The PDT-11/150 is one of Digital’s new family of intelligent or programmable data terminals. Based on powerful microcomputers, they offer exceptional data processing and communications capability and are user-programmable.
The
AUSTRALIAN COMPUTER
JOURNAL

Volume 10

February, 1978
to
November, 1978
<table>
<thead>
<tr>
<th>AUTHORS</th>
<th>TITLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>K. Reed</td>
<td>A Practical Method of Comparing Computer Bureau Costs</td>
<td>3-10</td>
</tr>
<tr>
<td>M. J. McLean</td>
<td>A Survey of Interactive Graphics Software</td>
<td>11-22</td>
</tr>
<tr>
<td>J. L. Keedy</td>
<td>On Structuring Operating Systems with Monitors</td>
<td>23-27</td>
</tr>
<tr>
<td>I. T. Hawryszkiewycz</td>
<td>A Formal Approach to File and Data Base Design</td>
<td>28-36</td>
</tr>
<tr>
<td>P. L. Juliff</td>
<td>Effective Structuring of COBOL Programs</td>
<td>43-49</td>
</tr>
<tr>
<td>A. W. Goldsworthy</td>
<td>8th World Computer Congress</td>
<td>50-52</td>
</tr>
<tr>
<td>J. Underwood and A. M. McMahon</td>
<td>Teaching Systems Concepts Using a Hospital Admissions Model</td>
<td>53-57</td>
</tr>
<tr>
<td>A. H. J. Sale</td>
<td>Stylistics in Languages with Compound Statements</td>
<td>58-59</td>
</tr>
<tr>
<td>C. W. Johnson</td>
<td>Simula 67 Through Run-Time Structure (Short Communication)</td>
<td>60</td>
</tr>
<tr>
<td>J. M. Yohe</td>
<td>An Overview of Programming Practices</td>
<td>61-74 and 101</td>
</tr>
<tr>
<td>D. J. J. van der Knijff</td>
<td>Software Physics and Program Analysis</td>
<td>82-86</td>
</tr>
<tr>
<td>C. Lakos and A. H. J. Sale</td>
<td>Is Disciplined Programming Transferable, and is it Insightful?</td>
<td>87-97</td>
</tr>
<tr>
<td>B. J. Austin</td>
<td>Port Management in the CYBER76-CSIRONET Interface</td>
<td>98-101</td>
</tr>
<tr>
<td>J. D. Blackburn</td>
<td>A SNOBOL 4 Program to Clean Up FORTRAN Source</td>
<td>102-114</td>
</tr>
<tr>
<td>D. A. Ratkowsky and G. N. Lance</td>
<td>A Criterion for Determining the Number of Groups in a Classification</td>
<td>115-117</td>
</tr>
<tr>
<td>R. Davenport</td>
<td>Data Analysis for Database Design</td>
<td>122-137</td>
</tr>
<tr>
<td>B. Cohen and C. Sammut</td>
<td>A Structural Concept Learning System</td>
<td>138-144</td>
</tr>
<tr>
<td>I. A. Parkin</td>
<td>On Determining the Prime Implicants of a Boolean Function without Recourse to its Minterm Form</td>
<td>153-156</td>
</tr>
<tr>
<td>Editorial</td>
<td>The Australian Universities Computer Science Seminar 1978</td>
<td>2</td>
</tr>
<tr>
<td>Editorial</td>
<td>About This Issue</td>
<td>42</td>
</tr>
<tr>
<td>Miscellaneous</td>
<td>Book Reviews</td>
<td>10, 22, 27, 36, 52, 74, 75-78, 80, 86, 114, 152 and 160</td>
</tr>
<tr>
<td></td>
<td>Corrigenda</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Computer Science D.Sc., Ph.D. and Masters Theses Submitted at Australian Universities</td>
<td>37-39</td>
</tr>
<tr>
<td></td>
<td>Forthcoming National and International Events</td>
<td>40, 79-80, 119 and 159-160</td>
</tr>
<tr>
<td></td>
<td>Letters to the Editor</td>
<td>118-119, 157-158</td>
</tr>
</tbody>
</table>
Index for the Australian Computer Journal, Vol. 10

1. Index of Titles of Papers, Books Reviewed, etc.

About This Issue. (Editorial). 42
A Criterion for Determining the Number of Groups in a Classification. 115
Adaptive Information Processing. 36
A Formal Approach to File and Database Design. 28
Algorithms and Complexity: New Directions and Recent Results. 76
Algorithms for the Computation of Mathematical Functions. 78
An Overview of Programming Practices. 61, 101
A Practical Method of Comparing Bureau Costs. 3
A Relational Database Management System for Microcomputers. 114
A SNOBOL 4 Program to Clean Up FORTRAN Source. 102
A Survey of Interactive Graphics Software. 11
Chess Skill in Man and Machine. 10
Computer-Based Information Retrieval Services, with Special Reference to their Availability in Australia. 78
Computer Methods in Operations Research. 114
Computer Programs for Building Perspectives, Evaluation Report. 52
Computers and Communications: Implications for Education. 152
Computer Science and Scientific Computing. 22
Computer Science D.Sc., Ph.D. and Masters Theses Submitted at Australian Universities. 37
Computers in Society. 77
CONFUCIUS: A Structural Concept Learning System. 138
Data Analysis for Database Design. 122
Data Base Organisation for Data Management. 27
DATA BASE PROCESSING: Fundamentals, Modelling, Applications. 137
Data Structures and Programming Techniques. 74
Data Structures, Computer Graphics and Pattern Recognition. 75
Distributed Processing. Infotech State of the Art Report. 75
Effective Structuring of COBOL Programs. 43
8th World Computer Congress. 50
Every Object is a System: A Cognitive Basis for System Description. 75
Forthcoming National and International Events. 40, 79, 119, 159
Fortran Programming: A Supplement for Calculus Courses. 76
Interactive Systems. LNCS Vol. 49. 160
Is Disciplined Programming Transferable, and is it Insightful? 87
Mathematical Foundations of Computer Science. 86
Matrix Eigensystem Routines — EISPACK Guide Extension. 77
On Determining the Prime Implicants of a Boolean Function without Recourse to its Minterm Form. 153
On Structuring Operating Systems with Monitors. 23
Port Management in the CYBER76-CSIRONET Interface. 98
Simula 67 Through Run-Time Structure. 60
Software Engineering. 144
Software Physics and Program Analysis. 82
Stylistics in Languages with Compound Statements. 58
Teaching Systems Concepts Using a Hospital Admissions Model. 53
The Australian Universities Computer Science Seminar, 1978 (Editorial). 2
Topological Design of a Small Resource Sharing Network. 142

2. Index to Authors and Other Contributors

KEY Name only — paper
E — editorial
BR — book review
L — letter to the editor
SC — short communication

Andrew, A. L. (BR) 77
Austin, B. J. 98
Blackburn, J. D. 115
Bobiro, A. (L) 157
Campbell, J. S. (L) 157
Cohen, B. 138
Coote, A. (L) 119
Davenport, R. 122
Evans, K. McR. (BR) 76
Foo, N. Y. (BR) 76
Gledhill, V. X. (BR) 160
Goldsworthy, A. W. 50
Gupta, G. K. (BR) 22, (BR) 77, (BR) 152
Harvey, I. J. (BR) 52
Hawryszkiewycz, I. T. 28
Herbison-Evans, D. (BR) 75
Herman, P. M. (BR) 10, (BR) 74
Holmes, W. N. (BR) 75
Johnson, B. (L) 158
Johnson, C. W. (SC) 60
Juliff, F. L. 43, (L) 119
Keesey, J. L. 23
Lakos, C. 87
Lance, G. N. (BR) 75, 115
Lings, B. J. 145
McLean, M. J. 11
McMahon, A. M. 53
May, D. R. (BR) 78
Montgomery, A. Y. (BR) 27, (BR) 114, (L) 118, (BR) 137
Osborne, M. R. (BR) 78
Payne, A. D. (BR) 36, 145
Parkin, I. A. 153, (BR) 144
Prime, C. M. (L) 118
Pritchard, P. (BR) 86
Ratkowsky, D. A. 115
Reed, K. 3
Ross, I. (L) 118
Sale, A. H. J. 58, 87, (L) 158
Summitt, C. 138
Underwood, J. 53
Van der Knaif, D. 82
Willis, R. J. (BR) 114
Yohe, J. M. 61, 101

3. Computing Reviews Categories Index

1. GENERAL TOPICS AND EDUCATION
1.3 Introductory and Survey Articles. 11, 61
1.5 Education
1.59 Miscellaneous. 53, 61
2. COMPUTING MILIEU
2.2 Professional Aspects. 61
2.44 Equipment Evaluation. 3
2.9 Miscellaneous. 98
3. APPLICATIONS
3.3 Social, Behavioral Sciences.
3.30 General. 53
3.34 Medicine, Health. 53
3.5 Management Data Processing.
3.59 Miscellaneous. 122
3.6 Artificial Intelligence.
3.62 Learning and Adaptive Systems. 138
3.7 Information Retrieval.
3.71 Content Analysis. 28
3.74 Searching. 87

4. SOFTWARE
4.0 General. 61, 87
4.2 Programming Languages.
4.20 General. 58
4.21 Machine-oriented Languages. 23
4.22 Procedure- and Problem-oriented Languages. 11, 43, 58, 102
4.3 Supervisory Systems. 98
4.31 Basic Monitors. 23
4.32 Multiprogramming; Multiprocessing. 23
4.33 Data Base. 28, 122
4.34 Data Structures. 122, 145
4.35 Operating Systems. 23, 145

4.39 Miscellaneous. 3
4.4 Utility Programs.
4.41 Input/Output. 11, 102
4.6 Software Evaluation, Tests and Measurements. 3, 82

5. MATHEMATICS OF COMPUTATION
5.2 Metatheory. 82
5.23 Formal Languages: nondeterministic processors; grammars; parsing and translation, abstract families of languages. 153
5.24 Analysis of Programs: schemata; semantics; correctness. 87
5.25 Computational Complexity; machine-based; machine-independent; efficiency of algorithms. 153
5.3 Combinatorial and Discrete Mathematics.
5.32 Graph Theory. 142

8. FUNCTIONS
8.1 Simulation and Modelling. 53
8.2 Graphics. 11
DIGITAL INTRODUCES PDT-11 INTELLIGENT TERMINAL FAMILY

The first family of intelligent terminals designed to be fully compatible with the PDP-11 computer line has just been introduced by Digital. The new PDT-11 family of programmable data terminals has provisions for multiple workstations and a choice of communication formats. The three family members, the PDT-11/110, 11/130, and 11/150, incorporate Digital’s LSI-11 microcomputer.

PDT-11 terminals share the same instruction set as PDP-11 computers, programs developed to run on the terminals can run with equal facility on larger minicomputer systems.

PDT-11 terminals are available with varying amounts of main memory. A PDT-11/110 can be easily upgraded in the field to a PDT-11/130 by adding magnetic tape storage devices; all terminals can accept memory in increments to a maximum of 60K bytes. Hardware upgradability as well as software migration permits users to expand the capabilities of their intelligent terminals to match their changing needs.

FIELD EFFECT LIQUID CRYSTAL DISPLAYS

Crystaloid Electronics have just released a range of standard 6”-12” high digit Liquid Crystal Displays.

These displays operate over a wide temperature range from -20 to +80 degrees C, with operating voltages of 2.5, 3 and 4 volt minimum. They are available in transmissive, reflective or transflective modes and feature high contrast with long life and reliability.

Applications envisaged for these displays include clocks, scoreboard displays, time devices, destination signs, pricing and advertising displays.

Crystaloid products are available from the exclusive distributor — Tecnico Electronics, Marrickville, NSW.

ALTERNATE TIMEBASE AND TRIGGER-VIEW CHANNEL IN 100 MHZ OSCILLOSCOPE

A compact dual-channel 100 MHz oscilloscope has been introduced by Philips Test & Measuring Instruments. Known as PM3262 this oscilloscope is unique in providing both trigger-view and alternative timebase operating modes. In addition, the new PM 3262 features a 5mV sensitivity over the full bandwidth, with a high 2mV sensitivity up to 35 MHz.

“I’m Denis Healey and have I got a smorgasbord for your system.”

Here at Sigma Data General Products, we have complete stocks of CFI Memories tapes, disks, cartridges and ribbons.

Although I’m now on the supplier side of the business, I’ve seen plenty from the customer side too.

I know only too well the frustration in finding your EDP supplies.

We’ve solved that problem with OASIS, our own computer system linked to our complete product range, to tell us what’s compatible with what and what’s in stock.

Next time you need EDP supplies, remember, your system will be refreshed best with products from our smorgasbord.

SIGMA DATA CORPORATION


Centronics 6000 Series Released
Sigma Data Corporation announces the release of Centronics 6000 series of line printers.

The new Centronics 6300 line band printer, one of the 6000 series.

Centronics 6000 is a series of back impact printing, fully formed character line printers featuring a horizontal moving one piece print band. The series is composed of four models covering a print speed range of 75 to 600 lines per minute of a full 132 columns. Each member of the series is fully upwards compatible.

Included in the standard features of the range are operator changeable fonts, cassette ribbon and parallel interface. The printers all have fully formed characters, 132 per line and 64 USASCII character set and micro processor electronics.

Each model of the 6000 series will accommodate one to six part forms that are 4” to 19” wide. Six or eight lines per inch vertical spacing is operator selectable. The paper is moved in a vertical direction at fifteen inches per second and is stacked in a convenient paper tray. An out-of-paper sensor automatically alerts the operator of the situation.

Optional features available include standard industry interfaces, Centronics communications interface, line counter, paper jam sensor, Electronic VFU and 136 columns.

The compact, reliable mechanism of the 6000 series is constructed to meet the rigors of a broad range of EDP applications. The series features an 85% spare parts commonality.

The Centronics 6000 series is available through the General Products Division of Sigma Data Corporation.

DATA GENERAL ENHANCES ECLIPSE SOFTWARE
Data General Australia Pty. Ltd. has announced major enhancements to the software capabilities of the Eclipse data system family.

An automatic segmentation facility for the firm’s Cobol language permits the operation of extremely large programs, while enhancements to INFOS file management software broadens its use to applications written in the PL/I, Fortran 5, and DG/L languages.

Running under Data General’s Advanced Operating System (AOS), the Cobol language complies with the American National Standards Institute (ANSI) 1974 standard, with all implemented modules at the highest level.

The new segmentation facility allows the execution of very large programs, whose memory requirements exceed the normal memory addressing capacity of Eclipse computers.

AOS INFOS is a data base-oriented file management system that provides multiple-keyed access to information in a multiple terminal, batch, or communications environment. The new extensions permit access to these data bases for applications written in the PL/I, Fortran 5, and DG/L languages.

AOS PL/I is the first implementation of the PL/I structured programming language from a major minicomputer supplier. AOS PL/I is based on the ANSI 1976 PL/I programming language standard.

AOS Fortran 5 produces highly-optimised code for fast runtime execution.

Data General’s DG/L is an Algol-derivative structured programming language designed for a wide range of applications. DG/L is an ideal development tool for systems-level software such as compilers, assemblers, sort/merge, and other utilities.

CSA Training Course to Begin
Computer Sciences of Australia Pty. Ltd. has had a very strong response to a campaign to recruit trainees, according to Mr Phil Molony, the company’s Assistant General Manager and manager of CSA’s Network Services Division.

A series of advertisements brought a total of 120 applications, of which nine were accepted.

This group, together with two applicants who are already on CSA staff, commenced an eight-week training course in Sydney.

Those completing the course will begin work immediately as Marketing Representatives for CSA as a first step in a planned career path.

Mr Molony said: “The level for applicants was deliberately set high. We set as minimum qualifications a
Poor line supply, mains surges or transients need no longer be a problem. With modern electronic equipment, line variations caused by the switching of large items of equipment can wreak havoc with control circuits. Philips now offer two methods to solve this problem, depending on the degree of stability required, and the load involved. They are the PE1400 series of constant voltage transformers and the PE1610 range of AC voltage stabilizers.

**Constant Voltage Transformers**
The PE1400 series are magnetic systems offering 1% line stability and very high transient suppression for short pulse widths. Alteration factors of 280 and higher are possible. Output distortion is completely independent of line distortion with sinewave output at less than 3% distortion. Other features include mains ‘hold’ for short trip outs, a condition which is disastrous for computers and digital equipment if unchecked; compact ‘monocore’ construction, automatic overload protection.

**AC Voltage Stabilizers**
The PE1600 series combines the advantages of electronic control plus the magnetic system in the PE1400 series. The result is a high 0.1% stability regardless of line voltage, frequency and load variations, combined with good transient suppression, and gives the possibility of varying the combination of stability and transient suppression to suit the application.

Automatic remote control is available. Operation of the stabilizers are designed to perform as constant voltage transformers and incorporate a special feedback control circuit. A sensing circuit provides a DC voltage which is proportional to the r.m.s. value of the stabilizer output. Maximum output powers of 1, 2 and 4kVA are available. Contact us for a demonstration or ask for our "DC Power Supplies and AC Voltage Stabilizers" Catalogue.

Philips Electronic Systems — Scientific & Industrial Equipment, P.O. Box 119, North Ryde, N.S.W. 2113 or phone Sydney 888 8222; Melbourne 699 0300; Adelaide 223 4022; Perth 277 4199; Brisbane 264 1946.
successful record in tertiary studies and also in their current employment, which in most cases is in fields such as banking, accountancy, science and the public service, including local government.

"The comparatively low acceptance rate from what was a generally high level of applicant indicates the standards which we expect to see met by the students in this group."

Mr Molony said CSA had assembled a training panel of 20 senior CSA staff members.

"This group can claim a very high level of expertise and involvement in the computer industry," he said.

"Their average length of service with CSA is six years, and their service in the computer industry is generally considerably longer."

The course has been designed by Mr John Allison, senior training officer in CSA's Melbourne office, who also will act as chief instructor during the course.

CSA plans to run a similar school next year, and is preparing plans for courses for technical trainees.

LATEST BWD CATALOGUE

BWD Electronics Pty. Ltd., have released their latest Shortform Catalogue which includes a general range of oscilloscopes, oscillators, power supplies and the BWD Mini Lab and describes one of their latest new products, the BWD 540/701 DC-100MHz Oscilloscope/Video Line Selector Dual Instrument Package.

Other new instruments are the BWD 539D DC-25MHz Dual Trace Oscilloscope, BWD 845 Dual Trace DC-30MHz Variable Persistence Storage Oscilloscope, BWD 711 Video Line Selector and the BWD 6F Plug In Video Line Selector, used with the BWD 525 DC-50MHz Oscilloscope. The latest version of the Minilab BWD 603B has an increased function generator bandwidth and facilities for sweeping the function generator over two decades on any range. It also provides an additional power supply fixed 5 Volt 1 Amp.

All of BWD Products are illustrated extensively, but more detailed technical information is available on individual instrument Data Sheets, if required. Also available is information on their wide range of power supplies. A comprehensive six page Data Sheet covers all specifications which are fully detailed for ease of selection.

Catalogues are available from BWD Electronics Pty. Ltd., Mulgrave, Victoria.

NEW SYSTEM FOR THE '80's

A general-purpose computer with significant advances in technology and programming — IBM System/38 — was announced recently by IBM Australia.
Central Data

Get into computing the economical, expandable way

FEATURES:

* Signetics 2650 microprocessor based —
* All processor signals buffered for TTL fan out of 10
* Supervisor programme in 1k of PROM
* 730 bytes of RAM available to the user
* Provision for 3k of additional PROM on the board
* Cassette Interface on the board using 1200/2400 Hz Kansas City standard
* Composite Video output with 16 lines of 80 characters display format
* Two parallel input ports, and one bit selectable output port

Central Data dynamic RAM boards are available with 16k, 24k, or 32k bytes of memory.

Central Data software includes an Assembler/Editor and an 8k BASIC tape. A Debugger, 12k BASIC and Assembly Language Package is coming soon.

Other hardware now available and on the way includes Central Data Computer Mainframe with Power Supply, ASCII keyboard with solid-state low-profile keyswitches and +5 volt operation and Floppy Disc controller with one, two or three drives.

For general and specific information:

TECNICO ELECTRONICS

Premier Street, Marrickville, N.S.W. 2204. Tel. 55 0411.
2 High Street, Northcote, Vic. 3070. Tel. 489 9322.
System/38's hardware and programming innovations expand and speed the flow of information while enabling many functions associated with large computers to be combined in a compact system.

System/38 also offers improved price performance with new semi-conductor technologies. The system introduces a dense random access memory chip storing 32,000 bits of information, used in combination with IBM's recently announced 64,000 and 18,000 bit chips, and a new logic chip with up to 704 circuits.

Compared with System/3, the new system's processor logic and random access memory chips have up to 28 and 32 times the circuits and storage cells, respectively.

