesses; what right have we to make their lives harder by presenting them with anything less than the best? They may simply decide they'd rather not use the system at all.

7. Conclusions.

The development of a system such as RESOLVE has required a wide range of skills, in particular an attention to detail and an ability to balance the technical possibilities (and temptations!) against the realities of running a large organisation. I am sure that there must be others who have had to face similar issues, no doubt in different application contexts, and I am keen to establish a forum where they can be discussed. Any interested parties are invited to contribute to the debate, either by contacting myself directly or through contributions to VECTOR.

References.

1. *"Graphics for Decision-Makers" by David Preedy, VECTOR Vol 1, No 2, Oct 1984, pp83-88.*

2. *"Rudiments of Numeracy" by A.S.C.Ehrenberg, J.R.Statist.Soc.A Vol 140, pp277-297.*

3. *"The Visual Display of Quantitative Information" by Edward R Tufte, (1983), published by Graphics Press, Cheshire, Connecticut.*

The Charlton Chase Package

by Paul Barnetson

Introduction

The package was written to help control the "Charlton Chase", an annual event organised by West Sussex County Council Youth Office. The event consists of teams of four young people (15-19 years) walking from checkpoint to checkpoint, back and forward across the South Downs, overnight and in the middle of winter. The weather is usually bad and the event demands map reading and night walking skill of the highest quality from the participants.

In previous years, the event was controlled manually with the obvious results that the HQ staff were never sure if a team was late at a checkpoint; arithmetical errors were made in calculating results; final results were not available until a fortnight after the event finished etc.

In 1987 a computer was used for the first time. This was a great success. The organisers decided always to use it in future. It automatically provided warning information on teams that were late at a checkpoint during the night, teams were given their printed result immediately on their return, interim results were available while teams were returning; final results were available immediately the last team was in, etc.

The package was written in PC APL to run on an IBM PC.

An Offer

I have been advised by IBM that the package belongs to me as I wrote it in my own time. It was written and run – with IBM's permission – on equipment belonging to IBM.

Whilst retaining ownership and copyright of the program, I am happy to make it freely available to any youth or other charitable organisation who would like to use it. An IBM PC or equivalent machine, running PC APL will be required. If other organisations or individuals wish to purchase the right to use the program I will be happy to negotiate with them.

The Package

Input starts by recording the teams as they register. This can either be done before the event or early in the evening of the event. Teams then acquire the status 'Entered' and are allocated a team number.

When the teams arrive for the event this is recorded, their status is changed to 'Arrived' and they are allocated a route.

When the teams have had their kit checked, they are started on their route. Their status is changed to 'Walking' and the program sets up the necessary variables to record the teams' times and points.

During the night as the radio messages come in from checkpoints, each team's arrival and departure times are entered into the computer for each checkpoint. The time for each leg of the walk is calculated as each arrival time is entered.

If a team retires during the night this is also recorded and its status is changed to 'Retired'.

When a walking team returns its status is changed to 'Finished' and the return time and the points it won at checkpoints and the optional orienteering points are all recorded.

When a retired team is brought back its status is changed to 'Back' and the return time and the points gained are all recorded.

There are facilities for entering and updating the names of the checkpoint staff and the locations of the support vehicles.

Checking

All possible checks are done. For example a team can't arrive if it was not entered, nor start walking if it has not arrived. It can't finish if it was not walking nor can it be reported back if it was not retired.

Zero and negative leg times are rejected as are negative periods at checkpoints.

Teams can only walk through checkpoints on their allocated route and if a team is reported as going through a checkpoint before we hear that it has gone through a previous checkpoint (perhaps because of radio breakdown), a warning message is given. So when a team is reported as going through a checkpoint not only is the previous leg time calculated but a check is made to see if the following leg time should also be calculated.

Reports

The reports described below are all available on screen or on paper.

A complete list of all teams that have entered or those of any particular status, sorted alphabetically or by team number or by order of arrival within route.

A team report listing team details, times through checkpoints, leg times, a message (if it is walking and late) saying it is late for the next checkpoint and full details of all points gained (if the team is finished or back). This was given to each team on its return and was very popular.

A checkpoint report listing the teams due through the checkpoint on all routes containing the checkpoint and the actual times for all teams as so far reported.

A route report listing the above details for all checkpoints on a route with details for each leg on the route. These include the expected time, the panic time (if a team exceeds this it is reported late) and the minimum, maximum and average of the actual times so far reported.

A late report listing all teams that are then late.

Other reports list staff at any checkpoint or the location of any support vehicle.

Results

There are two sorts of result:

Interim — Once the first team returned interim results were produced at regular intervals, and placed teams in order within their categories (finished, retired and back and still walking). These were posted on a notice board and were very popular with the participants.

Final — When all the teams were either finished or back the final report listed all finished teams and all the retired teams each in order according to the number of points they had gained. The ability to do this at once when all teams were back was also very popular.

Conclusions

Several suggestions for improvements were made by staff during the night and these will be incorporated for the 1988 event.

For example the new status of 'Scratched' and (unfortunately) 'Disqualified' will have to be added. The layouts of the various reports will be changed and one or two new ones will be added. Finally there was the old and unresolved discussion on what is the right order for sorting numbers and letters.

I was told that the use of the computer was an "unqualified success" and that it will definitely be asked for again next year.

Further Information

Whilst retaining ownership and copyright of the package, I am happy to make it available to any youth or other charitable organisation that feels it could use it. For more information on this and a more technical description of the package, contact me, Paul Barnetson, at my home address: 22 Ferndale Road, Chichester, West Sussex, PO19 4QJ

Bell's Inequality

by Sylvia Camacho

Intellectual communication with my daughter, Alison, became difficult some years ago when I realized that mathematical expressions which I would describe as 'smeared all over the page' held some meaning for her but none for me. My degree is in philosophy: she is just starting her final year in physics – although she did give a passing nod of recognition to philosophy as one of the side-shows in her first year. This explains how I came to browse among the philosophy and physics shelves in the bookshops of London, Oxford and Bristol and so added to my obsession with APL another with the philosophical problems of physics in general and quantum physics in particular. So, for those among VECTOR readers who enjoy philosophical in-fighting, here is an account of one of the books I found and how I came to use APL to help in thinking about the problem it expounds.

I knew I was hooked after reading 'Quantum Reality'. Nick Herbert is a professional physicist writing a popular account of how physicists came to 'lose their grip on reality'. He conveys the essence of the problem without resort to mathematics and follows it with a robust account of the various bizarre and incompatible models proposed by eminent physicists to give substance to the mathematical formalism on which they all were and are agreed. Perhaps I warm to Nick Herbert because he is not awed by Von Neumann's analysis; even going so far as to suggest a similarity with a cartoon showing a mathematician before a blackboard on which there is 'the usual incomprehensible symbols but near the end there is a gap in which he's written, "And then a miracle occurs". The math then resumes and proceeds to its logical conclusion. The mathematicians colleague is pointing to the gap and saying, "I think you should be more explicit here in Step 2".'

Nick Herbert relates how he finished his formal education, 'Blissfully ignorant concerning the real issues surrounding the quantum reality question' until in 1970 he was shown a paper by John Bell expounding a theorem of which he says, 'Bell's theorem is easy to understand but hard to believe. This theorem says that reality must be non-local."Non-local" means,. . . . . , that the atom's measured attributes are determined not just by events happening at the actual measurement site but by events arbitrarily distant, including events outside the light cone – that is events so far away that to reach the measurement site their influence must travel faster than light.' What is more, says Herbert, this theorem is expressed in simple arithmetic and, even more confounding, the test for non-locality suggested by it has been made and is confirmed by the experiments.

So I came to the chapter that expounds Bell's theorem and tried (and failed) to understand its significance. What I needed was a model in which I could point to the exact place at which 'the miracle occurs'. I set the model up and then re-opened communication with my daughter by means of the letter reproduced below together with the simulated results of the tests of reality that Bell had proposed – is it classical (Newtonian) or quantum (non-local)? As you will see, the APL I used to set up the model is very simple being used simply as a tool of thought.

Boxing Day 1986

Dear Alison, About 18 months ago – I believe on the occasion of the British APL Association AGM in the Spring of 1985 – I bought several books in Dillon's which fall into the category of popularisations of the concepts of quantum physics. In particular they analyse the difficulty that arises in the interpretation of the experimental evidence (& the mathematical underpinning) of models of quantum systems based on particles as opposed to models based on waves. I found that the book by a physicist, Nick Herbert, called 'Quantum Reality' was the clearest explanation of the difficulty. He explains that he wrote the book to bring to a wider public the findings of some comparatively recent experiments by Alain Aspect. These experiments were performed because a paper by the theoretical physicist, J.S. Bell, suggested how the hypothesis of non-local effects might be tested. This is a development of the 'thought experiments' which Einstein proposed among others when the unexpected and, to Einstein, unnacceptable consequences of the mathematical formalism of quantum effects was realised.

The thought experiments are concerned with measurements of the characteristics of pairs of particles which can be generated in what is called a 'phase-entangled' state. According to Nick Herbert this means that these particles can be considered to be twins – identical in all their characteristics. Two beams of these particles can be directed at meters which can be any distance apart and any distance from the particle source. The particle characteristic which is measured is polarity and each meter can measure one of two conditions, 'up' and 'down'. However, the meter can measure polarity in several planes. Bell suggested that polarity should be measured at -30, 0 & 30 degrees and that comparisons be made between the results at the meters with both set at 0, one at 0 with the other at -30 or 30 and lastly with one set to -30 and the other to 30 degrees.

It is well established that beams can be produced which are completely 'unpolarized' – that is, the proportion of 'up' to 'down' is always 50:50. Using such beams and setting both meters to the same plane, quantum theory predicts that the match between the meter readings will be perfect. That is to say an 'up' on the first meter will always correspond with an 'up' on the second and likewise 'down' with 'down' although the sequence of up and down appears to be unpatterned (i.e. random).

However, if one meter is set to 0 and the other to -30 (or 30) the perfect match is lost and the measured mismatch is 25%. From this one would expect that the mismatch when one meter is set to -30 and the other to 30 will not exceed 50%. Bell pointed out that this 'classical' expectation is not met by quantum systems, for when the meters are set to -30 and 30 the quantum prediction is that the mismatch will be 75%. Alain Aspect confirmed this prediction by setting up an experiment to perform these measurements.

How are we to interpret these results? It appears that the characteristics of the particles change as the meters are reset. This might be accepted and some 'classical' interpretation sought if it could be assumed that there is some means of communication between the meters such that setting them to -30 and 30 changes the environment for the particles in a

way which changes their measured characteristics. The puzzle is that the distance between the meters can be arranged so that the effect of changing the setting of a meter must be transmitted faster than the speed of light to account for the observed results.

I have set up an APL model of this experiment which can be run in 'classical' or in 'quantum' mode. The model generates particle beams by creating a sequence of random numbers chosen from the set 0 0 1 3 4 6 7 7. The binary equivalent of these numbers represents the three characteristics of the particle which may be measured – that is to say whether it will be 'up'(0) or 'down'(1) at -30 0 30 degrees. Thus a particle represented by the number 6 (1 1 0) will give down at -30 and at 0 but up at 30 degrees. Similarly, a particle of number 7 (1 1 1) will give down at all settings while a particle of number 0 will give up at all settings, and so on.

The model prints a set of results for all the possible settings of the two meters. The number of particles (random sequences of numbers as described) between changes of meter settings can be chosen, as can the type of 'reality' – classical or quantum. The model prints for each meter the proportion of up to down measurements and the 'mismatch' ratio between meter A and meter B. If set to 'classical' the results are as expected – the mismatch at the -30 30 setting approximates to 0.5.

To simulate the quantum behaviour the model has a slight difference in coding when the meters are set to -30 30 (or 30 -30). In these cases the random sequence for the second meter is reset before display such that half (chosen at random) of all particles of number 0 (0 0 0) is reset to number 1 (0 0 1) and half (chosen at random) of all particles of number 7 (1 1 1) are reset to number 6 (1 1 0). This small change is all that is necessary to produce the quantum reality mismatch ratio of 0.75 and is the place where, as Nick Herbert would put it, 'the miracle occurs'.

The fact that this resetting is done to all the numbers in the -30 30 and 30 -30 sequences represents the faster than light communication between the meters. To model the results which would be expected if the -30 30 setting were a significant change in the environment but that the effect could only be transmitted at the speed of light, the second meter sequence would be reset as described but after some delay. Therefore I have a third, modified, table of results which I have headed 'M-QUANTUM'. The delay is represented in the model by excluding the first T particles from the 'quantum resetting algorithm' and this tends to reduce the mismatch ratio. The amount of the reduction depends on the ratio of T to the total number of particles S in the sequence. If S is 100 and T is 50 the mismatch ratio reduction will be 0.25 times T divide S which is a reduction of 0.125. If, however, T is small in relation to S the mismatch will be much less noticeable.

Let us imagine the model is run with this 'propagation time' adjustment. We have to consider what the effect would be if the first meter were now reset to measure the same characteristic as the second meter (say both set to 30 degrees). The change would have a propagation time which would be represented by continuing to reset T particles for the second meter as though during this 'time' the -30 30 settings still pertained and this would

spoil the perfect match expected when both meters are set to measure the same characteristic. As before, the size of the blemish will depend on the ratio of T to S and will be a mismatch of 0.125 for T=50 and S=100. I have not simulated this as my model does not allow the settings to be changed while it is running.

What I do not yet know is whether Alain Aspect's experiments are such as to exclude these possibilities of finding a 'modified' quantum effect. I doubt my ability yet to understand the paper in which it is described – but I'm working on it!

