FACS FACTS

The Newsletter of
the
BCS Formal Aspects of Computing Science SIG

Series II Vol I No I April 90.

I am writing this on 16 th March 1990 the 12 th anniversary of when FACS became an officially British Computer Society Specialist Interest Group. The first Chairman was Dan Simpson , with John Cooke as Secretary and David Blyth as Treasurer. Since then there have been a number of changes the latter half of the decade is reported in FACS at ten. One of the commitments for the future in that report was that "the newsletter will continue in its present guise".

Well I am sure most of you noticed that the newsletter has not appeared since December 1978. The short and simple response to that is Sorry!. There are many reasons for this, briefly we have been involved in a number of new initiatives. For instance, some of you will already be aware of the increased number of workshops. It is our intention to publish the proceedings of these. In addition, many of you will have also received the Formal Aspects of Computing - the international journal of formal methods.

Numerologists say good news comes in three's. The Berlin Wall has crumbled Nelson Mandela has been released and FACS FACTS is back.

To ensure that it continues to be sent to you I need your contributions. So if your interested in contributing in the Formal Aspects of Computing area why not send us papers, notices, workshop reports, research reports, F X Reid columns etc.

Jawed Siddiqi

Membership details.

Membership of FACS itself, which for BCS members is very cheap, ensures direct mailing of all our events, preferential rates for attendance at our meetings and copies of our periodic (spasmodic) Newsletter. We offer discounted subscription rates to the quarterly journal Formal Aspects of Computing published by Springer which has a target of 100 pages per issue. We also offer a way of paying for subscription to the European Association of Theoretical Computer Science (EATCS) in pounds sterling. This provides you with an issue of the EATCS Bulletin three times a year with over 400 A5 pages of technical reports and news per issue

As a rough guide the numbers involved are

Members of FACS	-	258
Subscriptions to FAC journal via FACS	-	89
Subscriptions to EATCS via FACS	-	164

The FACS membership year has traditionally been from 1 May to 30 April. However the membership period which started on 1 May 1989 will finish on 31 December 1989. From 1 January 1990 the membership year will coincide with the calendar year and also with the subscription year for the FAC journal. There are three possible membership start dates for EATCS depending on when you first pay (paid) your subscription. Their membership years begin on Feb 1, June 1 or Oct 1.

The following rates apply for 1990

1/12/90	to	31/12/90 FACS membership	£25 (discounted rate £8)
1/12/90	to	31/12/90 FAC journal	£18
1 year		EATCS (via FACS)	£6

FACS Officers

Chairman

Dr D J Cooke
John
Dept Computer Studies
Loughborough University of Technology
Loughborough
Leics LE11 3TU
Telephone: 0509 222676
Email: john@uk.ac.lut.cs-vaxa

Secretary

B T Denvir
Tim
37 Orpington Road
Winchmore Hill
London N21 3PD
Telephone: 01 882 5853(home), (0225) 444700
Email: btd@uk.co.praxis

Treasurer

Dr R G Stone
Roger
Dept Computer Studies
Loughborough University of Technology
Loughborough
Leics LE11 3TU
Telephone: 0509 222686
Email: roger@uk.ac.lut.cs-vaxa

Newsletter Editor

Dr J I A Siddiqi
Jawed
Dept. of Computer Studies
Sheffield City Polytechnic
Pond Street
Sheffield S1 1WB
Telephone: 0742 720911

Publicity

Dr B Q Monahan
Brian
Dept. of Computer Science
The University
Manchester M13 9PL
Tel: home 0223 314019 / 061 275 6137
Email: bqm@uk.ac.man.cs

Dr P N Scharbach
Peter
Information Technology Research Unit
BP Research Centre
Chertsey Road
Sunbury-On-Thames
Middlesex TW16 7LN
Telephone: (0932) 762570

Specialist Groups Management Committee Representative

Dr T H Axford
Tom
Computer Science Dept.,
University of Birmingham
PO BOX 363,
Birmingham
B15 2TT.
Telephone: 021 414 4779
Email: AxfordTH@uk.ac.bham

Dr T H Axford
Computer Science Dept.,
University of Birmingham
PO BOX 363,
Birmingham B15 2TT.

D Blyth
Incord Ltd
15, Sherwood Avenue
Ferndown
Wimborne, Dorset BH22 8JS

Dr D J Cooke
Dept Computer Studies
Loughborough University of Technology
Loughborough
Leics LE11 3TU

B T Denvir
37 Orpington Road
Winchmore Hill
London N21 3PD

Dr A J J Dick
Racal Research Ltd
Worton Drive
Reading RG2 0SB

Prof. S J Goldsack
Dept of Computing,
Imperial College
180 Queen's Gate,
London SW7 2BZ

Dr R J Mitchell
32 Lincoln Avenue
Peacehaven
E Sussex BN10 7JT

Dr B Monahan
Dept of Computer Science
The University
Manchester
M13 9PL

Dr A Norcliffe
Dept of Mathematical Sciences
Sheffield City Polytechnic
Sheffield S1 1WB

Dr P N Scharbach
Information Technology Research Unit
BP Research Centre
Chertsey Road
Sunbury-On-Thames
Middlesex TW16 7LN

R C Shaw
Praxis Systems PLC
20 Manvers Street
Bath BA1 1PX

Dr J I A Siddiqi
Dept. of Computer Studies
Sheffield City Polytechnic
Pond Street
Sheffield S1 1WB

Prof D Simpson
Dept of Computing
Brighton Polytechnic,
Moulsecoomb,
Brighton BN2 4AT

Dr R G Stone
Dept Computer Studies
Loughborough University of Technology
Loughborough
Leics LE11 3TU

FACS at ten

D J Cooke

Department of Computer Studies, Loughborough University of Technology

D Simpson

Department of Computing, Brighton Polytechnic

The second five years of the BCS-FACS group is reviewed, and the groups major activities are reported.

1. The FACS Group in the UK

The British Computer Society SIG in Formal Aspects of Computing Science (FACS) was inaugurated on the 16 March 1978. A report of the early activities of the group is given in [Sim84]. Before the group was formed there was no forum within the BCS (or elsewhere in the UK) for those working on the more theoretical aspects of the subject. However during the past five years the community has grown and a number of other groups have emerged with aims similar to those of FACS. The Institute of Mathematics and its Applications, the London Mathematical Society and the organisers of the British Theoretical Computer Science Colloquium all support groups whose interests intersect with FACS. FACS undertakes liaison with these groups to try to ensure that our activities are complementary.

Many of these groups have members in common and cordial relations are maintained. It is certainly true that UK work in the area is as healthy as it has ever been in the past ten years. It is equally important that our links with European groups have been strengthened within the period. One of the major services offered to members of FACS is the ability to easily join EATCS in conjunction with FACS membership. About 80% of FACS members take advantage of the scheme and we expect this percentage to rise even higher. Despite this interest in EATCS by our members it is still worrying to note that EATCS does not have any UK based institutional members.

The increase in the number of people and the number of groups in the area has caused FACS to change its emphasis; although the groups aims stay exactly as they were first formulated ten years ago. The group is for active workers in the area and provides a forum where computer scientists can meet to exchange ideas, discuss problems and even, now and again, propose solutions. It remains a group whose primary emphasis is on work. Technology transfer to the rest of the computing community is still undertaken but has become less of a focus within the group as other organisations, both academic and commercial, have increased their activities in this area. This is not to say that the technology transfer issue is seen as any less important and many FACS members are active in this field.

2. Changes over the past five years

During the period under review research has been stimulated by the Esprit and Alvey programmes. The FACS group was proud to accept the offer to become the Alvey SIG in Formal Methods. As the Alvey SIG we were able to arrange a number of meetings to bring together the ever widening community and to help newer, younger workers meet the 'old timers'. Help from Alvey also allowed us to improve the quality of the newsletter and it is interesting to note that FACS FACTS now appears in the list of references to some highly regarded published papers. The newsletter was always produced to allow the quick publication of draft papers and other information. We expect the newsletter to continue in this way but the new journal we are about to launch will allow publication of final, refereed papers.

The journal is called *Formal Aspects of Computing - the international journal of formal methods* and we are pleased that Cliff Jones has agreed to act as its editor in chief. Due to appear early in 1989, the journal will be produced in association with Springer-Verlag and initially each volume will consist of four issues of approximately 100 pages.

Over the period we have continued to hold a number of meetings; details of which are given in the next section. The tutorial meetings have been particularly popular and are seen as a good way of introducing members to new topics in the field. The workshops are a useful way for members to get together and actually work on a problem. Joint meetings with other groups tend to be less popular with our members but still form an important part of the technology transfer part of the groups activities. (Maybe the multiple focus of such meetings is viewed, wrongly, as necessitating the 'watering down' of the technical material.)

3. Summary of Activities

1984 started with the Vol 6(1) issue of FACS-FACTS appearing in what is now its familiar blue cover but it was not until Vol 6(2) that the first paper from F X Reid was published. It was also in this issue that we announced the start of our formal collaboration with EATCS.

3 July 1984 - University College, London
A Meeting on occam

W Roscoe (Oxford)	The Formal Semantics of occam
J Kerridge (Sheffield Polytechnic)	occam and other Concurrent Languages
A Fisher (Hull)	occam for parsing van Wijngaarden Grammars
R Shepherd (Inmos)	The many uses of occam

F X Reid was obviously aware of this meeting as his first paper was on occam. However, despite Reid achieving world fame, the current authors feel that the effect of Coombes has often been underestimated in the (Reid-Russel-Coombes) theorem. Reid's first paper at least proved that someone read the newsletter as the response to the paper was greater than the response from members to the questionnaire on their technical interests. This had two consequences, the next issue of the newsletter contained a Biography of Reid and the UK government became so upset at the interests of academics and the state of research within the UK that the Alvey programme was initiated. FACS published the Alvey Formal Methods Strategy and everyone went underground to prepare proposals. This meant that no further meetings were held until Christmas. The Christmas meeting was (unofficially) jointly sponsored by FACS and the London Mathematical Society.

17, 18 December 1984 - Imperial College, London
Computer Science - Some current mathematical approaches
(joint meeting with LMS)

P W Dell (BT)	Formal Aspects in Industry
H M L Holcombe (Queens, Belfast)	Systems, Machines and Algebra
J McDermid (SD)	A view of the Software Development Process
J R Abrial (Paris)	Formalisation of Programming Concepts
L Paulson (Cambridge)	A Survey of Natural Deduction Proofs in LCF
M Hennessy (Edinburgh)	Algebraic Theories of Concurrent Systems
H D Ehrich (Braunschweig)	Algebraic Concepts for the specification of data types
M Shields (Kent)	Deterministic Asynchronous Automata

This meeting was reported in Vol 7(3) by Maurice Clint (Belfast) and the papers by Paulson and Shields were published in the Computer Journal ([Pau85] and [Shi85]). Following the meeting further discussions were held with members of LMS, this led to a number of LMS sponsored colloquia. This meeting also provided the impetus which led to the ongoing sequence of UK-based conferences on Category theory and computing [PAPR85], [PPR87].

Probably the most important event for FACS during 1985 was being invited to become the Alvey SIG 'Formal Methods'. This not only allowed us to hold more meetings and expand the newsletter but Alvey allowed us to provide these extra services to members at (almost) no extra cost. Some FACS members complained that we organised meetings for Alvey which were not open to all our members. This was true, they were Alvey club metings and attendance was limited to holders of Alvey contracts, but the meetings were always reported in the newsletter. Accordingly, two special issues of the newsletter were produced and circulated:- Vol 8(1) on Formal Aspects of Specification and Vol 8(3) on Abstract Interpretation of Declarative Languages.

In 1985 we also reinstated our Easter workshop; with grateful thanks to SERC for funding the overseas speakers and John Darlington for organising the event. The topic revisited that discussed in July 1979.