Because System/38 offers advanced functions that are easy to use, it is practical for both experienced and first-time computer users.

The new system combines large system functions, including data base, virtual storage, online programmer services and up to 40 directly attached workstations.

KEEPSING THE COMPANY'S MEMORY SECURE

Designers of Chubb's computer data protection cabinets have devised double compartment storage specifically engineered to keep temperatures within the cabinet below the critical level.

Humidity is as damaging as heat to data software; therefore the outer cabinet designed to protect paper records contains a specially insulated inner compartment with an hermetically sealed door to prevent the entry of steam and moisture into the tape storage area.

Single door, double door and double depth cabinets are available with 72, 96 and 144 tape capacity. Optional fittings such as storage trays and drawers allow cabinets to be virtually custom-detailed to suit particular requirements.

More modest needs are also catered for by microfilm containers, insert drawers and trays designed to go inside a parent Chubb Record Protection File or Cabinet.

Human interference poses almost as great a risk as fire to the EDP centre, where it may take the form of malicious sabotage, accidental or mischievous tampering.

Because of the high concentration of costly sophisticated equipment in the data centre, the area must be kept out of bounds except to authorised staff members.

A system which is designed to fulfill these tight security specifications is the MAC 530 installed by Chubb's.

This is an electronic card access system which gives centralised on-line control for up to 20,000 people - each person's card individually programmed into the controller's memory.

It can provide access for up to 256 entrances at widely separated locations, including roller shutter vehicle doors or vehicle boom gates.

But perhaps its most important quality is that it can be utilised over 127 different time intervals.

By contrast with the lock and key system of entry, where keys may be duplicated, Mr David McLaren, National Sales Manager of Chubb's Integrated Systems Division says of the MAC 530: "It is virtually impossible to duplicate the card."

"If a card is lost or stolen, or if an employee leaves the company the card's code is simply voided from the system, allowing it to operate without interruption."

A feature exclusive to the MAC system is the ability...
to unlock and lock doors automatically at certain times. It may, for example, unlock the front door to a building just before office hours, then automatically lock it and change to card entry when office hours cease.

It can also be equipped with a wide variety of options, including monitoring the use of office equipment such as copiers. Contact Condition monitors can also be linked to the system to provide an on-line capability for fire and smoke detection, door and window alarm switches, high pressure warnings, power failure warnings, equipment overload indicators.

COMPUTER GROUP BACKS DP JOBS FOR DISABLED

The computer industry has an obligation to train the physically disabled in skills which will allow them to be productive contributors to the community’s prosperity, Mr R.D. Rosengreen, managing director of Commercial Computer Centre Pty Ltd, said in Sydney.

He was commenting on the election of one of CCC’s employees, Miss Janine Lewis, as Miss Nadow 1978.

The Miss Nadow quest, organised by the National Association for the training of the Disabled in Office Work, is staged annually to raise funds for and to increase awareness of the ability of many disabled people to be trained for office work.

CCC, which has its headquarters in Parramatta and offices in the city of Sydney, at Redfern, at St Leonards and in Melbourne, is a leading data preparation and contract processing bureau. It is also a Nadow associate and a member of the Nadow Computer Training Scheme (NCTS).

“Nadow is not just another charity”, Mr Rosengreen said.

“It gives disabled people a chance to help themselves. Money given to Nadow is not given to the disabled. It is given to train them.

“It helps them to live normal, productive lives.”

Miss Lewis, 18, a brown-eyed brunette, is a receptionist at CCC’s head office in Parramatta.

SOCIAL SECURITY USING FAST FICHE DUPLICATOR

The South Australian bureau of the Department of Social Security will speed distribution of microform records with the latest US-developed fast fiche-to-fiche duplicator.

The high speed automated machine purchased from Plessey Communication Systems is identical with two being used for duplicating microfiche by the British police centre, Scotland Yard.

The Photomatrix Series 800 is capable of making 800 duplicates an hour from multiple master fiche or 1200 from one master.

The Plessey duplicator was chosen after Australian Government micrographics experts had compared the Photomatrix with other duplicators.

---

TELERAY Series 10 1061
MICROPROCESSOR CONTROLLED

$1195

FEATURES:
- Block/Character transmission
- Scroll/page mode
- Direct X Y Cursor Addressing & Sensing
- Insert/Delete Character & Line
- Transmit Page Message, and Line
- Clear Page, EOP, EOL
- Protected Fields
- Highlighting Dim, Blink, inverse underline — Any combination
- Programmable I/o 50-9600 baud
- Programmable peripheral Port
- Typewriter Keyboard U/L case
- Eight Function Keys
- 128 character set
- Programmable wide/narrow character display

ANDERSON DIGITAL EQUIPMENT PTY LTD
P.O. Box 322, MT. WAVERLEY, VIC. AUST. 3149, Phone: (03) 543 2077. P.O. Box 294, RYDE, N.S.W. AUST. 2112, Phone (02) 808 1444. Adelaide: 79 9211, Perth: 325 5722, Hobart: 34 4522, Brisbane: 59 8436, Darwin: 81 2505, Canberra: 58 1811, N.Z., Wellington: 64 4585, Auckland: 87 6570.
Contents:

122-137  Data Analysis for Database Design
R.A. DAVENPORT

138-144  CONFUCIUS: A Structural Concept Learning System
B.L. COHEN and C.A. SAMMUT

145-152  Topological Design of a Small Resource Sharing Network
A.D. PAYNE and B.J. LINGS

153-156  On Determining the Prime Implicants of a Boolean Function Without Recourse to its Minterm Form
I.A. PARKIN

Special Features:

137,144,152, 160  Book Reviews

157-159  Letters to the Editor

159-160  Forthcoming National and International Events
Data Analysis for Database Design

By R. A. Davenport*

The primary purpose of data analysis is to determine the fundamental nature of an organisation's data resources and to organise and document all relevant facts concerning this data. This paper describes a methodology for data analysis which has been applied in a number of actual situations and has been found to provide a very effective means of communication with users and a sound basis for the subsequent database design. The methodology employed is based on representing the environment in terms of entities and their attributes and relationships. The methodology when used in preparation for database design, involves two distinct phases — entity analysis and functional analysis. The inherent nature of the environment and its representation in the form of data, is analysed distinctly (although not independently) from an analysis of the business activities and the ways in which data is employed.

KEY WORDS AND PHRASES: data analysis, data resource, data model, database design, entity analysis, entity model, functional analysis.

CR CATEGORIES: 3.50, 4.33, 4.34

INTRODUCTION

As an increasing number of organisations implement systems employing database technology, a new attitude has been developing towards data processing. This new attitude has been termed the database approach. Database technology, Martin (1975), Palmer (1975), Date (1977), permits the data resources to be separated from the computer programs. In effect, the technology permits a separation of data from function. As a result, it has been predicted Nolan (1977), that organisation structure and control will shift to facilitate the management of the technology. Attention will shift to the management of the organisation's data resources rather than to the management of the computer. There will be recognition and management of the data resources function in a manner similar to the recognition of the manufacturing, marketing and finance functions at the present time.

The literature of systems analysis and design, with the odd notable exception, Waters (1974), has tended to emphasise, during the systems analysis phase, the 'processing'; with the database approach the emphasis changes towards 'data' on which the application systems are based. Previously data structures have been designed for particular applications, 'application dedicated files'; the database approach is centred around the 'shared data resource' with data structured independently of individual application requirements.

A fundamental part of the database approach is data analysis, which is concerned with identifying the data resources of an organisation. Although methodologies for data analysis have stemmed from the need for a new approach to systems design in a database environment, experience has shown that the concept of data analysis has a wider applicability, whether or not database software is involved. The approach to data analysis, the time scale and the emphasis put on the various tasks that need to be done, depend very much on the objectives of the project. Data analysis has been used to:

1. Determine the fundamental data resources of an organisation.

Data analysis provides a disciplined approach towards cataloguing the existing data in terms of the entities and relationships it represents. Without such an understanding of that part of the organisation being analysed, it is more difficult to establish whether and where a database could be effectively installed. Data analysis provides a very effective means of communicating with non-data processing users as it deals only with things that the users are familiar with and not with objects such as files and records.

2. Permit the design of flexible file structures capable of supporting a number of related applications.

Whenever data is to be shared between applications it is essential to analyse the inherent structure of that data independently from the details of the applications. This applies whether the shared data environment will be implemented using a database or a number of conventional files.

3. Aid application development or conversion by providing a fundamental understanding of the data involved.

Sound design of transactions and of program structure depends very much on the structure of the data to be processed. When converting from one data organisation to another (such as from conventional files to a database) both the source data structure and the target data structure can usefully be represented as entity models. The comparison of the two models indicates the simplest possible conversion process.

4. Form a basis for data control, security and auditing systems.

Unless the fundamental nature of the data is known, such that all interactions and redundancies can be documented, any control systems implemented are likely to be inadequate.

5. Organise all relevant facts concerning the organisation's data.

Data analysis is very much a documentation exercise and as such it may well be supported by the use of a data dictionary, Lefkovits (1977), BCS (1977), Canning (1978).

6. Aid the unification of an organisation by indicating the commonality between its divisions.

If data analysis is undertaken independently in each

*Copyright © 1978, Australian Computer Society Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that ACI’s copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society.*

*London School of Economics, Houghton Street, London WC2A 2AE. Manuscript received 17th November 1977 and in revised form 12th September 1978*
of the divisions of an organisation the resulting entity models can be compared to indicate the common entities and relationships and the areas where differences between the divisions need to be reconciled. As a result, inconsistency in the way the same events and facts are described in different parts of the organisation can be avoided.

7. To provide a basis for evaluating the structuring capability of competing database management systems.

The organisation's entity model serves as a point of reference against which a logical database structure for each of the database management systems can be designed. The relative ease of mapping from the entity model to the database management system supported structure, provides a good measure of its flexibility and its suitability to handle the organisation's data.

The primary interest in data analysis tends to be as providing a sound basis for database design. This paper describes a methodology for data analysis which has been applied in a number of existing organisations, Palmer (1978a). The methodology is illustrated by means of a case study which, although simplified, is based on an actual situation.

DATA ANALYSIS

Data analysis is regarded as consisting of two dependent projects — entity analysis and functional analysis. (Figure 1).

Entity Analysis provides a means of understanding and documenting a complex environment in terms of its entities and their attributes and relationships. The emphasis is on the things which make up the environment, rather than in the details of how these things are used. These things or objects and concepts of the real world are not dealt with directly; instead user level "name structures", Senko (1976) that are designed to stand for them are operated upon. This user level has been referred to by Stamper (1977) as the discourse system and by Langefors and Sundgren (1975) as the infological level.

Functional Analysis is concerned with an understanding and documentation of the basic business activities with which the organisation is concerned. Typically, there is a hierarchy of functions but the basic activities at the foot of the hierarchy are triggered by events occurring in the organisation and are supported by transactions in the data processing system. The entities and relationships involved in each fundamental function can usefully be represented as functional entity models. These in turn can be mapped to subschemas and will determine program structure.

ENTITIES, ATTRIBUTES AND RELATIONSHIPS

Three major data models (where 'data model' is used in the terms outlined by Sibley and Fry [1976]) have been proposed in recent years: the network model, Bachman (1969), the relational model, Codd (1970) and the entity set model, Senko et al (1973). These models have their own strengths and weaknesses. The network model, which is typically represented diagrammatically by means of a data structure diagram, provides a more natural view of data by separating entities and relationships (to a certain extent) but its capability to achieve data independence has been challenged, Codd and Date (1974). The relational model is based on relational theory and can achieve a high degree of data independence, but it may lose some important semantic information about the real world, Schmid and Swenson (1975). Also, as pointed out by Hall, Owlett and Todd (1976), the relational model cannot denote an individual object independently of its attributes. The entity set model, which is based on set theory, also achieves a high degree of data independence but as Chen (1976) points out, its viewing of values such as '3' or 'red' may not be natural to some people. A number of conferences recently, in particular Nijssen (1976), have been devoted to exploration of other forms of data models.

Recently Chen (1976) introduced the entity-relationship model, which is a simple extension to the data structure diagram. It is simple and easy to conceptualise and claimed to be semantically richer than the relational and the entity set model. The entity-relationship model represents re views by means of entities, relationships and the attributes of each of them. However, it does not allow the representation of relationships between two relationships or between an entity and a relationship.

The data analysis methodology described here relies on a description of an organisation in terms of its types of entities, the types of relationships between them and the types of attributes which describe each entity. The data is represented in a form similar to the entity-relationship model of Chen, which in turn is an extension to the data structure diagram of Bachman. The concepts can be defined as follows:

1. An entity is a fundamental thing of interest to an organisation.

An entity may be a person, place, thing, concept or
event, that is real or abstract, ANSI (1975). There is a distinction between entity type and entity occurrence. "Customer" is the name of an entity type while a particular occurrence of a customer, e.g., named 'Smith', is an entity.

2. An attribute is a descriptive value or property associated with an individual entity.

One type of attribute may describe entities of more than one type. One descriptive value or attribute occurrence can be associated with only one entity occurrence. An entity can exist in its own right without having any attributes defined for it. Examples of attributes of a customer entity are name, address, credit rating, etc. Attributes (or properties) can be classified by one or more roles, Bachman (1974). An attribute can: a) describe an entity, b) uniquely identify an entity, c) be used to represent relationships or connections between entities in the real world, d) have a value which is derived from other entity attributes and e) be used to supply information necessary to operate in a specific system environment.

3. A relationship is an association between two or more entities.

Different relationship occurrences of the same type may involve differing numbers of entities. All relationships of the one type are associations between entities of the same one or two types, or, in other words, describes how entities are logically connected. For example, when one says that a person called J. Smith 'lives in' some house called 10 Pine Avenue, 'lives in' is a relationship between person and house.

THE ENTITY MODEL

In the methodology described in this paper, each element in the environment must be classified as either an entity, an attribute or a relationship as far as the entity model is concerned. No element may be classified in more than one way, such that at times almost arbitrary decisions must be made as to the most convenient way of representing certain aspects of the environment.

Unlike other methodologies, such as the entity-relationship model of Chen (1976) and the ENALIM model of Nijssen (1977b), it has been funconvenient to introduce the restriction that only entities and not relationships may have attributes. Hence anything which may have descriptive values must be treated as an entity. Allowing the additional choice of it being treated as a relationship, adds even greater arbitrariness into the entity analysis process, increasing the number of iterations needed before an acceptable entity model is attained. Similarly, relationships may only exist between entities and are restricted to associations of entities of not more than two types. Of course, all attributes of an entity are related by virtue of the fact that they describe that entity, but relationships are not defined between attributes.

Entity analysis can result in the recognition and definition of many hundreds of types of entities and relationships. This complexity is best represented diagrammatically, Vincent (1977). The convention used for representing entity models are an evolution of the Bachman diagrams or data structure diagrams originally introduced by C.W. Bachman (1969), for expressing the structure of a network database. Two graphical symbols are employed: a box to represent an entity type and a line to represent a relationship type.

An arrow is added to the end of the line to indicate that more than one entity of the type named in the box may participate in an occurrence of the relationship. This is, of course, the degree of the relationship. Relationship types of degree many-to-many are represented by two arrows whereas relationship types of degree one-to-one need no arrows. Examples are shown in Figure 2.

The relationship 'married to' involves one husband and one wife. The degree of the relationship is one-to-one. The relationship between a father and his children, while involving only one father, may involve several children. Many employees can work on the one project, while one employee can work on many projects. The degree of the relationship is many-to-many. Often there is some useful information associated with a many-to-many relationship other than its existence. However, since there is the restriction that relationships cannot have attributes, it is necessary to introduce a new entity type to contain the attributes. In the example of employees and projects, one may wish to record 'time on project' so a new entity type "activity" is introduced. Instead of a many-to-many relationship between employee and project, there is now a one-to-many relationship between employee and activity and a one-to-many relationship between project and activity.

The majority of the relationships involve entities of two types. However, it has been found that most entity models do also include relationships between entities of only one type. This is the so-called involuted structure, a concept which is not supported directly in pure Bachman diagram (nor is the concept of many-to-many relationship directly supported). In an alternative methodology, Bachman (1977), this may be handled by introducing the concept of the role of the entity, which differs depending on whether it participates in the relationship as its owner or

---

Figure 2 Representation of Data Structures

Figure 3 Involutd Relationships

Figure 4 Redundant One-to-Many Relationships

as a member. But if an entity is to take a particular role this must depend on the existence of other entities taking another role. This constraint is best embodied in the association or relationship between the entities. For example, Figure 3, an employee entity cannot take a supervisor role unless it is related to employee entities taking the supervisee role.

A significant difficulty in defining relationships and representing them in the entity model, is determining which relationships are directly significant and which are redundant. This can be done only with a detailed understanding of the environment as there are no mathematical rules that can be applied but merely patterns in the entity model which prompt further investigation. For example, in Figure 4 the relationship ac appears to be the sum of the relationships ab and bc, but it is not necessarily redundant. A ‘receives’ relationship is likely to be as

Figure 5 Redundant Many-to-Many Relationships

redundant as normally a delivery is not made to a customer location unless an order is placed, such that all the delivery entities in a ‘receives’ relationship are associated with the same customer location by way of the combination of the ‘supplied-as’ and ‘places’ relationships. The same questions arise with Figure 5. But in this case the ‘available-with’ relationship associates all permissible combinations of products and packages such that those related by way of order lines must be a subset of these combinations. To determine the existence of relationships, the following procedure can be employed:

1. Take each attribute type and determine which entity type it describes, whether it could describe any other entity type and are these entity types related.
2. Take each entity type and pair it with another and determine if a meaningful question can be asked.
3. Finally determine if the relationship is relevant.

No less difficult is the decision concerning each element, as to whether it should be treated as an attribute type of an entity type or as a second entity type related to the first. As a guideline it has been found that an attribute of an entity-1 is best treated instead as entity-2 related to entity-1 if:

1. The attribute itself is found to have further relevant attributes.
2. The resulting entity-2 is itself of significance to the organisation.
3. The attribute in fact identifies entity-2.
4. Entity-2 could be related to several occurrences of entity-1.
5. Entity-2 is seen to be related to entity types other than entity-1.

Thus in Figure 6, ‘customer location’ is seen not to be an attribute of ‘customer’ as a customer may have several locations and as each location has its own attributes, such as postal code.

During the entity modelling phase the most significant entity types and relationship types are defined. But inevitably a model will be extended or modified during the detailed analysis phase as a result of re-examining the
attributes. The difficulties of establishing the entity types and the relationship types during the analysis phase have been very well discussed by Kent (1976, 1977).

**PROPERTIES OF RELATIONSHIPS**

It has been found that in understanding and documenting an environment leading to database design and the better control of the data resource, the relationship types are no less important than the entity types. As a result each relationship type is named, carefully defined, and a variety of its properties analysed. These include:

- The degree of the relationship
- Its optionality
- Its exclusivity
- Whether it is included in some other relationships
- Properties of sequence

**FULLY OPTIONAL**

1. Fully optional
   Entities of either of the related types may exist without participating in a relationship. In Figure 7 for example a 'house' need not participate in the 'occupied by' relationship because it is empty and similarly a 'person' need not participate in an 'occupied by' relationship because he has nowhere to live. This property is indicated in the entity model by the use of a dashed line.

2. Contingent
   The existence of an entity of one type is contingent on the prior existence of an entity of the related type. In other words, all entities from one type (but not both types) must participate in some occurrence of the relationship type. An example where all entities must own a relationship is that of the relationship 'made from' where each 'assembly' entity must be related to one or more 'parts' entities. An example where all entities must be members of the relationship is the 'ordered by' relationship where each 'order line' entity must be related to some 'product' entity. The relationship is drawn with dashes against the entity type which is optional and solid adjacent to the entity type which must participate. (Figure 7).

3. Mandatory
   For entities of either type to exist they must

**FULLY OPTIONAL**

- Occupied-by
- Assembly
- Made-from
- Ordered-as

**Contingent**

- Person
- Part
- Order Line

**Mandatory**

- House
- House
- Product
- Product

Figure 6 Replacing Attributes by Relationships

Figure 7 Optional Relationships

*The Australian Computer Journal, Vol. 10, No. 4, November, 1978*
participate in some occurrences of the relationship type. For example, an order is meaningless unless it consists of at least one order line and conversely every order line must be part of an order. This is indicated in the entity model by a single solid line.

The optionality property is important when it comes to maintaining control over the consistency of the contents of the database. The simple conventions described have been found to be a very powerful method of summarising some of the complexities of the environment.

Similar constraints in the environment are categorised by means of exclusivity between relationships. That is to say an entity occurrence may participate in one, but in not more than one, of a number of exclusive relationships. For example, in Figure 8, a store room is used for either storing raw materials or for storing finished goods, but never for both.