love,

Sylvia

```
    ∇ T BELL N
[1] ⍝ Model of all possible meter pair
[2] ⍝ settings for 0, 30 and ¯30 degrees
[3] ⍝ for runs with sample size of N
[4] ⍝ propagation time equivalent to sample T
[5] ⍝
[6]  20 1⍴' '
[7]  ¯50↑'Reality is ''Classical'''
[8]  ''
[9]  'C' UPDOWN N
[10] SCREEN
[11] ¯50↑'Reality is ''Quantum'''
[12] 'Q' UPDOWN N
[13] SCREEN
[14] ¯50↑'Reality is ''M-Quantum'''
[15] 'M' UPDOWN N,T
[16] SCREEN
    ∇

    ∇ R←M MATCH S
[1] ⍝ Match according to meter settings M
[2] ⍝ returning mismatch ratio of sample S
[3] ⍝
[4]  →(M='LRA')/LEFT,RIGHT,ACROSS
[5] ⍝
[6] LEFT:R←(+/S∈ 3 4)÷⍴S ⋄ →0
[7] RIGHT:R←(+/S∈ 1 6)÷⍴S ⋄ →0
[8] ACROSS:R←(+/S∈ 1 3 4 6)÷⍴S
    ∇

    ∇ R←QUANTUM S;N1;N2
[1] ⍝ Changes members of sample S if
[2] ⍝ 0 or 7 to 1 or 6
[3] ⍝ half such members only are changed
[4] ⍝ and they are chosen at random
[5] ⍝
[6]  S[(S=0)/⍳⍴S]←(0 1)[?(+/S=0)⍴2]
[7]  S[(S=7)/⍳⍴S]←(7 6)[?(+/S=7)⍴2]
[8]  R←S
    ∇

    ∇ R←ROLL N
[1] ⍝ Returns random sequence of N numbers
[2] ⍝ chosen from 0 0 1 3 4 6 7 7
[3] ⍝
[4]  R←(0 0 1 3 4 6 7 7)[?N⍴8]
    ∇

    ∇ SCREEN;A;B
[1] ⍝ Print screen headings and results
[2] ⍝ UDA and UDB are global vectors containing the meter up/down ratios
[3] ⍝ CV is a global vector of the mismatch ratio of meter A and B readings
[4] ⍝
[5]  (¯25↑'Meter A'),(¯20↑17↑'Mismatch Ratio'),'   Meter B'
[6] (15↑'U:D Ratio'),(¯10↑'Angle'),(20↑' '),(10↑'Angle'),'U:D Ratio'
[7]  ''
[8]  A← 0 30 ¯30 0 30 0 ¯30 ¯30 30
[9]  B← 0 30 ¯30 30 0 ¯30 0 30 ¯30
[10] 12 4 12 0 12 4 12 0 12 4 ⍉(UDA⍤.+,0),(A⍤.+,0),(CV⍤.+,0),(B⍤.+,0),UDB⍤.+,0
    ∇
```

```
    ∇ R←T TQUANTUM S;N1;N2
[1] ⍝ Makes random changes to members
[2] ⍝ of sample S if member is 0 or 7
[3] ⍝ in the same way as function QUANTUM
[4] ⍝ but excluding first T of S
[5] ⍝ to simulate propagation time
[6] ⍝
[7]   R←T↓S
[8]   R[(R=0)/⍳⍴R]←(0 1)[?(+/R=0)⍴2]
[9]   R[(R=7)/⍳⍴R]←(7 6)[?(+/R=7)⍴2]
[10]  R←(T↑S),R
    ∇

    ∇ TYP UPDOWN N;S;T
[1] ⍝ Determine the 'up/down' ratio
[2] ⍝ of the polarity of the samples
[3] ⍝
[4]   T←¯1↓N ⋄ N←1↓N ⋄ UDA←UDB←CV←⍳0
[5] ⍝
[6]   UDA←UDA,(+/(ROLL N)< 0 1 4)÷N
[7]   UDA←UDA,(+/(ROLL N)< 0 4 6)÷N
[8]   UDB←UDA←UDA,(+/(ROLL N)< 0 1 3)÷N ⋄ CV←3⍴0
[9]   UDA←UDA,(+/(S←ROLL N)< 0 1 4)÷N
[10]  UDB←UDB,(+/S< 0 4 6)÷N ⋄ CV←CV,'R' MATCH S
[11]  UDA←UDA,(+/(S←ROLL N)< 0 4 6)÷N
[12]  UDB←UDB,(+/S< 0 1 4)÷N ⋄ CV←CV,'R' MATCH S
[13]  UDA←UDA,(+/(S←ROLL N)< 0 1 4)÷N
[14]  UDB←UDB,(+/S< 0 1 3)÷N ⋄ CV←CV,'L' MATCH S
[15]  UDA←UDA,(+/(S←ROLL N)< 0 1 3)÷N
[16]  UDB←UDB,(+/S< 0 1 4)÷N ⋄ CV←CV,'L' MATCH S
[17]  →(TYP='CQM')/REALC,REALQ,REALM
[18] REALC:
[19]  UDA←UDA,(+/(S←ROLL N)< 0 1 3)÷N
[20]  UDB←UDB,(+/S< 0 4 6)÷N ⋄ CV←CV,'A' MATCH S
[21]  UDA←UDA,(+/(S←ROLL N)< 0 4 6)÷N
[22]  UDB←UDB,(+/S< 0 1 3)÷N ⋄ CV←CV,'A' MATCH S ⋄ →0
[23] REALQ:
[24]  UDA←UDA,(+/(S←ROLL N)< 0 1 3)÷N
[25]  UDB←UDB,(+/(S←QUANTUM S)< 0 4 6)÷N ⋄ CV←CV,'A' MATCH S
[26]  UDA←UDA,(+/(S←ROLL N)< 0 4 6)÷N
[27]  UDB←UDB,(+/(S←QUANTUM S)< 0 1 3)÷N ⋄ CV←CV,'A' MATCH S ⋄ →0
[28] REALM:
[29]  UDA←UDA,(+/(S←ROLL N)< 0 1 3)÷N
[30]  UDB←UDB,(+/(S←T TQUANTUM S)< 0 4 6)÷N ⋄ CV←CV,'A' MATCH S
[31]  UDA←UDA,(+/(S←ROLL N)< 0 4 6)÷N
[32]  UDB←UDB,(+/(S←T TQUANTUM S)< 0 1 3)÷N ⋄ CV←CV,'A' MATCH S
    ∇
```

HOTBRAY LTD.

57 Acre Lane • London SW2 5TN

ANALYST / PROGRAMMER (APL)

THE JOB

- Reporting directly to the General Manager, to take complete charge of our multi-user micro system using APL. Main duties will include:

  Analysis and further development of current system.

  To research requirements and make proposals for the replacement of existing hardware.

  To liaise with major car and truck manufacturers to ensure system compatibility.

  System backup and maintenance.

THE COMPANY

- A highly profitable, long established Trading House exporting automotive products worldwide. The products and the marketplace are very competitive and the company has to operate with sophisticated systems support.

QUALIFICATIONS

- APL knowledge with preferably experience in multi-user micro systems. A desire to work in a commercial environment.

THE REWARDS

- A highly negotiable salary + bonus.

For further details and an application form please contact Richard Hall, telephone 01 737 0044 or send your C.V. to the above address.

The British APL Association Survey of Members

Analysis by Christine McCree and Dick Bowman

Opinion by Dick Bowman

It's important that the BAA committee have some feeling for who are our members and for what the membership wants, because what we dream up in the seclusion of committee meetings can represent the ideas of a non-typical section of the membership.

So we thought we'd ask the membership a few questions (people who send back replies are another non-typical section – if what we find doesn't reflect your views and you didn't reply to the questionnaire you know who to blame); we did something similar a few years ago to try and see what we could do about meetings, this time we cast the net wider.

The results are presented in percentage terms as far as possible, not all adding up to 100% for the customary reasons, and we also show the total number of responses for each question.

Comments on the results are Dick Bowman's alone. To get the analysis printed as soon as we could meant that there wasn't time for a committee consensus to be gathered. In essence the committee would like to run the Association the way the members want, but sometimes it is difficult to choose between alternative ideas. I hope these comments will give you some better idea of what I believe we need to do. At the time of writing this there has been no committee discussion of the results – but it should be in progress by the time this appears. This is your Association and if you want things to happen your way – participate (in whatever way you can).

General Information

What type of BAA Member are you?

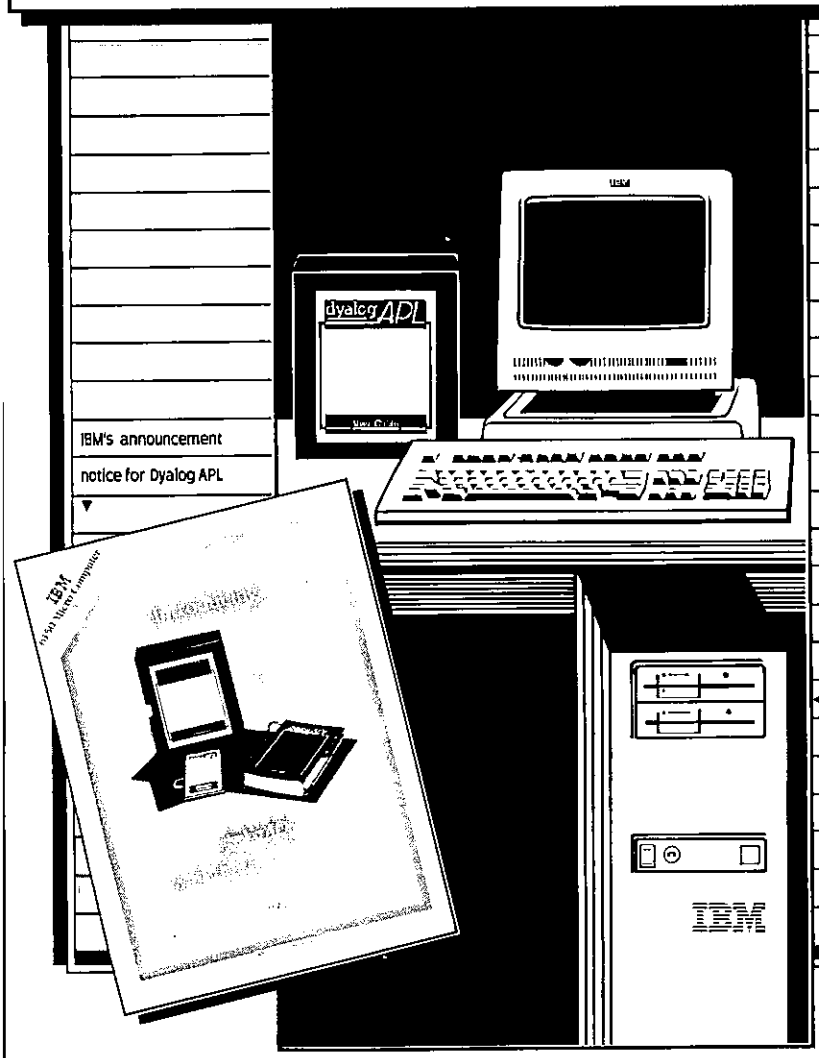
Total Replies	**88**
Individual UK members	63%
Individual overseas members	22%
UK corporate	11%
Other (US corporate)	2%
(Student)	2%
(European corporate)	1%

Fairly obviously the British APL Association isn't just British; we have a significant foreign membership and for this to continue we need to offer a blend of services which satisfies all. Interestingly we have no replies from Sustaining Members.

dyalog APL

IBM 6150

IBM ANNOUNCES DYAL



IBM's announcement notice for Dyalog APL

◀ IBM 6150 Model 125

4.5 MIPS

High Speed Floating Point (1650 Kwips)

High Performance disk (1.08 Mb/sec)

Dyadic is pleased to announce another distributor for Dyalog APL

Dyalog APL is a second generation APL that includes nested arrays, defined operators and other APL2 extensions. The 6150 implementation also provides support for standard IBM consoles, terminals and printers, graphics facilities and an interface to SQL/6150.

OG APL FOR THE 6150

IBM UK outlets for **Dyalog APL**

Over 66 authorised IBM 6150 dealers and your local IBM branch.

Announced as a **Vendor Logo** product by:

IBM Austria
IBM Belgium
IBM Denmark
IBM Finland
IBM France
IBM Germany
IBM Iceland
IBM Ireland
IBM Israel
IBM Netherlands
IBM Norway
IBM Portugal
IBM Saudi Arabia
IBM Spain
IBM Sweden
IBM Switzerland
IBM Turkey
IBM UK

Find out why **IBM** chose **Dyalog APL**

dyalog APL

Sales Department
Dyadic Systems Limited
Park House, The High Street, Alton, Hampshire, GU34 1EN, UK
Tel: (0420) 87024

When did you join?

Total replies	82
Before 1984	32%
1984	22%
1985	22%
1986	20%
1987	5%

No great surprise here.

What other APL interest groups do you belong to?

Total replies	23
SIGAPL	52%
SWAPL	13%
NY/SIGAPL	9%
Others (1 reply each)	26%

Other groups named included New England, FinnAPL, Swiss, Quebec, APLBUG and QL/APL. Interestingly only around 25% of our respondents belong to any other group.

What professional bodies do you belong to?

Total replies	82
British Computer Society	20%
(Statistics combined)	17%
ACM	12%
(Actuaries combined)	10%
(Accountants combined)	10%
(Operations Research combined)	10%
(Mathematics combined)	6%
IEEE	5%
(Economics combined)	5%
(Other computing combined)	4%
(Mechanical Engineering)	2%

We combined bodies which represented similar interests; multiple memberships were common. Wouldn't it be nice to have these percentages symmetrical (20% of BCS members also joining BAA)? Clearly we see a tendency for APLers to be involved with disciplines outside of computing.

What implementations of APL do you use?

Total replies	**160**
STSC APL*PLUS/PC	28%
VSAPL	19%
STSC APL*PLUS/mainframe	10%
APL2	9%
APL.68000	9%
Sharp APL/mainframe	7%
Dyalog	7%
IBM APL/PC	3%
None	3%
VIZ::APL	3%
Sharp APL/PC	2%
Other (1 reply each)	3%

(STSC/UNIX, PortaAPL, CDC, Siemens)

Access to multiple APLs was common and it's nice to see that we can attract members who have no APL. I personally was surprised by the level reached by APL*PLUS/PC; are these people brought into APL by the PC and the product, or are they longstanding APLers using the PC alongside a mainframe implementation or even deserting other APLs completely. We make no claims for this being any form of market share survey – outraged vendors please note; but the BAA would be more than happy to receive and publish a comprehensive APL market study.

What areas do you work in?

Total replies	**153**
Management Information Systems	32%
Finance	24%
Scientific/Engineering	21%
Education	7%
Others	16%

Again, multiple responses were common, and the spread is good; we can feel assured that APL is able to cross application boundaries (OK, we all knew it – now we have some evidence). We got 15 'other' areas.

How many people are there in your company using APL?

Total replies	**85**

None	4%
1	25%
2-4	24%
5-10	15%
More than 10	33%

We even got a multiple response here. The saddle effect isn't surprising, what's gratifying is just what a high proportion of respondents fell into the highest category. Maybe we should also have asked for an exact number – just what is the highest number in any one organisation?

Are they all BAA members?

Total replies	**84**
Yes	37%
No	63%

Clearly we have some way to go before BAA membership is seen as an essential part of being an APL user; remember also that this survey couldn't contact places where the BAA has no member.

Is the use of APL in your company increasing/decreasing/static?

Total replies	**83**
Increasing	33%
Decreasing	12%
Static	55%

This looks pretty good.

If the use is decreasing, what is replacing APL?

Total replies	**22**
4GLs	32%
Lotus	27%
Symphony	18%
Other	23%

'Other' included Cobol and RPG! Note the low response level – we appear to face no 'universal APL killer' and I wonder whether we need to get too upset about the nature of some of the work going elsewhere (was it that good a match for APL originally?).

Do you use APL at home/work?

Total replies	**112**
Home	32%
Work	68%

Introducing our new membership category – domestic members! Perhaps I should charge twice the individual member rate until you show me the receipt for your home APL interpreter. My congratulations also to the person who uses APL2 'at home'. One thing that's missing in this analysis is any attempt to correlate replies between questions – it ought to be done really, and volunteers are welcomed.

Vector	**Quantity**				**Quality**		
	Total	**More**	**Same**	**Less**	**Total**	**Int.**	**Dull**
Technical articles	76	69%	39%	0%	69	88%	12%
Application articles	77	57%	38%	5%	65	83%	17%
BAA meeting reports	71	8%	70%	21%	61	59%	41% (b)
Vendor news	74	8%	77%	15%	62	65%	35% (c)
Product guide	75	16%	69%	15%	62	65%	35% (d)
Software reviews	75	48%	51%	1%	58	90%	10%
Hardware reviews	75	40%	52%	8%	62	81%	19%
Book list	70	17%	73%	10%	64	66%	34% (e)
Book reviews	70	29%	61%	10%	62	82%	18% (f)
Public Domain Software	65	62%	32%	6%	52	83%	17% (g)
Competitions	72	15%	58%	26%	60	67%	33% (h)

a) Nice to see Vector going down well, the general tone of response indicating that it well satisfies the technical APLer.

b) Clearly a problem area. What's happening with meeting reports is that speakers are relying on others to pass on their words in writing; what is supposed to happen is that speakers supply their own article to Vector – speaking at a meeting is analogous to presentation of a paper. As a speaker you should not be happy leaving it to someone else to represent what you have to say to the much wider audience of Vector readers?

c) Vendors please note.

d) How do we make a product guide interesting – suggestions?

e) What would make the book list more interesting? Economic reality tells us that the BAA's USP in this area is making publications available which aren't available via normal commercial avenues. More books based around APL would seem welcome.

f) We can only review what's published.

g) Clearly PDSL is going down well, even on the little information you have so far.

h) Competitions are also a problem area – they take a lot of effort to devise and even more to judge.

Suggestions

We received many and they're being looked into; why not put a bit more flesh around your ideas and/or suggest authors (why not yourself)?

How many people read your copy of Vector?