31 March - 3 April 1985 - Reading University
Program Transformation

J Darlington (Imperial)	Introduction to Transformations
W Scherlis (CMU)	A view of Program Transformation
H Partsh (Munich)	A General Survey of the Munich CIP Project
J Darlington (Imperial)	Languages, Transformations and Programming Environments
R Dewar (New York)	Transformational programming in SETL
W Scherlis (CMU)	Specialisation and Program Derivation
J Boyle (Argonne)	An Introduction to the TAMPR System
H Partsh (Munich)	Transformation Systems
J Boyle (Argonne)	Transforming LISP to FORTRAN
T Axford (Birmingham)	Writing Re-usable Programs in a Functional Language
S Babiker (STL)	The LTS VLSI Design System
P Hodler (Oxford)	A Decision Procedure for a class of Transformational Problems
R Fleming (STL)	Program Transformation in a Polymorphic Language
R Dewar (New York)	A brand new algorithm, via transformations
C Runcimann (York)	Program and function inversion
Discussion sessions	Languages and Methodologies for Transformations. The Practicality of the Transformational Based Approach

The workshop was for those with a close interest in the topic and a feeling for the atmosphere can be obtained by considering that Bob Dewar was able to give an impromptu talk on an algorithm he developed during the workshop. But, in keeping with our aims, the next meeting was a tutorial for those just starting in this subject. In fact both John and Tim were able to present material from their books [CoB84], [Den86].

5 June 1985 - SOAS, London
Mathematics for Programmers
A Tutorial by D J Cooke (Loughborough) and B T Denvir (STL)

The Christmas workshop followed the well-tried format comprising a number of lectures on a particular theme. The popularity of Christmas shopping in London was again proved when a number of overseas delegates simply appeared for the meeting after being total previously that it was fully booked--but they were, as always, most welcome.

16, 17 December 1985 - Imperial College, London
Software Tools for Formal Methods

I Cottam (Manchester)	VDM and the MULE System
D Coleman (HP Labs)	Data Type Specification
M-C Gaudel (Paris)	PLUSS and ASSPEGIQUE
F Knowles (Gould)	A discussion of Formal Models
J Cunningham (Imperial)	The AVER Environment
I Cottam (Manchester)	Grapl
D Coleman (H P Labs)	Program Development from Executable Specifications
M-C Gaudel (Paris)	Test Set Generation from Algebraic Specifications using Logic Programming
F Knowles (Gould)	An experiment in code level Verification.

The February 1986 newsletter Vol 8(2) contained a report of this meeting, an evaluation of the Alvey Formal Methods programme at its halfway stage and book reviews by Reid. In 1986 the newsletter also started to syndicate 'Elektronick Brane'. Later issues of Vol 8 included Roland Backhouse's early papers on Type Theory which led to a later FACS meeting and wider publication by EATCS [BCM88]. Due to Alvey funding the 1986 newsletter ran to six issues.

In February we held an Alvey club meeting with the Dutch Concurrency Group. This meeting was to exchange ideas on Dutch and UK work which led to a larger technical meeting in 1987 and many more spin offs in terms of visits and closer cooperation between the two groups.

27 February 1986 - BCS, HQ(London)
Joint meeting with Dutch Concurrency Group

Presentations were given by:-
G Rozenberg (Leiden),
J W de Bakker (Amsterdam),
W-P de Roever (Eindhoven),
R Milner (Edinburgh),
W Roscoe (Oxford),
H Barringer (Manchester) and
M Shields (Kent).

For many years FACS has been a sponsor of the European Net Workshop and, although not directly involved in the organisation, we were pleased to be associated with the event when it was held in the UK at Oxford in June 1986.

The Easter meeting on OBJ was organised by Vicky Stavridou and sponsored by Alvey but run under the auspices of FACS.

18 April 1986 - Imperial College, London
The Specification Language OBJ and Applications
An Alvey SIG-FM Meeting

V Stavridou (UMIST)	Introduction to OBJ
R M Gallimore (H P Labs)	Rigours Software Development Using OBJ
C P Gerrard (UMIST)	Applications of OBJ to Clarify Functions
M Diss (STL)	Use of OBJ in protocol Specification
J A Goguen (SRI)	Aspects of the Past, Present to Future of OBJ
D Duce (RAL)	Experience of Specifying Graphics Software in OBJ
M Loomes & M W Nichols (Hatfield Polytechnic)	Specification of Concurrent Systems in OBJ

The FACS year ended in May with two important events. Roger Shaw became editor of the newsletter and John Cooke became Chairman of FACS (one of the current authors was pleased).

The Christmas meeting continued the tradition of tutorials for those working in the field and is reported in Vol 9(1).

15, 16 December 1986 - Imperial College, London
Classical and Non-Classical Logics
A course given by D Gabbay and M Sadler

The first issue of Vol 9 of the newsletter not only contained the report of the Christmas 1986 meeting but also a large set of book reviews (which we now syndicate to EATCS), and a report of the first meeting for 1987. This was a collaborative effort with yet another UK organisation with serious computing interests.

30 March - IEE, HQ(London)
Theorem Provers in Theory and Practice
(Joint meeting with IEE)

M Gordon (Cambridge)	Hardware Verification by mechanised formal proof
M Spivey (Oxford)	Mechanised reasoning about Z specifications
L Paulson (Cambridge)	The next 200 Theorem Provers
L Walkers (Edinburgh)	Formulating proof systems for automated deduction
B Ritchie (RAL)	Interactive proof construction
P Lindsay (Manchester)	Formal Reasoning in Software Engineering
I Mason (Edinburgh)	The Edinburgh LCF - A framework for defining logics

A paper by Larry Paulson on logics in HOL was in FACS-FACTS Vol 9(3).

Our next meeting furthered our work with the Dutch Concurrency Group and is reported in Vol 9(2).

14, 15 May 1987 - London Zoo (ie The London Zoological Society)
The Dutch Concurrency Research programme

I Aalbersberg and G Rozenberg	Some Recent Results in the theory of traces
I Kesmaat	Vector Synchronized systems and other generalisations of Path Expressions
P America and J de Bakker	Designing Equivalent Semantic Models for process creation
W-P de Roever	Different styles of compositional proof systems
R Koymans	Specifying message passing systems without extended temporal logic
R Gerth and A Boucher	A Timed Failure Model of Extended Communicating Processes
P America	Proof Theory for Object Oriented Languages
P America	A Proof Theory for a Sequential version of Pool

We also took advantage of a visit to the UK of an old friend now working in the Netherlands to bring together some of the earlier papers from the newsletter and obtain an update on his recent work.

14 September 1987 - Imperial College, London
The Martin Löf Theory of Types - a computer scientists perspective
A Tutorial by Roland Backhouse (Groningen)

Unfortunately, just when we were obtaining some good papers for the newsletter, Vol 9(3) contained another communication from Reid.

The year ended with a further mathematical tutorial for theoretical computer scientists.

21, 22 December 1987 - Imperial College, London
The Ins and Outs of Abstract Algebra
- A basic introduction for computer folk
A tutorial given by D Pitt (Surrey), M Hennessy (Sussex), A Poigné (GMD)

The tenth birthday was on 16 March 1988 and was celebrated at BCS HQ after the FACS AGM held on 19 May. Peter Landin (QMC) provided an interesting and enjoyable retrospective of the development of theoretical computing science. Dan Simpson (Brighton Polytechnic) was mercifully brief in his ten year retrospective of computing in the UK which allowed us all to enjoy the small party provided by the funds all members have paid in the past ten years.

4. The Future

We envisage that the group will change little over the next five years as the topic becomes increasingly important to those outside the field and more specialist groups are established. The group will remain as a focal point for those working in the area. We plan to remain a focus for the interchange of ideas and we hope to expand the opportunities for newer and more established researchers to get together and discuss their work prior to polishing it for formal publication.

We shall continue to hold meetings on a similar basis to those given above. We shall also continue liaison with other UK and European groups and we would hope to hold more joint meetings with these groups, there is a danger of fragmentation as the topic grows and we shall do our best to ensure that the fertile interchange of ideas continues. Despite good links with European Organisations we could perhaps have done more to disseminate results from Esprit projects and would wish to rectify this omission in years to come. There is also a need to consider the interface between formal methods and other aspects of Software Engineering, perhaps encompassing other branches of mathematics such as statistics.

The largest change in FACS activities will be in publications. The newsletter will continue in its present guise. We hope and expect to report in five years time that the journal has established itself as a major publication in the field. We are always pleased to receive material for either of these publications.

EXPLOITING FORMAL METHODS

This meeting was held at Imperial College on 25 May 1989, and was well attended by 56 people. There were six talks, all from industry, on the theme of examples of how formal methods are being exploited in industry. Abstracts of the talks are given below.

Formal Methods in the Development of CICS
John Wordsworth, IBM UK Laboratories Ltd
(See abstract)

The Use of VDM in the development of nuclear reactor software
Steve Sadler, Rolls Royce and Associates

The speaker described the development cycle of this type of software, and the use made of formal methods (VDM) in it. The systems are small and simple, giving little scope for refinement steps. The development includes cycles involving static analysis and testing. There is also a cycle in which the specification is documented in order to check it in comparison with the requirements. This animation is produced by rapid prototyping in code. The animation is used to derive test data. Hazard Analysis is also carried out. Tool support was needed for the techniques used over the full life-cycle.

For and Against Formal Methods for Secure Systems
Tom Farr, Logica Space and Defence Systems Limited UK
(see abstract)

The Use of HOL in the Development of Secure Systems
Roger Ives, ICL Defence Systems
(see abstract)

The above abstracts are the speaker's own except in the case of Steve Sadler. The main talks were followed by a "Poster Session" which included the following items:

Jawed Siddiqi from Sheffield City Polytechnic reported on experiences from teaching the course "Essential Mathematics for Software Engineering". Most of their students are not experienced in Mathematics and are easily alarmed by the notation. However, they soon become confident. Abstraction is central to the use of Mathematics in Software Engineering. Refinement seems to be difficult. Producing a formal specification itself is worthwhile. They have animated Z specifications using Lisp by producing an environment for validating formal specification prototypes.

Brian Monahan from IST described the facilities of Speedbox, a support tool for VDM hosted on an IBM PC and a SUN 3.

Tim Denvir from Praxis described the courses on VDM, Z and Discrete Mathematics offered through the NCC.

The meeting finished at 5.00 pm and was followed by the FACS AGM at 5.15 pm.
Tim Denvir

The Use of Z in Tool Development

David Brownbridge
Praxis Systems plc

ABSTRACT

This talk presents my experiences working with formal methods on client projects at Praxis. These are real software production projects not demonstrator projects for formal methods. For this reason my perspective is different from the dyed-in-the-wool purely formal approach and concentrates very much on the practical gains which can be made right now. The main issue is that maths can be used to improve software quality.

One of these projects produced tools for display and storage of documents in a distributed system. A specification of the tools was produced in the Z notation and is now being implemented. I will describe the relationship between formal methods and the Software Engineering process as observed on this project.

The use of HOL in the Development of Secure Systems

Roger Jones
ICL Defence Systems

Since 1985 ICL Defence Systems has had a small unit specialising in the application of formal methods to the development of very highly assured secure systems.

One distinctive feature of the work done by the unit has been the extent to which machine checked formal proofs have been employed in these developments.

The specification language, logic and proof development tool which has enabled this is HOL (Higher Order Logic), which was developed by Dr. M.J.C.Gordon and his collaborators at the University of Cambridge in 1985.

The special requirements of high assurance secure applications have demanded, and the characteristics of HOL have supported, the development and use of methods and techniques which differ in important respects from those which have been advocated for more general applications.

In this presentation some of these methodological issues will be examined. Special requirements of the application domain will be identified, methods enabling these requirements to be satisfied will be explained, and features of HOL which support the use of these methods will be described.