Of less significance but still useful in expressing some complexities of the environment is the inclusive relationship. In this case, an entity can participate in a relationship of one type only if it also participates in a relationship of a second type. The second relationship is said to include the first. The entities participating in the second must be a subset of those participating in the first. For example, Figure 9 indicates that a person may drive several vehicles and he may also own several vehicles, but although he may be entitled to drive vehicles he does not own, he is entitled to drive every vehicle which he does own.

Occasionally, a relationship can be more than merely an association between entities but imply an ordered sequence of these entities or at least the member entities of the relationships. For example, in Figure 10 each activity for which the shipment is used is related to another activity by virtue of the fact that it precedes it by a time sequence. This is best represented using the ‘S’ symbol on the ‘is used for’ relationship rather than by introducing an involuted one-to-one relationship. In practice, most relationships of this type are sequenced on time. This must not be confused with some logical sequence based on an entity identifier for the convenience of a data processing system.

There is another aspect of time which causes practical difficulties during the entity modelling phase. The degree of a relationship depends on whether the environment is viewed as at an instant in time or whether the entity model reflects the dynamic nature of the environment. For example, in most organisations the degree of the relationship between the employee and the position he holds is one-to-one at any instant in time but if the career of the employee is considered over a longer period the degree of the relationship becomes one-to-many. Other examples in Figure 11 are taken from the steel industry where the combinations of moulds and plates used together vary from time to time, and from the consultancy business where in the short term the consultant works on one project and over a period on several. The data analyst must take care that the entity model reflects either a short term view or a long term view and not a mixture of the two. Bubenko (1977) and Grindley (1975) have proposed different approaches for dealing with the problem of time.
Data Analysis

Bubenko has analysed how time is treated in different approaches to data modelling.

The final property of relationships is their use in uniquely identifying entities. The methodology permits the identification of entities in three ways: (Figure 12).

1. **By one or more attributes.**
   - The simplest case is where each occurrence of the attribute has a unique value which is used to identify the entity. Combinations of attributes may also be used, such as when employees are identified by their name, together with the date they joined the company. It may be useful to include the names of the identifying attributes in the entity type boxes on the entity model if it does not lead to excessive congestion.

2. **By the combination of a relationship with one or more attributes.**
   - The members of the relationship are often uniquely identified within that relationship by the values of the attribute type, but for uniqueness within the system the owner of the relationship needs also to be known. In effect, it is the relationship occurrence as identified by its owner, which is contributing to the unique identification of its members. As shown in Figure 12, this is indicated on the entity model by a stroke across the relationship line.

3. **By two or more relationships.**
   - In this case, the entity is identified only by the relationship in which it participates. In effect, it is the owners of the relationships which determine the existence of the entity.

There remains one difficulty frequently encountered during entity analysis which cannot adequately be handled by relationships. This is the problem which arises when different functions wish to use clearly identifiable subsets of the total population of an entity type. The question then arises as to whether the entity type, as defined, is taking too global a view and is better considered as being several entity types. In general it may be preferable to treat entities as being of different types if they have either:

1. significant differences in their attributes; or
2. different means of identification; or
3. participation in different types of relationships.

One alternative method of dealing with the above problem which has been suggested by Palmer (1978b) is the concept of the entity subtypes which permits the entity model to express both viewpoints. An entity subtype is a classification of an entity type where the value of a classical attribute is used to determine the subtype of an entity occurrence. Entity types may be classified in more than one way, such that there may be more than one group of subtypes and so more than one classifying attribute. An entity subtype may itself be classified so that a classification hierarchy of subtypes may be defined as shown in Figure 13.

Another alternative is the role model approach of Bachman (1977). The role model would treat Applicant, Manager, Worker and Pensioner as roles of the entity type Person. The subtype concept appears to be complementary to the ideas incorporated in the role model. In particular it permits a hierarchal classification of an entity type into entity subtypes, some of which, but not all of which, may be represented as role types of the entity type.
Data Analysis

Figure 13 Hierarchy of Entity Sub-Types

generalisation of the database abstraction methodology, Smith and Smith (1977) is similar to the subtype concept. Where different entity subtypes participate in different relationship types it is possible to depict the situation in the entity model. With two level classifications it is convenient to use entity subtype boxes drawn within a larger entity type box. For example, in Figure 14 where manager, permanent staff and temporary staff are all subtypes of employee, all employees are assigned to a department but only temporary staff are hired from an agency, and whereas permanent staff must be members of a union, this is optional for managers.

These various conventions, as described, have been found useful for extending the value of the entity model for communication purposes and as a convenient summary of the complex interactions in the organisation. Not all the conventions need to be included on all versions of the entity model. For example, in discussing their environment with the users it may be necessary to keep the model as simple as possible. The global entity model maintained by the data administrator, Palmer (1975), and reflecting the full complexities of the organisation does need to contain all the properties of entity types and relationship types as previously described.

ENTITY ANALYSIS METHODOLOGY

Entity analysis can itself be a major project involving a team of several analysts and users and requiring several months to cover merely part of an organisation. In fact, entity analysis can best be regarded as an on-going activity since even when all parts of the organisation have been analysed, they continue to change and these changes need to be reflected in the entity model. Entity analysis can be divided into the following phases.

1. Preliminary

It is essential to define the objectives and purpose of entity analysis and the area of data to be analysed. The data area can best be chosen by identifying each of the main areas that the system is to handle, either now or in the foreseeable future. The data area can best be specified in terms of data subjects or global entities, although its precise boundaries are certain to be vague at this stage and will be clarified as the analysis proceeds. Examples of data areas

Figure 14 Entity Sub-Types in the Entity Model

would be order processing, inventory control, customer accounting, etc. It will obviously be necessary to clarify within each area the terminology employed by users. The choice of areas for entity analysis will depend upon a number of factors. Some of the factors concerning the choice of area are:

(a) Small enough to be manageable
(b) Not so complex that an overview cannot be obtained
(c) Clear cut with definable boundaries
(d) Discrete with a minimum of interaction with other areas
(e) Required for applications that are to be implemented.

Regarding the last factor, it must be stressed that if the data area is defined in terms of applications or functions in the organisation, the resulting entity model will be biased towards them, rather than providing an inherent or independent viewpoint.

2. Entity Modelling

During this phase the entity types and the types of relationships between them are defined. They are immediately represented diagrammatically in an entity model, as this provides a means of understanding a complex environment and forms a basis for communication between analysts and users. The entity model is similar to the conceptual model of ANSI/SPARC, ANSI (1975), but it is meant to model all the data held by the organisation not just that data in the database. Many versions of the entity model are issued as new types of entities and relationships are added and as all viewpoints are gradually incorporated.

The only types of attributes considered at this stage are those needed to uniquely identify one entity occurrence from another as each entity must be uniquely identifiable as demonstrated by Grindley (1975) and Engles (1972).

3. Detailed Entity Analysis

The relevant attributes are recorded and defined for each entity type. This may lead to identification of new entity types or to the sub-divisions of existing ones. It also enables the boundaries of the data area to be defined more precisely. Once the entity model is reasonably complete, explicit checks need to be made to detect redundant relationships which unnecessarily increase its complexity. Similarly the rules of normalisation from relational theory, Codd (1971), are applied to the attributes of each entity type, in order to detect any hidden relationships and to ensure that the entity model represents the most fundamental understanding of the environment. Normalisation requires three actions to be performed on the attributes of an entity type. First, repeating groups are resumed. Next, attributes are removed which are dependent on only some of the identifying attributes. Finally, any attributes are removed which are not directly dependent upon the identifying attributes. This results in all attributes being in third normal form. If 'man' had been identified as one of the entity types of interest and address was identified as one of the attributes of man, the normalisation process may show (depending upon the particular environment) that there are relationships 'hidden' inside the entity type, i.e., several men may reside at the same address or a man may have several addresses. Normalisation would have then produced a new entity type 'property'. Normalisation ensures that each attribute type has only a single value for each occurrence of the entity type concerned, i.e., that repeating groups of attribute types are shown as entity types in their own right. Also, it ensures that the value of each attribute is really independent. This means that the value is not affected by any change in relationship between the entity concerned and any other entity.

4. Integration of data areas

As the entity analysis project proceeds, a number of data areas will be analysed. An entity model will be produced for each data area analysed and these individual entity models need to be merged together to produce an integrated entity model which will satisfy all data areas. This integration phase initially involves some editing to remove inconsistencies. Inconsistencies may arise concerning the type of attributes, entities or relationships. These inconsistencies may be in the form of one name referring to different components (homonyms) or different names referring to the same components (synonyms). The stages followed in integration (two areas at a time) are the following:

a) Identify any synonyms or homonyms in the different data areas. If a data dictionary is being employed, this task will be considerably eased. However, if only a clerical system of documentation is available, the task will have to be performed by inspection. Components which have homonyms will have to be renamed. Components which are synonyms will have to be referred to by a single name.

b) Entity models for the two data areas are integrated by super-imposing the identical or similar entity types in the different entity models. This may increase the total number of attribute types in the entity type which is the result of superimposing, as identical entity types in each data model have been concerned with different subsets of the total group of properties. When integration takes place, it may be advisable when superimposing the entity types to define entity subtypes of the final entity type. An individual subtype would satisfy the requirements of an individual data area. There may of course already be subtypes identified. Therefore integration may increase the number of levels of the classification hierarchy.

c) As a result of the integration, the composite entity model may contain redundant relationships, i.e., a particular relationship is a composition of two others. This redundant relationship may be eliminated. However, as has been discussed, determining which relationships are directly significant and which are redundant can present difficulties which can only be solved by an understanding of the environment.

5. Consistency with functional analysis

The functional entity models, representing each of the business activities analysed must be checked against the global entity model representing the organisation as a whole. Inevitably, certain aspects of the environment are detected during functional analysis which have been missed during entity analysis; in particular, certain attributes of entities which are relevant only in the context of the particular function. This phase ensures that the views of the data analysts and the function analysts are consistent and that the entity model does support all the application requirements.

ENTITY ANALYSIS CASE STUDY

As an example of the application of entity analysis, a
brief description of an area of data follows. The area of data to be considered is that of a typical order processing environment. The company possesses a number of goods which may be sold in a bulk or packaged form. Stocks of these goods are stored in 12 depots. All depots can supply packaged goods but only two can supply goods in bulk. The company supplies some 40,000 customers, which vary from small retail outlets to large nationwide corporations. Hence, a given customer may have many points to which goods are delivered. However, a delivery point is always supplied from the same depots; one for goods in bulk and one for packaged goods. In certain cases a delivery point may be supplied with both bulk and packaged goods by the same depot. Orders are placed by customers, each for delivery from a depot to a specific point. An order may contain several lines each for a different type of packaged goods or goods in bulk. Where part of an order cannot be supplied, it is split, a back order being created which is processed as soon as the stock becomes available. An entry is created in the sales ledger, debited against the appropriate customer for each order supplied. Entries in the ledger are created whenever payments are received and credited to a customer.

The entity types and relationship types, apparent from the above description, are represented in the form of an entity model, Figure 15, using the conventions previously described. Note that ‘filled order’ and ‘back order’ are entity subtypes of the entity type ‘order’ and ‘credit entry’ and ‘debit entry’ are entity subtypes of the entity type ‘ledger entry’. For reasons of simplicity the entity subtypes have not been included in the entity model.

A second area of data is identified as being concerned with manufacturing. The company manufactures some 3000 products each of which may be sold in bulk or a packaged form. Most products are available in a variety of packages (e.g. one litre cans, five litre bottles, etc.). Stocks of these products, either in bulk or packaged, are stored in 12 depots, two of which also stock raw materials as they supply manufacturing plants on the same site. Each product is made from a variety of raw materials and then packaged. The term goods is used to refer to a particular product-package combination. The entity types and relationship types apparent from the above description are represented in another entity model, Figure 16.

The two entity models are integrated to produce a composite entity model, Figure 17, by superimposing the entity types depot, goods and stock. Note that the relationship raw materials-stock is not redundant because of the exclusivity between it and the relationship goods-stock.
FUNCTIONAL ANALYSIS

Closely connected with entity analysis is functional analysis, which is concerned with an understanding and documentation of the basic business activities with which the organisation is concerned. Functional analysis has the following objectives.

1. To determine how entities are used so as to increase understanding of the entity model.
   Functional analysis may reveal attribute types of entity types which had not been detected during entity analysis. Similarly, relationships between entity types which had not been considered meaningful may be found to be required by certain functions. As a result the original entity model may be modified.

2. To provide a firm basis for transaction design.
   The basic functions identified in functional analysis would be expected to be translated into transaction types in the data processing system.

3. To gather estimates of data usage for database design.
   The most heavily utilised access paths are identified. This provides a basis for determining which access paths should be made most efficient.

Functional analysis can be divided into the following phases.

1. Preliminary.
   As for data analysis, the objectives need to be defined as well as the application area to be analysed. The application area may coincide with a data area examined in data analysis or it may cross several data areas.

2. Develop a Framework.
   This phase identifies the events and the functions. Typically, there is a hierarchy of functions but the basic activities at the foot of the hierarchy are initiated by events occurring in the organisation. An event (referred to by Grindley [1975] as a trigger), can be defined to be a stimulus to the organisation and functions can be defined to be the tasks that must be carried out as a direct result of an event, e.g., 'an order is placed' is an event while 'record the order', 'schedule production' and 'produce the invoice' are functions. Each event should be described and the functions carried out as a direct result of the event identified. Each function should be described in enough detail to allow the next phase to be carried out.

3. Access Path Analysis.
   From the description of the function it is possible to determine the entities and attributes involved in each function and the relationships used to perform the processing. This allows a functional entity model to be created. A functional entity model summarises diagrammatically the entity types and relationship types used by the function. The functional entity model is similar to the external model of ANSI/SPARC, ANSI (1975). An external model, Date (1977), is the information content of the database as it is viewed by some particular user. In this case, the user is a function. The functional entity model will be a subset of the total entity model. The access path for the function shows how, and the order in which the function uses the entity types and relationships. This provides a firm basis for database design. Access path analysis for each function follows the following stages:

   a) From the function description list the entity types used in the order in which they are assessed.
b) Record the selection criteria for each entity type, i.e., is a particular occurrence of an entity type selected by means of relationships or by the value of an attribute.

c) Record whether each occurrence of each entity type accessed is retrieved, modified, created or deleted.

d) Similarly record whether the relationships used are created, connected, disconnected or transferred.

Examples of each such relationship possibility are shown in Figure 18. With creation all entities participating in the relationship are stored for the first time. With connection, the relationship exists but a new entity participates in the relationship. Disconnection occurs when an entity ceases to participate in a relationship. Transfer denotes that an entity which is participating in one relationship changes to participation in another relationship. As functions are concerned with the usage of the data, it has been found useful to depict this information in the functional entity model. Where an occurrence of an entity type is accessed directly by means of the value of a particular attribute (typically referred to as a key) this is indicated by means of a broad arrow. Where relationships are used as access paths to access particular entities, a small triangle is used to indicate the direction of access. Where an entity is being stored for the first time the entity box is dashed.

(Figure 19).

**FUNCTIONAL ANALYSIS EXAMPLE**

One of the functions identified as being carried out in the order processing area previously described, is 'order entry'. An order is received from a delivery point. The depot that will make the delivery is selected depending on whether the goods are bulk or packaged. The order is recorded and related to the delivery point and the depot. The goods specified in each order line are validated and the stocks of the goods on hand are amended. Where stocks are insufficient to meet the quantities in one or more lines on the order, a second order, a back order, is created. The order lines are recorded and linked to the goods and to the order, or back order as appropriate. The entity types and relationship types used by the functions and the way that they are employed, as apparent from the above description, are represented in the form of a functional entity model, Figure 20, using the conventions previously described.

**DOCUMENTATION**

An essential outcome of data analysis is the documentation for entity types, relationship types, attribute types, functions and events. This documentation is in addition to the entity model and functional entity models. Where the volumes and complexity are low a clerical system has been found to be adequate, but in the longer term and in a dynamic environment, the use of a good data dictionary is advisable. Regrettably few currently available data dictionaries make the distribution between the constructs of the entity model and those of the logical
Figure 20 Functional Entity Model for Order Entry

This paper has described a methodology for data analysis and applied it to a simple example. It has attempted to highlight some of the questions that arise during data analysis and some of the ways in which an understanding of the data resource can be rationalised. Practical difficulties met during data analysis can be

<table>
<thead>
<tr>
<th>ENTITY ANALYSIS DOCUMENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
</tr>
<tr>
<td>SYNONYMS</td>
</tr>
<tr>
<td>DEFINITION</td>
</tr>
<tr>
<td>IDENTIFIERS</td>
</tr>
<tr>
<td>OCCURRENCES</td>
</tr>
<tr>
<td>SPECIAL VARIATIONS</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>AUTHORISED TO CREATE</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>CONFIDENTIALITY</td>
</tr>
<tr>
<td>SUBTYPES</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>DEFAULT FOR ATTRIBUTE DETAILS</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Figure 21
Data Analysis

ENTITY ANALYSIS DOCUMENTATION

ATTRIBUTE TYPE

NAME Credit limit

SYNONYMS

DEFINITION If the customer's balance exceeds this value no further orders will be accepted unless paid in advance

ENTITY TYPES DESCRIBED

Customer

WITHIN ATTRIBUTE GROUP

OCCURRENCES

NUMBER OF DIFFERENT VALUES

MINIMUM – 30,000

AVERAGE 5,000

MAXIMUM – 50,000

GROWTH 5%

SPECIAL VARIATIONS

AUTHORISED TO

CREATE Accounts Receivable

MODIFY Finance Director

RETRIEVE Salesman

SUMMARISE

CONFIDENTIALITY

Type 2

PERMITTED VALUES

FORMAT 5 numeric digits

RANGE £10,000 to £25,000

MEANING

CONSISTENCY

Only for customers with over 3 previous orders

AVAILABILITY Online

STORAGE

Updated daily

Figure 22

ENTITY ANALYSIS DOCUMENTATION

RELATIONSHIP TYPE

NAME Places

SYNONYMS

DEFINITION The customer has indicated by phone or by post that he wishes to buy specified products

DEGREE

AN ENTITY OF TYPE Delivery Point is always sometimes (70%) Related to Minimum Maximum 1 300

AN ENTITY OF TYPE Order is always sometimes (60%) Related to Minimum Maximum 1 1

ENTITIES OF TYPE Order Delivery Point

AUTHORISED TO

CONNECT Order Processing Department

DISCONNECT Order Processing Department

TRANSFER Prohibited

CONFIDENTIALITY

Type 3

CONSISTENCY

AVAILABILITY Online

REPRESENTATION

Contiguity in the Customer File

TIMELINESS Update daily

Figure 23

overcome if the process is well understood and the conventions for classifying and documenting are followed. The methodology described has proven to be a successful approach in a wide variety of situations and in providing confidence that any resulting database design will meet the longer term needs of the organisation.

ACKNOWLEDGEMENT

The author would like to acknowledge the efforts of
FUNCTION ANALYSIS DOCUMENTATION – ACCESS PATH

<table>
<thead>
<tr>
<th>FUNCTION NAME</th>
<th>RESPONSE REQUIRED</th>
<th>INITIALS</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Entry</td>
<td>5 sec AVG</td>
<td>RAD</td>
<td>1.8.78</td>
</tr>
</tbody>
</table>

FREQUENCY ASSUMPTIONS:

PER DAY AVERAGE: 4000 MAX: 6000 GROWTH: 10% per annum

ENTITY (E) / RELATIONSHIP (R) ACCESSED

<table>
<thead>
<tr>
<th>E/R</th>
<th>SELECTION CRITERIA</th>
<th>ACTION</th>
<th>VOLUME</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AVG</td>
<td>MAX</td>
<td>NO</td>
</tr>
<tr>
<td>Delivery Point E</td>
<td>Delivery Point Name R</td>
<td>1 1</td>
<td>1 1</td>
<td></td>
</tr>
<tr>
<td>Bulk/Package R</td>
<td>Bulk/Package R</td>
<td>1 1</td>
<td>1 1</td>
<td>1</td>
</tr>
<tr>
<td>Depot E</td>
<td>Via Relationship R</td>
<td>1 1</td>
<td>1 1</td>
<td>1</td>
</tr>
<tr>
<td>Order E</td>
<td>Order No. Delivery Point Name CON</td>
<td>1 1</td>
<td>1 1</td>
<td>1</td>
</tr>
<tr>
<td>Order/Depot R</td>
<td>Order No. Depot Name CON</td>
<td>1 1</td>
<td>1 1</td>
<td>1</td>
</tr>
<tr>
<td>Goods E</td>
<td>Goods Code R</td>
<td>10 30</td>
<td>10 30</td>
<td>1</td>
</tr>
<tr>
<td>Goods/Stock R</td>
<td>Goods Code R</td>
<td>10 30</td>
<td>10 30</td>
<td>1</td>
</tr>
<tr>
<td>Depot/Stock E</td>
<td>Depot Name R</td>
<td>10 30</td>
<td>10 30</td>
<td>1</td>
</tr>
<tr>
<td>Stock E</td>
<td>Depot and Goods Relationships M</td>
<td>9.5 28.5</td>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td>Back Order E</td>
<td>Back Order No. S</td>
<td>0.5 1.5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Order/Depot R</td>
<td>Back Order No. Delivery Point Name CON</td>
<td>0.5 1.5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Order Line E</td>
<td>Back Order No. Depot Name CON</td>
<td>0.5 1.5</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

NOTES
1. Assume 95% of orders fully satisfied

ACTION (ENTITY) RETRIEVE, MODIFY, STORE, DELETE; (RELATIONSHIP) RETRIEVE, CREATE, CONNECT, DISCONNECT, TRANSFER.