Total replies	**89**
1	56%
2	19%
3-5	13%
More than 5	11%

Caught you; this was a trap question. It's a little disturbing that nearly half the copies of Vector get read by more than one person, because that means that there are lots of people out there getting the benefits for free. The only thing that lets us publish is our income and the freeloaders don't give us any; if you're lending out your copy of Vector you're damaging your own interests because it stops us from growing and improving our service. What I'm happy about is a loan that turns into a membership, but if you pass your copy around the same people time after time perhaps it would be interesting to see what would happen if you stopped. *Ed: Many of us on the committee hold the other view. Please show Vector to more people, not less. If 100 people read each copy of Vector I would be delighted – we'd find it easier to recruit more members and sell more advertisements (which contribute a major part of the cost of producing Vector). Lending out Vector, far from damaging the BAA, does us good, you good and the future of APL good. Ignore Dick Bowman on this please.*

BAA Technical Meetings

		Quantity			**Quality**		
	Total	**More**	**Same**	**Less**	**Total**	**Int.**	**Dull**
Technical information	32	50%	50%	0%	24	92%	8%
Applications	32	50%	44%	6%	24	83%	17%
Software information	40	60%	38%	3%	20	80%	20%
Hardware information	30	33%	50%	17%	21	62%	38%
Panel discussions	29	10%	59%	31%	21	43%	57%
Workshops	28	32%	61%	7%	22	77%	23%
Talks	27	19%	70%	11%	20	70%	30%

Quantity of response about meetings was much lower because many of our respondents couldn't attend. Clearly we need to improve the meetings – the ratings are in general far too low. Panels are particularly bad – the problem here is that we've resorted to a panel discussion when ideas got a bit thin; in theory they should work (get a few people with strong viewpoints to argue), in practice they've tended to have too little preparation put into them. Workshops you can't fake – to be any good everybody knows that they need a lot of work. Talks could also be better prepared, presented and written up in Vector.

What we need here is participation again – lots of you gave us ideas and we'd like to see you take them a step further; the platform's there if you want to use it. If you think things are dull then surely you could do better.

Do you attend the BAA technical meetings?

Total replies	83
Regularly	14%
Sometimes	33%
Never	53%

Meetings are a minority activity, but valuable to those who can attend and should get with material for Vector *Ed: and allow members direct access to committee members.*

If you don't attend many meetings, why?

Total replies	75
Location	48%
Too busy/pressure of work	31%
Not interested	7%
Inconvenient timing	7%
Not relevant	4%
Poor publicity	4%

Given the geographic spread of our membership this isn't very surprising. At least the meeting topics seem relevant.

Would you attend more meetings if we changed the time?

Total replies	59
Yes	29%
No	71%
Evenings	14
Weekends	1
Mornings	1
Different day	2

Perhaps we should try the occasional evening meeting.

Would you prefer meetings to be held in a different location?

Total replies	**41**
Yes	27%
No	73%

The bare numbers suggest we could try somewhere else, your suggestions make it more difficult. I am happy to arrange meetings in San Jose, France, Helsinki, Frankfurt, Texas, New York and Luxembourg provided that all expenses are met by the audience; closer to home we've had meetings in places like Birmingham and I believe we did something in Manchester also. The problem is how many (or few) members we can find within travelling distance of a venue – if you want to set something up let us know and we'll help if we can (we're interested in supporting local groups).

Suggestions

There were plenty. If you were really keen on yours please take it up with the committee by describing it to them in more detail.

Other Areas of BAA Activity

Which existing BAA services are of value to you?

	Total	Value	None
Vector	80	100%	0%
Technical meetings	66	59%	41%
Public Domain Software	61	80%	20%
Other	11		

The main 'other' suggestion was contact with other APL users. We need to improve the value attached to technical meetings particularly – as I said above, it's a seed from which much more of what we do grows.

How would you allocate the annual budget?

	Min	Max	Avg	Act
Vector	10%	100%	41%	30%
Other publications	0%	30%	4%	0%
Regular meetings	0%	30%	12%	17%
Special events	0%	35%	8%	9%
Education	0%	50%	12%	9%
Bare-faced propaganda	0%	50%	8%	4%
Projects	0%	20%	6%	9%
Other	0%	25%	1%	16%
Administration	0%	25%	8%	6%

The 'actual' figures are based on 1986/7 budget estimates (as they were allocated to different categories I have done a reallocation as well as I can). The balance between Vector and meetings is different and I hope that what you've seen above explains something of why this happens. Overall the resemblance between the average and what we actually do isn't at all bad.

If the BAA were to drop one of these topics, which?

Total replies	**48**
Vector	0%
Other publications	19%
Regular meetings	6%
Special events	8%
Education	6%
Bare-faced propaganda	44%
Projects	13%
Other	0%
Administration	4%

Not that we have any plans to drop anything . . . Interestingly two things which haven't really come to fruition are prime candidates for being dropped (other publications and projects) – how do you know what they'll manifest themselves as. Something else we've done very little of is straightforward propaganda. Interesting though that propaganda was the single biggest positive suggestion in the next question. *Ed: perhaps describing it as "Barefaced" encouraged people to disapprove of it and if the questionnaire had offered "Intelligent promotion of APL" there would have been a different response.* It is important that the BAA should decide whether we are happy just to serve the needs of the APL user or whether we should spend effort on spreading the word? How do we get the message of APL across to a wider audience? We (neither the BAA nor APL as a whole) aren't going to grow by being introspective.

Suggestions/comments

Again lots of them, which we'll be following up

The BAA committee is grateful to everyone who took the time and trouble to send us a reply; it's given us a lot to think about. I hope that this analysis will provoke you to further positive action in making the BAA a better organisation.

See page 36 for Prize Draw results

I.P. Sharp Associates

APL ANALYST PROGRAMMERS
c£10-18,000

IP Sharp Associates (IPSA) is an international computer services company with 50 offices in more than 20 countries. IPSA offers advanced APL products within an integrated application development environment and, with over 500 staff, IPSA can justly lay claim to be the world's largest APL development team. APL is the basis for all IPSA application software which covers generic information centre products and utilities as well as specialised financial applications, IPSA pushes APL to the very limits.

Following on from Reuters' acquisition of IPSA, our London office is undergoing a major expansion. We need APL analyst programmers at a number of levels to work on a variety of large applications, mainly related to international trading operations.

The successful candidates are likely to be graduates with a year or more APL experience, preferably in a financial environment. In return IPSA offer the opportunity to broaden your APL experience, to learn about the world of finance and provide real career progression within APL development or by moving into project management or account development.

Please write or phone to Martin Turner, Managing Consultant at IP Sharp Associates, European Headquarters, Heron House, 1 Dean Farrar St., London SW1H 0DX. Tel 01-222 7033.

TECHNICAL SECTION

This section of VECTOR is aimed principally at those of our readers who already know APL. It will contain items to interest people with differing degrees of fluency in APL.

Tree Processing Algorithms

by Anne D Wilson

Incredulity

One occasionally comes across a method of solving a problem that is so simple that one's first reaction is that of incredulity. However, if one pursues the subject the explanation is usually obvious; the data is held in a form that is ideally suited to the questions being asked of it. For example if I want to find which Wilsons have a telephone I look up a telephone directory, not an electoral roll; but the electoral roll would be better if I was looking for all the Robsons in a particular road.

I have written several programs in PL/I and Basic to process data describing trees; there the use of linked lists yielded reasonable algorithms. But when using tree data in APL it was necessary to find another method of storing the data, one that would use the vector processing capabilities of APL. When I stored the data in the way described below as "traipze order", both storing the data initially and processing it brought forth the incredulity mentioned above.

The processes performed on the data include finding the size of a "subtree", copying subtrees, removing subtrees, cutting and pasting subtrees, finding the ancestors of a "node" and finding which nodes are on the same level. Only one of these requires more than three APL statements.

Why Trees

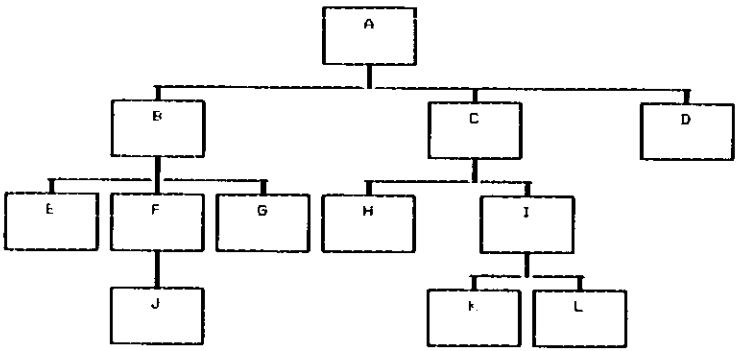
Trees used in computing come in two forms. This article is not about binary trees; they are simpler to process than non-binary trees and are widely used when studying efficiency in storing and accessing data. The trees under discussion here are non-binary ones; they are used to represent program code, description of programs and systems, and real world data. They are used in hierarchical decomposition, that is starting with something large and subdividing it, then subdividing it again, and again . . . . . ; this is how most people manage to address large problems without being lost in a welter of facts.

Any reader familiar with the programming methodologies of Warnier or Jackson will be accustomed to describing programs and data in tree format. With industrial processes where materials are composed of other materials, which in turn are composed of others, trees are a natural way of describing the components. It is possible for a recursive situation to arise when a material contains itself, for example waste paper and waste glass are recycled; compositions of this type can be highly complex. Questions may be asked about the derivation of materials, the components which are "elementary" (that is have no components themselves), and the quantities of these; tree processing programs are used to provide these answers.

Nomenclature

Most people are familiar with the terminology used to describe trees, it is a mixture of biological, genealogical and mathematical terms.

Node	– shown as a box in diagrams
Branches	– lines joining nodes
Terminal node	– node with no children
Root node	– the original node, one and only one always exists
Selecting node	– chooses one of its children
Iterated node	– is repeated
Ancestor	– as in a family tree
Child	– as in a family tree
Parent	– as in a family tree
Subtree	– part of a tree considered on its own, it behaves like a complete tree
Subroot	– the root of a subtree



A Typical Tree

Traversing Trees

Traversing is the word used to describe a logical method of picking some, or all the nodes of the tree, in a predetermined order. The three methods that are used in my tree processing programs are:

level by level
bubble up
traipze order.

Level by level picks out all the nodes on the same level, e.g. E-F-G-H-I are at level 3. This method of traversal is used for printing tree diagrams. Level numbers are transitory, in that the removal or addition of a node affects the level numbers of all nodes in its subtree, so that nodes originally on the same level do not necessarily remain so. It picks out a subset of the nodes.

Bubble up starts with a node and picks out its ancestors, e.g. K-I-C-A and F-B-A. This is used when the derivation of a node is required. It picks out a subset of the nodes.

Traipze order picks all the nodes. Each node is first picked before the nodes in its subtree and before its brother nodes on the right; if it has a non empty subtree it is picked again immediately after the nodes on its subtree, e.g. A-B-E-F-J-(F)-G-(B)-C-H-I-K-L-(I)-(C)-D-(A). This is the order used when moving chronologically through program code, or sequentially through data on a file. It must start with the root node (because the subtree of the root is the entire tree).

Input and Storage of Data

In my APL system I have stored the data in traipze order, with repeated nodes stored once only. Without any loss of generality the data may be considered to be a vector of keys for the purpose of the following discussion. In addition a second vector of LEVEL NUMBERS is kept. As mentioned above the level numbers are transitory, however they can easily be updated, cf the COPY and CUT and PASTE functions given below. The repetition of nodes, such as F and B is not required because their existence is implied by the level numbers, see "size of a subroot".

The input of data in traipze order is very simple. It is assumed that for every node there is a record containing its children; this may be null. Two pairs of vectors are set up:

IDI – data read but not yet processed
LEVI – level numbers for data held in IDI
IDX – data store
LEVX – level numbers for data held in IDX

Storage algorithm

```
1. Set IDX, LEVX as empty vectors
2. IDI = root node (A)
3. LEVI = 1
4. Move first element from IDI to end of IDX
5. Move first element from LEVI to end of LEVX (value l)
6. Read data for the element just moved, if empty go to 9
7. Store the data at start of IDI
8. Add an equal number of elements to start of LEVI, value (l+1)
9. If IDI is non empty go to 4.
```

```
*      IDI          LEVI         IDX                        LEVX
       A            1
       B C D        2 2 2        A                          1
       E F G C D    3 3 3 2 2    A B                        1 2
       F G C D      3 3 2 2      A B E                      1 2 3
       J G C D      4 3 2 2      A B E F                    1 2 3 3
       G C D        3 2 2        A B E F J                  1 2 3 3 4
       C D          2 2          A B E F J G                1 2 3 3 4 3
       H I D        3 3 2        A B E F J G C              1 2 3 3 4 3 2
       I D          3 2          A B E F J G C H            1 2 3 3 4 3 2 3
       K L D        4 4 2        A B E F J G C H I          1 2 3 3 4 3 2 3 3
       L D          4 2          A B E F J G C H I K        1 2 3 3 4 3 2 3 3 4
       D            2            A B E F J G C H I K L      1 2 3 3 4 3 2 3 3 4 4
                                 A B E F J G C H I K L D    1 2 3 3 4 3 2 3 3 4 4 2
```

Building up the Data on Input

Alternate Definitions

The simplicity of the functions required to perform the tree processing is explained by the alternate definitions given below, these interpret the functions in terms of the order of the nodes and their level values.

Level by level – all nodes in the tree with the same level number.

Size of a subtree – all the nodes whose level number is greater than that of the subroot, and follow the subroot until terminated by a brother or uncle of the subroot, whose level will either be equal or greater than that of the subroot.

Ancestors of a node – the last occurrence of each level number for the nodes which precede that particular node.

CUT off a subtree – remove all reference to the nodes belonging to that subtree, as defined under 'size of a subtree'.

COPY a subtree to a new node – take the nodes 'cut off' as described above and insert them after the new node, so that they are added as the first part of its subtree.

CUT and PASTE a subtree – a combination of CUT and COPY, leaving the original subtree unaltered.