Formal Methods in the Development of CICS

John Wordsworth
IBM United Kingdom Laboratories Ltd

IBM's laboratory at Hursley has been investigating the use of formal methods for the development of CICS since 1980. CICS is a twenty-year-old transaction processing system whose growth has created a complexity that only formal methods could control. The talk reviews the history of the application of Z, a specification language developed at the PRG in Oxford University, and the perceived benefits. Current interest in the use of proofs in the development process and on concurrent refinement will be reviewed.

For and Against Formal Methods for Secure Systems

Tom Farr
Logica Space and Defence Systems Ltd UK

The paper considers the role of formal specifications in the development of security critical systems, based on the author's implementation experiences.

Formal methods are found to be a useful aid, but are not the whole answer, due to a lack of correspondence between specification refinement and design decomposition. The fundamental problem is that it is not currently practicable to make formal connections between a high-level specification and the low-level module specifications.

CASE STUDIES IN VDM

FACS Christmas Workshop held at Imperial College on 20 December 1989

Two new books on VDM are appearing early in 1990. One is the second edition of Cliff Jones' "Systematic Software Development Using VDN". The other is a collection of VDM Case Studies edited by Roger Shaw and Cliff Jones. the second of these contains 12 case studies covering the specification and development of systems over application areas from Data Base systems, to object-oriented languages to the design of User Interfaces.

The FACS Christmas meeting was based on the case studies book. 59 people attended, there were six talks, of which abstracts are given below. The full text of a report on BSI VDN standardisation is included.

What is a specification? Cliff Jones.

In this keynote talk, Cliff Jones emphasised the role of a specification as a text which has to be read by various users. For this an agreed language is needed: hence the BSI standardisation effort. A specification must be such that whether a program fulfils it is 'testable' in Poppers' sense. Specifications reduce waste in a development, and need to be composable and usable at different levels in the development. Abstraction enables one to think of the needs of the system. He talked about the MURAL theorem proving assistant, modularisation, and some of the remaining current problems which formal specifications have not yet solved.

BSI/VDM Standardisation
John Dawes
(see report?)

VDM Modules and Semantics
Stephen Bear

The speaker described his proposal for modules in the VDM specification language. He described the module syntax with import and export contracts, and how the state in a module may be used to specify a type. He then outlined how the semantics of modules were related to the semantics of the core language, in terms of models of a specification. He finally gave an overview of parametrised modules and their instantiation.

A Store Management System
Chris George

NDB: a binary relational database
Ann Walshe

Specifying Garbage Collection using VDM
Mario Wolczko

For a summary of these talks see the brief abstracts overleaf.

Tim Denvir

Heap Storage

Chris W. George

The specification describes the *NEW* and *DISPOSE* operations of the heap storage in Pascal. It contains several levels of specification, each intended to implement the previous one. It shows how an efficient implementation may be gradually created from an abstract specification by successive commitments to data structures and algorithms: VDM is used to capture design decisions one at a time. The example was originally created as an exercise for a VDM course. It has the advantage of being a problem many programmers are aware of while not being trivial.

NDB: The Formal Specification and Rigorous Design of a Single-user Database System

Ann Walshe

This specification of a general-purpose database system provides a good illustration of the usefulness of model-oriented specification techniques for systems. The chosen system (NDB) also has intrinsic interest. This chapter explains the derivation of the appropriate state; after this is found, writing pre- and post-conditions for the operations is relatively straightforward. The starting point for this specification exercise was an informal description which made heavy use of pictures. It was also couched too much in terms of the implementation to be abstract enough for a concise formal specification. As well as the specification itself, this chapter provides a good example of the development method (particularly data reification).

Garbage Collection

Mario I. Wolczko

Like the preceding chapter, this specification is concerned with storage management. In this case, the topic of garbage collection algorithms is discussed. Standard algorithms such as reference counting and mark-sweep are related to an abstract VDM specification. These specifications show how to record a body of knowledge about algorithms: VDM can be used to describe algorithms at a level of abstraction which makes their reimplementation in various languages straightforward.

BSI/VDM Standardisation — Status Report, December 1989

John Dawes, ICL

1 Background and History

The BSI/VDM standardisation project started in January 1986, when the time seemed ripe: the use of VDM was growing, a number of toolset projects were starting or under way, and there was the danger of proliferating dialects.

The project began with an initiative by STL, the research division of STC; STL had part in an Alvey project to produce a VDM toolset, which led to a definition of a VDM dialect called STC Reference Language, STC-RL. The initiative gained a widespread positive response from industry, from UK academia including of course Cliff Jones at the University of Manchester, and from Government establishments — both the National Physical Laboratory and the Rutherford Appleton Laboratory have played an active part. An international aspect was present from the start with the participation of DDC (Dansk Datamatik Center) and the Technical University of Denmark, the centres of VDM development in Denmark, and soon after of the University of Kiel. It has been strengthened since by significant contributions from the Technical University of Delft, and recently we have welcomed a member from Bull S.A. in France. We have also had significant support from VDM-Europe, an organisation sponsored by the Commission of the European Communities to promulgate the use and development of VDM in Europe.

The first meeting took place in February 1986. In March 1986 the first application was made to the British Standards Institution (BSI) for accreditation; this failed at a high level in BSI, but a revised application was successful in September 1987. After that we were an official BSI Panel, IST/5/50: VDM Specification Language, or VDM-SL. IST refers to information technology, and IST/5 is the committee responsible for programming languages and similar things.

An application was then made through the appropriate channels to the International Organisation for Standards (ISO) to start work on an international standard in parallel; this is done by proposing a New Work Item (NWI) to the appropriate subcommittee, in this case SC22, which is paralleled by BSI IST/5. Once again the first attempt was unsuccessful, and a revised application was made. Balloting closed at the end of November 1989, but the result is not yet known.

The first "Proto-Standard" was published in January 1987; it happened to be document no. 40, and document no. 40 it remained until recently (it is now just the Proto-Standard, without a document number). This is the main work document of the Panel; it will evolve eventually into the standard, though two major sections (the context conditions and the dynamic semantics) are at present separate documents. In May 1989 the existing versions of the Proto-Standard, the context conditions, and the dynamic semantics, were issued through the CEC as a "Draft for Comment", principally to inform the world of the way things were going. As some sets were sent out in a confused state, it might be worth mentioning that the titles of the four documents that should have been received are:

- Proto-Standard and Working Papers
- Proto-Standard
- Context Conditions
- The Dynamic Semantics of the BSI/VDM Specification Language

The first three are dated 24 April 1989. I am unsure of the date on the last (I have mislaid my copy).

Another significant event occurred in early 1989 when VDM-Europe set up the "Review Board for BSI/VDM Formal Semantics ", chaired by Professor Andrzej Blikle. The purpose was to submit the formally defined semantics of BSI/VDM-SL to a detailed scrutiny, starting with the dynamic semantics. The Review Board met three times in 1989: a preliminary meeting in March, a review meeting in August, and a final review meeting in December. Their findings have had a significant effect on the standardisation effort.

2 Objectives

The objectives of the standardisation project were clear from the outset and have not changed, though they have been refined a little. In essence they are:

- To standardise the specification language. Other aspects of the method, for instance the process of refinement whereby a specification is converted into an implementation, were considered unsuitable for standardisation as yet. The language would be called VDM-SL; the standard would define the following aspects.
  - The abstract syntax, i.e. the structure of the language without the details of its representation in actual characters.
  - One or more concrete syntaxes, i.e. representations in characters.
  - The static semantics or, as we now generally call it, the context conditions; this is the context-dependent part of the syntax.
  - The dynamic semantics, which defines the meaning of a specification.

  Verification conditions, i.e. the conditions that must be proved to hold for a specification to be well-founded, were added later, and work was started; but the amount of effort needed to derive them formally from the semantics has not been available.

  It was soon decided that all parts of the language definition should be formal, using appropriate formalisms. It was also decided that two standard concrete syntaxes were needed: a style using mathematical notation (the "mathematical" syntax") for publication, and one based on a restricted character set (nicknamed the "ASCII" syntax) mainly for information exchange.
- To harmonise the main existing styles of VDM language so as to cover the main applications. In practice this meant to accommodate the features of the Danish Meta-IV language and the UK dialects, notably STCRL and the version in use at the University of Manchester, to allow both imperative and applicative styles, and to support both system specification and language definition.
- To add any necessary features that were missing from existing dialects. Two significant missing features were identified. The first is modularisation: the ability to structure large specifications as collections of separate but related modules. The second is the ability to write reusable generic components of specifications, which can be instantiated appropriately for particular contexts; the required features can be called "polymorphism".
- Eventually to achieve international acceptance: i.e. to aim for an International Standard.

3 Status and Future Plans

3.1 The Language Definition

The Draft for Comment in May 1989 had complete abstract and mathematical syntaxes and dynamic semantics, and partial context conditions. Since then:

- The abstract syntax has been reviewed in detail, and is now in a very stable condition. It is defined in VDM, but not (yet) in the notation defined in the Proto-Standard.
- The mathematical syntax likewise, except for the lexical details and the precedence of operators. Proposals for these aspects were agreed at the last Panel meeting (Monday 18 December), but need to be reviewed. The mathematical syntax is defined in the British Standard metasyntactic notation.
- The "ASCII" syntax has undergone considerable review, and a proposal was adopted at the December 18 meeting, but again it will need to be reviewed. It has been agreed that, with very few exceptions, the "ASCII" syntax should be identical to the mathematical syntax down to the lexical level, so that it is defined simply by giving a table of equivalences for symbols.
- The context conditions published in May 1989 covered only statements and expressions. Much work has been done since then by a team from the NPL and the Technical University of Delft, and a complete draft is expected very soon. It is defined in VDM-SL, and takes the form of well-formedness predicates defined over constructs of the abstract syntax.
- The dynamic semantics was written by members of the Technical University of Denmark, and has been revised by then several times as a result of recommendations by the Review Board, and to stay consistent with the evolving abstract syntax. It is now fairly stable, but may be changed as a result of recommendations from the last meeting of the Semantics Review Board (16-17 December 1989). It is essentially a mathematical definition of a collection of "domains", representing all the entities that can be defined in VDM-SL, and a relation between specifications (again using the abstract syntax) and models that satisfy them, where a model is a mapping from identifiers to domains.

A number of outstanding problems were identified in the Introduction to the Draft for Comment. Most have been solved; the main remaining outstanding problems are as follows.

- Modules. See below.
- Error values. This concerns the semantics of erroneous specifications, e.g the value of a function application to arguments that do not satisfy the preconditions. Should such an error value be the same as the undefined or bottom value, or should it be a new value intermediate between bottom and the other values? How far should the error value spread outwards through a specification? These questions are under investigation at present.
- Loose specifications. This concerns expressions with incompletely specified values, e.g.

  let $x \notin$ **in** $\{1, 2\}$

  This can be treated as *nondeterministic*, i.e. its value (in any one model) is both 1 and 2; or as *underspecified*, i.e. its value is either 1 or 2. The present decision is that operations are treated as nondeterministic, while functions are treated as underspecified. This has two disadvantages; it means that functions cannot be easily refined to operations, as one might wish

to do when moving from an applicative specification to a more operational one (nearer to the implementation); and it is not possible in general to substitute its defining expression for a function application within an operation. These questions are still being addressed by the Panel.

3.2 Harmonisation of Danish and UK Styles

This has proved an almost complete success. BSI/VDM-SL allows both applicative and operational styles, and contains the most useful features of both Meta-IV and STCRL. For example:

- Statements are available, but in operations only. Expressions are guaranteed pure, i.e. cannot affect the state. This allows an operational style using (appropriately) operations for closeness to implementation, and an applicative style using functions for verifiability.

- Functions and operations can be specified either explicitly by defining their results, or implicitly by postconditions.

- At a lower (but still contentious!) level, the concrete syntaxes in general allow only one form for each construct; sometimes advantage is taken of the existence of both the mathematical and "ASCII" syntaxes to incorporate two traditional forms, e.g.:

mathematical	ASCII
X^*	set of X
$X \xrightarrow{m} Y$	map X to Y

(These are the cases where the "ASCII" and mathematical syntaxes are not exactly parallel.)