Figure 24

FUNCTIONAL ANALYSIS DOCUMENTATION – SUMMARY OF ACCESS TO AN ENTITY TYPE

ENTITY NAME: Goods

<table>
<thead>
<tr>
<th>ATTRAIBUTES</th>
<th>SELECTION CRITERIA</th>
<th>RETRIEVE AVG</th>
<th>MAX</th>
<th>MODIFY AVG</th>
<th>MAX</th>
<th>STORE AVG</th>
<th>MAX</th>
<th>DELETE AVG</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Goods Code</td>
<td>8 x 10^4</td>
<td>10^5</td>
<td>50</td>
<td>100</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit of issue</td>
<td>Goods Code</td>
<td>10 15</td>
<td>10 15</td>
<td>10 15</td>
<td>10 15</td>
<td>10 15</td>
<td>10 15</td>
<td>10 15</td>
<td>10 15</td>
</tr>
<tr>
<td>Cost Price</td>
<td>Goods Code</td>
<td>260 300</td>
<td>50 80</td>
<td>10 10</td>
<td>50 10</td>
<td>50 10</td>
<td>50 10</td>
<td>50 10</td>
<td>50 10</td>
</tr>
<tr>
<td>Sales Prices</td>
<td>Goods Code</td>
<td>8 x 10^4</td>
<td>10^5</td>
<td>50</td>
<td>100</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VIA RELATIONSHIP

<table>
<thead>
<tr>
<th>RELATIONSHIP</th>
<th>SELECTION CRITERIA</th>
<th>RETRIEVE AVG</th>
<th>MAX</th>
<th>TRANSFER AVG</th>
<th>MAX</th>
<th>CONNECT AVG</th>
<th>MAX</th>
<th>DISCONNECT AVG</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stock/Goods</td>
<td>Goods delivered</td>
<td>4 x 10^4</td>
<td>5 x 10^4</td>
<td>20</td>
<td>30</td>
<td>80</td>
<td>100</td>
<td>50</td>
<td>70</td>
</tr>
<tr>
<td>Order Line/Goods</td>
<td>Goods of amended order</td>
<td>20</td>
<td>30</td>
<td>80</td>
<td>100</td>
<td>50</td>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Order Line/Goods</td>
<td>Goods of cancelled order</td>
<td>20</td>
<td>30</td>
<td>80</td>
<td>100</td>
<td>50</td>
<td>70</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

REFERENCES


BACHMAN, C.W. (1969), Data structure diagrams, Database 1, 2, pp. 4-10.


CHEN, P.P.S. (1976), The entity-relationship model – Toward a unified view of data, ACM Transactions on Database Systems 1, 1, pp. 9-36.

**Book Review**


This is an introductory text on Data Base processing. It presents material on physical representation, modelling, commercial systems and implementation, with about equal emphasis.

Part I provides background material. Part II concerns database models. Part III is devoted to the processing of Databases. Part IV deals with Database implementation.

Whilst finding the Chapters on Physical Data Structures, Data Base Models (including examples from hierarchical, network and relational systems) Database Analysis and Design and the Role of the Database Administrator entirely satisfactory, I got most interest out of the Description of Commercial Database Systems. Whilst far from exhaustive — as might be expected in an introductory text — the description of ADABAS, SYSTEM 2000, TOTAL, IDMS, IMS

and MAGNUM give a good insight into how the systems work and give opportunity for the more experienced reader to impute the relative strengths and weaknesses of these systems.

I found the subject of Backup and Recovery — about which the author rightly says "Backup and Recovery is one of the most critical aspects of database processing, yet it does not get much attention" — is given too little attention! I also found that the Bibliography, which one would hope in an introductory text to lead readers to exploring topics of interest in greater depth and breadth, contains but 40 references of which at least two are unobtainable by Australian readers.

However the book does have some case studies and is provided with an Instructors' Guide.

In summary I thoroughly recommend this book to practitioners wishing to get an initial accurate overview of DBMS to teachers of courses pitched at the competent introductory level.

A.Y. Montgomery, Monash University
CONFUCIUS: A Structural Concept Learning System

By B.L. Cohen* and C.A.Sammut*

This paper describes a general structural object recognition and learning system. It introduces CODE, a structural description language based on boolean logic and set theory. Internal machine representations of concepts and objects are given for these which directly influence the efficiency of the system. In particular, a data structure called the "graft," is described. Two concept learning strategies, LSI and LS2, used in CONFUCIUS are also discussed. Both algorithms are generalizations of the "Conservative Focusing Strategy" developed by Bruner et al. The generalizations consist of using the high level language, CODE and the introduction of an "implication net" which enables CONFUCIUS to work in a more complex environment.

CR CATEGORIES: 3.60, 3.61, 3.62, 3.63 KEYWORDS: Learning and adaptive systems

1. Structural Description Languages and Concept Learning

Given the description of an unidentified object, the task of a recognition device is to give the object an appropriate name, that is, to classify it as belonging to a particular class of objects. Such a class is called a "concept".

In recognition problems such as scene analysis, the objects under consideration are often quite complex. Thus, it is important that the representations of objects and concepts in a computer are powerful enough to describe complex concepts as well as permitting economy both in memory space and in the time required by the recognition process. The structural representation of concepts is very attractive for these reasons.

As an example, consider the scene consisting of a small blue pyramid sitting on top of a big red block which is inside a big green box. This may be described by the graph shown above.

This scene is described by naming its component parts and the relationships between them. Each object is described in terms of a number of properties: shape, size and colour.

If some simple concepts are already known then the representation of a more complex one may be simplified by describing it in terms of the known concepts. Thus a hierarchical representation, based on the structure of an object, is constructed.

If a system is capable of learning new concepts and adding these to its memory, then it is possible to use this newly acquired knowledge to simplify the description of still more complex concepts. In this way a system may become capable of recognizing and learning more and more difficult patterns by starting with simple problems and progressing to the more difficult ones just as humans do. Such a system is said to be capable of growth.

New concepts may be introduced into memory in a number of ways. Of course, there is the brute force method of providing a complete description of the new concept directly to the system. The more interesting approaches are those which enable the computer to actively learn a concept. This may be done by providing it with a number of positive examples of objects which do belong to the concept and a number of negative examples of ones which do not. Or just one positive example may be shown and the system asks questions which the trainer may answer by a "yes" or a "no".

Several programs have already been written which are capable of learning concepts and recognizing objects which belong to them, for example, Winston (1970) and Michalski (1974). The CONFUCIUS system seeks to improve upon previous work by providing a relatively powerful concept description language which exhibits growth, an efficient object recognition system and a learning algorithm which is a generalization of a psychological model. The description language, CODE, is based upon a language developed by Banerji (1969).

2. CODE — The Structural Description Language

Given the task of recognizing an object or learning a new concept, CONFUCIUS will at some stage ask the trainer to show an object. This object is written in the form (object name (property1.value1) (property2.value2) ...), that is, the objects in CODE are described as a list of property/value pairs and a reference name, for example, (BLOCK (COLOUR.RED) (SHAPE.CUBE) (SIZE.BIG)) describes an object which is a big, red cube and is called a

*Department of Computer Science, University of New South Wales. Manuscript received 7th September 1978

"Copyright © 1978, Australian Computer Society Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that ACJ's copyright notice is given and that reference is made to the publication, its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society."
block. In this example, the property values are atomic. However, we would like to be able to give a structural, or hierarchical description of objects. If we wished to describe a “steeple”, which consists of a pyramid supported by a cube, it might be done as follows:

(STEEPLE (TOP SPIRE) (BOTTOM BLOCK))
(SPIRE (COLOUR BLUE) (SHAPE PYRAMID) (SIZE SMALL))
(BLOCK (COLOUR RED) (SHAPE CUBE) (SIZE BIG))

In the description of “steeple”, the property values are non-atomic. That is, they are themselves objects, which must be described as well.

As well as describing single objects, CODE allows us to describe sets of objects. For example, if we want to know if two people satisfy a “parent-child” concept, the set we might show is:

(PAIRPEOPLE FRED LUCY)
(FRED (AGE 40) (SEX MALE) (HOUSE WILLOWS))
(LUCY (AGE 10) (SEX FEMALE) (HOUSE WILLOWS))
(WILLOWS (SIZE BIG) (COLOUR WHITE))

The set type is useful for describing associations of objects where it is inappropriate to classify them as “properties” such as the above case. The set form also allows us to describe “n-ary” relations which involve more than two objects. Set operations may also be introduced (element-of, not-element-of).

Suppose we have a concept, C, which involves two objects. If the first object is big and red or the colour of the first is red and the size of the second equals the size of the first then this object-type is recognized by C. The concept may be described in CODE as

(OR
 (AND (EQUAL (OBJECT 1 COLOUR) RED)
      (EQUAL (OBJECT 1 SIZE) BIG))
 (AND (EQUAL (OBJECT 1 COLOUR) RED)
      (EQUAL (OBJECT 1 SIZE) (OBJECT 2 SIZE))))

This expression is self-explanatory except, perhaps, for the term

(OBJECT <property list>)

If CONFUCIUS is given two objects and the first is

(BLOCK (SHAPE CUBE) (COLOUR RED))

then (OBJECT 1 COLOUR) has the value RED, thus the statement

(EQUAL <term> <value>)

would be true in this case, so part of the requirement is satisfied.

On the other hand, since the size was not specified (OBJECT 1 SIZE) is unidentified and therefore the statement it is in is false, so the object fails the recognition test.

The statements which are possible in CODE include: giving the value of a property; equality between properties or objects; set operations such as “element-of” and “sub-set”; and negations of these. An important statement is that of the form (IN a CON), which is an explicit statement of object recognition. It states that the value of “a” is recognized by (contained in) the concept, CON. “a” may be a simple term or it may be a list of terms which forces a set to be constructed. This latter situation gives CODE the ability to recognize n-ary relationships.

An example will demonstrate a very important aspect of CODE — its recursive nature. Concepts may be described in terms of other concepts or in terms of itself. Consider the concept of binary numbers. Before NUMBER can be defined we must first describe the concept of a binary digit, DIGIT.

A number may be defined recursively as consisting of a head which is a number or null and a tail which must be a digit if the head is a number or a one only if the head is null. No leading zeros are allowed. Thus,

(OR
 (AND (EQUAL (OBJECT HEAD) NULL)
      (EQUAL (OBJECT TAIL VAL) 1))
 (AND (IN (OBJECT HEAD) NUMBER)
      (IN (OBJECT TAIL) DIGIT)))

To describe an object-type which is a number we use

(TWO (HEAD . ONE) (TAIL . ZERO))
(ONE (HEAD . NONE) (TAIL . UNITY))
(UNITY (VAL . 1))
(ZERO (VAL . 0))

Since CONFUCIUS is written in LISP, the internal representation of objects and sets is almost the same as the external form. Objects are lists of dotted pairs with a reference name attached. Sets are list of references to the objects within the set as well as a reference name for the set itself. A schematic diagram of PAIRPEOPLE is shown below.

The internal representation of concepts is quite different to their external representation. For example, the concept

(OR
 (AND (EQUAL (OBJECT HEAD) NULL)
      (EQUAL (OBJECT TAIL VAL) 1))
 (AND (IN (OBJECT HEAD) NUMBER)
      (IN (OBJECT TAIL) DIGIT)))

is represented as shown in the following figure.

This structure is called a graft because it is the fusion of two trees:
1. The statement tree — this stores the predicates in the expression.
2. The logic tree — this links nodes of the statement tree to represent the logical (AND, OR) connectives between statements.

Note that identical statements that occur in different parts of the external description appear only once in the statement tree. Also, statements with some initial part of their list of qualifying properties in common, share a branch of the tree up to the point where their properties differ. This gives the structure its economy and promotes the efficiency of the recognition and learning procedures.
The structure for storing the grafts is a table referenced by properties. Concepts to test for the recognition of an object are restricted to those which are pointed to by properties of the object. In this way irrelevant concepts are not considered. For example, a binary number would not be tested for containment for the concept describing submarines, as they have no common properties.

A complete description of CODE and its internal representations are given in Cohen (1978).

3. Object Recognition

Having formulated internal representations for object-types and concepts and a procedure for storing and accessing concept descriptions, we now discuss a procedure for discovering those concepts which recognize a given object-type.

After determining the set of eligible concepts using the procedure given in the previous section, the recognition process considers each of these concepts in turn. The process starts at the root of the statement tree and proceeds, depth-first, down this tree.

Referring to the graft shown earlier, we note that the tree search may encounter two types of node, a property name, or an operator.

If the current node is a property name, $p$, then the current object is searched for this property. The value, $v$, of $p$ is returned and used in examining the descendant nodes of $p$. If a property cannot be found or its value is of wrong type (e.g. integer when an atom is expected) then all operator nodes in the tree that are rooted at the property node are evaluated as false.

Each operator has associated with it a function. When the recognition procedure reaches an operator node in the statement tree then the appropriate function is called. For example, if EQUAL is reached then EQUALSTAT is called. This operator function compares the property value of the object-type with that stored in the concept description and returns true or false accordingly.

The results of evaluating an operator are passed on to those nodes in the logic tree pointed to by the operator. If the value of the operator is true then:

1. If the node is an OR operator, the value, true, is passed up to the parent of this node.
2. If the node is an AND then we must determine whether or not all conjuncts are true making the whole conjunction true. To do this a count of unevaluated conjuncts is kept. Each time a “true” is returned this count is decremented. If the count becomes zero the value, true, is passed up to the parent node.

If the value of the operator is false, then the same procedure as for true is carried but with OR and AND interchanged and with “true” replaced by “false”.

If a logic node has no parent, that is, it is the root of the logic tree, then the value of the whole recognition process stops with this value. The logic tree is processed each time a statement operator is evaluated. Therefore the recognition procedure may terminate without evaluating all concept statements.

To illustrate the recognition procedure we now show a trace of the object

\[(EX \ A \ B)\]

\[(A \ (COLOUR.RED) \ (SIZE.SMALL))\]

\[(B \ (COLOUR.BLUE) \ (SIZE.SMALL))\]

being recognized by the concept, $C$, described earlier, i.e. (IN EX C) is traced.

The function INCON is a recursive procedure which carries out the tree search. When an operator is reached it calls the operator function for that node. When control returns to INCON it calls FIXUP which then passes the result of the evaluation of the operator to the logic tree.
4. Concept Learning

Having discussed the basic elements of a pattern recognition system, we are now in a position to consider how such a system can grow, namely, by learning new concepts.

The learning algorithms are based on the “Conservative Focusing Strategy”, of Bruner, Goodnow and Austin (1956) which deals with conjunctive concepts only. In this method, the trainer gives the subject a “focus object”, that is, an instance of an object which belongs to the concept. The subject then makes a sequence of choices each of which alters only one property of the focus. If a property value is altered and the focus is still in the concept then that value is known to be essential to the concept description. However, if the object is no longer in the concept then that value is known to be essential to the concept description.

As an example, suppose we are given a big, red cube and told that it is a positive instance of a concept. We might then ask, “Is a small, red cube in the concept?”. The trainer responds with a “yes” or a “no”. If the answer is “yes” then the size of the object is irrelevant. Now we ask, “Is a green cube in the concept?”. The answer is “no”, so red is an essential part of the description. When a red pyramid also fails to be contained then we know that the concept is “red cube”.

When an object is presented as a positive instance of the concept, we build a concept to describe the object, that is, we generate a trial concept which becomes our first approximation to the concept to be learnt (call it the target concept). When the object is to be altered to test for recognition by the target concept, rather than changing one of the property values, a statement in the trial concept description is removed. For example (EQUAL OBJECT SIZE SMALL) is removed from the description when a red cube is to be tested. Note that by removing a statement, we relax the restrictions on objects which are contained in the concept. In the case of the specification of the size, when that statement is taken out of the description, the concept then contains objects of any size. Thus the removal of statements generalizes a concept description.

5. Learning Strategy: LSI

The first learning algorithm to be considered, LSI, generalizes the conservative focusing strategy in a number of ways.

Firstly, Bruner’s method only allows conjunctive concepts to be learnt, these are concepts described by statements connected by a single AND. However, LSI can learn disjunctive concepts as well, i.e. concept descriptions which include OR’s. The concepts are learnt in disjunctive normal form.

If the trainer presents CONFUCIUS with a positive instance of a disjunctive concept, then that example provides only enough information to learn one disjunct. In order to learn another disjunct, a new example satisfying this new sub-concept must be shown. Thus, the strategy is to learn one conjunctive sub-concept at a time and take the disjunction of all of them to be the final concept description.

A further generalization arises from the fact that LSI is able to operate in a more complex environment from Bruner’s subjects enabling it to learn relational and hierarchical concepts as described in CODE. This difference can be explained by an example. Consider the following statements

(EQUAL (OBJECT 1 SIZE) SMALL)
(EQUAL (OBJECT 2 SIZE) SMALL)
and (EQUAL (OBJECT 1 SIZE) (OBJECT 2 SIZE))

Note that each statement is directly implied by the other two, that is, the statements in a concept description need not be independent in CONFUCIUS, although this was not allowed by Bruner.

Suppose, in learning a conjunctive concept which has inter-dependent statements, one statement is removed for testing. If this statement is implied by others still resident in the concept description, then in fact, no new concept has been generated since the information contained in the removed statement is still present in its implicants.

In order to solve this problem, an implication network is constructed before the learning procedure begins. The network is superimposed onto the graft, providing direct implication information for each statement. Each statement points to those groups of statements which directly imply it. In this way it is possible to determine whether a new concept has been created by testing the implication information.

In order to produce a new concept, LSI first attempts to remove only independent statements. However, the case often arises when there are no such statements left. In this situation, groups of statements are arbitrarily removed until a new concept is produced. This concept is
then tested for containment in the target concept by asking the trainer. If it is not contained, then new combinations are tried. When it is contained then it is known that all those statements removed from the concept description, which are not implied by the remaining statements, definitely are not part of the description.

When all statements have been tested, the trainer is asked if the resulting concept is equivalent to the target. If the answer is "yes" then the concept has been learnt, if not the concept must be disjunctive. Thus, the partial concept is stored and another conjunction learnt.

If statements are to be removed from and replaced into the trial concepts as often as described, a great deal of processing would be involved—linking lists in and deleting them, storing them temporarily elsewhere, etc. Actually no statements are physically removed until the whole concept has been learnt. Instead a statement may exist in a number of states. The main ones are:

1. IS—definitely in the concept description.
2. UNCLASSifed—yet to be tested but temporarily assumed to be in the concept description.
3. TEMP—temporarily "removed", considered not in the concept description.
4. NO—definitely not in the concept description, "permanently removed".


In order to produce a new concept, LS1 removes statements arbitrarily and uses the implication information to test whether this removal has in fact changed the concept. Because a new concept cannot be guaranteed, LS1 must perform a number of passes through the statement list, L.

LS2 represents a significant improvement over LS1 in that it requires only one pass. Rather than using implications merely as a device for testing concepts, LS2 uses them to guide the removal of statements so that when they are taken out, we are guaranteed a new concept.

As before, the basic idea is to learn one disjunct at a time by removing statements and testing the new concept for containment in the target. In this way we can classify the statements removed as either belonging to the concept or not.

We have already seen that it may not be sufficient to remove one statement to produce a new concept because it could be implied by statements still in L. However, if we remove, not just one statement but also some of those that imply it, then we can guarantee that the concept formed in this way is independent of the old one.

To understand LS2 we must first describe the implication structures associated with each statement. Such a structure is shown below.

![Implication Structure Diagram]

It consists of a list where the first element is the state of the current statement (IS, NO, TEMP, UNCLASS), followed by the list of groups of statements which imply it.

The groups themselves are lists of references to other statements.