Sometimes the subroot is treated as part of the subtree; this depends on the actual problem being solved. The difference can be seen in the functions CUT, where it is left, and COPY where it is moved with the the subtree.

```
      FIG1

    SIZE←SUBTREE⍋SIZE N
    -------------------

[1]   ⍝ LOOK ONLY AT THE LEVEL OF RECORDS AFTER THE SUBROOT.
[2]   ⍝ TAKE ALL THOSE UNTIL LEVEL IS EQUAL TO (BROTHER)
[3]   ⍝          OR LESS THAN (UNCLE) THAT OF THE SUBROOT.
[4]   ⍝

[5]    SIZE←→/⍵⍋LEVX[N]<N↑LEVX
      ∇

    ANC←ANCESTORS N
    ---------------

[1]   ⍝ FIND THE INDEX VALUE OF ALL NODES WHICH ARE ANCESTORS OF THE NODE
[2]   ⍝      WITH INDEX VALUE N (INCLUDING ITSELF).
[3]   ⍝ ONLY THE LAST INSTANCE OF EACH LEVEL BELONGS TO AN ANCESTOR.
[4]   ⍝ LOOK ONLY AT THE LEVEL OF RECORDS BEFORE THE SUBROOT.
[5]   ⍝ THE ⌽ IS TO PROCESS THE DATA FROM RIGHT TO LEFT.
[6]   ⍝ THE ⌊ GIVES TO ALL THE NODES OF EARLIER SUBTREES, WHOSE SUBROOT
[7]   ⍝      IS AN UNCLE OF N, THE LEVEL VALUE OF THEIR SUBROOT.
[8]   ⍝

[9]    ANC←(X<¯1↑999,X←⌊⍀LEVX[⌽⍳N])/⌽⍳N
      ∇
```

```
    RC←LEVEL⍋BY⍋LEVEL L
    ------------------

[1]  ⍝ PICK OUT ALL RECORDS WITH THE LEVEL L.
[2]  ⍝
[3]   RC←(LEVX=L)/⍳1↑⍴LEVX
     ∇

    CUT N;SIZE
    ----------

[1]  ⍝ CUT OUT THE SUBTREE OF THE NODE WITH INDEX N, BUT LEAVE N IN.
[2]  ⍝ FIRSTLY FIND THE SIZE OF THE SUBTREE OF NODE WITH INDEX N.
[3]  ⍝
[4]   SIZE←SUBTREE⍋SIZE N
[5]   'NEW IDS ARE ',(N+IDX),(N+SIZE)↓IDX
[6]   'NEW LEVELS ARE ',⍕(N+LEVX),(N+SIZE)↓LEVX
     ∇

    M COPY N;SIZE
    -------------

[1]  ⍝ COPY THE SUBTREE OF THE NODE WITH INDEX VALUE N, INCLUDING N,
[2]  ⍝      SO THAT N IS THE FIRST CHILD OF THE NODE WITH INDEX VALUE M.
[3]  ⍝ FIRSTLY FIND THE SIZE OF THE SUBTREE OF NODE WITH INDEX N.
[4]  ⍝

[5]   SIZE←SUBTREE⍋SIZE N

[6]  ⍝

[7]   'NEW IDS ARE ',(M+IDX),IDX[N,N+⍳SIZE],(M+IDX)

[8]  ⍝
[9]  ⍝ ADJUST THE LEVELS OF THE COPIED NODES.

[10]  'NEW LEVELS ARE ',⍕(M+LEVX),(LEVX[N,N+⍳SIZE]+1+LEVX[M]-LEVX[N]),(M+LEVX)
     ∇

    N CUT⍋AND⍋PASTE N;LEVDIFF;NEWIND;NEWLEV;SIZE;X
    ----------------------------------------------

[1]      ⍝ COPY THE SUBTREE OF THE NODE WITH INDEX VALUE N, INCLUDING N,
[2]      ⍝      SO THAT N IS THE FIRST CHILD OF THE NODE WITH INDEX VALUE M.
[3]      ⍝ CUT OUT THE SUBTREE OF THE NODE WITH INDEX N, INCLUDING N.
[4]      ⍝ FIRSTLY FIND THE SIZE OF THE SUBTREE OF NODE WITH INDEX N.
[5]      ⍝ AND CHECK THAT IT IS NOT PASTED INTO ITSELF.
[6]      ⍝

[7]       SIZE←SUBTREE⍋SIZE N
[8]       →(~M∈N+⍳SIZE)/L10
[9]       'CANNOT CUT AND THEN PASTE INTO THE VOID'
[10]      →0
[11] L10: ⍝ N ITSELF IS TO BE MOVED, SO ADD 1 TO ITS SUBTREE SIZE.

[12]      ⍝ CALCULATE THE AMOUNT TO ALTER THE LEVEL NUMBERS.

[13]       LEVDIFF←1+LEVX[M]-LEVX[N]

[14]      ⍝
[15]      ⍝ BUILD UP THE INDEX IN NEWIND, REFERRING TO THE OLD TREE.

[16]       NEWIND←(⍳N-1),(N+SIZE)+⍳⍴LEVX
```

```
[17]    ⍝
[18]    ⍝ FIND WHERE M OCCURS IN THE NEW INDEX.

[19]     X←NEWIND⍳M

[20]    ⍝
[21]    ⍝ INSERT N AND ITS SUBTREE AS MS FIRST CHILD.

[22]     NEWIND←(X↑NEWIND),(N,N+⍳SIZE),X↓NEWIND
[23]     'NEW IDS ARE ',IDX[NEWIND]

[24]    ⍝
[25]    ⍝ SIMILARILY BUILD UP THE NEW LEVEL NUMBERS, ADJUSTING FOR NS SUBTREE
        ⍝
[26]     NEWLEV←LEVX[NEWIND]
[27]     NEWLEV[X+⍳1+SIZE]←NEWLEV[X+⍳1+SIZE]+LEVDIFF
[28]     'NEW LEVELS ARE ',⍕NEWLEV
     ∇
```

FIG2
SIZE OF SUBTREE:-

```
of A - 11
of B - 4
of E - 0
of F - 1
of J - 0
of G - 0
of C - 4
of H - 0
of I - 2
of K - 0
of L - 0
of D - 0
```

FIG3
NODES ON:-

```
level 1 - A
level 2 - BCD
level 3 - EFGHI
level 4 - JKL
```

FIG4
ANCESTORS:-

```
of A - A
of B - BA
of E - EBA
of F - FBA
of J - JFBA
of G - GBA
of C - CA
of H - HCA
of I - ICA
of K - KICA
of L - LICA
of D - DA
```

FIG5
CUT SUBTREE:-

```
remove subtree of A -
     New ids are A
     New levels are 1
remove subtree of B -
     New ids are ABCHIKLD
     New levels are 1 2 2 3 3 4 4 2
remove subtree of E -
     New ids are ABEFJGCHIKLD
     New levels are 1 2 3 3 4 3 2 3 3 4 4 2
remove subtree of F -
     New ids are ABEFGCHIKLD
     New levels are 1 2 3 3 3 2 3 3 4 4 2
remove subtree of J -
     New ids are ABEFJGCHIKLD
     New levels are 1 2 3 3 4 3 2 3 3 4 4 2
remove subtree of G -
     New ids are ABEFJGCHIKLD
     New levels are 1 2 3 3 4 3 2 3 3 4 4 2
remove subtree of C -
     New ids are ABEFJGCD
     New levels are 1 2 3 3 4 3 2 2
remove subtree of H -
     New ids are ABEFJGCHIKLD
     New levels are 1 2 3 3 4 3 2 3 3 4 4 2
remove subtree of I -
     New ids are ABEFJGCHID
     New levels are 1 2 3 3 4 3 2 3 3 2
remove subtree of K -
     New ids are ABEFJGCHIKLD
     New levels are 1 2 3 3 4 3 2 3 3 4 4 2
remove subtree of L -
     New ids are ABEFJGCHIKLD
     New levels are 1 2 3 3 4 3 2 3 3 4 4 2
remove subtree of D -
     New ids are ABEFJGCHIKLD
     New levels are 1 2 3 3 4 3 2 3 3 4 4 2
```

FIG6

COPY:-

```
copy from A to D -
     New ids are ABEFJGCHIKLDABEFJGCHIKLD
     New levels are 1 2 3 3 4 3 2 3 3 4 4 2 3 4 5 5 6 5 4 5 5 6 6 4
copy from B to L -
     New ids are ABEFJGCHIKLBEFJGD
     New levels are 1 2 3 3 4 3 2 3 3 4 4 5 6 6 7 6 2
copy from E to K -
     New ids are ABEFJGCHIKELD
     New levels are 1 2 3 3 4 3 2 3 3 4 5 4 2
copy from F to I -
     New ids are ABEFJGCHIFJKLD
     New levels are 1 2 3 3 4 3 2 3 3 4 5 4 4 2
copy from J to H -
     New ids are ABEFJGCHJIKLD
     New levels are 1 2 3 3 4 3 2 3 4 3 4 4 2
copy from G to C -
     New ids are ABEFJGCGHIKLD
     New levels are 1 2 3 3 4 3 2 3 3 3 4 4 2
copy from C to G -
     New ids are ABEFJGCHIKLCHIKLD
     New levels are 1 2 3 3 4 3 4 5 5 6 6 2 3 3 4 4 2
copy from H to J -
     New ids are ABEFJHGCHIKLD
     New levels are 1 2 3 3 4 5 3 2 3 3 4 4 2
copy from I to F -
     New ids are ABEFIKLJGCHIKLD
     New levels are 1 2 3 3 4 5 5 4 3 2 3 3 4 4 2
copy from K to E -
     New ids are ABEKFJGCHIKLD
     New levels are 1 2 3 4 3 4 3 2 3 3 4 4 2
copy from L to B -
     New ids are ABLEFJGCHIKLD
     New levels are 1 2 3 3 3 4 3 2 3 3 4 4 2
copy from D to A -
     New ids are ADBEFJGCHIKLD
     New levels are 1 2 2 3 3 4 3 2 3 3 4 4 2
```

FIG7

CUT AND PASTE :-

```
cut A and paste on D -
     Cannot cut and then paste into the void
cut B and paste on L -
     New ids are ACHIKLBEFJGD
     New levels are 1 2 3 3 4 4 5 6 6 7 6 2
cut E and paste on K -
     New ids are ABFJGCHIKELD
     New levels are 1 2 3 4 3 2 3 3 4 5 4 2
cut F and paste on I -
     New ids are ABEGCHIFJKLD
     New levels are 1 2 3 3 2 3 3 4 5 4 4 2
cut J and paste on H -
     New ids are ABEFGCHJIKLD
     New levels are 1 2 3 3 3 2 3 4 3 4 4 2
cut G and paste on C -
     New ids are ABEFJCGHIKLD
     New levels are 1 2 3 3 4 2 3 3 3 4 4 2
cut C and paste on G -
     New ids are ABEFJGCHIKLD
     New levels are 1 2 3 3 4 3 4 5 5 6 6 2
```

FIG7 Continued

```
cut H and paste on J -
     New ids are ABEFJHGCIKLD
     New levels are 1 2 3 3 4 5 3 2 3 4 4 2
cut I and paste on F -
     New ids are ABEFIKLJGCHD
     New levels are 1 2 3 3 4 5 5 4 3 2 3 2
cut K and paste on E -
     New ids are ABEKFJGCHILD
     New levels are 1 2 3 4 3 4 3 2 3 3 4 2
cut L and paste on B -
     New ids are ABLEFJGCHIKD
     New levels are 1 2 3 3 3 4 3 2 3 3 4 2
cut D and paste on A -
     New ids are ADBEFJGCHIKL
     New levels are 1 2 2 3 3 4 3 2 3 3 4 4
```

Faster File Input In TSO

by John Sullivan

After talking to various people at the February BAA meeting it is obvious that the following notes need circulation to a wider audience. One of the biggest problems with APL is the large amount of CPU time required to read a single record from a QSAM file into the workspace through auxiliary processor 111. This paper sets out to find a way of speeding this process up and at the same time make use of APL's array handling facilities.
The standard method of file processing in FORTRAN, COBOL and the like is as follows:

Read a single record
Process it
Repeat until end of file

which works fine because these languages are scalar oriented in that they can only work on one value at a time and they are compiled so that looping code can run very fast. In APL we prefer to save our values until we have either got all of the file into the workspace or, if the file is too big, as much of it as possible to reduce the looping process to the minimum. If we use AP111 as described in the APL2 System Services Reference manual we get:

Read a single record
Read another single record
. . . etc until we can't get any more into the workspace
Process what we've got
Repeat until end of file

which still contains a loop that is executed once for each record in the file.

When I first wrote code like this and ran it, I seemed to be waiting for ages for the function to finish running so I decided to time it. I discovered that in the following program the loop takes about 3 milliseconds to execute and any other processing included within this loop took much less time than the file read operation.

```
      ∇TEST;I;R
      ...
      ⍝ Share variable R with AP111
      ...
[n-1]  I←1
[n]    LABEL:X←R
[n+1]  →(1000≥I←I+1)/LABEL
      ...
      ∇
```

When I tested this function on files with different logical record lengths I discovered something that seemed strange to me at the time and is the key to what follows: it doesn't matter how long the record is; it always takes AP111 about 3 milliseconds to read it. (The first test was of the average of 1000 reads which could have concealed wide variations, but timing individual record read operations showed that the variations were minor – 3 milliseconds is reasonably constant).

Having discovered that it takes just as long to read 10 bytes into the workspace as it does to read 10K bytes, my next thought was 'How do I read multiple records at a time into the workspace?' The obvious way is to make the logical record length of the data the size of the space available for processing. Although this is great in theory, it is impossible in practice for various reasons, viz:

- In MVS, the largest block size (and hence the largest record size for fixed length records) is 32600 bytes.
- You must have enough sharespace allocated to hold the record when it is being read from the file and passed to APL. This comes out of your available region size, effectively reducing the size of your active workspace.
- The owner of the data might object to your reformatting of his/her dataset and the DP department might object to your request for a couple of hundred megabytes to duplicate a dataset in your desired format.

We can get around these problems by using the Great British Compromise. If we override the values in the dataset control block (DSCB) then we do not need to reformat or copy the data. However, we cannot change the block size so we might not read as much data at a time as we could. We could consider various options:

- Specify the logical record length equal to the block size. Objections:
  - with fixed length records this will usually give a wrong-length record error on the last block.
  - with variable length records you must make the LRECL 4 bytes less than the blocksize to allow for the block-length prefix.
- Our first thought for getting round these objections was to specify the record format as VB (varying-length records, blocked). This doesn't work with fixed length record files (you get an 0C4 abend) because the system is looking for a block-length prefix and of course it doesn't find one.
- The final chance is to specify the record format as U (undefined). This works, for both fixed and varying-length records, although you have to know the type of records in the file in advance since processing is different for fixed length records and varying length records. However you can write a generalised program or a set of programs to do the work for you and if your system has the APL DDI and APL DSI commands everything can be automated.

In order to override the record format on the DSCB you use one of the following sequences of commands via AP100:

In JCL you code

```
//ddname DD DSN=dsname,DISP=SHR,DCB=RECFM=U
```

For TSO/E you can issue the following command

```
ALLOC DD(ddname) DS(dsname) SHR REUSE RECFM(U)
```

Otherwise for TSO you will have to use the following sequence

```
FREE DD(ddname attrname)
ATTR attrname RECFM(U)
ALLOC DD(ddname) DS(dsname) SHR USING(attrname)
```

If the dataset you are trying to read was set up with blocksize 0 then you will also have to specify the blocksize. You can do this even if the blocksize is not defined as zero, as long as the new blocksize is greater than or equal to the old blocksize. This is because the value you specify tells AP111 how much buffer space to use, and extra buffer space is immaterial even though it is never used. Although TSO/E should accept the keyword combination SHR BLKSIZE(value) on the ALLOC command, in fact this does not allow ALLOC to complete properly. Thus in TSO/E you code:

```
FREE DD(attrname)
ATTR attrname BLKSIZE(32600) RECFM(U)
ALLOC DD(ddname) DS(dsname) SHR USING(attrname)
```

If you haven't got TSO/E you use the same code as before, replacing the ATTR command with the new one just shown. Note that we specify the maximum block size that MVS can handle here; this means that our code will work for any dataset, although we might have an awful lot of unused buffer space. Of course, if you have it, you could use the APL DSI command to determine the DSCB values and use those. APL DSI is also useful for determining the record format as mentioned above, so that you can deal with the data appropriately – see later on for amplification of this.

Of course, once you have got the data into the workspace it is up to you to process it in whatever way is appropriate for your application.

It should be emphasised that this method of deceiving the system into believing that you are reading a file with different attributes from those declared in the data control block is not peculiar to APL but is a feature of the operating system. It is a feature that IBM are not exactly over-generous with references to in their manuals. I have looked through several of them and I have only come up with the following references, for those of you who want to develop this line of thought.

PL/I Optimising Compiler Programmer's Guide, Chapter 4 (Data Sets and Files), pp 117-118

OS/VS2 MVS JCL pp 203-215 (The DD statement, DCB parameter)

Another Useful Idea

Try the following. You have a dataset with the LRECL=80, BLKSIZE=6400, RECFM=FB. Allocate it in such a way as your program thinks it has LRECL=6400 (not difficult; override the DCB parameters in a similar manner to that shown above). Then open the dataset using AP210. What do you think you can do? . . That's right, you can read individual blocks in random order by specifying their relative record numbers in the control variable, and , what's more, you can update them in place. This is especially useful if your dataset is big (with a few hundred thousand records, for example) and you have only one or two amendments to make. Of course if you know you are going to want to do this you will pad out the last block of the dataset with blanks to the full 6400 bytes, won't you (because you can't access it if you don't).

Text Inequalities Under Dictionary Ordering

a note by Robert Pullman

A domain error results if one tries to apply greater than or less than (with or without 'or equals') to character data. As character data is often sorted in dictionary order it seems legitimate to allow such tests on character data. This is not to suggest that the primitives should do this directly. Such comparisons are bound to be application dependent.

When extended grade is available it is very easy to write defined functions to do these tests. They may use the ⎕AV sequence or allow a collating sequence as an argument.

The example extends the idea by making multiple comparisons at one time (putting one set of lines into a matrix). To keep it simple I have left out code to force the same number of columns in the matrix A and vector B.

```
C←A GT B
C←⍋ ⎕AV ⍋ A ⍪ B
C←(¯1↑C)>¯1↑C
```

Line [1] catenates the vector B onto the end of A, then grades the new matrix and puts the result in C.