3.3 Adding Necessary Missing Features

- Modules

  This has not yet been completely solved. Stephen Bear worked out an approach to a solution that allowed control of import and export, that allowed modules to be parameterised, and that did not affect the semantics of the rest of the language (the "flat" language), whichwas accepted by the Panel. However this is a new area for VDM and all the details need to be worked out, including the detailed semantics. Because it is the view of several members that modules are essential for the standard to be usable in industry, some form of modules will certainly be present in the standard. However because the modules feature may still be a little tentative, and in order not to delay the standard, modules may be given a different status from the rest of the standard, e.g. put into an Appendix; though I for one would hope we can avoid this.

- Polymorphism

  This has been largely achieved at the component level. Polymorphic functions are catered for in the syntax and the semantics. The only outstanding problem identified is type inference, i.e. how to define the condition that an expression has the correct type for its context. This is not just a problem of polymorphism, but polymorphism complicates it. This is being studied by the Panel.

3.4 Internationalisation

As noted, the result of the ballot on the NWI is awaited. Assuming it is successful, a Working Group will be set up to work jointly with the BSI Panel. The arrangements have been accepted by SC22. Interest in participating has been shown by Canada, the USSR, and Japan, as well as Denmark, Germany, France, the Netherlands, and of course the UK, so it is hoped that the proposal will attract enough support to succeed.

3.5 Future Plans

The next draft for comment will be available for presentation at VDM'90; it should be complete and consistent, though no doubt there will still be outstanding problems in some areas. It is hoped to submit it to BSI for publication as a Draft for Public Comment (DPC) and to ISO for publication as a Draft Publication (DP) at the same time. After that the work will proceed at its own pace to culmination in publication of a BSI and/or an ISO standard.

4 The Panel (BSI IST/5/50 — VDM Specification Language)

Convenor: Derek Andrews
Computing Studies Unit
University of Leicester
University Road
Leicester LE1 7RH

telephone: 0533 523400
email: ajd@vax.le.ac.uk

Secretary: John Dawes
ICL
Eskdale Road
Winnersh
Wokingham
Berkshire RG11 5TT

telephone: 0734 639191
email: sjd@win.icl.stc.co.uk
John_Dawes_ICL@eurokom.ie

5 Disclaimer

All opinions expressed are those of the author, and not necessarily of his employer nor of BSI IST/5/50.

THIRD REFINEMENT WORKSHOP

The meeting was held at IBM Hursley on 9-11 January 1990. It was organised jointly by Oxford University PRG, IBM Hursley and BCS-PACS. Its proceedings are to be published by Springer-Verlag in the BCS Workshop series. For that reason, a brief report only is included here.

The first day, Tuesday 9 January, consisted of a tutorial. After an introduction to IBM Hursley by Mike McMorran and a welcoming address by the Director of the Laboratories, G W Robinson, we had three tutorial talks:

Refinement to Sequential Programs	Carol Morgan
Refinement to Concurrent Programs	Jim Woodcock
Refinement to Functional Programs	Geraint Jones

These talks were particularly well co-ordinated, and the speakers had looked ahead to the papers being presented subsequently in the workshop and had laid the necessary groundwork in order to make them more easily understood by the delegates. I was impressed by the attention to detail which the PRG team had shown in this respect.

The remainder of the Workshop started with a keynote address by Tony Hoare. In this he demonstrated the proof of a compiler for a high level language subset of occam, which was part of the PROCOS project. The whole approach was geared to ensuring that there is no gap between the assumptions of the compiler designers and those of the hardware designers. The compiler is specified by a predicate l on a source program p, l being defined recursively on the structure of p. l is correct if the interpretation of the direct code has the same effect as the course code. The proof is by induction on the structure of p. The extreme economy of his approach was impressive.

The other papers presented were as follows:

Mary Sheeran: Relations and Refinement in Circuit Design

Don Sanella: Formal Program Development in extended ML

Chris George: Refinement in RAISE

Chris Sennett: Using Refinement to Convince

Ralph Beck: Deriving an occam (implementation of Action Systems

Joseph Morris: A Methodology for Designing and Refining Specifications

Cliff Jones: Support Tools for Rectification

In addition, five discussion groups were held on the following themes:

Sequential Refinement
Concurrent Refinement
Functional Refinement
Comparison of Refinement in different methods
Refinement in VDN

The Workshop was very well attended by 105 delegates from seven countries, although it was not publicised particularly as an international event IBM provided lunches and refreshments and secretarial support throughout the Workshop and also an evening meal on Tuesday, 9 January. Oxford University PRG staff arranged the technical programme and FACS provided other co-ordination, support and publicity, FACS will co-ordinate a fourth workshop in about a year's time.

Tim Denvir

Structured & Formal Methods Workshop
Leeds Polytechnic & British Telecom
December 5-6 1989

Lesley Semmens & Tony Bryant, Leeds Polytechnic

Introduction

The workshop arose from activities of the Structured & Formal Methods Research Project initiated by members of the Computing Science Dept. of Leeds Polytechnic and, since 1987, funded jointly by the Polytechnic and BT. The major concern of the project is the use of Formal and Structured Methods in the development of information systems.

Presentations

Two introductory papers aimed to set out the objectives of the workshop. These were followed by a series of invited papers.

Setting the Agenda. Paul McGrath (Leeds Polytechnic) saw quality in systems encompassing the development process and the products as a common interest. The gathering aimed to decide on areas of agreement and to arrive at a basis for continued exchange of ideas.

Structured & Formal Methods; Developing a Framework for Synthesis & Investigation. Tony Bryant (Leeds Polytechnic) discussed the paradox that despite two decades of technological advances organisations are experiencing more problems than ever before in IS development. Methodologies were packaged for sale rather than use; the response to criticism was to add new features making them even more unweildy; and discussion on issues other than those related to the 'market' has been curtailed. Two factors would alter this:
First, the position of SSADM made it a prime target for criticism and among its critics are some with a degree of detachment. Key issues will be raised and the discussion will move to include scrutiny of fundamental methodological issues.
Second, experienced practitioners are using methodologies in non-standard ways and developments towards a 'Euromethod' were encouraging this intelligent fragmentation on the basis of requirements and needs. These two trends aspire to a common cause: a global view of IS development incorporating a 'methodological map', a universal model.
The aims and strategies of the Leeds/BT project were outlined. The ultimate aim of this is to design a way of merging or linking structured and formal approaches within the context of the putative methodological map.

Methods in the Commercial Environment. Alan Stoddart (Divisional Manager Systems & Software Engineering Division, BT Research Laboratories) described his role within BT which includes responsibility for identifying and transferring the best technologies relevant to BT's business. He therefore has a strong commitment to research both short/medium term and long term.
The interest in notations and methods is medium term wheras the long term perspective encompassed the system as a whole. Existing methods are incomplete & ambiguous and offer little or no tool support. BT's requirements from a method are that it covers real-time, data processing and transaction processing and that it covers the complete life cycle from initiation to obsolescence. There must be tool support suited both to users and developers. Formal notations are seen as a route towards such a method, but there must be 'seamless' transition between a formal notation and the graphic or user-oriented facets of the approach.

Standardising Methods - Euromethod. Neil Glover (CCTA) outlined the CCTA's involvement with SSADM and the initiatives in developing a European method.
SSADM was aimed at increasing staff effectiveness, project communication, and user involvement and providing mechanisms for better management and quality. Euromethod, however, is not an individual method but a basis for procurement specification and perhaps later a basis for harmonisation of key techniques. The framework will take the form of a series of templates defining points of the life cycle with mandatory deliverables for evaluation and review.

Do we need a Universal Systems Model?. Prof. John McDermid (York University) - the short answer is 'yes'! If you are going to unify formal and structured methods you do need some underlying canonical systems model. There are some key questions to be answered:
- do we need both static and dynamic models of a system?
- are the static/dynamic models sufficient ? do they let us reason about

similar. Questions to be asked were: Will it work? and Will it be enforced? In answer to the first the introduction and gradual acceptance of Ada was used as an example on the basis of which 1995 was predicted as a date for stabilisation of the standards and 1997 for full acceptance. To the second, despite pressure from industry, he felt that formal methods would remain part of the standard and that the best route forward was to begin using basic techniques on existing projects and, crucially, monitoring the results enabling an answer to be given to the question 'Has it worked?'

Discussion

The discussions took a wide variety of directions. A major concern was who structured and formal methods were for. It was agreed that a major problem was that no single notation could encompass the whole of systems development. Another problem was the way the current workforce thought about systems: they were to ready to start from the operational level. There is therefore a need to persuade developers to think denotationally in addition to educating them in terms of new notations.

Additional Material

In addition to the formal presentations other papers and outlines of current research interests were contributed:

- Ideas for a Z methods handbook (part of a larger IED project (ZIP))
- Automatic documentation of Z schema specifications
- Representing Real World Knowledge at the stage of Requirements Capture
- Structuring Physical Design in SSADM
- Generating Database Update Programs from ELH Analysis
- Translating Data Flow Diagrams into Z
- The role of intuition in systems development

Further Information

For a fuller version of this report, together with copies of papers and/or 'overheads' contact: Tony Bryant, Division of Informatics, Leeds Polytechnic, Beckett Park, Leeds LS6 3QS; email (JANET) tonyb@uk.ac.leeds-poly

security and safety ?
- is one model sufficient for all types of system ?

We might use a formal method to underpin a structured one. The formal method would give a frame of reference which might permit comparisons between different forms of specification. The example of 'objects' where their external behaviour could be seen in terms of CSP traces or similar, producing a detailed set of transactions and their internal state could be represented using a model-based notation such as Z was cited.

A Strategic View of Formal Methods. Anthony Hall (Praxis) stepped back to pose basic questions about methods. 'What are methods for?' - this must be discussed first and only then can we ask 'What should we look for in a method ?', 'What problems do methods solve?' and 'How do methods solve problems?'.

Agreeing with Parnas that software development is an inherently difficult intellectual task, methods exist to help people think. A method must help developers think: think about the right things, and think about things right, in that order.

In general it is the hardest and biggest problems we should be addressing and not those which can be most easily solved. The key element of any method is its notation and that notation should convey meaning; it must be abstract; it must be user or application-oriented rather than machine-oriented; it must denote what is to be done rather than how; and should have an appropriate level of detail.

He then posed questions as to how structured and formal methods conformed to this archetype and saw them as differing markedly but noted that both had been brought to bear on the critical early stages of the development process. The future convergence between real-time and commercial systems was highlighted. The combination of formal techniques to form a single method in the way that structured techniques had been was seen as a possible way forward.

Def.Stan.00-55. John Robinson (KBSL) outlined some aspects of the standard, together with some predictions based on what he considered to bé a similar case of the reception and acceptance of Ada.

The standard was about more than just formal methods and although it concerned the military sector he recognised the needs of the civil sector to be

Computer Aided Transformation of Z into Prolog

A. J. J. Dick (Racal Research Ltd.)

P. J. Krause (University of Surrey)[†]

J. Cozens (University of Surrey)

[†]Now at Biomedical Computing Dept., Imperial Cancer Research Fund

ABSTRACT

A strategy for the rapid prototyping of Z specifications is described. The semantics of Z are captured in a library of Prolog rules based on a standard specification of the language constructs. Each Z schema is translated directly into a Prolog rule which is then transformed for efficiency. A classical generate-and-test form is used to mirror the structure of a schema; goals resulting from the schema variable declarations are used to generate candidate instantiations, and goals resulting from the schema predicate part are used to test the instantiations for required properties. The incompleteness of the Prolog search strategy limits the approach to the use of finite sets. Negation by failure places limitations on the kinds of transformation that can be applied. A set of tools is being developed to explore the potential of this approach for the animation of Z specifications. These tools include a Z-to-Prolog translator, a transformation aid, and an animator. Initial experiments with the transformation tool suggest that the approach can offer more than simple animation; execution of the Prolog can also be used to check logical consistency and non-determinism.