Implication groups are also assigned several states:

1. SUPERACTIVE—all statements in the group are marked IS.
2. ACTIVE—the statements are marked IS or UNCLASS.
3. INACTIVE—at least one statement is in TEMP or NO.

A PASCAL-like description of the higher level procedures in LS2 follows:

```pascal
procedure ls2;
procedure evaluate(i);
begin
  i.state := TEMP;
  case examine(i.subgroups) of
    superactive : i.state := IS;
    none_active : if A <= C then i.state := NO
      else i.state := IS;
    else : begin
      deactivate(i.subgroups);
      if a superactive group has formed then
        i.state := IS
      else begin
        while (A<<C) and more combinations do check;
        if A <= C then
          set all temporarily removed statements to NO
        else i.state := IS
        end
    end;
end;
end; (evaluate)
```

The main difference between LS1 and LS2 is that the inner repeat loop in LS2 represents the only pass through L. In LS1, when a statement was reached which had not yet been classified as IS or NO and it was not implied, then that statement would be skipped over. LS2 however, goes ahead and tries to classify all statements. This is the purpose of the procedure, EVALUATE.

A statement is classified by "de-activating" its implication subgroups, that is, by removing a statement from each of its active subgroups so that it is no longer implied by the remaining statements in L. The concept so formed is independent of the old one, and so may be tested for containment.

If the new concept is contained in the target then all the statements which have been removed temporarily may...

be set to NO. Note that none of these statements are implied by others remaining in L. Such a set is said to be "closed".

Since a statement may be implied by several combinations of other statements, some of which may be necessary in the concept description, the de-activation process may be carried out a number of times until either the new concept is contained in the target or there is no new combination of statements to remove.

Let us now consider the operation of EVALUATE. EVALUATE begins by calling EXAMINE. This procedure checks each subgroup of the statement, \( L_i \), to determine whether it is active, superactive or inactive.

If \( L_i \) has a superactive subgroup, that is, \( L_i \) is implied by statements which are known to be in the concept, then \( L_i \) definitely is in the concept.

If there are no active subgroups, that is, \( L_i \) is not implied by any other statement, then by removing \( L_i \) we create a new concept which we test for containment in the concept, \( C \), to be learned.

However, if neither of these conditions hold then the classification problem is not so easy to solve. Because some of the statements in the active subgroups are themselves unclassified, we must try to classify them, which means classifying the statements in their active subgroups, and so on.

Clearly then, EVALUATE will involve a tree search procedure. A path in the tree terminates when a statement has either an active subgroup which is superactive or it has no active subgroups at all. These results are backed up the tree to classify statements higher up in the search.

Now, eventually we may find that the original statement, \( L_i \), has a superactive subgroup after backing up, so again, \( L_i \) is placed in the IS list, that is, definitely in the concept. Otherwise the containment question is asked of the concept formed by removing \( L_i \) and all its descendants which have been temporarily removed during the tree search. This produces a completely independent concept, as the removed set, by construction, is closed.

If the new concept is contained in \( C \) then all those statements which have been temporarily removed may be removed permanently.

However, if the answer is no then some of the statements removed did belong to the concept. When each subgroup was processed, one unclassified statement was chosen for removal. Failure of the containment test requires at least one new statement to be removed. Thus a different combination of implications is removed, creating another independent concept. The procedure CHECK is called to produce the different set of temporarily removed statements each time.

Suppose every subset produced by CHECK results in a new concept which is not contained in \( C \). Since \( L_i \) is always in the removed set, it must be responsible for the failure of the test, so it can be put in the IS list as definitely part of the concept or implied by some statements that are. In this way, LS2 always guarantees that information can be gleaned from the containment question.

Both LS1 and LS2 can operate in a non-interactive mode by supplying a list of positive and a list of negative instances. The containment question is replaced by testing each of the negative instances for recognition in the concept. If they fail then the containment question is satisfied. Otherwise, it is not satisfied as the trial concept has clearly been over-generalized in that it recognizes a negative example (cf. the "near-misses" of Winston [1970]). The equivalence question is satisfied if all positive instances are recognized by the trial-concept. A complete description of all aspects of LS1 and LS2, including flow charts can be found in Cohen (1978) and Sammut (1977).

7. Conclusion

As can be seen from the table of results in the appendix LS2 would seem to be much more efficient (based on the criteria chosen) than LS1. This is not unexpected, as has been pointed out LS2 has much more direction in its search and always guarantees the gaining of some information from tests it performs. In any case, both LS1 and LS2 are powerful learning strategies in that they enable disjunctive as well as conjunctive concepts to be learnt and, in particular, can learn concepts involving n-ary relations, hierarchies and recursions. In comparison, Winston's (1970) algorithm is restricted to conjunctive concepts involving, at most binary relations. Michalski's (1974) VL1 system also cannot learn n-ary relations.

Therefore the learning algorithm, LS2, in conjunction with the description language, CODE, and the object recognition procedure constitute a powerful learning system - called CONFUCIUS. Such a system is a pre-requisite for any intelligent system, artificial or otherwise.

References


Appendix

From the table of results, it is evident that LS2 is more efficient than LS1 over all criteria used. LS2 takes less time, asks fewer containment questions, gains more information per containment question (i.e. finds more in and out of the target) and usually removes fewer statements to form a question.

Note, the times shown below are adversely affected by a number of extraneous system dependent variables. These include the program being interpreted, not compiled; and a restriction on the size of core storage which necessitates the use of the software virtual memory facility of UT-LISP, resulting in some thrashing. The size restriction is also responsible for a large number of garbage collections.
### Table of Results

<table>
<thead>
<tr>
<th>Concept</th>
<th>Form</th>
<th>Init</th>
<th>Time</th>
<th>Cont</th>
<th>Time Cont</th>
<th>In Cont</th>
<th>Out Cont</th>
<th>Temp Cont</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARK</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>3</td>
<td>3.0</td>
<td>1.7</td>
<td>0.3</td>
</tr>
<tr>
<td>M1 1/0</td>
<td>1 : 1</td>
<td>14 : 14</td>
<td>64</td>
<td>100</td>
<td>15</td>
<td>18</td>
<td>4.3</td>
<td>5.6</td>
</tr>
<tr>
<td>M1 0/1</td>
<td>2</td>
<td>14</td>
<td>31</td>
<td>53</td>
<td>9</td>
<td>11</td>
<td>3.4</td>
<td>4.8</td>
</tr>
<tr>
<td>M2 1/0</td>
<td>2 : 2</td>
<td>10 : 10</td>
<td>52</td>
<td>63</td>
<td>14</td>
<td>14</td>
<td>4.8</td>
<td>4.5</td>
</tr>
<tr>
<td>M2 0/1</td>
<td>2 : 2</td>
<td>10 : 10</td>
<td>67</td>
<td>63</td>
<td>14</td>
<td>14</td>
<td>2.7</td>
<td>4.5</td>
</tr>
<tr>
<td>DIGIT</td>
<td>1 : 1</td>
<td>2 : 2</td>
<td>6</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>6.0</td>
<td>8.0</td>
</tr>
<tr>
<td>NUMBER</td>
<td>2 : 2</td>
<td>5 : 12</td>
<td>68</td>
<td>108</td>
<td>11</td>
<td>14</td>
<td>6.2</td>
<td>7.7</td>
</tr>
<tr>
<td>LESSD</td>
<td>2</td>
<td>6</td>
<td>17</td>
<td>64</td>
<td>3</td>
<td>3</td>
<td>5.7</td>
<td>21.1</td>
</tr>
<tr>
<td>LESS</td>
<td>2 : 2 : 1</td>
<td>6 : 13 : 10</td>
<td>330</td>
<td>*</td>
<td>23</td>
<td>*</td>
<td>14.7</td>
<td>*</td>
</tr>
<tr>
<td>H - R</td>
<td>1</td>
<td>15</td>
<td>69</td>
<td>117</td>
<td>6</td>
<td>7</td>
<td>11.5</td>
<td>16.7</td>
</tr>
<tr>
<td>ARCH</td>
<td>3</td>
<td>16</td>
<td>85</td>
<td>*</td>
<td>9</td>
<td>*</td>
<td>9.4</td>
<td>*</td>
</tr>
</tbody>
</table>

**Legend**

**FORM:** Structure of the learned concept, e.g. 1 : 3 is a disjunctive concept, whose first disjunct has one statement and the second has three.

**INIT:** The number of statements in the initial trial concept for each disjunct.

**TIME:** The total time taken, in seconds, to learn the concept for LS2 and LSI, given as LS2 | LSI.

**CONT:** Total number of containment questions asked for LS2 | LSI.

**TIME CONT:** Average time taken to form a containment question for LS2 | LSI.

**TEMP CONT:** Average number of statements temporarily removed during the formation of a containment question LS2 | LSI.

**NOTE:** The * indicates that LSI had not learnt the concept after 512 CPU seconds. For the concept LESS, this occurred at the first disjunct.

---

**Book Review**


The term Software Engineering, for readers unfamiliar with it, refers to the business of computer software design and development practised and taught as an engineering discipline, rather than as an intuitive craft as it has sometimes been regarded.

The book records the papers of the British Computer Society's Symposium on Software Engineering held at the Queen's University of Belfast in April 1976. The contributors are all Europeans and discuss European, and for the main part British, experience in software construction. A short selection of topics and speakers will indicate the scope of the discussion: C.A.R. Hoare provides the introduction and background; we have OS/360 at Cambridge University, a Modular One system at Strathclyde; Michael Jackson on design and usage of Cobol, K.V. Roberts of Culham Laboratory on Fortran and extensions to it; G. Newell discusses I.C.L.'s 1900 Cobol and George operating systems and Niklaus Wirth reviews the design of the Pascal Language and the development of its compilers.

The overall, frequently stated, message of the book is that Software Engineering, as a discipline, is as yet largely unstructured and without strong guidelines for sound practice. This message seems to have determined the mood of the book and the symposium which it reports. As a result most of the discussion is anecdotal rather than analytic — "what we did was ..." rather than "what should be done is ...". The opportunity appears, in the main, to have been missed of evaluating with hindsight, over the considerable number of software products discussed, the relative merits of the design and development options adopted and those rejected. The accepted goal of the contributors seems to have been to record, rather than to interpret, the product design and development histories presented. Specifically, the measured effectiveness of a product in the user's hands and the constraints of development project resource management are rarely mentioned in quantified terms as determinants in design decisions.

An exception to the above is the contribution of Wirth, who sets out with the intention of identifying criteria for assessment (of a programming language) and concludes with a suitable set of simple and stringent principles of design. He gives, into the bargain, quantified details of a compiler development which establish his principles as realisable and provide a bench mark for similar projects. Wirth's contribution should be read, in particular, not only for the facts it contains but also as an approach to delineating the problems which software engineering addresses.

In summary, the book is not a reference text but can be read, with considerable interest, as a "tale of our times" which deals, as the dust-jacket states, with "... the problems of earlier designs ...", the further claim that it shows "... how to remedy them" is not fulfilled.

Dr I.A. Parkin, Bassett Department of Computer Science, University of Sydney
Topological Design of a Small Resource Sharing Network

By A.D. Payne* and B.J. Lings*

In recent years there has been a rapid growth in the development and implementation of networks to achieve efficient sharing of geographically distributed computer resources. It is the purpose of this paper to detail an approach which, using available performance data on existing networks, derives an optimum topology for a network interconnecting the computing facilities of Australian research institutions.

KEYWORDS AND PHRASES: resource sharing network, network reliability, topological design, error control.

CR CATEGORIES: 2.40, 3.51, 3.81.

SHORT FORM TITLE: Network topology design.

1. Introduction

It has recently been proposed (Payne and Lings, 1978) that an Australian Universities and Research Organization Network (AURONET) be constructed to facilitate, initially, the interconnection of host computers belonging to these institutions. At present the significant and possibly unique computing resources of any particular institution are restricted to its local user group. Implementation of AURONET would extend the availability of such resources to all users with access to the network. This would ensure a substantial improvement in the range of computing facilities and services available to support education and research activities. At first, the network would, at least, provide new opportunities for users to obtain needed access to specialised resources such as software (compilers, application packages), data (files, data bases), hardware (special purpose processors, COM) as well as access to other researchers via electronic mail. Subsequently, AURONET's role as a nation wide data communications network would encourage a demand for its use for purposes not directly associated with teaching or research. For example, there is at present an unfulfilled need for interlibrary communication for interrogation of catalogue files which could be satisfied through a network of the type proposed here. It could be expected that network computing might only supply a small fraction, perhaps about 5 percent of the total computing load. However, this would be a very important fraction supplying to users specialised services which would not otherwise be available.

The establishment and operation of a nation-wide distributed computing network would of course present a number of particular problems. There is a technical problem of network design which is dealt with in some detail in this paper. The resolution of implementation and operating costs is essential to an economic appraisal of the project. A procedure for the selection of a network configuration to minimize recurrant communication costs is explained later. The initial hardware and programming costs are not considered. Also political differences will undoubtedly force administrative barriers to the availability of funds for the purchase of network services. The same barrier will cause difficulties in relation to the determination of a satisfactory accounting and billing procedure for network users. None of these problems can be regarded as insuperable but their existence can be expected to lead to a situation of lower initial network traffic and possibly to an overall reduction in traffic growth.

Circuit switching, message switching and packet switching are the three distinct communications (sub)network technologies that have been employed for the interconnection of autonomous host computers. In a circuit-switched network a dedicated circuit is provided to support user-host and host-host communication. However, with this technology, the time required to establish a new point-to-point circuit is in the order of seconds which must be considered too slow for the maintenance of dynamic communication between hosts. Message switching involves the transmission of a logical unit of information called a message from source to destination by a store-and-forward technique. The message is stored at each intermediate switching node until the proper outgoing line is available when it is then forwarded. Unless a message length priority system is implemented, this type of subnetwork technology can result in a relatively poor service to users sending short messages. Difficulties can also arise in finding sufficient buffer space at the switching nodes because of the frequently large variation in message lengths. Further, there is a high probability that an error will occur in long messages and the consequent error detection and message retransmission process can produce excessive delays. The problems associated with message switching can be alleviated or removed by subdividing a message in to one or more fixed length packets which can then be transmitted as separate independent entities over possibly different routes in the subnetwork. The message is reassembled either in the destination host or in the node to which the host is attached. Packet switching obviously has the capability to simultaneously support real-time, interactive and file transfer traffic modes. It is therefore the generally preferred technology for subnetwork implementation for resource sharing networks.

The largest and most successful packet-switched


"Copyright © 1978, Australian Computer Society Inc. General permission to republish, but not for profit, all or part of this material is granted, provided that ACJ's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society."
Topological Design

The network is the Advanced Research Projects Agency Network or ARPANET. It currently consists of approximately one hundred host computers interconnected by a subnetwork constructed with about fifty switching nodes. Extensive statistical measurements of ARPANET performance and operation parameters are available (Frank et. al., 1972; Kleinrock and Naylor, 1974; Kleinrock et. al., 1976) and for this reason the approach to the design of AURONET is based on the ARPANET design philosophy. All network design procedures begin with a specification of the fixed switching node locations. The relative geographic positions of Australian Universities strongly suggest that the switching nodes of AURONET be located at Sydney, Melbourne, Canberra, Adelaide, Brisbane and Armidale. Long-haul part-time leased lines or switched lines may be used to connect Hobart, Perth and Townsville to the corresponding nearest node of the network.

2. Network Reliability

If users are to be attracted to the facilities offered by a multi-host network they must be guaranteed a consistent availability of any particular host. This situation can only exist when the network contains sufficient line redundancy to sustain communication between operative nodes in the event of failure of network components. The fraction of all pairs of nodes which can be expected to communicate is a natural definition of network reliability for it measures the extent of possible host-host communication. A highly reliable network can be derived with enough component redundancy but a trade-off between cost and achievable improvement in reliability is, of course, essential.

The structure of the communications subnetwork can be modelled by a connected undirected graph $G=(V,E)$. In AURONET the size of the node set is fixed at $p=6$ while the size of the line set, $q$, is a design parameter. It is now possible to propose a number of feasible AURONET topologies with each specified in terms of its graph theoretic characteristics. Various network topologies are shown by the graphs of Fig. 1. Here, $\kappa$ is the graph connectivity, $\lambda$ is the line connectivity and $\delta$ is the minimum degree of the graph. These graph parameters are related (Harary, 1969) by the inequality

$$0 \leq \kappa(G) \leq \lambda(G) \leq \delta(G) \leq \left[ \frac{2q}{p} \right]$$

Using the parameter values of Fig. 1 it is seen that topologies (b), (d) and (e) are all $\kappa$-optimal and $\lambda$-optimal since in these cases $\kappa=\lambda=2q/p$.

If the line and node outage rates are denoted respectively as $P_l$ and $P_n$ then the expected number of pairs of non-communicating nodes can be written

$$N(P_l, P_n) = \sum_{i=1}^{6} n(i) \left( \frac{q}{p} \right)^{i-1} + \sum_{i=1}^{q} m(i) \left( \frac{1}{p} \right)^{i-1}$$

where $n(i)$, $m(i)$ are the average numbers of non-communicating pairs due respectively to $i$ node failures and $i$ line failures. Here $m(i)=0$ for $i=0, 1, \ldots, \lambda-1$. We define the network reliability to be $R=1-N/15$ since 15 is the total possible number of communicating pairs. There is a considerable variation in line outage rates with geographic location. Further, due to the common routing of local loops connecting a user (node) to an exchange, lines in this type of network do not fail independently. However, specific data on line outages is not available and consequently the simplifying assumption is made that nodes and lines fail independently with the same network-wide outage rates. Under these circumstances, the two contributions of $N$ specified in equation (2) can be determined by a combinatorial analysis. The results are shown in Fig. 2 where the expected number of non-communicating pairs of nodes is plotted against the element failure probability $P=P_l=P_n$ for each considered topology. As well, the network reliability is shown on the right-hand vertical scale.

Experience with ARPANET (Frank et. al., 1972) indicates an expected failure probability of $P=0.2$ for both nodes and lines. This figure is reasonably consistent with the common carrier quoted outage rates for local loops which are usually the major source of circuit failure. The plotted curves of Fig. 2 indicate that for small anticipated

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Some feasible topologies for networks of six nodes.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{The expected numbers of non-communicating nodes and network reliability for the five topologies of Figure 1. Nodes and links are assumed to have the same outage rates.}
\end{figure}
failure rates AURONET network reliability will be most affected by the occurrence of node failures. It is also clear that a 1-connected network will exhibit unacceptable reliability. On the other hand, only a small improvement in network reliability is achievable with networks having greater topological complexity than that of Fig. 1(b) and, on reliability consideration alone, the extra line costs for these networks would not be warranted. A sufficiently reliable network can therefore be derived with a minimal 2-connected topology.

3. Message Characteristics

A packet-switching network provides a transportation mechanism for the delivery of data between host computers that are interconnected via the subnetwork. In order for the data to be meaningfully and properly interpreted it is necessary to establish a system of message formats by defining appropriate communication protocols. The initiation and maintenance of communication is the primary goal of protocol design, however, protocol parameters also influence network performance measures such as communication efficiency and reliability. The direct consequence of protocols is to induce a packet transmission overhead which may be regarded as all those characters transmitted that are not exchanged between user processes.

It is assumed that a Network Control Program (NCP) is resident in each host with the responsibility for the transmission and acceptance of messages on behalf of the processes it serves. A user process is consequently considered to be any process that makes use of the system calls provided for communication with the NCP. It should be noted that alternative terminology for NCP is current. In the CYCLADES network (Zimmerman and Elie, 1974) it is referred to as a TS (transport station) which is the same nomenclature advanced by Cerf et al., 1976. The acronym TCP for transport control program is used by Cerf and Kahn (1974).

All user processes within a host communicate with the NCP through several input-output paths or ports. Interprocess communication is then initiated by establishing an association of ports in the NCP's to which the processes are attached. Accomplishing this task requires that ports have a network wide unique identification and this is arranged by defining an intermediate process name space which is mapped by each host into its own internal process name space. An element in this name space is formed by the concatenation of at least three numbers—a user number, host number and port number. In ARPANET the elements are called sockets and the port associations are half-duplex links.

In a packet switched network, a number of communication levels can be identified and each of these is implemented with an associated protocol. Broadly, three levels can be specified:

1. Level-0: communication between nodes of the subnetwork.
2. Level-1: communication between a node and an attached host.
3. Level-2: communication between the network hosts.