Line [2] compares the grade for line B with the grade for each other line. As the grades are numeric there is no problem with the comparison primitives. The result is boolean, with a 1 for each line of A that is greater than B. Since B is catenated onto the end of A lines of a that are equal to B will be sorted before B and so will give a zero in the result.

Here is another example, this time for greater than or equal to.

```
C←A GE B
C ← ⍋ ⎕AV ⍋ B ⍪ A
C ← (1↑C) > 1↑C
```

By catenating A onto B the lines that are equal in A will be graded after B and so will give the required 1 in the result.

This plan can easily be modified to allow a collating sequence other than ⎕AV or even to replace the dyadic grade with a user defined function if the APL in use does not have that feature built-in.

It would not be difficult to cater for scalars or vectors in either argument. Then character arguments can be used in expressions such as:

```
LESS THAN IS LIKE GE WITH < INSTEAD OF >
LESS THAN OR EQUAL TO IS LIKE GT WITH < INSTEAD OF >

A←(A LT B) ≠ A

D ← ((A GE B)∧A LE C)≠A
```

Extending IBM's Logic Algorithms in Dyalog

by Allan G. Prys Williams

Department of Management Science and Statistics
University of Wales, Swansea

Abstract

The algorithms developed by Brown *et al* for use in APL2 can be converted to run in Dyalog APL. Various minor improvements are possible. A by-product of this is a set of functions to convert nested arrays of logical data into simple character vectors. These are developed to provide ways of testing logical data for consistency with notational rules, correcting errors in data entry, and recovering formal-logic versions of APL logic data.

1. Simple Substitutions

1.1 The Original Material

In their APL86 tutorial [1] James Brown and his colleagues give a very useful and comprehensive account of the ways and means of embedding artificial intelligence algorithms into APL2. Their reference for the underlying theory is the book by Charniak and McDermott [2] and I have indeed found that a very effective way of learning about AI is to work through their APL functions and examples first, and then read Charniak once one can visualise, from the APL, what is going on. However, since my own computer is an RT running Dyalog APL, it was necessary to adapt the IBM functions to fit. Not only are the dialects different, but the published functions contain IBM proprietary ⎕functions that had to be circumvented.

Thus the first section of this paper contains an account of the minimal adaptations needed to get the IBM system running under Dyalog, in the sense of doing what Brown *et al* say in their tutorial that it should do. The rest of the work consists of improvements (perhaps) and adaptations.

Throughout, I have had to assume some familiarity with the underlying ideas. The original tutorial runs to some sixty pages, and I can't repeat more than a fraction of it. I have, however, tried to ensure that someone who knows AI, but not the tutorial, will have a clear idea of what is going on.

1.2 Data Structures

The data structures used in AI work are nested character vectors of considerable complexity. In IBM's terminology the highest level is the *database*, which is a vector of *clauses*, each of which is taken to be asserted. A clause is a two element vector of *lists* of *predicates*:

```
A,B,... if P,Q,...
```

which is interpreted as: "at least one of A,B,... is true if all of P,Q,... are true", or as "either at least one of A,B... is true or at least one of P,Q,... is false". These interpretations are equivalent in classical logic. Brown *et al* prefer the second, which brings out the rule that an unconditional fact is coded by a clause with the empty character vector as the second element (with statements of falsity going the other way round).

A predicate is a statement which is essentially indivisible, such as:

```
Socrates is a man
```

which is made up of a *relation* 'is_ a_man' and a *term* 'Socrates'. The general form is:

```
relation term1 term2 ....
```

where terms may be variable or fixed. Predicates are the building blocks of logic statements. Thus the clause asserting that "all men are mortal" is represented semi-formally as:

```
all Xs are mortal if all Xs are men
```

being the pair

```
<is_mortal X>,<is_man X>
```

In APL all *variables* (the 'X's inside predicates) get a 'delta' in front of them, and 'is' and 'are' disappear. Thus typical predicates are:

```
'man' 'socrates'
 'mortal' 'ΔX'
```

while terms can be complex structures

```
'man' ('husband' 'xanthippe')
```

and clauses are always vectors of rank 2, for example:

```
('married' 'ΔX' 'ΔY')    ('married' 'ΔY' 'ΔX')
```

means "X is married to Y, if Y is married to X", which is the kind of assertion that often has to be carefully spelled out in logical databases. 'Married' is a reflexive relationship, unlike (say) `'is_husband_of'`. You know this, but the machine has to be told.

1.3 Inference Functions

There are two core functions in the approach to AI that is discussed here. They are *unification* and *resolution*. The job of unification is to compare two statements (usually half-clauses) and say whether they are identical, or could be identical if some or all of their variables took particular values. If so, the UNIFY function should return these values. For instance

```
man socrates
```

unifies with

```
man X
```

under the substitution

```
X=socrates
```

A more complex APL example is:

```
      X
 ΔX   f ΔX    ΔY
      Y
a ΔZ   g ΔZ
      X UNIFY Y
1    ΔX←'a'    ΔY←('g' 'fa' )   ΔZ←'fa'
```

Resolution takes two clauses and uses unification to find their common ground, if any. Thus (remembering that a simple fact is a clause with an empty second element) the clauses

```
man socrates if <NULL>
```

and

```
mortal X if man X
```

have the resolvant

```
mortal socrates if <NULL>
```

using the same substitution of 'socrates' for 'X'. To give an APL example, the sort of thing that we really want is:

```
    ST2
on  box   table
    ST3
by  ΔZ   ΔY       by   ΔX   ΔY     on   ΔZ   ΔX
    ST2 RESOLVE ST3
  by   box   ΔY       by   table   ΔY       ΔX←'table'    ΔZ←'box'
```

However, there is no assurance, in the absence of special precautions, that the resolvant is not more obscure than the originals.

```
    ST3 RESOLVE ST3
by  ΔX   ΔY       by   ΔX   ΔY     on   ΔX   ΔX     on   ΔX   ΔX       ΔZ←'ΔX'
```

Careful consideration will show that the result is in fact trivially true. The OBVIOUS function in section 2.1 detects such nuisance results.

1.4 The Unification Algorithms

```
     ∇ Z←X UNIFYA Y      ⍝ unify X with Y
[1]    →((0=⍴X)∧(0=⍴Y))/FAIL                ⍝fail if both clauses empty
[2]    X Y←EVAL DEPTH1 (X Y) ◊ →(X≡Y)/GOOD⍝do substitutions
[3]    X Y←(1=≡Y)⌽X Y                       ⍝put atom first, if any
[4]    →((0=≡X)∧(0=≡Y)∧(~X≡Y))/FAIL   ⍝avoid infinite loops of 'a' swop 'b'
[5]    →(1≠≡X)/RECUR                 ⍝if not an atom apply to each; else:
[6]    X Y←('Δ'≡⊃Y)⌽X Y                     ⍝put variable first, if any
[7]    →(~'Δ'≡⊃X)/FAIL               ⍝if no variable, items are different
[8]    →(X SEARCH Y)/FAIL
[9]    ⍎X,'←Y' ◊ →GOOD                 ⍝do substitution
[10]  RECUR:→(~(⍴X)≡⍴Y)/FAIL
[11]   →(1=X(UNIFYA UNTIL 0)Y)/GOOD
[12]  FAIL:Z←0
[13]  GOOD:Z←1

     ∇ Z←X UNIFY Y;T;USUBS
[1]    ⍝Unification main algorithm: Z is 2-item vector
[2]    ⍝<0 or 1 for failure or success> <the substitutions>
[3]    T←⎕EX'Δ'⎕NL 2 ◊ Z←X UNIFYA Y
[4]    T←↓'Δ'⎕NL 2 ◊ →(0=⍴USUBS←T)/END
[5]    USUBS←(T⍳',','←'), SCRIPT ⍝ T
[6]   END:Z←Z USUBS
```

Except for a certain amount of compression, these are the same as in [1] apart from lines 8 and 11 of UNIFYA and lines 4 and 5 of UNIFY. Sameness is modulo direct 'machine translation' between APL2 and Dyalog in as much as the same functions sometimes have different symbols, of which by far the commonest in this particular workspace is 'disclose':

```
↑ in APL2 ; ⊃ in Dyalog.
```

Line 11 of UNIFYA is a speeded-up version according to IBM's own suggestion. Line 8 is the 'occurs check' to avoid perverse substitutions. In the original, this uses the idiom 'epsilon-underline monadic epsilon', neither element of which occurs in Dyalog. An equivalent effect, using a different principle, is given by:

```
     ∇ Z←X SEARCH Y      ⍝models APL2 'X∈∈Y' for use in the UNIFYA function
[1]    Z←~(''≡DEPTH1 Y)≡(X≡DEPTH1 Y)
```

In line 9 of UNIFYA substitution is made by giving a value to a particular variable. This value is stored as an instruction inside USUBS, so that the variable can be deleted at the start of the next call to UNIFY. The IBM original uses

```
USUBS←2 ⎕TF c[2]'A' ⎕NL 2
```

which is a proprietary function. The Dyalog XFER functions work very differently, so we introduce

```
     ∇ Z←SCRIPT A;BL;BR;Q;E                    ⍝ A is a CHARACTER variable
[1]    BL←'(' ◊ BR←')' ◊ Q←'''' ◊ E←',c'       ⍝ set up working variables
[2]    ⍝ A is the material, BL, BR the brackets, Q quote mark, E enclosure
[3]    →(''≡DECON A)/NULL ◊ →(1=≡A)/VEC ◊ →(0=≡A)/SCAL ◊ →(2≤⍴A)/REC ◊ →ENC
[4]    REC:Z←BL,(⊃,/SCRIPT A),BR,◊ →0          ⍝ Result is simple vector
[5]    ENC:Z←BL,E,(SCRIPT⊃A),BR ◊ →0           ⍝ Scalar is 1-element vector
[6]    VEC:→(1≥⍴A~' ')/NEXT ◊ Z←Q,(A~' '),Q,' ' ◊ →0
       ⍝ Only one word per vector
[7]    NEXT:→(0=⍴A)/NULL ◊ Z←BL,',',Q,(A~' '),Q,BR ◊ →0 ⍝ No trailing zeros
[8]    NULL:Z←Q,Q ◊ →0                          ⍝ Null clause is empty vector
[9]    SCAL:Z←BL,',',Q,A,Q,BR ◊ →0              ⍝ Scalar is 1-element vector
[10]   ⍝ Replaces APL2 '2 ⎕TF A' for use in the UNIFY function
```

The DECON function cited in line 3 is intended to be an equivalent for IBM's monadic epsilon, i.e. it destroys structure and just returns contents.

```
     ∇ R←DECON X ⍝ Tries to remove all structure from X
[1]    →(1<|≡X)/REC ◊ R←,X ◊ →0 ⍝ Ravels simple objects
[2]    REC:⍎((,⊂)≡⍴X)/'R←''''' ◊ →0' ◊ R←⊃,/DECON X
       ⍝ Deals recursively with complex objects
```

SCRIPT has several intended side-effects, of which more in subsequent sections. Indeed, most of its elaborations are never used during unification, which only involves very simple objects: usually ordinary vectors, sometimes vectors of depth 2. There is therefore no penalty in terms of cpu time. However SCRIPT, though introduced out of necessity in the first place and only extended later, proves to have an amazing range of uses.

The unification algorithms are the core of this sort of AI. It is surprising to what degree any troubles at higher levels tend to refer back to unresolved problems in this group of functions.

1.5 The Resolution Algorithms

The resolution algorithms are a straightforward case of machine translation except for what appears to be a misprint in IBM's RESOLVANT. As it is given in [1], this does not seem to pass on the substitutions as well as the resolvant itself. This omission has no effect on the FORWARD goal- proving function, but it plays havoc with the PROLOG group of functions. We therefore quote:

```
     ∇ Z←AR RESOLVANT BR;T
[1]   ⍝UNIFY AR with BR, Z is 0 or the implied resolution.
[2]   ⍝A & B are global
[3]    Z←0     ⍝Assume failure
[4]   →(0≠T←AR UNIFY BR)⍴0     ⍝return with 0 if no UNIFY
[5]    Z←(((1⊃A)~⊂AR),1⊃B)((2⊃A),(2⊃B)~⊂BR)
[6]    Z←(EVAL DEPTH1 Z)(2⊃T)
     ∇
     ∇ Z←A RESOLVE B
[1]    Z←,(1⊃A)∘.RESOLVANT(2⊃B)
[2]    B A←A B
[3]    Z←Z,,(1⊃A)∘.RESOLVANT(2⊃B)
[4]    Z←Z~0
     ∇
     ∇ Z←A RESOLVEGOAL B
[1]    Z←,(1⊃A)∘.RESOLVANT(2⊃B)
[2]    Z←Z~0
     ∇
```

where line 6 of RESOLVANT is the only significant change.

1.6 Higher Levels

The front-end functions work more or less on sight, after 'machine translation', provided that one has faith. For instance, the depth-first search algorithm

```
     ∇ Z←CGS DF DB;DBI;GI;ASUBS;GSUBS;SG;T;GOAL;NEW
[1]    GOAL ASUBS←2↑⊂CGS           ⍝ divide argument
[2]    SG←SPLITGOAL⊃GOAL           ⍝get simple goals
[3]    →(3>⍴CGS)/START
[4]    GI GSUBS DBI←3⊃⊂CGS
[5]    DBI[(GI-1)⍳⍴DBI]←1 ⋄ →BACK
[6]   START:GI←1 ⋄ GSUBS←(⍴SG)⍴⊂'0' ⋄ DBI←(⍴SG)⍴1 ⍝first goal, initial subs
[7]   NEXT:T←⎕EX'A''⎕NL 2 ⋄ T←⍎DEPTH1 ASUBS GSUBS
[8]    T←EVAL DEPTH1(DBI[GI]⊃DB) ⍝ select next rule
[9]    NEW←T RESOLVEGOAL EVAL DEPTH1 GI⊃SG
[10]   →(0≠⍴NEW)/RES              ⍝ branch if something found
[11]  NEXTDB:→((⍴DB)≥(DBI[GI]←DBI[GI]+1))/NEXT  ⍝ try next rule
[12]   DBI[GI]←1                   ⍝ initialize index again
[13]  BACK:→(0=(GI←GI-1))/Z←0        ⍝ back up to previous goal
[14]   GSUBS[⊂GI]←'0' ⋄ →NEXTDB   ⍝ forget old subs, find another proof
[15]  RES:GSUBS[⊂GI]←⊂2⊃⊃NEW      ⍝ record substitutions
[16]   →(0=⍴1↓2⊃NEW)/NEXTG        ⍝ branch if proved
[17]   T←((VRENAME⊃NEW)[1])(ASUBS GSUBS)DF DB ⍝ do sub-goal
[18]   →(2⊃T)/SGOK                ⍝ branch sub-goal OK
[19]   GSUBS[⊂GI]←'0' ⋄ →NEXTDB   ⍝ forget old subs, find another proof
[20]  SGOK:GSUBS[⊂GI]←,⊂GSUBS[⊂GI],2⊃T ⍝ record new substitutions
[21]  NEXTG:GI+←1 ⋄ →(GI≤⍴SG)/NEXT ⍝ on to the next goal, if any
[22]  DONE:Z←1 GSUBS(GI GSUBS DBI)
```

and its user-friendly front end

```
     ∇ Z←L PROLOG R;T
[1]    →(~''≡⊃L)/START ◊ L←⊂L
[2]   START:Z←1 ◊ →(⊃T←(L'0')DF R)/GOOD
[3]    →Z←0
[4]   GOOD:'VALUES: '(2⊃T) ◊ →(';'≠⊃□)/0
[5]    →(⊃T←(L'0'(3⊃T))DF R)/GOOD
```

otherwise differ from their originals only in lines 14,15 (=19) and 20 of DF and in the beginning of line 1 of PROLOG.

```
      GO2
     john  loves  ΔX     ΔX  is  female
      †DATA2
   john  loves  food
   jane  is  female
   john  loves  jane
      GO2 PROLOG DATA2
VALUES:      ΔX←'jane'

1
      SCRIPT GO2
(''(('john' 'loves' 'ΔX' )('ΔX' 'is' 'female' )))
      †SCRIPT DATA2
((,⊂('john' 'loves' 'food' ))'')
((,⊂('jane' 'is' 'female' ))'')
((,⊂('john' 'loves' 'jane' ))'')
```

Exhibit 1. Use of PROLOG.