ACKNOWLEDGEMENTS

This work is based on a project initiated by Ron Knott, University of Surrey, whose assistance in development of the library of Prolog rules was invaluable. His continued technical support is gratefully acknowledged. We also thank David Pitt, Steve Schuman and Paddy Byers for useful discussions on the formal specification side of the work.

1. Introduction

Rapid prototyping is seen as important method of validating a specification against its informally perceived requirements. This is especially so when the specification language used is not easily understood by the non-specialist reader who, nevertheless, has a strong interest in the consequences of the specification.

By their very nature, specification languages tend to be non-algorithmic, which inhibits direct execution. Rapid prototyping, therefore, involves making an interpretation of the specification in some language that does lend itself to direct execution. Having invested in a formal specification, it is highly desirable that the process of interpretation should itself have a formal basis. Otherwise, the interpretation may not truly reflect the consequences of the specification. Equally, the process should be as automatic as possible, to minimise the possibility of human error.

This paper reports some preliminary work on the automated rapid prototyping of Z specifications by a direct translation into Prolog. It builds on the work initiated by [Knott/Krause'88a] as part of the Alvey FORSITE project (SE/065), whose approach is reviewed here. Taking the basic types sets, sequences and relations, their manipulative environment is modelled using a library of Prolog rules whose specification is taken from the notation in the Glossary of *Specification Case Studies* [Hayes'87]. Z schemas follow the format of a declaration of variables grouped with predicates constraining the values of those variables. This can be translated directly into Prolog as a generate and test cycle, with the declarations generating candidate values of variables and the predicates checking for correctness. This will, however, lead to grossly inefficient programs and some thought to filter promotion is almost always required before the programs will run in a "reasonable" time.

In the preliminary work the translation of the Z specification into Prolog was carried out "by hand", although following fixed rules. Correctness preserving transformations were then also carried out by hand to improve the efficiency of the resulting program. Work is well under way to develop a syntax-directed editor and a translation program to enable a formal specification to be input into the computer and automatically translated into Prolog. Work has also been carried out to semi-automate the transformational part of this rapid prototyping process [Dick/Krause'89]. Some further details of this work can be found in Section 6. It is intended that this be the first stage of a specification development environment written in Prolog with a NeWS user interface.

In what follows, no attempt is made to explain the Z notation in detail; readers may wish to refer to [Spivey'88, Spivey'89] for a detailed description. A knowledge of Prolog [Stirling/Shapiro'86] is also assumed of the reader, although some operational aspects of the language are briefly explained below.

2. Construction of Prolog Rules from Z Schemas

We give here a brief characterisation of the approach taken to translating a Z specification into Prolog rules. Further details can be found in [Knott/Krause'88]. We concentrate for the time being on schemas, leaving a discussion of other aspects of the Z notation, such as axiomatic definitions and language extensions, to Section 5.

2.1 Basic Types

Z is a strongly typed set-theoretic language. A Z schema describes a collection of named variables and the relationships that must hold between them. To model the Z type system in Prolog, two basic patterns are defined which characterise *sets* and *tuples*. This enables the declaration of Prolog representations for *relations*, *functions* and *sequences*. A library of Prolog rules then define the manipulative environment, these rules being derived from the notation in the Z Glossary found in [Hayes'87].

We take the default Prolog representation for sets, namely ordered lists using the standard lexicographical ordering. Many of the operations involving sets can then be implemented in Prolog using the built in predicate `setof`. For example, set intersection may be defined as:

```
intersect (Xs,Ys,Zs) :-
        setof (X, (member (X,Xs),member (X,Ys)), Zs).
```

Note in passing that this rule has a restricted modality; `Xs` and `Ys` must both be instantiated when the rule is called.

Ordered n-tuples are represented in the conventional way: an ordered n-tuple of $t_1, t_2, \ldots t_n$ being represented by :

$(t_1, t_2, \ldots t_n)$

Structures representing relations, functions and sequences may now be defined in an obvious way. For example, the ubiquitous biblical parent relation is given as the set of ordered pairs:

```
[(abraham,isaac), (haran,lot), (haran,micah), (haran,yiscah)]
```

2.2 Strategy

A Z schema consists of a declaration part and a predicate part which constrain the possible states described. This format can be mirrored in Prolog by the classic generate and test cycle. The declarations are translated into Prolog goals which can be used to generate candidate solutions, and the predicate part into goals used to check the solutions for correctness.

Thus a schema becomes a Prolog rule with an argument for each state variable declared in the schema declarations, and an argument for every given set used in the schema. Execution is achieved by calling the rule with the given sets instantiated with "representative elements" of the required type. The rule then generates an instantiation of the state variables which satisfies the schema predicates, using Prolog's depth first search strategy.

This approach leads to two immediate problems :

1. The inherent unfairness of the depth first search strategy may lead to incompleteness of the Prolog rule.
2. Many iterations of the generate and test cycle may be required before the correct solution is found.

By only considering finite sets, the first problem should not arise (but see Section 3 for some further comments on this). The second problem is one of efficiency, and may be addressed by promoting the test goals into the generator wherever possible. The ability to do this is limited by the modality of the rules in the library used in the translation. Many of the rules are incomplete when certain combinations of arguments are uninstantiated. These limitations are discussed in detail in Section 3.

This provides the basis for a strategy for generating Prolog programs from a Z specification. In summary :

1. The *given sets* are instantiated with "representative elements" of the required type.
2. A "first order" Prolog rule is defined with goals derived from the schema declarations acting as generators, goals derived from predicate parts as tests.
3. Filter promotion and other transformations are applied, *when possible*, to improve the efficiency of the Prolog rule.

2.3 An Example

As a simple example of the construction of a Prolog rule from a Z schema, consider the specification of a *File Update* given in *Specification Case Studies* [Hayes'87, p.43] :

File Update

$f, f' : Key \pfun Record;$
$d? : Record;$
$u? : Key \pfun Record$

$d? \subseteq dom(f) \land$
$d? \cap dom(u) = \{\} \land$
$f' = (d? \ndres f) \oplus u?$

Each record in the file is indexed by a key, so that the file is modelled as a partial function from keys to records. A transaction involves specifying a set of keys of records to be deleted from the file, and/or a partial function giving the keys to be updated together with their new records. A literal line-by-line translation of the schema using the Prolog library yields the following rule:

```
file_update(Keys,Records,F,Fp,Di,Ui) :-
        partial_fn(F,Keys,Records),
        partial_fn(Fp,Keys,Records),
        powerset(Keys,Pset), member(Di,Pset),
        partial_fn(Ui,Keys,Records),
        dom(F,Dom), has_subset(Di,Dom),
        domain(Ui,Doml), intersect(Di,Doml,[]),
        domain_sub(F,Di,Temp), overriding(Temp,Ui,Fp).
```

This rule will execute very inefficiently because instantiations of `Fp` are generated in the second goal, and not checked until the last goal. If the modality of the `overriding` goal is such that it can be used to instantiate `Fp`, then the order of the goals can be changed so that the `overriding` goal acts as the generator. Checking the modality of the `overriding` goal, we can be sure that, if the two parameters `Temp` and `Ui` are ground when the rule is called, a ground instantiation of `Fp` will result. Hence the second goal can be demoted to a position following the `overriding` goal, to give the revised rule:

```
file_update(Keys,Records,F,Fp,Di,Ui) :-
        partial_fn(F,Keys,Records),
        powerset(Keys,Pset), member(Di,Pset),
        partial_fn(Ui,Keys,Records),
        dom(F,Dom), has_subset(Di,Dom),
        domain(Ui,Doml), intersect(Di,Doml,[]),
        domain_sub(F,Di,Temp), overriding(Temp,Ui,Fp),
        partial_fn(Fp,Keys,Records).
```

We could further improve the efficiency of this rule by replacing the two goals

```
powerset(Keys,Pset), member(Di,Pset),
```

with the equivalent goal `has_subset(Di,Keys)`. This avoids the time consuming generation of a powerset.

The two given sets `Keys` and `Records` contain the primitive type information for the file update. Prior to calling the `file_update` rule, they must be instantiated with "all possible" elements of type Key and type Record respectively.

The following is a transcription of a file update operation using the Prolog rule defined above :

```
|?-Keys = [k1,k2,k3,k4,k5,k6],
|: Records = [r1,r2,r3,r4,r5,r6],
|: F = [(k1,r1),(k2,r2),(k3,r3),(k4,r4)],
|: Ui = [(k3,r5),(k5,r6)],
|: Di = [k2, k4],
|: file_update(Keys,Records,F,Fp,Di,Ui).

     Fp = [(k1, r1),(k3,r5),(k5,r6)]
```

A more thorough example of the rapid prototyping of a Z specification in Prolog can be found in the report *LIBRARY SYSTEM: An example of rapid prototyping of a Z specification in PROLOG.* [Knott/Krause'88b].

3. Limitations

3.1 Finite Sets

Mention was made in the previous section that sets should be finite. This is because of the following problem which arises as a result of Prolog's depth first search strategy. Suppose we require a program that generates all the triples of natural numbers `(x,y,h)` such that $x^2 + y^2 = h^2$. This may be specified in Z as

Pythag Triads

$x!, y!, h! : \mathbb{N}$

$(x*x) + (y*y) = (h*h)$

If we define a Prolog rule `posnum` which generates the infinite set of natural numbers:

```
posnum(1).
posnum(N) :- posnum(N1), N is N1 + 1.
```

Then we may naively translate the Z specification into :

```
pythagtri(X,Y,H) :-
        posnum(X),posnum(Y),posnum(Z),
        0 is H*H - X*X - Y*Y.
```

Although the solution `X=3, Y=4, H=5` is in the meaning of the program, Prolog's depth first search strategy will never find this, or any other solution. Having tried and failed with `X=1, Y=1, H=1`, on backtracking it will try the solution `X=1, Y=1, H=2`, then `H=3,4,5`... and so on without terminating.

Thus Prolog's depth first search rule may lead to the incompleteness of a Prolog program. For example, Prolog could never show that `pythagtri(X,Y,H),X=3, Y=4, H=5` was a logical consequence of the above program. Yet given the query

```
|? - pythagtri(3,4,5).
```

it will verify that the given parameters do have the pythagorean property. One way of overcoming this problem in this case would be to define a Prolog rule to generate successive triples of natural numbers with an ordering based on diagonalisation. This is fairly easy to do in this case, but more work is needed to obtain a more general "safe" strategy for handling infinite sets.

For current purposes we limit ourselves to the use of finite given sets. However, it would be premature to say that the use of finite sets completely removes problems of incompleteness due to "unfairness" of a depth first search strategy. One may for example refer to the simple program in [Lloyd'84] :

```
p(a,b).
p(c,b).
p(x,z) :- p(x,y),p(y,z).
p(x,y) :- p(y,x).
```

The query `p(a,c)` will result in a stack overflow in Prolog. It is not immediately clear that one may not be able to write a Z schema that would produce the same behaviour when translated into Prolog.

3.2 Negation by Failure

The general translation strategy described above works because the Prolog goals arising from variable declarations never contain negation, and can therefore be guaranteed to instantiate the variables. The Prolog goals arising from predicates may contain negation, but this is no problem as long as all variables in the negated goals have been previously instantiated. Prolog's negation as failure cannot be used to instantiate a variable.