In ARPANET line overheads for different protocol levels are described by Kleinrock et al. (1976). The level-0 protocol is responsible for two overhead categories. One corresponds to the protocol for packet transmission between adjacent nodes and includes characters for clock synchronization, packet delimiting, and checksum and acknowledge characters for error control. The other overhead category determined by level-0 protocol consists of all characters necessary for message control in the subnet i.e., transmission control between the source and destination nodes. In user packets, this overhead includes characters for source and destination specification and packet and message numbers to facilitate message reassembly at the destination node. A substantial contribution to the second category overheads of level-0 protocols is imposed by subnet control messages which have the principle purpose of ensuring that sufficient buffers are available at the destination node to accommodate arriving packets. The rate at which messages are transmitted from a source node is regulated by the subnet control messages. A message is not sent until notification is received from the destination node of buffer availability. When a message has been correctly reassembled at the destination node and in turn passed on to a host a Ready-For-Next-Message (RFNM) is returned to the source node certifying completion of message delivery to the destination node. Transmission of the next user message is not initiated until an ALLOCATE message is received from the destination node signalling allocation of buffers for message reception. If sufficient reassembly buffer space is available when an RFNM is sent it “piggy-backs” an ALLOCATE message. Observed behaviour of subnet control messages in ARPANET indicate that the ALLOCATE message is invariably piggy-backed on a RFNM.

Transmission overheads for the communications subnet do not include the overheads associated with level-1 protocols. Indeed, in networks of the ARPANET type where messages are fragmented and reassembled in the nodes rather than the hosts, the protocol design at this level is largely determined by the particular host machine and can be regarded as a technical implementation problem.

Communication at level-2 is handled in ARPANET with a host-host protocol specified for each message and host-host control commands. The host-host protocol is required for reference to processes in foreign hosts and it defines a port as described earlier. Control commands are needed to open, maintain and close connections (ports) between user processes and to establish message space in the receiving host.

The measured ARPANET overheads are summarised in Table 1. The third row in this Table refers to the characteristics of the first packet in a user message while the fourth row refers to subsequent packets which do not contain 40 bits of host-host protocol. It is apparent that approximately 87 percent of all user messages are single packet. After subtracting out overheads, the average user packet contains 226 data bits.

### Table 1. Packet characteristics in ARPANET

<table>
<thead>
<tr>
<th>Packet Type</th>
<th>No. of bits</th>
<th>% total packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet control</td>
<td>168</td>
<td>49</td>
</tr>
<tr>
<td>Host-host control</td>
<td>318</td>
<td>29</td>
</tr>
<tr>
<td>First user packet</td>
<td>450</td>
<td>26</td>
</tr>
<tr>
<td>Subsequent user</td>
<td>410</td>
<td>4</td>
</tr>
</tbody>
</table>

4. Network Traffic

The major parameters influencing the assignment of line capacities in the communications subnet are the expected node to node packet traffic rates. Values for these
parameters can be easily determined when the traffic between each pair of hosts is known. It has been shown in the last section that the majority of user messages, which constitute about 30 percent of message traffic, are single packet. Since all other network (control) messages are also single packet it will henceforth be assumed on a close approximation, that every message is single packet.

Suppose that the one-way packet transfer rate between host k attached to node i and host l attached to node j is proportional to the product of their respective “weights” $w_{ik}$ and $w_{jk}$. In regard to network traffic, a suitable measure of host weight might be the size of its local user population or the amount of funds its users have for network computing. Assuming symmetrical traffic flow between each pair of hosts, it follows that the one-way traffic (packets/sec) between nodes i and j at time t can be written

$$\gamma_{ij}(t) = R(t) \sum_k w_{ik}W_{jk}$$

$$= R(t)W_{ij}$$

where $W_{ij} = \sum w_{ik}W_{jk}$ is a node-weighting and R(t) is a time dependent function. This equation implies that any changes in packet traffic rates with time are uniform throughout the network. The total number of packets entering the entire network per second is then

$$\gamma(t) = \sum_{i,j} \gamma_{ij}(t) + \sum_i \gamma_{ii}(t)$$

$$\sum_j \gamma_{ij} = \sum_i \gamma_{ii}$$

where the first summation assumes $\gamma_{ii} = \gamma_{jj}$ and the second summation quantifies the amount of intranodal or incestuous traffic in the network. Combining equations (3) and (4) yields

$$\gamma_{ij} = \frac{\gamma_{Wij}}{\sum_{ij} w_{ij} + \sum_i w_{ii}}$$

After normalizing, the denominator of the expression on the right hand side of this equation simplifies to

$$\gamma_{ij} = \gamma W_{ij}$$

Now the $w_{ij}$ which are listed in Table 2 represent the fraction of all host-host traffic input to the network which are transmitted from node i to node j. These weights are calculated from institute staff establishments.

### TABLE 2. One-way traffic rates as proportions of total traffic.

<table>
<thead>
<tr>
<th></th>
<th>Sdy</th>
<th>Melb</th>
<th>Canb</th>
<th>Bris</th>
<th>Adel</th>
<th>Armidale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sydney</td>
<td>0.093</td>
<td>0.051</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Melbourne</td>
<td>0.097</td>
<td>0.021</td>
<td>0.004</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Canberra</td>
<td>0.029</td>
<td>0.021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Brisbane</td>
<td>0.042</td>
<td>0.031</td>
<td>0.009</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adelaide</td>
<td>0.039</td>
<td>0.026</td>
<td>0.008</td>
<td>0.011</td>
<td>0.004</td>
<td></td>
</tr>
<tr>
<td>Armidale</td>
<td>0.015</td>
<td>0.012</td>
<td>0.003</td>
<td>0.004</td>
<td>0.004</td>
<td>0</td>
</tr>
</tbody>
</table>

A reasonable estimate of the initial packet traffic and its subsequent growth on AURONET can be made from available data on measured ARPANET traffic rates. In the latter case the initial packet input rate to the network was about 0.05 packets/sec/host from 20 connected hosts, which expand to a rate of about 1.0 packets/sec/host two years later from 50 hosts. Translating these figures to AURONET would mean an initial total packet input rate of $\gamma=1$ growing to a rate of $\gamma=35$ in three years.

Host-host communication when both hosts are connected to the same node is said to be incestuous. This type of communication does not require the packet switching facilities of the subnet but does require some reassembly storage in the node. Some relatively large volumes for intranodal weights listed in Table 2 indicate that it may be of interest to ascertain the expected extent of incestuous traffic in AURONET. Equation (6) shows that the fraction of all packets entering the network that are incestuous is

$$I = \frac{1}{2} \sum_i \gamma_{ii} = 2 \sum_i W_{ii}$$

Doubling the sum of the diagonal entries in Table 2 yields $I=0.30$ for AURONET which compares with the measured incest of $I=0.22$ on ARPANET. The average number of hosts attached to each node in these networks is about two and three respectively which, together with the uneven distribution of hosts in AURONET, accounts for its higher expected incest.

### 5. Topological Design

A number of algorithms have been proposed (Gerla and Kleinrock, 1977) for the least-cost topological design of distributed packet-switched networks. In most cases the node locations are assumed fixed and known. It is then required to minimize the total line cost for a specified set of design variables which could include the actual network topology, channel capacities, and the traffic routing strategy. The derived minimum cost is usually subject to design constraints such as average packet delay and reliability requirements. No efficient techniques exist for the exact solution of this topological design problem. However, heuristics which have been developed provide good suboptimal solutions.

It has been proposed that AURONET have only six switching nodes and it is therefore possible to find the required least-cost topology by exhaustive search methods. Although this technique is feasible here it becomes rapidly intractable as the number of nodes increases. For example, assuming all lines in the network have the same capacity, the number of different 2-connected topologies for six nodes is 60 while for ten nodes the number is 181,440! The following formulation of the AURONET design problem may be specified:

Given

- Fixed node locations
- Expected traffic between nodes
- Common carrier channel capacities and costs
- Total line cost
- Topology
- Channel capacities
- Routing strategy
- Average packet delay
- Maximum packet delay
- Line connectivity
- Longevity

A description of the computation procedures for the solution to this problem will be given in section 7. We first seek to develop appropriate and usable expressions for message delays in the network.
A suitable performance measure of network responsiveness is the round-trip delay for an external message input to the network. This delay may be defined as the elapsed time between a host-host message entrance to the network at a source node and reception at the same node of a request-for-next-message returned by the destination node. It is assumed that, as before, all messages are single packet and further that nodal processing delays are negligible. The average round-trip message or packet delay in the network may then be written

\[ D_r = \frac{1}{\gamma} \sum_{i,j} \gamma_{ij}(T_{ij} + T_{ji}) \]  

where \( T_{ij}, T_{ji} \) are respectively the expected propagation times for a host-host packet transmitted from node \( i \) to node \( j \) and for an RFNM transmitted from node \( j \) to node \( i \).

Now suppose that \( k \) and \( t \) are adjacent nodes and the line \( k \) belongs to the set \( L \) of lines which routes packets from source node \( i \) to destination node \( j \), the routing strategy being fixed. The forward propagation delay \( T_{ij} \) is then

\[ T_{ij} = \sum_{k \in L} t_{ik} \]

In order to evaluate the \( t_{ik} \) analytically the following specific assumptions are made: 1) Poisson arrivals at each node; 2) exponential packet length distribution; 3) negligible nodal processing time; and 4) error-free channels. Consequently the delay on line \( k \), which is the sum of transmission and queueing delays, is

\[ t_{ik} = \frac{1}{\mu C_{ik}} + \frac{n_k}{k C_{ik}} \]  

where \( C_{ik} \) is the channel capacity, \( n_k \) is the expected queue length at node \( k \) for packets being transmitted to node \( i \), the mean host-host packet length is \( 1/\mu \) and \( 1/k \) is the mean length of all packets. Similarly, the reverse propagation delay is

\[ T_{ji} = \sum_{k \in L'} t_{jk} \]  

where \( L' \) is the set of lines which routes packets from node \( j \) to node \( i \). The delay for an RFNM on line \( k \) is

\[ t'_{ik} = \frac{1}{\mu C_{ik}} + \frac{n_k}{k C_{ik}} \]  

where \( 1/\mu' \) is the mean length of subnet control packets and \( n_k \) is the expected queue length at node \( i \) for packets being transmitted to node \( k \).

All lines in the network must be full-duplex in order to support simultaneous both-way transmission. Also, the uniform bidirectional traffic flow imposed with the earlier assumption of \( \gamma_{ij} = \gamma_{ji} \) ensures identical traffic composition and rate for both channels of any particular line. In these circumstances the queues specified in equations (10) and (12) will be the same length and elementary queueing theory gives

\[ n_k = \frac{\nu_{kt}}{\bar{C}_{kt} \nu_{kt}} \]

where \( \nu_{kt} = \nu_{kt}/n_k \) is the arrival rate at node \( k \) of all packets to be forwarded to node \( i \). In the case of equal numbers of host-host and subnet control packets the mean packet lengths are related by \( 2l'/l' = 1/\mu + 1/\mu' \). Using this and the derived expression for \( n_k \) and \( n_i \) in equations (10) and (12) we find

\[ d_{kt} = t_{ik} + t'_{ik} \]

where \( d_{kt} \) can be considered a “round-trip” delay for adjacent nodes \( k \) and \( t \) and \( d_{kt} = \nu_{kt}/2 \) is the one-way host-host traffic on line \( k \) which can be determined from the predicted or given host-host traffic as described earlier. Thus, finally the average round-trip packet delay is

\[ D_r = \frac{1}{\gamma} \sum_{i,j} \gamma_{ij} \sum_{k \in L} d_{kt} \]  

which can be computed once the network topology, channel capacities and traffic routing strategy are specified.

6. Error Control and Packet Length

The node-node protocol uses retransmission to recover from loss or damage to packets. The occurrence of a damaged packet can be detected with a polynomial error-detecting code (e.g. Payne, 1977). A polynomial of order \( r \) allows detection of all error bursts of length less than \( r+1 \), of a burst of length \( r+1 \) with a probability \( 1-(1/2)^r \), and of a burst of length greater than \( r+1 \) with a probability \( 1-(1/2)^{r+1} \). For ARPANET \( r=24 \) which obviously means a very high degree of data protection. Line or equipment outages would result in lost packets and for this reason it is preferable to implement an error control system in which a packet is retransmitted if after a time-out period, say \( t_0 \), the sending node has not received a positive acknowledgement signal from the adjacent receiving node.

It is desirable to select a \( t_0 \) which is large enough to ensure that no bandwidth is wasted on unnecessary retransmissions. On the other hand a small \( t_0 \) will reduce packet transit times in the network because lost or damaged packets will be retransmitted sooner. It may be assumed that the ACK message generated on correct reception of a packet is piggy-backed, if possible, on a packet travelling on the reverse channel. If the fixed maximum packet size is \( L \) bits then \( L/C \), where \( C \) is the channel capacity, is the maximum wait encountered before an ACK is transmitted. Under these circumstances we set

\[ t_0 = 2(L/C + t_p) \]  

where \( t_p \) is the signal propagation delay on the line. For copper wire circuits \( t_p \approx 10 \) ms for a line distance of 2000 km. Allowing an average packet transmission time before a packet can be retransmitted, the total elapsed time from initiation of transmission of an individual packet to its successful receipt is the packet service-time.

\[ t_{ps} = t_p + 1/\xi C + E(t_o + 2/\xi C) \]  

where \( E \) is the expected number of packet retransmissions. Now it is easily shown that \( E=P_r/(1-P_r) \) where \( P_r \) is the retransmission probability. For a high order polynomial error-detecting code \( P_r = P_e = 10^{-9} \) and \( P_e \) is the mean packet corruption and BER is the mean
bit-error-rate. Consequently $E = \frac{BER}{(\xi - BER)}$. The data throughput efficiency due to transmitting a single packet on an individual line and normalized by the packet format efficiency is then

$$\eta = \frac{1}{I_{pr} \xi C}$$

Equation 18

Plots of normalized efficiencies as functions of the bit-error-rate are given in Fig. 3 for different maximum packet lengths and a channel capacity of 4800 bps. It is well known that on common carrier circuits errors in digital data are not uniformly distributed but instead tend to occur in bursts. Usually transmission conditions are good and the average bit-error-rate may be about $10^{-6}$. However in periods of poor conditions due to intense noise sources the average bit-error-rate could be $10^{-4}$ or higher. The graphs of Fig. 3 indicate that in these circumstances the throughput efficiency is significantly lower for larger $L$ thus suggesting the desirability of a small selected maximum packet length. Another advantage of a minimum $L$ is the resulting reduction in buffer storage requirements at the switching nodes. A suitable packet length is also suggested from the ARPNET measurements where the average packet length is only 282 bits. Further the very high proportion of single packet messages indicates that most transmission is in character-at-a-time mode with almost insignificant file transfer for which a large packet length would be useful. Consequently it is proposed that a maximum packet length of 512 bits would be appropriate for AURONET.

7. Derivation of Topologies.

In specifying constraints for the network it must be borne in mind that it is to support an interactive mode of operation. The minimum performance considered suitable in this context is an average round trip delay of under 200ms. If this is the only performance constraint specified then minimizing cost will result in a topology which penalizes light users of the system. This is considered to be unsatisfactory, and a further constraint is added limiting the maximum expected round trip delay for messages between any source and any destination to 500ms. Both delays can be calculated by applying equations (14) and (15) after the $\beta_i$ are known. The performance constraints are specified in terms of round trip delay for two reasons. Firstly, in an interactive environment the round trip delay must be small enough to cope with character echoing at reasonable typing rates. Secondly, round trip delay is a measure of performance which can be readily and regularly checked by the system for statistical purposes. The network is also to be designed under the longevity constraint that the chosen topology must be able to cope with expected traffic growth for a period of one year. It is imposed in order to overcome the problems associated with frequent topology changes which would otherwise arise.

A network optimization program has been written (Payne & Lings, 1978) to generate minimum cost topologies for given network parameters and subject to specified constraints (section 5); due to the small number of nodes in AURONET this program uses exhaustive search rather than heuristic techniques. It was determined in section 4 that an initial packet input rate to the network of $\gamma = 1$ can be anticipated. This figure together with inter-nodal fractions of traffic as specified in Table 2 provide the base data for the program. This former figure will be low because users will be essentially unaware of the flexibility of the new resource. During the first year familiarity with the network and its available facilities will grow, and there will be a natural increase in the workload imposed on the system. If we assume that user behaviour on AURONET will parallel that on ARPNET (Kleinrock & Naylor, 1974) then this will result in a figure of approximately 9 packets per second for total host-host traffic by the end of the first year of operation. During the second year ARPNET experienced a large growth in the number of host computers connected to the network, and frequent changes to the topology had to be made. This involved siting new nodes as well as changing inter-node connections. Although it is highly likely that similar growth would take place in AURONET, the geographical concentration of institutions suggests that the initial proposal for node siting will still be optimal. Therefore the projected growth in traffic to 33 packets per second by the end of the second year must be catered for by changes in node connections and line speeds. One possibility is to insert new links in order to relieve the saturated sections of the subnetwork. In AURONET, the network optimization program does not select this solution for low to medium traffic rates. With the scale of Telecom charges presently in operation (see Table 3) it is considerably cheaper to upgrade links than to insert new ones; in the former case increased charges are due almost entirely to the increased cost of the necessary modems. Consequently, in early years, the minimum cost configuration for the network will have minimal 2-connectivity. Such solutions will have acceptable reliability, as has been shown in section 2. The network optimization program therefore keeps node sitings fixed and manipulates links, line speeds and routing strategy in order to derive a minimum cost network under the given constraints. Because of the longevity constraint two
TABLE 3. Cost of full time Trunk Circuits (Nov. 1977)

<table>
<thead>
<tr>
<th></th>
<th>Adel</th>
<th>Armidale</th>
<th>Bris</th>
<th>Canberra</th>
<th>Melbourne</th>
<th>Sydney</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate</td>
<td>34126 (2400 bps)</td>
<td>35626 (4800 bps)</td>
<td>39627 (9600 bps)</td>
<td>183136 (48000 bps)</td>
<td>136260 (9600 bps)</td>
<td>161091 (9600 bps)</td>
</tr>
<tr>
<td>Rate</td>
<td>35626 (4800 bps)</td>
<td>39627 (9600 bps)</td>
<td>183136 (48000 bps)</td>
<td>136260 (9600 bps)</td>
<td>161091 (9600 bps)</td>
<td>161091 (9600 bps)</td>
</tr>
<tr>
<td>Rate</td>
<td>39586</td>
<td>13636</td>
<td>12866</td>
<td>44013</td>
<td>16017</td>
<td>89385</td>
</tr>
<tr>
<td>Rate</td>
<td>205458</td>
<td>62980</td>
<td>26744</td>
<td>17803</td>
<td>26917</td>
<td>105737</td>
</tr>
<tr>
<td>Rate</td>
<td>183136</td>
<td>89385</td>
<td>26744</td>
<td>17803</td>
<td>26917</td>
<td>105737</td>
</tr>
</tbody>
</table>

Optimization runs were performed, one for an input rate of 9 packets per second and the other for a rate of 33 packets per second.

Because of the existence of constraints on the average and maximum round trip delays, 2400 bps lines are found to be unacceptable for AURONET. The cost of full time trunk circuits plus modems for the initial topology (see Fig. 4) is $117565 for the year, representing a cost to each institution in the order of $7,000. This will be significantly reduced during the second and third years of operation as other institutions are encouraged to connect to what will then be a well established facility. Initial connection of hosts to the network would be the responsibility of individual institutions, although once the specifications have been met for one machine the task of connecting similar machines would be considerably reduced. For example, the significant number of DEC PDP-10's in use suggests considerable scope for cooperation and cost sharing.

The program generates the optimum fixed routing strategy to be employed. Although ARPANET has adopted dynamic routing, statistical studies of the system (Kleinrock & Naylor, 1974) have established that the vast majority of packets travel by the shortest route. The overhead involved in dynamic routing is therefore not considered warranted. However in the case of a node or line failure re-routing of messages would occur. The chosen routing algorithm maximizes possible throughput and hence increases longevity and decreases cost per megabit at the highest acceptable input rate. By the nature of the intended use of AURONET this figure is not of crucial importance. However it is worth commenting that as traffic increases, and with it the line capacities in the network, the figure drops dramatically. The average number of links traversed by packets in the system can readily be calculated from expected line usage figures; for topology 1 this value is 1.3. The figure is low because of a high expected rate of incest (30%) for this network.

The large volume of traffic between Sydney and Melbourne forces a topology change after one year, with a direct link being introduced between these two cities. Cost minimization still results in a minimally 2-connected network, but all lines must now be upgraded to 9600 bps. Having incorporated a direct Melbourne-Sydney line the average hop distance drops even further, to 1.14, but the high utilization of this link (32%) by the end of the year will necessitate a further topology change. In this case upgrading capacities would require the introduction of 48000 bps lines, which are expensive and have less reliable modems. The alternative is to introduce extra links to cut down the high line utilizations. At this stage it would not be useful to predict traffic growth for the third year of operation.
network operation. However, the effect of introducing a direct Sydney-Melbourne link into the minimally 2-connected topology of Fig. 4, at the same time upgrading all lines to 9600 bps, can be seen in Fig. 5 (topology B1). Line and modem costs for this network amount to $160000 per annum, and it can handle packet input rates of up to 58 packets per second: an extremely high rate of input.