Prolog logic progamming[3] assumes that all data can be represented by Horn clauses (clauses with only one element on the left hand side), in which case its system of depth-first searching and backtracking is an extremely effective way of testing existence statements. In exhibit 1, the goal is to test the assertion

there exists an X such that John loves X and X is female

and return with the correct answer 'Jane'. Prolog handles such statements by putting them on the right hand side of a clause. This is equivalent to:

<NULL> if John loves X and X is female

i.e.

not John loves X and X is female

but since X is a free variable, this is equivalent to

for all X not John loves X and X is female

so if a contradiction appears during the search we have

not for all X not John loves X and X is female

which is equivalent in classical logic to

there exists X such that John loves X and X is female

In other words, we add the negation of the thing that we want, look for a contradiction, and report what caused it.

The clever part of the Prolog approach is the ability to forget false starts. In the example, any resolution-based system will lock first of all onto X=food before getting into trouble trying to show that "food is female". Prolog does too, but then discards the first statement

and starts again. Furthermore, the PROLOG cover function allows the user to send the machine back to find other examples, if any. The other method used in [1] is forward chaining. In this instance, we have altered the way it works, and so it is discussed in the next section.

2. Minor Improvements

2.1 Eliminating Tautologies

In the course of resolving clauses, a logic program will on occasion come up with a statement that is true, but trivially so; that is, a tautology. In particular, if the same assertion appears on both sides of a clause, then that clause is a tautology. Remember that a clause holds if *any* element on the left hand side follows from the right hand side. Such a tautology can be detected by:

```
     ∇ Z←OBVIOUS X ⍝ Checks to see if a predicate occurs both sides of X
[1]    →(2≠⍴X)/Z←0                ⍝ Precaution: is X a clause?
[2]    →(1=∨/(⊂'')≡ X)/Z←0        ⍝ Precaution: is X a fact (pos or neg)
[3]    Z←∨/,∋((⍳.≡)/X)            ⍝ Matches each of X[1] to each of X[2]
```

This function is used in the new version of the forward chaining function:

```
     ∇ PZ←GOAL FORWARD DB;NEW;NEW2
[1]    PZ←1 ◊ VCOUNT←1
       ⍝ assume goal will be found. VCOUNT external
[2]    NEW←DB                     ⍝ DB against itself first time
[3]    L1:→((⊂GOAL)∈NEW)/0        ⍝ done if goal is found
[4]    NEW2←NEW∘.RESOLVE DB       ⍝ resolve everything
[5]    DB←DB,NEW                  ⍝ add last set of results to DB
[6]    DB←∪DB                     ⍝ delete duplicates
[7]    NEW←∋⍪/,NEW2               ⍝ bring out new inferences
[8]    NEW←∪NEW                   ⍝ delete duplicates
[9]    NEW←(~OBVIOUS NEW)/NEW     ⍝ attempt to eliminate tautologies
[10]   NEW←VRENAME NEW            ⍝ rename variables
[11]   →(0=⍴NEW)/FAIL ◊ →L1
       ⍝ fail if nothing new, else redo resolution
[12]   FAIL:PZ←0
```

Lines 6 and 8 of FORWARD have been altered to use the Dyalog 'unique' function. Line 9 is a pure addition to eliminate tautologies. Exhibit 2 shows the effects of omitting line 9 to give FORWARD1. The display is achieved by altering the function to print out NEW at each stage. FORWARD generates two items in the first pass and four in the second, before recognising that one of them is the thing it was asked to prove. FORWARD1 generates three items in the first pass and seven in the second, including an annoying wrap-round. The extra items are offspring of the tautology allowed through in the first pass.

For completeness, we also show the effect of using the PROLOG function, remembering to reverse the goal first. PROLOG reports the individual values involved, whereas FORWARD (except when, as here, it has been altered to print out the intermediate working) simply reports that the goal is true. On the RT, PROLOG is more than twice as fast on this problem as the terse version of FORWARD. However FORWARD is much more robust. PROLOG can be thrown simply by permuting DATABASE. Possible precautions are discussed in the next section.

VCOUNT controls the renaming of variables in line 10 of FORWARD and line 17 of DF. The fact that a variable X is mentioned in two different clauses does not mean that it will take the same value in both. Renaming variables ensures that no two machine-generated clauses have the same variables, and so no interference during substitution will occur. In defining the initial database, this may need to be done by hand.

```
      GO
   by  box  window
      †DATABASE
   by  table  window
   on  box  table
   by  ΔZ  ΔY              by  ΔX  ΔY     on  ΔZ  ΔX
      GO FORWARD DATABASE
   by  ΔZ  window        on  ΔZ  table
   by  box  ΔY           by  table  ΔY
   by  box  window
   by  ΔZ  window         on  ΔX  table     on  ΔZ  ΔX
   by  ΔZ  ΔY             by  table  ΔY     on  ΔZ  box
   by  box  ΔY3           by  ΔX  ΔY3     on  table  ΔX
1
      GO FORWARD1 DATABASE
   by  ΔZ  window        on  ΔZ  table
   by  box  ΔY           by  table  ΔY
   by  ΔX  ΔY            by  ΔX  ΔY     on  ΔX  ΔX     on  ΔX  ΔX
   by  box  window
   by  ΔZ  window           on  ΔX  table     on  ΔZ  ΔX
   by  ΔZ  ΔY               by  table  ΔY     on  ΔZ  box
   by  box  ΔY3             by  ΔX  ΔY3     on  table  ΔX
   by  table  window        on  table  table     on  table  table
   by  ΔZ  ΔY               by  ΔX  ΔY     on  ΔX  ΔX     on  ΔX  ΔX     on
   by  ΔX4  ΔY4             by  ΔX  ΔY4     on  ΔX4  ΔX     on  ΔX4  ΔX4

      ΔZ  ΔX
      on  ΔX4  ΔX4
1
      VCOUNT←1
      (0GO) PROLOG DATABASE
VALUES:      ΔY←'window'    ΔZ←'box'    ΔX2←'table'

1
```

Exhibit 2. Two variants of the FORWARD function.

2.2 Stabilising PROLOG

In section 2.1 we stated that PROLOG was better than twice as fast, on a test problem, as FORWARD. But it may fail if we reorder the assertions in the database. To see why, we trace line 2 of DF, which lists the subgoal(s) at each stage. D is a vector of all the orderings of the variable DATABASE as used in exhibit 2.

```
      2 ⍞TRACE 'DF'
      (0GO) PROLOG 1↑D
DF[2]         by  box  window
DF[2]         by  ΔX1  window          on  box  ΔX1
VALUES:      ΔY←'window'    ΔZ←'box'    ΔX1←'table'

1
```

The system decides that in order to prove 'by box window' it must simultaneously prove the next two assertions: the two sub-goals. This it does, and returns the successful values.

But another permutation has to be stopped by an interrupt.

```
     (0GO) PROLOG 3DD
DF[2]          by  box   window
DF[2]          by  ΔX2   window          on  box   ΔX2
DF[2]          by  ΔX3   window          on  box   ΔX3
DF[2]          by  ΔX4   window          on  box   ΔX4
```

The system finds the same subgoals (give or take renumbering) as before. It then finds them again as subgoals of the first subgoal, and goes into an infinite branching process. The first sub-goal is, in full:

```
<NULL> if by X2 window
```

and the machine has the general rule

```
by Z Y if by X Y and on Z X
```

which, together with the current substitutions, gives

```
by box window if by X window and on box X
```

RESOLVE legitimately matches X2 with 'box' and DF tries, yet again, to prove the right hand side of the second expression. This is an explosive process. It will happen for any ordering that presents the general formula before the fact

```
by table window if <NULL>
```

After the final substitution, the remaining checking is safe enough.

One safeguard is always to ensure that all facts come before the general rules. This is good Prolog style, but is not a complete protection. Another possibility is to alter line 17 of the DF function to:

```
[17]   T←((VRENAME⊃NEW)[1])(ASUBS GSUBS)DF 10DB ⍝ do sub-goal
```

This shifts the database along one when testing the subgoals (the only point at which it can be touched), and again when testing sub-subgoals, and so on. It seems to be successful in practice, as long as there is a route through the database and only one line is causing the trouble. Sooner or later that line will arrive at the end and never be reached. It fails if there is no such route at all. An example of an unreachable goal is 'by(box, table)':

```
      CONSTRUE NUMB
~by(box, table)
      NUMB PROLOG DATABASE
 DF[2]          by  box   table
 DF[2]          by  ΔX16  table          on  box   ΔX16
 DF[2]          by  ΔX17  table          on  box   ΔX17
 DF[2]          by  ΔX18  table          on  box   ΔX18
```

This is using the improved version of DF. With an unprovable assertion to be tested, there is no safe route through the database for the permutations to bring to the front. FORWARD also fails to halt (we use the chatty version that prints NEW at each stage):

```
    (0NUMB) FORWARD DATABASE
by  ΔZ  window       on  ΔZ  table
by  box  ΔY         by  table  ΔY
by  box  window
by  ΔZ  window        on  ΔX  table     on  ΔZ  ΔX
by  ΔZ  ΔY           by  table  ΔY     on  ΔZ  box
by  box  ΔY3         by  ΔX  ΔY3     on  table  ΔX
```

Again an interrupt is necessary. The very powerful nature of the general rule means that intermediate rules can be generated ad infinitum. It seems more promising to attack PROLOG again, as the fault here is much more specific.

We first write a function that discovers if the variables have only been relabelled, so that at least one of the new goals is in reality the same as the original.

```
     ∇ Z←X FRAUD Y;T ⍝Determines if X is the same as Y modulo labelling
[1]    Z←0 ◊ →(X≡Y)/0 ⍝ 0 means fraud. Exit if identical
[2]    T←⎕EX'Δ'⎕NL 2 ◊ T←X UNIFYΔ Y ◊ Δ(T≡0)/'Z←1 ◊ →0'
[3]   ⍝ If X and Y do not unify, no fraud. Else look for substitutions
[4]    T←↓'Δ'⎕NL 2 ◊ Δ(0=⍴T)/'Z←1 ◊ →0' ⍝ Precaution against null case
[5]    T←Δ T ◊ →(∧/'Δ'≡⊃T)/0 ⍝ If all substitutions are simple variables
[6]    Z←1 ⍝ then fraud, else pass it as having some content.
```

We then feed it into the function RESOLVEGOAL:

```
     ∇ Z←A RESOLVEGOAL B
[1]    Z←,(⊂A)⍤.RESOLVANT(⊂B)
[2]    Z←Z~0 ◊ →(0=⍴Z)/0 ⍝ Exit if empty, else check for fraudulent subs.
[3]    Z←(∧/(⊂⊃B)FRAUD ⊂⊃⊃Z)/Z
```

This modifies the behaviour of DF and hence that of PROLOG:

```
      (⊕GO) PROLOG 3⊃⊃
 VALUES:      ΔY←'window'    ΔZ←'box'    ΔX22←'table'

1
      NUMB PROLOG DATABASE
0
```

There is little value in also using the permutation suggestion for line 17 of DF as this reduces the consistency of timing over permutations of the database. The negative result takes roughly the same average time as the positive, but is even more consistent. FORWARD shows the least variation in speed, but the average time is twice as long as PROLOG for the positive search. As we saw, the negative search never terminates.

FRAUD allows substitutions of the form 'f(x)' for 'x'. We leave a potential for infinite branching, because it is possible that a complex substitution into a simple rule may give a useful instance.

2.3 Consistency of Notation

```
      X
   p 'a    p  ΔX   q
      Y
 r     p  a
      Y1
 r     p  a
      →C←X RESOLVE Y
      p  ΔX   q  r              p  a   q  r       ΔX←(,'a')
      →C1←X RESOLVE Y1
      p  ΔX   q  r              p  a   q  r       ΔX←(,'a')
      C≡C1
1
      Y≡Y1
0
      YY
 r    p  a
      X RESOLVE YY
```

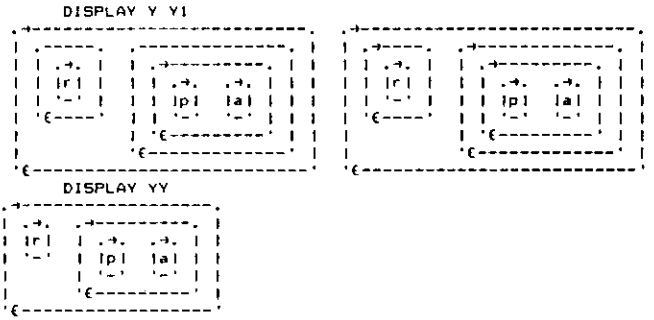


Exhibit 3. Effect of Variations in Notation.

In exhibit 3, we have an example from [1], p.149 (Test 2). Using the 'pure' logic notation in which single letters can stand for words and brackets are used to indicate application, so that 'p(a)' means "the property 'p' holds for the individual 'a'", X stands for:

```
p(varX) or p(a) or q if <NULL>
```

and Y, Y1, YY are all formulations of:

```
r if p(a)
```

so that X RESOLVE Y gives the two immediate resolutions:

```
p(a) or r or q if <NULL>;
```

using the substitution `'varX = a'` to match the first term in X with the right hand side of Y and

```
p(varX) or r or q if <NULL>;
```

matching the second term in X likewise

The same confusion between a one-element vector and a scalar gives rise to the difference between Y and Y1 in exhibit 3. Both do the job and get exactly the the same outcome, but the APL cannot recognise that they are the same. Conversely, it is very hard to spot the difference by eye. It is necessary to use the full DISPLAY function (risking advanced wrap-round if the clause is a long one) and then spot a single character difference in the result.

YY is another attempt to encode the same clause, and follows the rules in that it has two elements, the first containing the consequence and the second the condition. In fact, YY is just 'disclose each of Y' (or of Y1). Unfortunately, it doesn't resolve with X, because the system attempts to match the predicate on its right hand side with the whole of the left hand side of X, instead of with any one individual predicate. The obvious solution is to use 'enclose-each' on YY to get the depth up to that of X, but the result of this is Y, not the more correct Y1. To ensure that every element, at every level, is a vector it is necessary to use 'ravel-jot-enclose-each'.

The problems with the exact way in which statements are to be coded into APL begin at the lowest level. In Dyalog, at least, it is not possible to enclose a scalar, so that when we try to enter separate one-letter statements they tend to stick together.

```
      'equal' 'a' 'b'
equal ab
```

This can be overcome by use of IBM's ENCODE principle by putting an 'a' on the front of everything. However I find that this approach makes ordinary words unreadable.

The choice of rule is governed by the fact that the standard format (thorn) operator gives a one-element vector when applied to a one-digit number. If the unification algorithm is to recognise machine-generated values when they occur in rules, the SCRIPT function must be adapted to match, the line dealing with scalars being:

```
[9]    SCAL:Z←BL,',',⍕A,Q,BR ⋄ →0          ⍝ Scalar is 1-element vector
```

An analogous rule using 'ravel-enclose' ensures that non-simple scalars are turned into one-element vectors.

In general, the system as developed so far is fairly tolerant of any consistent approach to the APL representation of statements. For instance, an individual user is unlikely to use both the forms Y and Y1, while the failure of YY is precisely because of failure to ensure similarity of depth in objects of the same nature. However, I have a nasty feeling that real-life systems will be used by more than one person, so that some form of consistency check seems essential. But in fact I have found that the one that I propose below is life-saving as a diagnostic aid even in single-user conditions.