Where negation as failure becomes a real limitation is in the ability to transform Prolog rules for greater efficiency. To illustrate this, consider the following event for inserting a new identifier into a set of identifiers:

File Insert

$x : \mathbb{N}$;
$X, X' : \mathbb{P}\,\mathbb{N}$

$x \notin X$;
$X' = X \cup \{x\}$;

The derived Prolog rule

```
file_insert(_x,X,Xp,NAT) :-
        member(_x,NAT),
        has_subset(X,NAT),
        has_subset(Xp,NAT),
        not member(_x,X),
        union(X,[_x],Xp).
```

will generate an as yet unused identifier and insert it into an updated file. An attempt to promote the goals `has_subset(X,NAT)` and `not member(_x,X)` results in a rule that will behave quite differently:

```
file_insert(_x,X,Xp,NAT) :-
        has_subset(X,NAT),
        not member(_x,X),
        member(_x,NAT),
        has_subset(Xp,NAT),
        union(X,[_x],Xp).
```

With `X` instantiated as any non empty set of identifiers and `_x` uninstantiated, the call to `member(_x,X)` will succeed and hence the negated goal will fail. The promotion of these goals has caused a negated goal to be called before all its variables have been instantiated. This becomes particularly problematic with the use of universal quantification in a schema, if implemented in Prolog as :

```
for_all(Generator,Predicate) :-
        not(call(Generator), not call(Predicate)).
```

Promotion of such a `for_all` goal may lead to less obviously incorrect behaviour. If `Predicate` contains an uninstantiated output parameter, rather than fail, the `for_all` goal may succeed, but return a still uninstantiated variable, and hence provides no constraint on its value. A universally quantified predicate cannot be used to generate values for state variables.

4. Modal Analysis in Rule Transformation

To determine the correctness of various transformations, two types of modal information have to be known: the valid static modes of all the library and system defined Prolog rules, and the transformational mode under which the a rule is being transformed.

The latter of these relates to the mode in which a specification is to be animated. The nature of Prolog is such that one may often be able to use a rule to determine the "inputs" that result in a specified "output" as well as determining the "outputs" that result from a specified "input". This is mirrored in the animation of Z specifications by allowing the animator to explore the consequences of event schemas by deriving the pre-states that result in a particular post-state, and *vice versa*.

If it is known in advance the mode in which a rule will be executed (that is, which arguments will be instantiated on call), there may be transformations that can be performed that are not possible in more general cases. Indeed, several different efficient versions of a rule may be created depending on the mode of use.

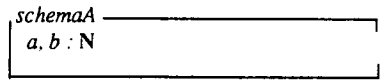
For instance, in the last example of `file_insert`, the proposed transformation would be correct if the variable `_x` was instantiated on call.

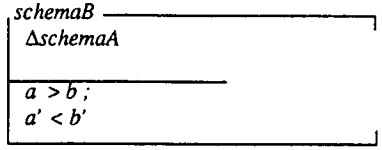
5. Other Constructs in the Z Notation

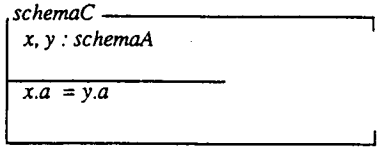
So far we have only considered simple Z schemas. Axiomatic descriptions and generic schemas involve other considerations which have not been fully worked out. Nor have we considered more complex forms of schema referencing, and the schema combinators. This section sketches some ideas in this area.

5.1 Schema Referencing.

Consider the following schemas, in which *schemaB* is a *schemaA* event, and *schemaC* names two occurrences of *schemaA*:







The Prolog translation of *schemaA* is performed in the way previously discussed:

```
schemaA(Nat,_a,_b) :-
        member(_a,Nat),
        member(_b,Nat).
```

The references to *schemaA* in *schemaB*, one of them decorated, are mirrored in the Prolog by two calls to *schemaA* in the obvious way:

```
schemaB(Nat,_a,_b,_ap,_bp) :-
        schemaA(Nat,_a,_b),
        schemaA(Nat,_ap,_bp),
        _a > _b,
        _ap < _bp.
```

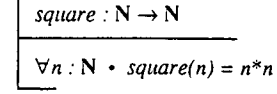
In the Prolog translation of schema C, the variables *x* and *y* may be treated as tuples as follows:

```
schemaC(Nat,_x,_y) :-
        schemaA(Nat,_xa,_xb),
        _x = (_xa,_xb),
        schemaA(Nat,_ya,_yb),
        _y = (_ya,_yb)
        _xa = _ya.
```

This treatment reflects the Z concept of an instance of a schema being a tuple consisting of instantiations of the components of the schema. Note that the numbers of arguments in *schemaB* and *schemaC* mirror the number of named state variables in each, plus the given set of naturals.

5.2 Axiomatic Descriptions

Axiomatic descriptions can be treated as per the following example:



These may be translated to:

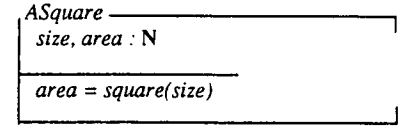
```
square(Nat,_square) :-
        total_fn(Nat,Nat,_square),
        for_all(member(N,Nat),
                ( is_related(_square,N,Temp),
                  Temp is N*N
                )).
```

Since axiomatic descriptions are often given using universal quantification, making such definitions efficient in the Prolog frequently presents a problem. In this case, all possible partial functions will be generated until the square function is found — extremely inefficient. A possible transformation is to take advantage of the knowledge that *square* is a function, and eliminate the *for_all* by treating it as follows:

```
square(N,NN) :-
        NN is N*N.
```

However, it is probably better, in general, to consider transformations of axiomatic definitions in the context of particular references.

Axiomatic descriptions can be referenced as, for instance, in the following schema:-



Using the Prolog library rule *is_related(Rel,X,Y)*, which instantiates related *X*, *Y* pairs in the relation *Rel*, the *ASquare* schema may be translated into:-

```
asquare(Nat,_size,_area) :-
        member(_size,Nat),
        member(_area,Nat),
        square(Nat,Square),
        is_related(Square,_size,_area).
```

By unfolding the definition of *square* into the body of *asquare*, one can envisage a series of transformations leading to the following rule which does not have to generate the entire *square* function:

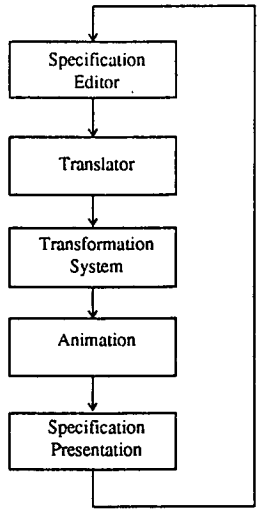
```
asquare(Nat,_size,_area) :-
        member(_size,Nat),
        member(_area,Nat),
        _area is _size*_size.
```

5.3 Generics

By making given sets used in Z schemas into arguments in Prolog rules, schema have, in a sense, already become generic in the translation — that is, parameterised by the basic sets used in the schema. Explicit generics are treated in just this fashion.

6. Support Tools

A specification tool set is being developed at the University of Surrey to investigate the usefulness of animation in the process of specification. The tool set being developed consists of five main components as shown in the figure below. At present the stages which have been implemented are the editor, translator and transformation system. Some initial ideas have been tried for the animator tool set.



The editor has been developed using the SUN network extensible windowing system (NeWS). It lies somewhere between a syntax directed editor and a text editor. To the level of entering formulae the editor is syntax directed, with schemas being generated and the desired fields added to these schemas. The formulae are then entered into the fields as text. Having created/modified some schemas these may be checked selectively for syntactic correctness. Correct schema inclusion is built up, this is then used by the translator to generate the first version of the executable code.

An advantage of the editor is that it allows schemas to be viewed in any context, within a specification, as it is window based and there is no linear ordering of the schemas as in a text document. This facility allows selected parts of the specification to be viewed easily.

The translator is based on the work of [Krause/Knott'88a] and generates a Prolog program from the database created by the editor.

The transformation system is described in detail in [Dick/Krause'89]. It allows a variety of transformations to be selected by the user for application to Prolog rules. The transformations supported include unfolding, goal promotion/demotion, and removal of duplicate goals. It uses a set of inference rules that encode knowledge about the library of Prolog rules used in the translation process to reason about the goals in the rule. Modal analysis is carried out to ensure the correctness of the transformations performed. The execution of rules may be portrayed on the screen in a way that vividly portrays undesired inefficiency. The tool is itself written in Prolog, and a new graphics front end is being developed which allows easy case selection of schemas, rules and goals using a mouse.

Having obtained a set of efficient Prolog programs, the objective is now to present the system as specified to the client. It is required that the presentation (execution) of the programs reflects the original system structure. In the formalism being used, specifications model a system having state with a number of events over the state. As a first attempt at animation, the state will be represented listing its components, these being partitioned according to the system structure. It will then be possible to initiate events over

this state and observe the way in which the state changes. A trace of the events may be recorded, along with reasons for event failures (*e.g.* when pre-condition not satisfied). The generation of such an animation should be largely automatic and should enable the designer to obtain some insight into the dynamic behaviour of the system specified. This presentation however is still not totally suitable as a way of presenting the specification to the client.

For presentation of an animated specification to the client, the designer should be able to start with the automatically generated presentation and enhance this with a more meaningful interface. This would involve connecting variables and events to graphical objects reflecting the intended interpretation of the specification. It is envisaged that to allow the designer to build such a presentation, an animation tool set will be developed. This will supply the designer with the uninterpreted animation, a set of graphical objects and the tools for connecting the animation to the objects. It should be possible for the designer to extend the set of graphical objects. This system would essentially be a high level graphical programming interface. Having constructed the interpreted animation this could then be presented to the client.

The client would now have an interface to the specification which can be used to assess whether the requirements have been met. On feedback from the client the designer can decide whether or not the underlying specification is correct. This will be dependent on the nature of the clients feedback, as some of the comments may simply require modifications to the presentation and not affect the underlying specification.

The development of a tool set for animating specifications is now being carried out in the SERC Animate project. This project started in November 1989 and involves the University of Surrey, Logica Cambridge Limited and Data Logic Limited.

7. Experiments

The main experiments that have been carried out have used

- the specification of Telephone Network from [HAYES'87].
- a lift system written in a language closely related to Z [Krause/Knott'89].
- a specification of a vending machine.

Experiments on the Telephone Network demonstrated the value of rapid prototyping, in that it revealed what we believe to be unintended properties of the specification. The system is specified so that all outstanding requests for a telephone connection are satisfied subject to the constraint that no 'phone may be engaged in more than one connection at a time. If a call is made to an already connected 'phone, that call is queued as an outstanding request until the first caller hangs up. Then the requested connection is made. A Call request in the Prolog prototype reproduces this behaviour. But a request for further solutions produces a state in which the first connection is broken, with connection made to the calling party and the originally connected 'phone being queued as an outstanding request. We suggest that this more perverse behaviour was not an intended feature of the specification.

By applying the kinds of transformations described above, prototyping of the lift specification was practical. Every necessary transformation could be carried out using the tool. It was very tedious carrying out the same transformations entirely by hand. The animation of the lift system prototype was completed by adding a simple interface which explained simply in English where the lift was, and what it was doing, by examining the state of the lift between events. Using this animated interface, it was possible to validate the specification by trying out various situations.

The vending machine example contained several Z language extensions, which seemed to make it harder to make any headway on increasing efficiency. By choosing a small enough set of the natural numbers, though, a prototype that ran in minutes rather than hours was eventually created. With the right inference rules, we felt that transformation would have been considerably easier.

This work highlighted some difficulties with the structures used in the transformation system. In particular, it turns out to be difficult to apply inference rules to sequences of goals that lie inside another goal, like *not/1*, for instance. At present, the goal list is assumed to be a flat structure, and it is difficult to get at goals which are parameters of other goals. A more sophisticated approach would allow transformations to be more generally applicable.

It became apparent during these experiments, that some combinations of transformations occurred frequently. For instance, repeated unfolding of schema references followed by removal of duplicate goals. A language for describing transformational tactics would be an advantage.

The transformation that seemed most often to have a dramatic effect on efficiency was promotion/demotion of goals. Unfolding frequently allowed duplication to be removed, which also helped. Many of the other transformations often had little more than an aesthetic value, only slightly affecting efficiency on the examples we were trying.