8. Conclusions
This paper has presented theoretical aspects of the design of resource sharing networks and the topological design of AURONET, a small network initially intended to fulfill the need of Australian education and research institutions. The behaviour of such a network in the Australian environment is unknown and measurements on similar networks overseas have been used to predict likely traffic patterns. Certain simplifications have had to be made; in particular traffic between any two nodes is assumed to depend only on institution sizes, not on the more nebulous grounds of popularity and resource availability. In fact measurements on ARPANET suggest a certain amount of favouritism between geographically close nodes, even though the network topology is transparent to users. Also certain 'service' hosts are particularly heavily used. The affects of these factors cannot be realistically estimated, and the simplifications of random destination assumed in the paper are considered justified in establishing an initial network proposal. The proposal has been presented in the paper, together with projections of likely usage over a two year period. It is likely that once AURONET has been established the level of use would rise consistently, and the class of users would expand far beyond the initial base of universities and similar research bodies.

9. References

---

**Topological Design**


These are the proceedings of an invitational conference entitled "Ten-Year Forecast for Computers and Communications: Implications for Education – 1985" held in Arlie, Virginia, U.S.A. in September, 1975.

The papers included in the book discuss computer technology (hardware, artificial intelligence, graphics, speech recognition, etc.), communications technology (cable TV, satellites), video disc technology and the impact of these technologies on education. The book is divided into five main parts which correspond to the following five sessions of the conference:

1. LSI Technology and 1985 Educational Computing Systems
2. Storage Technology: Effect on Education.
3. What do Developments in Communications imply for the Distribution of Educational Resources?
4. If the Machine Gets Smarter, Does the Student Learn More?
5. How will Improvements in Man-Machine Interface affect Learning?

In addition to the five sections, the book includes a chapter on "Technology Forecast: Participant Profile" and two short papers on Federal R & D programs. In all about thirty five papers are included in the book. There are ten papers discussing LSI technology. Eight papers discuss developments in communications, five papers discuss uses of artificial intelligence in CAI and another ten papers discuss its implications for CAI systems.

It is not possible to discuss each paper in this short review but the papers presented reflect the various faces of CAI. For example, Donald Bitzer (of PLATO fame) discusses his dream of a million terminal CAI system of 1985 using 250 processors tied together. According to Bitzer the project will cost about 500 million dollars and will most likely be funded by the industry. On a different level of resource requirements, Seymour Papert talks about changing the learning environment for children by using systems like LOGO in which children learn geometry by moving a computer controlled Turtle. Another technology which requires even less resources is the video disc technology and Arthur Leuhrman discusses its potential. Development of instructional material for video disc system also requires large investments.

Possibilities of using speech recognition systems as part of CAI systems are explored by N. Rex Dixon, Thomas B. Martin and Lee D. Erman.

Paul Baran discusses education at home using two way communication via cable TV and wonders "why is nothing happening?".

After reading the purpose of the conference, I expected these proceedings to answer questions like "How is CAI going to improve education?" and "How are the new techniques going to effect education in the next ten years?". Unfortunately not many authors face these issues. Among those who do, F.P. Brooks makes the following points:

1. It is very hard, but not impossible, to show that any change in technique gives better learning (higher performance level).
2. It is almost impossible to use CAI or other educational technology to lower the cost of instruction, unless a substantial cost attaches to student time.
3. It is not hard to harness CAI to give demonstrably faster learning (that is, attainment of given performance level in less student time).

Looking to the future, Seymour Papert predicts that fundamental changes in education will certainly not come within the next five years but will certainly come within the next twenty years.

G.K. Gupta
(Monash University)

On Determining the Prime Implicants of a Boolean Function Without Recourse to its Minterm Form

By I. A. Parkin*

A justification is given of a modification to a method for identifying the prime implicants of a Boolean expression without recourse to its minterm form. The method avoids certain redundancies of earlier methods and, in consequence, offers improved efficiencies in time and space utilisation.

KEYWORDS: boolean functions, boolean expressions, prime implicants, minimisation.
CR CATEGORIES: 6.1, 8.3.

1.1 Notation

The following notation is used throughout this paper.

The letters A, B, C, ... will be used to name clauses, each clause being a conjunction of Boolean literals.

The letters X, Y, Z, ... will name unspecified entries in specific literal positions within clauses. Where it is required to instance such an entry, in the X position for example, the symbols x, X and — (“don’t care”) will be used; x and X represent a particular instance and its negation without specifying whether these are 0, 1 or 1, 0.

The letters T, U, V, ..., represent conjunctions of literal entries within clauses where it is not required to nominate specific entries. Where it is required to indicate entry-positional correspondences between clauses, period (.) symbols are used to mark positional boundaries and are to be regarded as matched between clauses as the context requires; as in:

\[ U \cdot x \cdot y \cdot - \]
\[ V \cdot X \cdot y \cdot z \]

\( \phi \) represents the null clause.

1.2 Background

In certain small, interactive computing systems, for which naive users may nominate conditionally selected computational or command sequences, it is desirable to incorporate validation processes over Boolean expressions to enable such expressions in user input to be checked for self consistency and for system compatibility.

The design and implementation (in software) of such processes are simplified if carried out in terms of canonical forms of the expressions involved and constraints of memory size dictate that these forms should be compact. These considerations suggest the use of the prime implicant form as a standard for both storage and processing of expressions.

Within a user — interactive system it is to be expected that individual Boolean expressions which are input will frequently make reference to only a subset of the variables within the domain of discourse of the system; in effect the typical expression will have “don’t-care” entries in respect of many variables.

Given constraints on available memory size, it seems inappropriate therefore to use a method of prime implicant determination, such as those of Morreale (1970), Hwa (1974), which depend on initial expansion of a given expression to its minterm form.

Rather, it is desirable to use a method which, like that of Tison (1967), will identify prime implicants without recourse to the minterm form. In investigating such a process, a means has been found of avoiding the generation of certain redundant clauses which typically arise in such methods, as exemplified by Morreale (1967).

1.3 Introduction

We assume a Boolean expression of interest to be given in disjunctive normal form, that is, as the disjunction of the members of a set G of clauses where each clause is a conjunction of N literal entries; each literal entry being either a positive, negative or “don’t care” reference to one member of a set of N independent Boolean variables such that there is at most one reference to a given variable within any clause. We assume that the variables over which the given Boolean expression is defined are ordered (arbitrarily) so that when the entries in each clause are correspondingly ordered, the names of the variables can be abstracted to leave, in each literal position, one of the symbols:

0 : indicating a positive occurrence of the variable;
1 : indicating a negative occurrence of the variable;
— : “don’t care”, indicating that the clause is independent of the variable.

A minterm is a clause which contains no “—” symbols. The set of minterms of a clause A is the set of clauses obtained when the “—” symbols in A are each instanced by “0” or “1” in all possible combinations. The set of minterms of a given set G of clauses is the union of the sets of minterms of the members of G. Any clause A (which may or may not be a member of G) will be said to be defined on the minterms of G if every minterm of A is contained in the set of minterms of G.

It may be taken as definition of the set P of prime implicants of a given set G of clauses that:

— all clauses of P are defined on the minterms of G;
— any clause which is defined on the minterms of

---


Copyright © 1978, Australian Computer Society Inc.

General permission to republish, but not for profit, all or part of this material is granted; provided that ACJ’s copyright notice is given and that reference is made to the publication, its date of issue, and to the fact that reprinting privileges were granted by permission of the Australian Computer Society."
G is defined on the minterms of (at least) one member clause of \( P \) taken alone;

- no clause of \( P \) is defined solely on the minterms of one other clause of \( P \).

2. STANDARD OPERATIONS ON CLAUSES

The following definitions and results arise in a straightforward manner.

2.1 Standard Definitions

We define the \( \text{intersection} \ A \land B \) of two clauses \( A \) and \( B \) to be the clause (possibly null) each entry of which is obtained by applying Table 1 to the corresponding entries in clauses \( A \) and \( B \).

**TABLE 1: Intersection of Entries**

<table>
<thead>
<tr>
<th>( \land )</th>
<th>( x )</th>
<th>( x )</th>
<th>( - )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( x )</td>
<td>( \phi )</td>
<td>( x )</td>
</tr>
<tr>
<td>( \bar{x} )</td>
<td>( \phi )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( - )</td>
<td>( x )</td>
<td>( x )</td>
<td>( - )</td>
</tr>
</tbody>
</table>

The occurrence of the symbol \( \phi \) in any entry position of the result indicates that the given clauses have an empty intersection, written thus:

\[ A \land B = \phi \]

If for two clauses \( A \) and \( B \),

\[ (A \land B) = B \]

we say \( A \) covers \( B \) and write: \( A \gg B \).

For \( S \) a set of clauses we say \( B \) is covered within \( S \), written \( B \_e_S \), if there exists a clause \( A \) such that:

\[ A \gg B \] and \( A \in S \).

2.2 \( * \)-operation: Generation of new clauses

We define an operation \( * \)- for the generation of a new clause \( A*B \) from given clauses \( A \) and \( B \), as follows: let

\[ A = U \cdot x \]
\[ B = V \cdot x \]
where \( U \land V \neq \phi \), then

\[ A*B = (U \land V).- \]

otherwise

\[ A*B = \phi \]

The \( * \)-operation is equivalent to the consensus operation of order 2 of Tison (1967).

2.3 Observations

2.3.1 For any clauses \( A \), \( B \) and \( C \)

\[ A*B = B*A \]

In general,

\[ (A*B)*C \neq A*(B*C) \]

2.3.2 For any clauses \( A \) and \( B \)

\[ A*A = \phi \]
\[ A*(A*B) = \phi \]

2.3.3 If \( A \gg B \) then for any \( C \)

- either \( A \gg (B*C) \)
- or \( (A*C) \gg (B*C) \)

for if \( B = U \cdot x \),

so that \( A = T \cdot x \)

where \( T \gg U \) and \( X \gg x \),

and if \( C = V \cdot x \) such that \( V \lor U \neq \phi \)

then \( B*C = (U \lor V).- \)

for \( X=, A = T. \gg (B*C) \);

for \( X=x, A*C = (T \lor V).- (\neq \phi) \gg (B*C) \).

2.3.4 It follow that if

\[ A \gg B \]
\[ C \gg D \]

then one of the following holds:

- \( A \gg (B*D) \)
- \( C \gg (B*D) \)
- \( (A*C) \gg (B*D) \)

For from 2.3.3, either

\[ A \gg (B*D) \text{ or } (A*D) \gg (B*D) \]

and, also from 2.3.3, either

\[ C \gg (A*D) \text{ or } (A*C) \gg (A*D) \]

2.3.5 Thus, in particular, for a set \( K \) such that

\[ A, B \in K \Rightarrow (A*B) \in K \],

it follows that

\[ A, B \in K \Rightarrow (A*B) \in K \].

2.4 \( * \)-Closure of \( G \)

Let \( G \) be a given set of clauses; then the closure \( K \) of \( G \) under the \( * \)- operation is defined by:

\[ G \in K, \]
\[ A, B \in K \Rightarrow (A*B) \in K \]

We now show that for any clause \( A \) which is defined upon the minterms of \( G \),

\[ A \in K \]

**Proof:** The proof is by induction on the number \( n \) of \( " \) symbols in the clause \( A \).

If \( n=0 \) so that \( A \) is a minterm:

\[ A \in G, G \in K \Rightarrow A \in K \]

We now assert the truth of the proposition for clauses of \( n-1 \) \( " \) symbols and prove its truth for clauses of \( n \) \( " \) symbols; we postulate the existence of a clause \( A \) of \( n \) \( " \) symbols which is defined upon the minterms of \( G \) and such that \( A \notin K \), and seek a contradiction.

Let \( A = U \cdot x \) where \( U \) contains \( n-1 \) \( " \) symbols and consider the clauses \( B = U \cdot x, C = U \cdot x \). By inspection, the minterms of \( B \) and \( C \) are minterms of \( A \) and therefore lie in the set of minterms of \( G \). Thus, by hypothesis

\[ B, C \in K \]

But

\[ B*C = A \]

So, by result 2.3.5, \( A \in K \), a contradiction.

The preceding result is well-known (e.g., Tison (1967)). It indicates that the prime implicants of a given set \( G \) of clauses, which are certainly defined on the minterms of \( G \), can be found as a subset of the \( * \)- closure of \( G \). The prime implicant subset is, in fact, found by eliminating from the \( * \)- closure any clause which is covered by another member of the closure. The result 2.3.5 shows that the two processes, of removing covered clauses and of generating

*The Australian Computer Journal, Vol. 10, No. 4, November, 1978*
Prime Implicants further members of the closure by \(*\)-operations on existing members, can be freely commuted.

The \(*\)-operation will, in general, redundantly generate multiple copies of members of the \(*\)-closure in a manner similar to that exemplified by Morreale (1967).

In the following section a modified operator \(*'\) is defined which eliminates this redundancy.

3. NEW OPERATIONS

3.1 Definitions

A significant redundancy in the generation of new clauses is avoided as follows.

We represent by the symbol \("/\) the entry, previously written which is created in the clause resulting from the \(*\)-operation. Thus:

\[ A = U \cdot x, B = V \cdot x \]

yields

\[ A * B = (U A V) / \text{ for } U A V \neq \phi. \]

We first revise the intersection table (Table 1) to include the new symbol, thus:

<table>
<thead>
<tr>
<th>( \Lambda )</th>
<th>( x )</th>
<th>( \overline{x} )</th>
<th>( / )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( x )</td>
<td>( \phi )</td>
<td>( x )</td>
</tr>
<tr>
<td>( \overline{x} )</td>
<td>( \phi )</td>
<td>( x )</td>
<td>( \overline{x} )</td>
</tr>
<tr>
<td>( / )</td>
<td>( x )</td>
<td>( \overline{x} )</td>
<td>( / )</td>
</tr>
</tbody>
</table>

The validity of the preceding discussion is unimpaired and the new table is equivalent to the old if the distinction between the symbols \("/\) and \("/\) is ignored.

We then define a new intersection operation \( \Lambda' \) by repeating Table 2 with the sole modification that entries involving opposition of the symbol \("/\) and either \("x\) or \("x\) are made null; thus

<table>
<thead>
<tr>
<th>( \Lambda' )</th>
<th>( x )</th>
<th>( \overline{x} )</th>
<th>( / )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( x )</td>
<td>( \phi )</td>
<td>( x )</td>
</tr>
<tr>
<td>( \overline{x} )</td>
<td>( \phi )</td>
<td>( x )</td>
<td>( \overline{x} )</td>
</tr>
<tr>
<td>( / )</td>
<td>( x )</td>
<td>( \overline{x} )</td>
<td>( / )</td>
</tr>
</tbody>
</table>

We define a new \("/\) operation in terms of the above; thus for \( A = U \cdot x, B = V \cdot \overline{x} \):

\[ A * B = (U A V) / \text{ for } (U A V) \neq \phi; \]

\[ = \phi \text{ otherwise.} \]

3.2 \("/\)-Closure

Let \( G \) be a given set of clauses and let \( K' \) be the \("/\) closure of \( G \). We show that for any clause \( A \in K' \) and any other clause \( B; A * B \in K' \).

\[ \text{Proof: We consider } K' \text{ to be partitioned into subsets } K'_n, 0 \leq n < N, \text{ such that} \]

\[ K'_n = \{ A_n | A_n \in K' \text{ and } A_n \text{ contains n } "/\text{ symbols} \}. \]

In particular, \( G = K'_0 \) and

\[ K' = \bigcup_{i=0}^{N} K'_i \]

We prove by induction on \( n \), for any clause \( A_n \in K'_n \) and any other clause \( B \) that

\[ A_n * B \leq \min(n+1, N) K'_i \]

For \( n=0 \) the result follows immediately from the structure of Table 2.

We assert the truth of the result for clauses \( A_{n-1} \) of \( n-1 \) "f" symbols and prove the result for \( n \).

Let

\[
A_n = U_n \cdot [f]^n \cdot x
\]

where \( U_n \) contains no "f" symbols and \([f]^n\) represents a string of \( n \) "f" symbols.

Correspondingly, let

\[
B = V \cdot [Y_i]^n \cdot x
\]

in which representation we distinguish especially the entries \( Y_i, 1 \leq i \leq n \) which oppose the "f" symbols in \( A_n \).

Then, using Table 2,

\[
A_n \ast B = (U_n \land V) \cdot [Y_i]^n \cdot f
\]

Now if every entry \( Y_i, 1 \leq i \leq n \) is either a "-" or "f" symbol, application of Table 2 is identical in effect to application of Table 2, and

\[
A_n \ast B \in K_{n+1}
\]

Otherwise we consider the following cases.

Case \( i, 1 \leq i \leq n, Y_i = y_i \)

Since \( A_n \in K_n \) there are clauses

\[
A_{n-1}, A'_{n-1} \in K_{n-1}
\]

such that

\[
A_{n-1} \ast A'_{n-1} = A_n.
\]

The definition of the \( \ast \) operation indicates that these clauses can, without loss of generality, be written in the forms:

\[
\begin{align*}
A_{n-1} &= U_{n-1} \cdot [\land f]^n \cdot y_n \cdot X \\
A'_{n-1} &= U'_{n-1} \cdot [f \lor \land]^n \cdot y_n \cdot X'
\end{align*}
\]

where

- \( U_{n-1} \land U'_{n-1} = U \)
- \([\land f]^n\) and \([f \lor \land]^n\) are strings of "-" and "f" symbols such that in each of the entry positions which they span at least one of the strings contains an "f" symbol
- \( X \land X' = x \)

Sub-case \( Y_n = y_n \).

In this case

\[
A_n \ast B = (U_n \land V) \cdot [Y_i]_{i=1}^{n-1} \cdot y_n \cdot f
\]

For \( X = - \):

\[
A_{n-1} = U_{n-1} \cdot [\land f]_1^{n-1} \cdot y_n
\]

so that

\[
A_{n-1} \gg A_n \ast B
\]

thus

\[
A_n \ast B \in K'_{n-1}
\]

For \( X = x \):

\[
A_{n-1} \ast B = (U_{n-1} \land V) \cdot [Y_i]_{i=1}^{n-1} \cdot y_n \cdot f
\]

but, by the induction hypothesis,

\[
A_{n-1} \ast B \in \bigcup_{i=0}^{\min(n+N)} K_i
\]

so that

\[
A_n \ast B \in \bigcup_{i=0}^{\min(n+1,N)} K_i
\]

Sub-case \( Y_n \neq y_n \).

In this case

\[
A_{n-1} \ast B = (U_{n-1} \land V) \cdot [Y_i]_{i=1}^{n-1} \cdot y_n \cdot f
\]

For \( X = x' = x \):

\[
A_{n-1} \ast B = (U_{n-1} \land V) \cdot [Y_i]_{i=1}^{n-1} \cdot y_n \cdot f
\]

and

\[
(A_{n-1} \ast B)(A'_{n-1} \ast B) = (U_n \land V) \cdot [Y_i]_{i=1}^{n-1} \cdot f
\]

\[
= A_n \ast B
\]

but by the induction hypothesis there are clauses \( A_p \) and \( A_q \) such that:

\[
A_p \in K_p, 0 \leq p \leq n \quad A_p \gg (A_{n-1} \ast B),
\]

\[
A_q \in K_q, 0 \leq q \leq n \quad A_q \gg (A'_{n-1} \ast B);
\]

now, by 2.3.4, one of the clauses \( A_p, A_q \) or \((A_p \ast A_q)\) covers the result \((A_{n-1} \ast B)(A'_{n-1} \ast B)\);

but \( A_p \) and \( A_q \) are each members of some \( K_i, 0 \leq i \leq n \) and, by re-use of the induction hypothesis, \((A_p \ast A_q)\) is covered by some \( K'_{i+1}, 0 \leq i \leq n+1 \), thus

\[
A_n \ast B \in \bigcup_{i=0}^{\min(n+1,N)} K_i
\]

Sub-case \( Y_n = y_n \).

In this case

\[
A_{n-1} \ast (A'_{n-1} \ast B) = A_n \ast B
\]

and an argument similar to that for the preceding case establishes the desired result.

*The Australian Computer Journal, Vol. 10, No. 4, November, 1978*
Letters to the Editor

The Editor,
Australian Computer Journal,

Dear Sir,

The lecture by Barry Dwyer at ACS8 prompted me to read again the article on Structured Programming by Peter Juliff (Journal, May) and the following correspondence. I refer also to Barry’s article in the May '77 issue of the Journal.