2.4 The CHECK Function

The essence of the approach is to make use of the fact that SCRIPT has already been constructed to obey all the rules of layout. So powerfully so, that it automatically alters the shape of anything unorthodox that it meets, returning the same data in an improved layout. This motivates the simple-minded function:

```
     ∇ Z←CHECK A
[1]    Z←A≠⍎SCRIPT A
```

```
      DISPLAY C
┌→──────────────────┐
│ ┌→─────────┐      │
│ │ ┌→┐  ┌→┐ │   f  │
│ │ │p│  │q│ │   -  │
│ │ └─┘  └─┘ │      │
│ └←─────────┘      │
└←──────────────────┘
      CHECK C
0
```

```
      DISPLAY ⍎SCRIPT C
┌→──────────────────┐
│ ┌→──────────┐ ┌→┐ │
│ │ ┌→┐ ┌→┐ │ │f│ │
│ │ │p│ │q│ │ │ │ │
│ │ └─┘ └─┘ │ └─┘ │
│ └∈──────────┘     │
└∈──────────────────┘
      CHECK Y
0
      CHECK Y1
1
      Y1←⍎SCRIPT Y
1
      CHECK YY
1
```

Exhibit 4. Use of CHECK and SCRIPT.

As seen in exhibit 4, this enables us to isolate dodgy layout. I have not made it overwrite the original. That would be to assume that an inconsistency necessarily arose from an error in shaping, which may not be the case. For instance, 'execute-SCRIPT' will only occasionally rescue an attempt to enter a single-element vector without precautions. It works in the case of C in exhibit 4 because there is only one such object, but an attempt to use something of the form

```
'p' 'q' 'r'
```

will never recover because SCRIPT will refuse to split the resulting word 'pqr'. It might, after all, be a real word in somebody's universe.

The main use of 'execute-SCRIPT' is to convert more complex objects into their proper form. For instance, it will detect that it is Y rather than Y1 that is imperfect in exhibits 3 and 4, and then convert Y into Y1 without the user having to worry about the exact location of the error. Conversely, SCRIPT was written, or evolved, mainly to do this second job. It is possible that a much simpler function might be adequate for its original purpose as one of the Unification algorithms.

There is no way in which CHECK can spot the error in YY. That is because the inconsistency is with the coding of X, and not a failure to meet some more general principle. If CHECK also 'knew' that the object to be checked was a clause, then it might be able to run a test on the depth of the object. However, in a propositional context (with no variables or individuals involved) the depth of each half of a clause can legitimately be 3, as in YY. It is only if there is at least one variable or individual mentioned somewhere in the database that it is certain that at least one half-clause must have a depth of at least 4. The null statement, which makes up half of every factual statement, has, of course, much lower depth. All of this makes a depth check very far from simple even to visualise.

Another 'invisible' problem is the possibility of different standards for the empty statement. The logic functions will accept many formulations: the empty character vector; the enclosure of it; a vector containing it. It will sometimes fail even to reject inconsistent versions, although the form of the resolvant is then unpredictable. The null statement

ought to be standardised as the empty character vector, which motivates the presence in SCRIPT of the first element of

```
[3]     →(''≠DECON A)/NULL ◊ →(1=≠A)/VEC ◊ →(0=≠A)/SCAL ◊ →(2≤⍴A)/REC ◊ →ENC
        which branches to
[8]   NULL:Z←Q,Q ◊ →0
```

Line 6 of SCRIPT has a similar purpose:

```
[6]   VEC:→(1≥⍴A~' ')/NEXT ◊ Z←Q,(A~' '),Q,' ' ◊ →0
```

It is difficult to spot the difference between the following two statements by eye alone, but the first is probably a fatal error:

```
      ⍎A1←'function' 'word1 word2'
function  word1 word2
      ⍎A2←'function' 'word1' 'word2'
function  word1  word2
      CHECK A1 A2
0 1
```

CHECK detects this because SCRIPT throws out blanks characters within vectors. A user who needs embedded blanks can easily edit SCRIPT to allow them.

Monitoring of cpu time indicates that all this elaboration does not add to the time needed to do logic processing. The substitutions that are SCRIPTed by logic functions are very simple vectors and most lines of SCRIPT are never accessed. SCRIPTing an entire database is something that will only be done interactively and may involve a perceptible delay. It is extensive recursion that is expensive in terms of time.

3. Recovering Logical Form

Even when an expression has been checked for formal consistency, it often far from obvious what it actually says. The DISPLAY function is often needed, and takes up far too much of the screen. Even then, it may not be clear whether the variables are quantified by 'there exists' or by 'for all'. For instance, the expression

```
by(X,Y) if (by(Z,Y) and on(Z,X))
```

really means

```
(for all X)(for all Y)(by(X,Y) if (there exists Z)(by(Z,Y) and on(Z,X)))
```

In other words, I can choose any X and Y that I like, provided I can find some Z (by searching and unification) that fits into the right hand side.

Formal logic uses the standard symbols for *for all* and *there exists*, but they are not available in APL, being the letters A and E printed upside- down and reversed. Other symbols are also used, for instance the variable on its own in brackets may mean *for all*. However, the easiest thing in APL is to use the intersection sign for *for all* and the union sign for *there exists*. The connection is a standard one in model theory and in set theory, and should not be in any way confusing. Thus

```
(∩X)(∪Y)inverse(X,Y)
```

reads

```
for all X, there exists Y such that 'inverse(X,Y)' is true
```

and means that any (arbitrary) object X has at least one inverse (which may or may not be correct, depending on context; it is simply an assertion).

As well as putting in these quantifiers, we also have to choose a standard form in which to represent clauses. We apply the following rules: facts are unadorned; negatives have a *not* sign in front of them (the tilde); two-element clauses are treated as implications rather than as disjunctions of positive and negative elements. We try to minimise the number of brackets, but not at the risk of ambiguity.

First of all, we treat those cases in which the type of expression – database, clause, list, predicate – is taken to be known. Life becomes much harder once we ask the machine to work out the logical type for us.

3.1 Expressions of KnowKnowne

The underlying functions, whose use is shown in exhibit 5, are meant to be used inside a CONSTRUE cover-function that will check the object for correctness and, if not told by a parameter, make a guess at its type. The level-1 function for databases is very simple, merely laying things out for readability and getting rid of the 'del's in the variables.

```
     ∇ R←CONSTRUE1 RA    ⍝ Logical form of database
[1]    R←↑(⊂'∧')⍨∼ CONSTRUE2 RA
```

The level-2 function, for clauses, is much more complicated, since it needs to discover, without external prompting, what variables are present and then decide

```
      CONSTRUE1 DATABASE
by(table, window)
on(box, table)
(∩Z)(∩Y)(by(Z, Y) ← (UX)(by(X, Y) ∧ on(Z, X)))
      CONSTRUE2 ⊃⊃DATABASE
(∩ΔZ)(∩ΔY)(by(ΔZ, ΔY) ← (UΔX)(by(ΔX, ΔY) ∧ on(ΔZ, ΔX)))
      '∧' CONSTRUE3 2⊃⊃⊃DATABASE
(by(ΔX, ΔY) ∧ on(ΔZ, ΔX))
      CONSTRUE4 ⊃⊃⊃⊃DATABASE
by(ΔZ, ΔY)
```

Exhibit 5. Use of CONSTRUE Sub-Functions

whether they belong to one or other side, or to the whole. Most of the work, however, simply consists of editing brackets in and out as needed.

```
     ∇ R←CONSTRUE2 RA;T;V;LH;RH;F                    ⍝Logical form of clause
[1]    T V←DELDREDGE RA              ⍝Find the variables on each side
[2]    LH RH←'∨∧'CONSTRUE3 RA        ⍝Find the logical form of each side
[3]    F←T∩V ◊ R←T∼V ◊ V←V∼T ◊ T←R⍝F is the universals on both sides
[4]   ⍝V is the existentials on the right, T is left hand universals
[5]    ⍝(0≠⍴F)/'F←⊃,/,(⊂''(∩''),F,[1.1]'')'' '⍝Format universals in F
[6]    ⍝(0≠⍴T)/'T←⊃,/,(⊂''(∩''),T,[1.1]'')'' '⍝Format universals in T
[7]    ⍝(0≠⍴V)/'V←⊃,/,(⊂''(U''),V,[1.1]'')'' '⍝Format existentials in V
[8]    →(RH≡'')/fact ◊ →(LH≡'')/neg ◊ →imp ⍝Detect type of clause
[9]   fact:⍝((0=⍴T)∨∼'∨'∈LH)/'LH←1↓¯1↓LH ' ⍝Delete unwanted brackets
[10]   R←T,LH ◊ →0                    ⍝ add universal quantifier(s)
[11]  neg:⍝(∼'∧'∈RH)/'RH←1↓¯1↓RH ' ⍝Delete unwanted brackets
[12]   R←'∼',V,RH ◊ →0                ⍝Construct negative statement
[13]  imp:⍝(∼'∨'∈LH)/'LH←1↓¯1↓LH ' ⍝Delete unwanted brackets
[14]   ⍝(∼'∧'∈RH)/'RH←1↓¯1↓RH ' ⍝Delete unwanted brackets
[15]   ⍝(0=⍴F)/'R←T,LH,'' ← '',V,RH ◊ →0'⍝If no common variables
[16]   R←F,'(',T,LH,' ← ',V,RH,')'  ⍝Construct full-blown implication
```

The auxiliary function DELDREDGE for finding variables is discussed below. If the expression's notation is wrong (especially the wrong null clause) CONSTRUE2 misfires, but usually comes back with a comprehensible Prolog- like expression.

Below this level, variables are treated simply as strings, without bothering about quantification. The level-3 function, for lists of predicates, takes external prompting as to what symbol shall separate the elements of the list.

```
     ∇ R←LA CONSTRUE3 RA
                    ⍝ LA is separator, RA is list of predicates
[1]    ⍎(0=≢RA)/' R←'''' ◊ →0' ⍝Catch the null string
[2]    RA←CONSTRUE4 RA    ⍝ Get logical form of each predicate
[3]    R←'(',(¯3↓⊃,/,RA,[1.1]⊂' ',LA,' '),')' ⍝ Result will have brackets
```

The final function deals with predicates. The standard form of a predicate is: *relation, term1, term2* etc., where any term may be of the form *function, term11, term12,...* This means that CONSTRUE4 has to be recursive. On the other hand, an expression at this level may be a *proposition*, an assertion that does not have a sub-structure. In this case, a 'predicate' may be of depth 1 (and not, in fact, a proper predicate at all).

```
     ∇ R←CONSTRUE4 RA;T⍝ Logical form of predicates
[1]    ⍎(1=≡RA)/'R←RA⍝ '' ◊ →0'⍝ Leave simple strings alone
[2]    R←CONSTRUE4 RA       ⍝ Construct component parts
[3]    T←¯2↓⊃,/,(T←1↓R),[1.1]⊂', '⍝ Get inside terms separated by commas
[4]    R←(⊃R),'(',T,')'     ⍝ Put predicate in front, with brackets
```

Finally, we have the auxiliary functions from CONSTRUE2, the idea being that DREDGE1 dives down into the structure to look for variables and passes them back up to DELDREDGE.

```
     ∇ R←DELDREDGE A     ⍝ Look for variables in A
[1]    R←''              ⍝ Initialise list for variables
[2]    DREDGE1 A         ⍝ Search A. Variables are passed to R
[3]    R←∪R⌿⍨⊂''         ⍝ Delete duplicates and header

     ∇ DREDGE1 A         ⍝ Pass variables back to R in DREDGE
[1]    ⍎(1≥|≡A)/'R,←⊂(''Δ''=⊃A)/A ◊ →0' ⍝ If A simple & variable, keep it
[2]    DREDGE1 A         ⍝ If complex, look one stage lower
```

3.2 Guessing the Type

```
      CONSTRUE RTEST1
(p ∨ q ∨ r) ← s
q ← (p ∧ t)
      ≡RTEST1
4
      CONSTRUE RTEST7
(∩X)(∩Y)(p(X, f(a)) ∨ p(X, f(Y)) ∨ q(Y))
~(∪Z)(p(Z, f(a)) ∧ q(Z))
      ≡RTEST7
¯6
      ≡X
¯5
      CONSTRUE X
(∩ΔX)(∩ΔY)(p(ΔX, f(a)) ∨ p(ΔX, f(ΔY)) ∨ q(ΔY))
      CONSTRUE ⊃X
(p(ΔX, f(a))   p(ΔX, f(ΔY))   q(ΔY))
      CONSTRUE ⊃⊃X
p(ΔX, f(a))
      CONSTRUE ⊃⊃⊃X
p
```

Exhibit 6. CONSTRUE in action.

It is much more difficult to write a function that will recognise the type of assertion that it is dealing with. In fact, the result is long and imperfect. All we can guarantee is that it works on all the test items available, as in exhibit 6, but it is essentially a function that guesses. The reader may recognise examples from [1] pp148-50.

```
     ∇ R←≡LA⍝CONSTRUE RA;L;P              ⍝ Logical version of APL data
[1]    RA←⍎SCRIPT RA       ⍝ Make certain that the notation is reasonable
[2]    →(2≠⎕NC'LA')/guess          ⍝ If level not specified by LA
[3]   branch:⍎(LA=3)/'R←'' '' CONSTRUE3 RA ◊ →0' ⍝ Level 3 needs left arg.
[4]    R←⍎'CONSTRUE',(1 0⍕LA),' RA' ◊ →0     ⍝ Other levels as needed
[5]   guess:⍝ Rough heuristics to estimate the type of object
[6]    ⍎(1=≡RA)/'LA←4 ◊ → branch'    ⍝Only predicates can have depth 1
[7]    L←'∆'⍷⍕DECON RA ◊ P←14       ⍝L predicate logic, P possibilities
[8]    ⍎(∨/(⊂'')≡ RA)/'LA←2 ◊ → branch'    ⍝Only clauses contain ⟨null⟩
[9]    ⍎(∨/'∆'∈ RA)/'LA←4 ◊ → branch'    ⍝Only predicates contain ∆
[10]   ⍎((⊃∨/2≠⍴ RA)∨(L←4)>|≡RA)/'P←P~1'⍝Necessary conditions for database
[11]   ⍎((2≠⍴RA)∨(L←3)>|≡RA)/'P←P~2'    ⍝Necessary conditions for clause
[12]   ⍎((L←2)>|≡RA)/'P←P~3'            ⍝Necessary conditions for list
[13]   LA←⍳/P ◊ →branch             ⍝Make guess upwards
```

If it is given a left argument, then all it does is run the appropriate sub-function on a consistency-checked version of the object. The extra material is its way of guessing. It uses the following principles:

i. Simple vectors only appear at the lowest level.

ii. The null assertion can only appear in a clause.

iii. A variable must be inside a predicate.

iv. A database must be of depth at least 5 (or 4 if propositional) and each of its elements must have exactly two items.

v. A clause must contain exactly two items and be of depth at least 4 (or 3 if propositional).

vi. A list must be of depth at least 3 (or 2 if propositional).

vii. If in doubt, assume maximal complexity.

We remark that this function shows the problem attached to trying to use contrapositives, arguments of the form:

```
(not clause) if (not of rank 2)
```

Of course, they could be added to a logical database as positive implications, by turning them round, but then we have the problem of looking for a *unique* type for our object. Prolog might find a possible value, but how to be sure that it is unique?

3.3 The Need for Consistency

The logical algorithms themselves are very tolerant of variations in notation. For instance, exhibit 1 uses data lifted directly from [1] , section 4.5 at p.125, and it does exactly what it's supposed to. However, when we come to construe the bits, we find:

```
      CONSTRUE DATA2
john(loves, food)
jane(is, female)
john(loves, jane)
      CONSTRUE GD2
~(∪ΔX)(john(loves, ΔX) ^ ΔX(is, female))
```

It is, of course, nice to find that CONSTRUE has still got the structure dead right, but it is clear that it is far less tolerant than PROLOG is. Equally, it would clearly be fatal to mix DATA2 with anything entered according to the standard notation. I suspect that in any real-life situation with more than one user, or even with more than one development programmer, fatal clashes in notation would be very common. It is remarkably hard to be consistent even when on your own, indeed the functions in this section were developed in exactly that situation just to find out what was really going on.