8. Conclusions and Further Work

When viewed as a logic language, Prolog is an obvious candidate to use for the rapid prototyping of a specification written in first order predicate logic. It is quite straightforward to generate a Prolog program from a Z schema, given an understanding of Z and the Prolog language. See for example [Stepney/Lord'87]. What we have tried to produce is a general environment whereby a Z specification may be compiled automatically into a Prolog program that correctly prototypes the formal specification. For reasons given, in general, a number of correctness preserving transformations need to be applied to the resulting Prolog program to improve its efficiency. It may well be that a more advanced control structure, as available in Mu-Prolog [Lloyd'84] or one of the concurrent logic languages [Ringwood'89], may allow a more straightforward approach without the need for program transformations. However, at present the program transformation system that has been described in this paper is an essential component in the animation of a formal specification.

The transformations that may be applied to the Prolog rules will in general depend on the required modality. As they are generated by the translation process, only the given sets need be instantiated. This enables the Prolog prototype to be used to fully explore the properties of a specification. All initial states that produce a given output may be generated, as well as all outputs for a given initial state. Any non-determinism in a specification will be quite naturally reflected by the non-determinism in a Prolog prototype. This may be an intended property of the specification, but on the other hand further solutions may show up an unintended underconstraining of the problem by a specification. An example of the latter case was demonstrated, as mentioned, in an experimental prototype of the Telephone Network specification given in [Hayes'87]. Unintented non-deterministism is a common error in Z specifications, and can be detected as Prolog uses backtracking to generate all possible solutions to rules.

The execution of rules that prototype a Z specification can be used to explore its logical consistency. If, for example, the pre-condition of a schema can never be satisfied, the Prolog rule will always fail. Some of the transformations are forms of partial evaluation, and the simplification of schemas may be possible by studying the effect of transformations on the Prolog. In the case of unsatisfiability, a rule may reduce it to the single unsatisfiable goal, `false`.

Work is under way at Surrey to produce a general purpose interactive graphics front end that will enable the properties of a Prolog animation of a formal specification to be easily and fully explored. We have shown the feasibility and the value of automating the transformational part of a method for prototyping Z in Prolog.

The work has suggested the kinds of in-build knowledge required in such a system, such as modal knowledge relating to system and library rules, and inference rules for the underlying set theory used in the specification language. Useful investigations could be made into heuristics for the application of transformations, and a simple language devised for combining and repeating transformations.

References

A. J. J. Dick & P. J. Krause [1989], "Computer Aided Transformation of Prolog Specifications", Tech. Report 10-1702, Racal Research Ltd., Worton Drive, Reading

I. Hayes (Ed.) [1987], "Specification Case Studies", Prentice Hall

R. D. Knott & P. J. Krause [1988a], "An approach to animating Z using Prolog" Report A1.1, Alvey Project SE/065, Dept. Maths, Univ. Surrey.

R. D. Knott & P. J. Krause [1988b], "LIBRARY SYSTEM: an example of rapid prototyping of a Z specification in Prolog" Report A1.2, Alvey Project SE/065, Dept. Maths, Univ. Surrey.

P. J. Krause, R. D. Knott & P. J. Byers [1989], "Lift System Animation" Report A2.1, Alvey Project SE/065, Dept. Maths, Univ. Surrey.

J. W. Lloyd [1984] "Foundations of Logic Programming", Springer Verlag

G. A. Ringwood [1989], "A Comparative Study of Concurrent Logic Languages", Knowledge Engineering Review, 4 (to appear)

J. M. Spivey [1988], "Understanding Z: a Specification Language and its Formal Semantics" Cambridge Univ. Press

J. M. Spivey [1989], "The Z Notation: a Reference Manual", Prentice Hall

S. Stepney & S. P. Lord [1987], "Formal Specification of an Access Control System", Software Practice and Experience, 17, pp. 575-593

L. Stirling & E. Shapiro [1986], "The Art of Prolog", MIT Press

From Jan1990

9-11/January/90 Refinement workshop @ IBM Hursley

Spring 1990 Functional Languages @ Herriot Watt

May 1990 Annual General Meeting

23-25 July 1990 - Semantics for Concurrency @ University of Leicester

Summer 1990 - LOTOS

December 1990 - 1990 Christmas workshop

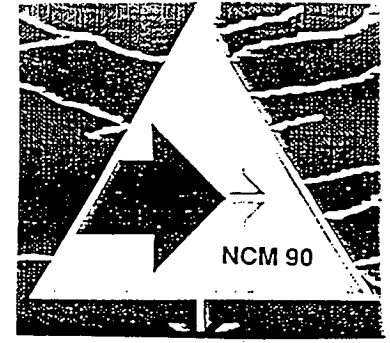
Topics under investigation for future meetings:
ERIL/AXIS etc
Topology in Computer Science

Other Meetings of potential/likely interest to FACS members

ICALP 90 - 16-20 July 1990 @ University of Warwick

TAPSOFT 91 - Easter 1991 @ Imperial College

British Neural Network Society



Neural Computing Meeting 90

Queen Elizabeth Hall, Kensington, London
18-20 April,1990

DAY1. Guest speaker: Professor S. Grossberg.(Boston,USA).
"Neural Nets for Learning and Memory of Cognitive Recognition Codes"

Tutorials:
Dynamical systems and Neural Networks.(Prof. J.G. Taylor).
Neuronal Modelling.(Dr. C.L.T. Mannion).
Kohonen Maps.(Dr. N. Allinson)
Hardware.(Dr. S. Garth)

DAY2. Contributed Papers (Abstracts by 20/3/90)

DAY3. Guest Speakers
Prof T. Kohonen .(Helsinki, Finland).
"The Self Organising Map".

Prof. K.E. Kurten.(Koln,FDR).
"Quasi-Optimized learning dynamics in sparsely connected Neural Network Models".

Dr. A. Mayes.(Manchester,U.K.).
"Nature of the Functional loss in Amnesia: Possible Role for a Highly Structured Neural Network".

Prof. C. Blakemore/Dr. A. Larkman. (Oxford,U.K)
"Real Neurons"

Prof. R. Cotterill(Copenhagen,Denmark).
(Title to be Announced)

Contact
Professor J.G.Taylor, Mathematics Dept., King's College, London WC2R 2LS.
Phone:01-873-2213; fax:01-836-1799; e-mail:udah057@uk.ac.kcl.cc.ash.
Dr. C.L.T. Mannion, Dept. of Electrical Engineering, University of Surrey, Guildford, Surrey GU2 5XH
Phone:0483-571281 X2304; fax:0483-34139; e-mail:conal@cs.surrey.ac.uk

BCS The Society of Information Systems Engineering

Patron: HRH The Duke of Kent KG GCMG GCVO ADC

CYBERNETICS MACHINE GROUP

30 NOVEMBER 1989

NEWSHEET

The REMIT of the CYBERNETIC MACHINE GROUP is COMPUTATION AS A PHYSICAL THEORY.

In this issue, the Chairman gives an analysis of progress in the field over the past three years. A redefinition of aims and activities is proposed - a sharper focus by means of which even greater progress can be made.

It is concluded that the revised remit of the Group can be nothing less than the discovery of the physical laws by means of which natural information processing, knowledge engineering and intelligence take place.

It is seen as significant that in the year when the Group's chosen theme is the nature of information, that the Journal Nature should publish W.H. Zurek's discovery of the information metric and that David Deutsch, whose original Royal Society paper on the Quantum Theory of Computation marked the inauguration of the Group, should publish another such paper on Quantum Computational Networks. A short paper by another of the Group's previous speaker Ian White is also reproduced, which focuses on the final frontier of computer science where the physical limitations of computation and complexity hold.

WHAT is INFORMATION? WHAT is KNOWLEDGE? WHAT is INTELLIGENCE?

These words are in commonplace use, but there are no agreed definitions. WHAT are the CRITERIA for SAFETY CRITICALITY?, for the efficient modelling Byzantine complexity in REAL TIME?, for THERMODYNAMICALLY OPTIMAL computation?, for PERCEPTION?, for COGNITION?

These criteria remain largely unfathomed by computer science, yet natural selection has established long ago physical mechanisms appropriate to them that are

The British Computer Society 13 Mansfield Street London W1M 0BP Telephone: 01-637 0471
Telex: 262284 Mono Ref: 1828 Facsimile: 01-631 1049
Founded: 1957 Incorporated by Royal Charter 1984

evolutionary triumphs of miniaturization. The machinery for their definition therefore already exists! It is not astonishing therefore that the study of such machinery is not a standard part of the computer science curriculum?

This year's theme is The Nature of Information.

Already in November, Gordon Scarrott FENG talked on this topic on which he published a paper with the title, "The Nature of Information" in a recent issue of the Computer Journal 32,3,1989.

In February Professor Tom Stonier of the University of Bradford will continue the theme with a talk entitled,

"Information: Has it reality or is it merely thought?"

This will take place as a joint meeting with the Cybernetics Society on Tuesday, February 6th at Birbeck College, rooms C/D at 1800.

Professor Stonier's book entitled "Information and the Internal Structure of the Universe" will be published in Springer-Verlag in the Spring, and his paper entitled "Towards a general theory of information II: Information and entropy" appears in Aslib.Proc. 41, 2, 41-55, Feb 1989.

A speaker on the same theme is being approached to speak at the AGM in May.

The programme of forthcoming meetings of the Cybernetics Society is published separately.

Recent references from the Chairman's file:

W.H. Zurek, Nature 341, 119-124, 14th September 1989 "Thermodynamic cost of computation, algorithmic complexity and the information metric"

Zurek's discovery of the information metric is seen as highly significant and Ian White has kindly supplied a comprehensive list of G.J. Chaitin's work on Algorithmic Complexity and Randomness to accompany it. However, the Chairman believes that Zurek's paper contains a highly significant invalid conclusion P.119, ".... the optimum thermodynamic efficiency is difficult to attain: the minimal programs needed to guarantee it cannot be found systematically because of the

undecidability of the halting problem - a Turing machine version of Godel's theorem". In fact the algorithms and similarly the minimal programs are known to be countable, and hence can be systematically ordered. The means for this is the information metric itself. It follows reductio ad absurdum that therefore the undecidability of the halting problem can be overcome! The Chairman would welcome correspondence on this issue. That thermodynamic optimal efficiency can be achieved systematically is shown by N. Sourlas, Nature 339, 693-695, 29th June 1989, "Spin - glass models as error-correcting codes" and B.E.P. Clement, Nature 340, 514, 17th August 1989, "Spin glass in a Whirl". Both papers show that codes are known that saturate Shannon's well known cost-performance bounds for information transfer.

Some insight into the machinery of the brain is provided by Sir John Eccles, "Do mental events cause neural events analogously to the probability fields of quantum mechanics?" Proc.Roy.Soc.Lond B227, 411-428, 1986, and by William C. Hoffman, "The Visual Cortex is a Contact Bundle" Applied Mathematics and Computation, 32, 137-167, 1989; into the eye itself, Richard H. Masland, "The functional architecture of the retina" Scientific American, 255, 6, 90-99, December 1987, and into the ear, A.J. Hudspeth, "How the ear works" Nature, 341, 397-404, 5th October 1989. The Journal Science published a series of articles on the Frontiers of Neuroscience in Vol.242, 4879, 641-745, 4th November 1988. Francis Crick specifically comments on the difference between natural and synthetic neural nets in Nature 337, 129-132, 12th January 1989, "The recent excitement about neural networks".

Both G.J. Chaitin see reference in Application to Biology, and W.C. Hoffman, "A system of axioms for mathematical biology" Mathematical Biosciences 16; 11-29, 1973, have proposed a mathematical basis for the description of living systems and the mechanism by which they might work, and further essays into the frontiers of computation, are given in, "The Universal Turing Machine - A Half Century Survey" Rolf Hertken (ed) Oxford University Press, 1988.