It is interesting and perhaps even slightly amusing to see people go to such lengths to try to improve COBOL, whose code is cumbersome, verbose, and despite its proponents claims, not always readily intelligible. It is a reflection on the inadequacy of COBOL that so much work is done in endeavours to make it easier to write, maintain, and understand.

Your readers may be interested in that a decision table based language known as FILETAB generated enormous interest at ACS8. It is applicable to computing problems in general, and some large organisations use it almost exclusively for all programming. It is available on many computers, from micros through to mainframes, such as IBM (360, 370, S3, S32), ICL (1900, 2903, 2900), Digital (LSI 11 to PDP 11/70), UNIVAC (1100, 90/30), and others.

The main feature of FILETAB is its ability to interpret decision tables as its normal code, and “compile” direct to machine code at 10 to 20 times the speed of COBOL. Secondly, it has a very powerful report generator built into it, and the combination of these features makes a dramatic impact on the amount of effort required to write programs. Typically a FILETAB program would require about a quarter of the effort of the equivalent COBOL, and in many cases less.

By way of illustration, below is the equivalent FILETAB code for some of the coding examples from Peter Juliff’s article.

Example 1 (Figure 4) — “Table Search”
* DETAB VALIDATE
C LOOKUP ITEMS CUSTOMER Y N
Note that it is not necessary to set a flag, as the code automatically returns a T or F reply to the “IF VALIDATE” question.

Example 2 (Figure 5) — “perform depending on”
* DETAB SELECT
C Code = ? 1 3 6 ELSE
A CALL ROUTINE? X Y Z E
The number of valid codes to be tested does not really matter, as the C line (condition test) is merely expanded to accommodate more. Thus the simplicity and elegance of FILETAB compared with COBOL is further emphasised.

Example 3 (Figure 6) — “Multiple rules”
* DETAB SELECT
C if COND-1 COND-2 COND-3 ELSE
A SELECTION MV 1 x x x x
SELECTION + 4 x - - -
SELECTION + 2 - x - -
SELECTION + 1 x - x -
GOTO TEST x x x x

Yours faithfully,
A. Borbiro,
M.A.C.S.

The Editor,
The Australian Computer Journal.

Dear Sir,

The article on “Computing as a Human Activity” (Sale, 1976) serves as a starting point for some comments I should like to offer on the APL language. Recognizing that computing is indeed a human activity, it seems clear to me that the cost in human time is every bit as important in evaluating computing procedures as is CPU time. My remarks will therefore pertain to the type of program that is typical in much of engineering, a program whose cost in manpower expended for its development exceeds the cost of computer time over the life of the program.

During a sabbatical leave at Princeton University, I was able to use their extensive facilities for interactive APL. Based on my experience, I would agree that APL has many of the advantages put forth by its proponents (Bent and Pessis, 1976; McKenna, 1976; Wittig and Eggensperger, 1976). The close correspondence between the mathematical statement of the problem and the APL program results in programs which have a minimum of peripheral detail (format, dimensions, element-by-element array processing, etc.). The availability of special operators for matrix and vector operations, and the ability to use functions and operators in an interactive manner, serve to reduce program development time in comparison with many commonly-used programming languages.

The workspace feature of APL assists one in carrying out check calculations for the critical portions of a numerical calculation. Additional output can be obtained after completion of a run; the need for such output might not have been foreseen in advance. The availability of workspace variables also introduces the necessity of an extra precaution. When programs are being checked out for the last time, they should be run using a clear workspace. Otherwise, variables having extraneous data might be used without producing an error message in a function that contains an undetected error in the sequence of statements. Despite the advantages of APL, it should be borne in mind that computational time is NOT in its favor (Alleman and Richardson, 1974). Also, the greater availability of mathematical software for FORTRAN will very likely inhibit the use of APL over the next five or so years.

As for documentation of APL software, it is urged that all procedures beyond five or 10 lines should be accompanied by clearly stated algorithms and references to the mathematical development. Although this seems obvious, it is common to find APL code presented by itself with no word description of the procedure. Because of the
fuzziness of most printed code, APL programs can seldom be read with anything approaching full confidence. However, due to the conciseness of APL, programs can often be exchanged in card form without excessive cost or inconvenience.

As for the need for an open mind on language choice, I would refer again to Professor Sale's article. Computer languages should serve not as a religion to be defended or propagated, but as alternative means for converting organized thought into organized computation.

Yours sincerely,
George S. Campbell,
Professor of Aerospace Engineering


**REPLY BY PROF. A.H.J. SALE**

I thank Professor Campbell for taking my article as a provocative starting point for his comments on APL as a programming language, for he gives me the opportunity to make a point. APL is recognizable one of the programming languages that have made it to the top and are in very wide use. My own checklist is quite short, and includes several very flawed languages in it: FORTRAN, Basic, COBOL, APL, PL/I, Algol 60, and Pascal of course. How did these languages get to where they are, and why? What influence does the conceptual structure of the language have (in PL/I for example)! And in particular APL?

I argue that APL’s success is due to three factors. The first two are direct human factors:

(a) The APL interactive system is so highly interactive that it must evoke sympathetic responses from its users, whatever language were embedded in it, and

(b) the arcane nature of APL notation and algorithms encourages something of the atmosphere of a secret society, or of a closed club.

The third factor is IBM support for APL, but that seems to follow the other factors, not precede them...

Interestingly, if I am right, we ought to preserve APL as a cultural heritage as enabling people to indulge themselves in an orgy of cunning tricks without harmful effects. I remember having lots of fun as an aspiring programmer writing very obscure code for very obscure reasons; now the nastiest corners of my mind only get exercised in devising validation programs for compilers. And yet I do not personally regret the insights those tricks gave me: yes, APL has a useful role.

But moving back to the mundane, sure, APL is a useful tool for people whose problems fit its facilities. Which mostly means those problems which fit within an array of numbers mould. It matches many scientific requirements well. It is, however, the exaggerated claims one must beware of — and Professor Campbell carefully made none.

After all, if you read the fine print at the back of *Pascal News*, you will find that even our latest wonder-toy has limits, expressed in the following terms:

"Pascal is a small, practical, and general purpose (but not all-purpose) programming language possessing algorithmic and data structures to aid systematic programming . . ."

The Editor,
Australian Computer Journal

Dear Sir,

I feel I must write in response to Professor Sale's article ("Stylistics in Languages With Compound Statements, May 1978") to congratulate him on daring to suggest that the begin-end pair be placed anywhere other than indented on a line of its own!

I have, quite independently, developed a similar approach for virtually the same reasons.

I found myself perturbed by what appeared to be inconsistencies, such as the following:

(i) consider the loop control constructs, while-do and repeat-until, of Pascal. In the latter, we may insert the statement list without the begin-end, because these are effectively absorbed into the convenient keywords of the structure.

The repeat introduces the structure to the compiler, which then seeks the until to finish it; anything intervening must be the enclosed statement list.

However, with the while-do, there is no such convenient keyword terminator for the end of the statement list. Whilst some languages (notably ALGOL 68) provide an od, Pascal imposes an apparently artificial begin-end bracketing.

(ii) ALGOL 60, unfortunately, made no great distinction between a compound statement and the block, and hence its implication on the execution environment has perhaps remained somewhat shrouded, particularly to the first-time user. As already indicated above, more recent language designs have attempted to overcome this problem by introducing such things as fi, esac, od and so on. In these languages, begin-end may be kept for blocks solely.

One reason for developing ALGOL-like languages has been to provide a meaningful and readable programming language. It would seem unreasonable, then, to allow limitations of the syntax to dictate the formatting technique to be used. If a language does not provide fi and esac, then we could at least attempt to simulate them, as suggested by the style under consideration.

Indeed, I would propose that Professor Sale has not gone far enough with his suggestions.

The best model we have is our own language; we should use this as a base, rather than the programming language syntax, from which to develop a formatting style.

As a demonstration of my inclinations on the matter, I offer my own version of Professor Sale's example. The reader is reminded of Professor Sale's comment that it may look difficult to follow, but it really depends upon what one is familiar with:

My reason for settling on this technique is, most importantly, because every statement starting at a particular nesting level has its first keyword/token at the same level or "tabstop". There is certainly not the rapid movement across the page of "regular" techniques.

Finally, I reiterate Professor Sale's request that all examine their technique of program layout. Perhaps a comparison of the two paragraphing techniques most commonly found in print (indent the first line, or, more significantly, indent all continuation lines after the first) would prove profitable.

Yours sincerely

Barry Johnson

Forthcoming National and International Events

<table>
<thead>
<tr>
<th>DATE</th>
<th>EVENT</th>
<th>LOCALE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1978</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>December</td>
<td>Winter Simulation Conference</td>
<td>Miami Beach, Florida USA</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>National Telecommunications Conference</td>
<td>Birmingham, Alabama, USA</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>ACM 1978</td>
<td>Washington, D.C. USA</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>CMG IX International Conference on “Management in Evaluation of Computer Performance”</td>
<td>San Francisco, California, USA</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>Working Conference on “Performance Evaluation of Numerical Software”</td>
<td>Baden, Austria</td>
<td>IFIP</td>
</tr>
<tr>
<td></td>
<td>International Computer Symposium</td>
<td>Taipei, Taiwan, Rep. of China</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>First Iranian Computer Congress</td>
<td>Teheran, Iran</td>
<td>IFIP</td>
</tr>
<tr>
<td>1979</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>January</td>
<td>Twelfth Hawaii International Conference on System Sciences</td>
<td>Honolulu, Hawaii</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>Pacific Telecommunications Conference</td>
<td>Honolulu, Hawaii</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>SOTAC 79. IFIP/IFAC Working Conference on “Socio-Technical Aspects of Computerization”</td>
<td>Budapest, Hungary</td>
<td>IFIP</td>
</tr>
<tr>
<td></td>
<td>Sixth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages</td>
<td>San Antonio, Texas</td>
<td>ACM</td>
</tr>
<tr>
<td>February</td>
<td>Fourth International Symposium on Computer System Modelling and Performance Evaluation</td>
<td>Vienna-Laxenburg, Austria</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>International Symposium on Flow Control in Computer Networks</td>
<td>Paris, France</td>
<td>IFIP</td>
</tr>
<tr>
<td></td>
<td>1st European Conference on Parallel and Distributed Processing</td>
<td>Toulouse, France</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>ACM Computer Science Conference</td>
<td>Dayton, Ohio, USA</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>A Technical Symposium on Computer Science Education</td>
<td>Dayton, Ohio, USA</td>
<td>ACM</td>
</tr>
<tr>
<td>March</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IAG Seminar “The Immediate Future of DBMS”</td>
<td>Munich, FRG</td>
<td>IFIP</td>
</tr>
<tr>
<td></td>
<td>Twelfth Annual Simulation Symposium</td>
<td>Tampa, Florida, USA</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>4th GI Conference on Theoretical Computer Science</td>
<td>Aachen, Germany</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>Second International Conference on Computational Methods in Non-Linear Mechanics</td>
<td>Austin, Texas</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>Conference on Numerical Ordinary Differential Equations</td>
<td>Champaign, Illinois, USA</td>
<td>ACM</td>
</tr>
<tr>
<td></td>
<td>Computers in Ophthalmology</td>
<td>St. Louis, Missouri, USA</td>
<td>IFIP</td>
</tr>
<tr>
<td></td>
<td>APOLOLO AGONISTES: The Humanities in a Computerized World</td>
<td>Albany, N.Y.</td>
<td>ACM</td>
</tr>
</tbody>
</table>
**Forthcoming Events**

<table>
<thead>
<tr>
<th>Month</th>
<th>Event Description</th>
<th>Location</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>May</td>
<td>Working Conference on &quot;Evaluation of Effectiveness of Medical Action, Methodology and Experiences&quot;</td>
<td>Bordeaux, France</td>
<td>IFIP</td>
</tr>
<tr>
<td>14 – 19</td>
<td>4th PROCOMAT Conference “PROCOMAT 79”</td>
<td>Michigan, USA</td>
<td>IFIP</td>
</tr>
<tr>
<td>June</td>
<td>National Computer Conference</td>
<td>New York City, USA</td>
<td>ACM</td>
</tr>
<tr>
<td>4 – 7</td>
<td>2nd IFIP Conference on “Human Choice and Computers”</td>
<td>Vienna, Austria</td>
<td>IFIP</td>
</tr>
<tr>
<td>4 – 16</td>
<td>IFIP/IFAC Symposium on “Software for Computer Control – SOCOCO”</td>
<td>Prague, CSSR</td>
<td>IFIP</td>
</tr>
<tr>
<td>11 – 13</td>
<td>Working Conference on “The Information Systems Environment”</td>
<td>Bonn, FRG</td>
<td>IFIP</td>
</tr>
<tr>
<td>11 – 13</td>
<td>International Conference “Tele-informatics 79”</td>
<td>Paris, France</td>
<td>IFIP</td>
</tr>
<tr>
<td>19 – 21</td>
<td>IFIP/IFAC Workshop “Real-Time Programming”</td>
<td>Prague, CSSR</td>
<td>IFIP</td>
</tr>
<tr>
<td>20 – 22</td>
<td>Ninth Annual International Symposium on Fault Tolerant Computing</td>
<td>Madison, Wisconsin, USA</td>
<td>ACM</td>
</tr>
<tr>
<td>July</td>
<td>VIII IFAC Symposium on Automatic Control in Space</td>
<td>Oxford, England</td>
<td>ACM</td>
</tr>
<tr>
<td>2 – 6</td>
<td>1979 Summer Computer Simulation Conference</td>
<td>Toronto, Ontario, Canada</td>
<td>ACM</td>
</tr>
<tr>
<td>16 – 20</td>
<td>Sixth International Colloquium on Automata, Languages, and Programming</td>
<td>Graz, Austria</td>
<td>ACM</td>
</tr>
<tr>
<td>August</td>
<td>Sixth International Joint Conference on Artificial Intelligence</td>
<td>Tokyo, Japan</td>
<td>ACM</td>
</tr>
<tr>
<td>20 – 24</td>
<td>9th Conference on “Optimization”</td>
<td>Warsaw, Poland</td>
<td>ACM</td>
</tr>
<tr>
<td>4 – 10</td>
<td>5th IFAC Symposium on Identification and System Parameter Estimation</td>
<td>Darmstadt, FRG</td>
<td>ACM</td>
</tr>
<tr>
<td>24 – 28</td>
<td>EURO-IFIP 79. IFIP European Conference on “Applied Information Technology”</td>
<td>London, UK</td>
<td>IFIP</td>
</tr>
<tr>
<td>1980</td>
<td>MEDINFO 80</td>
<td>Tokyo, Japan</td>
<td>IFIP</td>
</tr>
<tr>
<td>Sept. 29 – Oct. 3</td>
<td>8th World Computer Congress</td>
<td>Tokyo, Japan</td>
<td>IFIP</td>
</tr>
<tr>
<td>October</td>
<td>IFIP Congress 80</td>
<td>Melbourne, Australia</td>
<td>IFIP</td>
</tr>
</tbody>
</table>

---

**Book Review**


This book is Volume 49 of the "Lecture Notes in Computer Science" series produced by Professor Goos and reports the Proceedings of the 6th Informatik Symposium of IBM, Germany, held in September 1976.

The Symposium was divided into four sessions: Systems Design Considerations; Managing Interactive Systems; Application Development; Applications of Interactive Systems.

The session on Systems Design contained three papers. A.L. Scherr discussed design parameters, implementation techniques and future trends, while the role of interactive programming languages was discussed by G. Seegmueller, who provided a general classification and critique of current languages. Finally, H.J. Hoffmann investigated the design process of interactive systems.

Three papers in the second session on management cover the logical evolution of interactive systems within IBM (Kalisky), and performance evaluation of interactive systems (Bard). Finally, M. Achenbach described some of the practical considerations obtained in an industrial computer centre.

The third session on applications, described the current emphasis on transaction-oriented and educational systems. A number of the authors suggest that advances in technology will allow a shift into computer-aided design and training systems.

The final session presents five papers on some specific application systems.

The set of papers is by no means comprehensive, but does provide an historical sketch of the growth of interactive systems and describes a number of unique attributes of these systems. Such material dates very quickly, but is a worthwhile reference for anyone actively working in the field, or presenting courses on the subject.

V.X. Gledhill,
New South Wales Institute of Technology
P.G. Computer Products locally manufacture
- Back to Back modems, line drivers which allow terminals to be connected up to 300 metres away from the controller with only one “box” and low-cost cable.
- Modem sharing units (Telecom Approved).
- Clock buffer units for multi-stream services.

P.G. Computer Products makes computer cable to any specification. Let us quote for any one-off equipment or cables to suit your requirement.

All Data communication needs can be met from our range of locally manufactured equipment. Call now for a free brochure on our PG1002 Back to Back Modem or the PG1008 Synchronous Line Extender.

Service is no problem as we develop and manufacture locally.

P.G. Computer Products
Box 202, Toorak, Vic, 3142
Telephone (03) 63 3968

DECwriter Printer Terminal LA36 & LS120

ADE has been appointed by Digital Equipment Australia Pty. Ltd., as a reseller of DECwriters

Features:

* True 30 CPS throughput for LA36
* 180 CPS for LS120 true 1200 baud
* Low price, high reliability
* Office standard 128 character ASCII keyboard
* Full 132 character line length
* 90-day Warranty by ADE
* Manufactured by DEC

ANDERSON DIGITAL EQUIPMENT Pty Ltd
The Australian Computer Journal is an official publication of the Australian Computer Society Incorporated. Subscriptions and orders should be sent to PO Box 640, Crows Nest, 2065.

Office-Bearers
President: A.R. BENSON
Vice-President: A.H.J. SALE
Immediate Past President: R.S. NORTHCOTE
National Treasurer: V.G. WHITELEY
National Secretary: G.K. PIRIE
Chief Executive Officer: R.W. RUTLEDGE
P.O. Box 640, CROWS NEXT, N.S.W. 2065
Telephone: 929 2219

Editorial Committee:
Editor: A.Y. MONTGOMERY
Department of Computer Science, Monash University, Clayton, Victoria, 3168
Associate Editors: J.M. BENNETT, G.N. LANCE, T. PEARCEY, P.C. POOLE

Published by:

Represented by:
Victoria and South Australia: Derrick, Lewis & Associates, Ashley House, 409 St Kilda Road, Melbourne 3004. Phone 267-4022.
Italy: Publicitas, Via E. Filiberto 4, Milan.
France: Gustav Elm, 41 Avenue Montaigne, Paris 8.

Advertising inquiries should be referred to one of the above addresses.

Printed by:
Publicity Press Ltd., 29-31 Meagher Street, Chippendale, N.S.W. 2008.

SUBSCRIPTIONS: The annual subscription is $10.00. All subscriptions to the Journal are payable in advance and should be sent (in Australian currency) to the Australian Computer Society Inc., P.O. Box 640, Crows Nest, N.S.W. 2065.

PRICE TO NON-MEMBERS: There are now 4 issues per annum. The price of individual copies of back issues still available is $1.50. Some are already out of print. Issues for the current year are available at $3.50 per copy. All of these may be obtained from the National Secretariat, P.O. Box 640, Crows Nest, N.S.W. 2065. No trade discounts are given, and agents should recover their own handling charges. Special rates apply to members of other Computer Societies. Printed by: Publicity Press Ltd., 29-31 Meagher Street, Chippendale, N.S.W. 2008.

MEMBERS: The current issue of the Journal is supplied to personal members and to Corresponding Institutions. A member joining part-way through a calendar year is entitled to receive one copy of each issue of the Journal published earlier in that calendar year. Back numbers are supplied to members while supplies last, for a charge of $1.50 per copy. To ensure receipt of all issues, members should advise the Branch Honorary Secretary concerned, or the National Secretariat, promptly of any change of address.

REPRINTS: 50 copies of reprints will be provided to authors. Additional reprints can be obtained, according to the scale of charges supplied by the publishers with proofs. Reprints of individual papers may be purchased for 50 cents each from the Printers (Publicity Press).

PAPERS: Papers should be submitted to the Editor; authors should consult the notes published in Volume 9, pp. 45-49 (or request a copy from the National Secretariat).

MEMBERSHIP: Membership of the Society is via a Branch. Branches are autonomous in local matters, and may charge different membership subscriptions. Information may be obtained from the following Branch Honorary Secretaries. Canberra: P.O. Box 446, Canberra City, A.C.T. 2601. N.S.W.: Science House, 35-43 Clarence St, Sydney, N.S.W. 2000. Qld: Box 1484, G.P.O., Brisbane, Qld. 4001. S.A.: Box 2423, G.P.O., Adelaide, S.A. 5001. W.A.: Box F320, G.P.O., Perth, W.A. 6001. Vic.: P.O. Box 98, East Melbourne, Vic. 3002. Tas.: P.O. Box 216, Sandy Bay, Tas. 7005.

Copyright © 1978. Australian Computer Society Inc.