The obvious thing to do is to develop a front-end function that will convert any sentence in formal logic into its proper standard form in APL. But that is another story. It will be far from easy in the general case. Even to take in a simple contrapositive, it must recognise that it will need careful handling and considerable adaptation.

References

1. *Brown et al., 'Algorithms for Artificial Intelligence in APL2' in APL86 Tutorials, ed Camacho, London 1986, pp 87 – 150.*
2. *Charniak, E. and McDermott, D., Introduction to Artificial Intelligence, Addison Wesley, 1985.*
3. *Sterling, L. and Shapiro, E, The Art of Prolog, M.I.T., 1986.*

Appendix: The Remaining Functions.

```
     ∇ Z←EVAL R                          ⍝evaluate logic variables in R
[1]    Z←R                                ⍝initialize result
[2]    →(~'Δ'∊⍕R)/0                       ⍝exit if not a logic variable
[3]    →(2≠⎕NC R)/0                       ⍝exit if no value in variables
[4]    Z←EVAL DEPTH1⍎R                    ⍝replace variables with values
     ∇

     ∇ Z←SPLITGOAL G
[1]   ⍝ G is a single possibly conjunctive goal, Z a vector of simple goals
[2]    Z←(cc''), c c 2⊃G
     ∇

     ∇ Z←VRENAME R
[1]    VCOUNT←VCOUNT+1 ◊ Z←VRENAME1 DEPTH1 R
     ∇

     ∇ Z←VRENAME1 R
[1]    Z←R ◊ →('Δ'≠⍕⊃R)/0 ◊ Z←R,(⍕VCOUNT)
     ∇
```

```
     ∇ Z←(FN DEPTH1)B
[1]    →((0>≡B)∨(1<≡B))/rec
[2]    Z←FN B ◊ →0
[3]   rec:Z←FN DEPTH1 B
     ∇

     ∇ Z←L(F UNTIL THIS)R;I    ⍝EACH THAT QUITS ON TRUE/FALSE
[1]    →(0≠⍴⍴L)/L1               ⍝branch L not scalar
[2]    L←(⍴R)⍴L                  ⍝extend scalar left
[3]   L1:→(0≠⍴⍴R)/L2            ⍝branch R not scalar
[4]    R←(⍴L)⍴R                  ⍝extend scalar right
[5]   L2:Z←I←0                   ⍝initialize result and counter
[6]   LP:→(I≥⍴R)/0               ⍝exit when counter exceeds length
[7]    →(THIS≡Z←(⊃I↓L)F(⊃I↓R))/0 ⍝exit when result THIS
[8]    I←I+1 ◊ →LP                ⍝continue
     ∇
```

Submitting Articles for Vector

by Jonathan Barman and Anthony Camacho

The editors of Vector are all volunteers and carry out their work in their spare time. Everyone is busy, and getting Vector out on time can be a struggle when there is lots of 'real' work to be done. Any help in reducing the load on the editors would be very welcome, and will make the difference between getting an article into the current issue, or having to delay it.

When you send an article to Vector in typescript there may be a great deal to do to it before it is printed. It may be edited first and then it will have to be entered on a keyboard. If there is APL in it that too may have to be entered and tested before being printed out in camera ready form. We try to avoid any manual operations on code: what is printed should be a direct printout of what has just been working.

Then the text has to be sent to be typeset and the APL photographed and stripped into the text in the appropriate positions.

This article is intended to make it possible for authors who have access to a PC or clone to send us material which needs as little attention as possible.

The first thing you can do is to send us both the typescript and the text on a PC disk. This makes the greatest saving in our effort. As there are hundreds of different disk formats we have standardised on the IBM PC format (as has the Public Domain Software Library).

The general availability of the IBM PC and clones, with APL*PLUS PC, makes it easy for text to be swapped and manipulated in a standard way. If everyone submitted articles in a suitable format, then typesetting can be carried out quite simply and at a substantially lower cost.

Our current typesetters have a PC clone which they can link into their typesetting machine, and it is the demands of the typesetting machine which forces the format of the files. Typesetting is quite different from normal typing:

- there are many different fonts of varying sizes.
- each letter is variable in width and the space between letters is varied to justify a line. The left and right margins have to be specified, so that indented text can be produced, as in this paragraph.
- there are many additional characters. Alas, not yet all the APL characters, but our typesetters are adding a desktop publishing front end to their system, using a Hercules Plus card (see separate article) and the APL set should soon be achievable.

The font and size have to be specified in detail to the typesetting machine, but we have simplified the process by specifying three fonts, a standard and two specials for headings. The standard font is assumed unless a control character appears at the beginning of a paragraph; the specified font is then in force for the whole of that paragraph.

Most word processing programs carry out line justification by assuming every letter has the same width, and adding additional spaces between words to get an even right margin. Typists usually add an extra space or two after a full stop in order to improve the visual appearance of a letter and achieve indented text by padding out with spaces. Typesetting machines treat each paragraph (i.e. up to the next carriage return/line feed) as soft putty, squeezing it into the line width that has been previously specified, and stripping out all spaces except one between each word. It is not hard to imagine the lethal effect on a file full of indents, tabs and nicely spaced out and justified text. The ability of the machine to indent is in fact infinitely fine, but to simplify things we have specified a code that will produce a standard indent.

It can also produce fixed space text, with an 'OCR B' font, so text needing this treatment, such as listings, should be preceded by [OCR] and an [ocr] at the end to show where the normal font resumes.

One significant problem with typesetting is the quote marks. Printed text has a different open quote to a closing quote, both for singles and doubles, and it is important that the correct ones are specified otherwise the result looks quite horrid. The way in which this has been solved is to use an ASCII single quote for a right quote, so this looks OK for the possessive case (Vector's quotes), and the ASCII back quote (grave accent) is used for a printed left quote. The ASCII double quote goes through as an open double quote, so a different code is needed for 'close doubles' and for this we use two singles (''). Most typescripts so far have had a typed single quote throughout, which can be converted by assuming that all single quotes preceded by a space, or carriage return, or open bracket, are open quotes, and the remainder are right quotes.

APL text at present is a nuisance. The only way in which it can be catered for is by using some ancient terminal which produces beautiful APL characters, enlarging or reducing the printout photographically to the right size and getting the typesetters to glue the photographic print onto the page (where, of course, the right sized gap has to be left in the text). Authors should supply APL copy on white paper ready for photographing. But, as we say, we are working on it.

In the meanwhile, it is expensive and difficult to include any APL in a paragraph of text. We have had to do substantial editing to remove embedded APL. Please keep all your APL separate. All APL examples should be in between blocks of text.

There are a large number of word processing programs around, and no particular difficulties have been encountered so far with any of them. The main problems are that in copying files from one disk format to another wordwrap line ends ('soft' carriage returns) become hard. That is itself no problem, as line ends are now 'space, carriage return', and these can be stripped out by the typesetters, unless of course you are in the habit of hitting 'full stop, space bar, carriage return' at your ends of paragraphs . . . . Worst of all are the file transfers that produce carriage returns with *no* spaces at line ends, so that every line has to be unpicked. Moral: it can save a lot of grief if all wordwraps are removed before they leave you, so that in effect each paragraph is one long skinny line.

One benefit of using a word processor is getting the spelling right. It is amazing how many mistakes slip through the net. Proof reading is difficult and time consuming, and if there are many corrections to make, checking the corrections is a chore that is most unwelcome as we are usually right up against the time limit.

Text that arrives on diskette goes through at least two editing processes. First we make sure that all the control codes are in the correct places, and make sure that the text fits the style of Vector. All the articles received are then combined into a few files and sent to the typesetters, with a printed copy of the text. The typesetters then change all the control characters into the actual controls needed for their typesetting machine; most of this is automatic, but the tabs need special treatment. The proofs are sent back to us for correction, and the typesetters then edit their files for any mistakes or changes that are required.

The control characters are as follows. They can be accessed in APL*PLUS PC by using the ⎕AV indices given (index origin 0), or by using the APL character.

Large Centered Bold Heading.

^ D (AV 4 diamond) at beginning and end of text.

Small bold heading.

^ B (AV 2 not-equal overstruck by underbar) at beginning and end of text.

Italics.

^ S (AV 19 del-tilde) at beginning and end of text.

Centred italics.

^ X (AV 24 take) at beginning and end of text.

Centred text.

[c] at end of line to be centered, immediately before the carriage return/line feed. This can be used in combination with other control codes.

Left/right justified text (rare).

[lr] at the point where text is to be split and [j] at the end of the line. Can be used in combination with other control codes.

Indented paragraph.

[i] at beginning and end of paragraph to be indented. If a paragraph number is needed, then put the [i] between the number and the start of the paragraph, but Vector articles do not normally have numbered paragraphs.

Tabs.

ASCII split stile (AV 124). The actual amount to tab is set by the typesetters, who adjust it to look well and suit all the text in the columns required. If you must have absolutely precise positioning, specify the OCR font (see above), making sure there are no more than 90 characters per line.

If APL*PLUS PC is being used, then)EDIT is best for producing the text which can be in the form that looks good when printed. It is quite simple to change the resulting vector into the format needed by the typesetters, as follows:

Put APL diamonds around title and APL takes around the by-line.

Put APL inequivalent (not-equal overstruck by underbar) around the small bold headings.

Add [c] to the end of lines to be centered.

Put a [i] at the beginning and end of the indented paragraphs.

If a table is included, replace the spaces with an ASCII split stile.

Use the TEXTREPL function distributed in the ASMFNS workspace to replace the double carriage returns between paragraphs to a dummy character, replace all carriage returns with spaces, and replace the dummy character with carriage return.

Use the DEB function distributed in the ASMFNS workspace to remove all surplus spaces.

Use TEXTREPL to replace space-quote with space-backquote and cr-quote with cr-backquote. Do the equivalent for double quotes by replacing double quotes in double-quote-space or double-quote-punctuation-mark by two single quotes.

Use TEXTREPL to replace carriage returns with carriage return/line feed.

Visually check that the result looks ok.

Add an APL right arrow to the end of the vector, and write to file.

We have had lots of problems. The printed copy sent to the typesetters did not always match the text on file, sometimes the file was correct, sometimes the printed copy. The typesetters have been able to point out all sorts of spelling mistakes and grammatical errors, in spite of the technical nature of the articles. We have found errors in the original scripts at the proof reading stage, and worse, some blunders in the printed copy of Vector. Ideally, what you type in your article is what you get and if you really want a spelling mistake, then so be it. However, the editors are not happy producing a journal with spelling mistakes and bad grammar, so a little editing of files is inevitable. We hope this article will reduce the work to a minimum.

Index to Advertisers

All queries regarding advertising in VECTOR should be made to the advertising editor, Cathy Dargue, at the following address:

Cathy Dargue,
60 Downhall Ley,
Buntingford,
Herts SG9 9TL.
Tel: 0707–325161

Advertisements should be submitted in typeset, camera-ready A5 portrait format with a 20mm blank border. Illustrations should be black-and-white photographs or line drawings. Rates are £250 per page. A6 and A7 sizes (at £150 and £75 respectively) are available, subject to layout constraints.

BRITISH APL ASSOCIATION

Membership Application Form

Please read the membership information in the inside front cover of VECTOR before completing this form. Use photocopies of this form for multiple applications. The membership year runs from 1st May – 30th April.

Name: _____

Department: _____

Organisation: _____

Address Line 1: _____

Address Line 2: _____

Address Line 3: _____

Address Line 4: _____

Post or zip code: _____

Country: _____

Telephone Number: _____

Membership category applied for (tick one):	87/88	
Non-voting student membership (UK only)	£ 5	
UK private membership	£ 10	
Overseas private membership	£ 18	$ 27
Airmail supplement (not needed for Europe)	£ 8	$ 12
Corporate membership	£ 85	
Corporate membership Overseas	£140	$210
Sustaining membership	£360	

For student applicants:

Name of course: _____

Name and title of supervisor: _____

Signature of supervisor: _____

PAYMENT

Payment should be enclosed with membership applications in the form of a UK sterling cheque or postal order made payable to "The British APL Association". Corporate or sustaining member applicants should contact the Treasurer in advance if an invoice is required. Please enclose a stamped addressed envelope if you require a receipt.

Send the completed form to the Treasurer at this address:

Mel Chapman, 12 Garden Street, Stafford ST17 4BT, UK.

The British APL Association

The British APL Association is a Specialist Group of the British Computer Society. It is administered by a Committee of officers who are elected by the vote of Association members at the Annual General Meeting. Working groups are also established in areas such as activity planning and journal production. Offers of assistance and involvement with any Association matters are welcomed and should be addressed in the first instance to the Secretary.

1987/88 Committee

Chairman:	Dick Bowman 01-634 7639	CEGB, 85 Park Street, London SE1.
Secretary:	Graham Parkhouse 0483-571281 (ext 2379)	Dept. of Mech Eng, University of Surrey, Guildford, GU2 5XH.
Treasurer:	Mel Chapman 0785-53511	12 Garden Street, Stafford, ST17 4BT
Activities:	Phil Goacher 03727 21282	27 Downs Way, Epsom, Surrey.
Journal Editor:	Adrian Smith 0904-53071	Brook House Gilling East York.
Education:	Norman Thomson 0962-54433	IBM Mail Point 188, Hursley Park, Winchester, Hants., SO21 2JN.
Publicity:	Jill Moss 0225-62602	17 Barton Street, Bath, Avon.
Technical:	David Ziemann 01-493 6172	Cocking & Drury Ltd., 16 Berkeley Street, London, W1X 5AE.
Recruitment:	Valerie Lusmore 0225-62602	17 Barton Street, Bath, Avon.
Projects:	David Eastwood 01-622-0395	MicroAPL Ltd., Unit 1F, Nine Elms Industrial Estate, 87 Kirtling Street, London, SW8 5BP

Journal Working Group

Jonathan Barman	0374-588835
Anthony Camacho	0727-60130
Cathy Dargue	0707-32516
Val Lusmore	0225-62602
Adrian Smith	0904-53071
David Ziemann	01-493 6172

Activities Working Group

Phil Goacher	03727-21282
Maurice Jordan	01-562 3090
David Parker	01-834 2333

WP disks converted to photosetting by Capella House, Stowmarket (0449) 677663
Printed in England by Short-Run Press Ltd., Exeter.

VECTOR

VECTOR is the quarterly Journal of the British APL Association and is distributed to Association members in the UK and overseas. The British APL Association is a Specialist Group of the British Computer Society. APL stands for "A Programming Language" - an interactive computer programming language noted for its elegance, conciseness and fast development speed. It is supported on many time-sharing bureaux and on most mainframe, mini and micro computers.

SUSTAINING MEMBERS

The Committee of the British APL Association wish to acknowledge the generous financial support of the following Association Sustaining Members. In many cases these organisations also provide manpower and administrative assistance to the Association at their own cost.

APL People	17 Barton Street Bath, Avon Tel: 0225-62602
APL Software Technology	14 Rosewood Avenue, Alveston, Bristol, BS12 2PP Tel: 0454-415737
Cocking & Drury Ltd	16 Berkeley Street, London, W1X 5AE Tel: 01-4936172
Dyadic Systems Ltd	Park House, The High Street, Alton, Hants Tel: 0420 87024
H.M.W. Programming Consultants Ltd	142 Feltham Hill Road, Ashford, Middlesex, TW15 1HN Tel: 07842-41232
Inner Product Ltd	Eagle House, 73 Clapham Common Southside, London SW4 9DG Tel: 01-6733354
Mercia Software Ltd	Aston Science Park, Love Lane, Birmingham, B7 4BJ Tel: 021-359 5096
Meta Technics Systems Ltd	Unit 216, 62 Triton Road, London, SE21 8DE Tel: 01-6707959
MicroAPL Ltd	Unit 1F, Nine Elms Industrial Estate, 87 Kirtling Street, London, SW8 5BP Tel: 01-6220395
I P Sharp Associates	10 Dean Farrar Street, London, SW1H 0DX Tel: 01-2227033
Peter Cyrax Systems	213 Goldhurst Terrace, London, NW6 3ER Tel: 01-624 7013 (Answerphone) 0860-377963 (Mobile)

The British Computer Society, 13 Mansfield Street, London W1M 0BD.