There has been a recent spate of books, concerning quantum physics and mind, "The Emperor's New Mind: Concerning Computers, Minds and the Laws of Physics" Roger Penrose, Oxford University Press, 1989. Mind, Brain and the Quantum: The Compound 'I'. Michael Lockwood, Blackwell, 1989; No Ghost in the Machine: Modern Science and the Brain, the Mind, and the Soul. Rodney Cotterill, Heinemann 1989; The Quantum Self, Danah Zohar, Bloomsbury, 1990.

All these books turn on the debate now taking place in quantum physics between the traditionalists of the Bohr school, who believe that quantum theory is merely a mathematical subterfuge for calculating the consequences of experiments at the level of the quantum, and those of the de Broglie, Bohm school who believe that the probability fields of quantum theory described actual "subquantum" phenomena taking place in the vacuum which is only indirectly measurably as in for example the Lamb shift of atomic spectra, the Aharonov-Bohm effect and the Berry phase, see "How to send electrons round the twist" Nick Herbert New Scientist, 11th June 1987, 37-40, and "Spin, and non-locality in quantum mechanics" C. Dewdney et al, Nature 336, 536-544, 8th December 1988. If the de Broglie Bohm viewpoint prevails then there is no reason to reject the postulate that the Mind might be a self-organising wave construct taking place in the vacuum by means of which in living systems brains may be controlled. In the GSLT cosmology cited in the Chairman's analysis which follows, the Wheeler Meaning circuit constitutes a basis for the mind/brain model by means of Huygens' principle of secondary sources. All creative acts as secondary sources, and their mechanisms mind/brains are then related by means of evolution the creative process, to their source the Creation through the unified field.

The physical basis for this statement is provided in the paper, G. Resconi, P.J. Marcer, Physics Letters A 125, 6/7, 282-290, 23rd November 1987, "A novel representation of quantum cybernetics using Lie algebras"

Other papers in this area and quantum computation are, D. Deutsch, "Quantum computational networks", Proc.Roy.Soc.Lond. A425, 73-90, 1989.

B.J. Hiley, "Cosmology, EPR Correlations and separability" Bell's theorem, Quantu Theory and Conceptions of the Universe, 181-190, Kluwer, 1989 and "Vacuum or Holomovement" preprint - Dept. of Physics, Birbeck College, University of London, London WC1E 7HX.

An analysis of the Groups REMIT.

THE REMIT OF THE CYBERNETIC MACHINE GROUP is COMPUTATION AS A PHYSICAL THEORY.

Since the Group's establishment, it has become apparent that its remit looks beyond any of the man-made information or neural processing systems we understand. These systems mimic brains only in certain very limited conceptual particulars.

Brains give living systems the ability to efficiently simulate environments of Byzantine complexity for the purposes of perception, cognition and survival; by processes as yet unfathomed by computer science and artificial intelligence. Natural selection, the physical process of incremental evolution over time and space, has established the blueprints for brains in the genomes of the genetic code. Natural selection favours the establishment of optimal mechanisms and pathways; it has proceeded from the micro to the macroscopic. Can there be any doubt therefore that the microscopic mechanisms for information processing at work in molecular biology (for example in the genetic code itself) apply equally to brains, and that living systems operate at the final frontier of computer science where the physical limitations of computation and complexity hold? The unique ability of the eye and the ear to detect individual quanta or molecules respectively supports this hypothesis; both the eye and the ear are evolutionary triumphs of miniaturization beyond any conceivable man-made technology. The focus of the Group can therefore be nothing less than an understanding of how such natural information processing is achieved. Nature has thrown down the gauntlet and computer science must pick it up - such understanding addresses all the fundamental unsolved problems at the heart of computer science. Little progress can therefore be made without it.

Neurophysiology and neuropsychology furnish the experimentally validated transformational systems for information processing and intelligence. New mathematical models can surely furnish explanations of why brains are constructed the way they are and why therefore natural neural information processing takes place the way it does. Such models will provide the trustworthy conceptual foundations that computer science now lacks by explaining what physically constitutes the nature of information and of knowledge. Without these trustworthy conceptual foundations, information processing, knowledge engineering and aritificial intelligence will remain artforms and cannot become hard science.

Can it be mere coincidence that brains exercise their computational abilities through both graded and switching mechanisms, i.e. analogue-digitally through 'thermodynamic' and 'computational' degrees of freedom respectively?

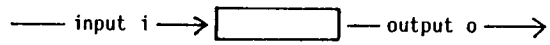
In a real-time control it is essential to iterate to problem or near problem solutions as rapidly as possible but in highly complex non-linear problems corresponding to most real systems, only analogue processes capture this computational ability. Similarly, in safety-critical systems, if the system can be shown to operate safely between extremes A and B, a control calculation must always return that control to this interval. Once again it is known that digital computational control systems do not guarantee the continuous behaviour that this criterion requires. Thus brains utilize their 'thermodynamic' degrees of freedom to reach these objectives, in the same way conceptually artificial neural nets do.

On the other hand the benefits that accrue to digital computation through programmability are well known and beyond dispute; yet brains as far as we know only simulate this ability. One must therefore postulate that natural languages as used by brains, represent the fundamental analogue-digital nature of their working. This concerns the boundary where the measure of information with respect to the 'computational' degrees of freedom conceived symbolically in terms of Os and Is for example, may be equivalanced with the

measure with respect to the 'thermodynamic' degrees conceived in terms of negentropy and Shannon's well known bounds for the transmission of information. One may postulate that a particular encoding of this natural language is the genetic code. A more general computational machinery than at present must therefore be envisaged which operates according to the full capabilities offered by physical laws. These laws concern the relationship between the continuous or analogue and the discrete or digital, and the interrelationship between energy, entropy and information, where information now becomes as important to our understanding of physics as our current understanding of energy.

It is not difficult to envisage the benefits that an understanding of the physical nature of information will bring, if it can be put on a par with that of our understanding of energy from which our knowledge of the physical sciences over the past centuries clearly flows. The physical sciences will be generalized allowing us to more fully comprehend how matter is organised in both equilibrium and non-equilibrium situations especially in living systems. It should also furnish us with answers to the Wheeler questions - "Why does quantum theory have to be?", "Why is universal Turing computation physically possible?" and "Why does constructive mathematics the theoretical methodology of science work?", and to the Wheeler Meaning Circuit - of how physical laws have allowed and facilitated the construction of observer/participators able to simulate or model by means of an incorporated brain, the meaning of the observed. Similarly, the knowledge of the relationship between energy, entropy and what it is that constitutes information, should explain the nature of the relationship between the reversible processes of physics, its dynamics and irreversibility and dissipativity which characterize at the macroscopic level its thermodynamics.

This relationship between energy, entropy and information may have been recently annunciated by W.H. Zurek in his paper "The thermodynamic cost of computation, algorithmic complexity and the information metric", Nature 341, 119-124, 14th September 1989, where it is shown that with respect to digital computation, dissipativity constitutes an information metric in terms of increasing algorithmic complexity. Zurek's derivation concerns a model of replacement computation by which an input i is replaced by an output o. Such a model presented categorically as



is the basis of the General System Theory of Mesarovic and Takahara, and a generalization of it, the General System Logical Theory (GSLT) of Resconi and Jessel. Such theories of input/output constitute universal models because of the universality of Turing computation, and thus are able to generate all the semantic representations of the theories they represent, and such semantic models are entirely appropriate to an abstract theory of knowledge engineering in accordance with Zurek's discovery that such categorical arrows constitute an information metric. It is equally appropriate that Jessel and Resconi have shown that the atomic categorical diagram fundamental to GSLT is appropriate to the representation of Huygens' principle of secondary sources. In GSLT therefore Huygens' principle constitutes a universal model of wavefield physics, i.e. one capable of generating all representations which Zurek's discovery of the information metric shows, should all have physical validity. This postulate receives full support from Jessel's original derivation in which by using a Heaviside-like operator to obtain the density of Huygens' secondary sources as a commutator of two operators acting on the wavefield to be analysed (the atom of GSLT is a commutator diagram) it is possible to obtain all the results usually achieved by means of integral formulas and Green's function identities. It is fully in accord with the conception of Huygens' Principle of secondary sources for the propagation of wavefields as a 'Gedankenexperiment', i.e., an experiment of pure reason needing no

experimental validation.

Recent mathematical research reveals that such universal models of a theory T in a language L of which the universes are sets, determine a surreal number field. Such number fields may be generated entirely recursively, and this construction gives them many remarkable properties. In particular each such field is, a complete binary tree; has a canonical power series structure so that computation within the field is finite and polynomially complex; mimics the arithmetic continua, and has corresponding Euclidean, hyperbolic and elliptic geometric counterparts. In particular the surreal field appropriate to the universal model GSLT concerns the field On_2 which J.H. Conway shows may be defined lexicographically, and is constructed in the simplest possible way and so may be computed in the smallest number of steps. On_2 also mimics all the properties of the number field No which describes all numbers great and small; ordinary, infinitesimal and transfinite. It therefore spans the whole of number, is computationally optimum, and is the source for example of binary Gray codes and Hamiltonian paths. Surreal number fields possess a unique birthorder automorphism among their many field automorphisms, where the birthorder specifies the height of an element in their binary tree. Such a unique birthorder automorphism appropriate to the otherwise reversible processes of wavefields in GSLT by means of On_2 provides an entirely satisfactory explanation of the nature of the Second Law as uniquely in On_2, its complexity a global property provides the only means to assess birthorder, and so natural number constitutes the implicit and only absolute measure on On_2, all other measures being relative. Zurek's information metric and GSLT explain why quantum theory has to be, because it is one among many of the physically valid representations of wavefields. Similarly universal Turing computation is physically possible because replacement computation is a universal submodel of GSLT; the information metric represented by GSLT categorical arrows explains why any system in GSLT where all such systems are physically valid systems has in principle a mathematical representation, although in practice these may be beyond any resources to compute, except by the Cosmos of which GSLT constitutes the universal model.

In such a Cosmos, the preferred mode of computation is semantic computation or modelling since it is naturally polynomially complex in contrast to Turing computation which is naturally exponentially complex. Semantic computation or knowledge engineering appropriate to On_2, computation in a minimum number of steps, represents the natural minima in the domain of all active or relational processes, and thus in terms of dissipativity which constitutes the information metric is the thermodynamically optimum. In other words the Wheeler meaning circuit is a natural consequence of the physical stability criteria operating in the universal model. Further, while the model is relational and no assumptions are therefore made as to domain or range of the functionality of its representations, its natural symmetries are appropriate to a 3+1 space-time and to Lorentz invariance. Huygens' Gedankenexperiment therefore contains that of Einstein for special relativity, and Resconi has shown for General relativity also. In a relational model however, General Relativity specifies only the domain bounds when the range of wavephysics is held invariant, and thus quantum theory may specify the range bounds when the domain is similarly held invariant, without any inconsistency. The universal model of wavefields therefore possesses the fundamental attributes for a cosmology of the universe we observe.

Additionally, the model indicates that this cosmology is closed (On_2 is a field or proper class), evolves for ever (the birthorder process for the elements of On_2 is denumerable without end), and generates itself from nothing (the birthorder process begins as a second order process by means of which the empty set is compared with itself). The Jessel Heaviside-like operator or corresponding Green's function specifying this initial comparison is thus implicitly a manifold or surface of secondary sources of a wavefield where the natural domain symmetry must be $t - r/c$ where t is the continuous variable corresponding to the birthorder, c is a universal constant and $r = \sqrt{x^2 + y^2 + z^2}$. It is therefore possible to postulate that a 3+1 space-time constitutes the only stable stationary domain basis for this cosmology. It follows similarly from requirements for finiteness intrinsic to the model that the usual inverse square laws quantify the range basis. It therefore appears that the electromagnetic and gravitational fields associated with these inverse square laws constitute a description of the complementary range and domain phenomena of the fundamental wavephysics of this universal model, i.e. the range and domain aspects of the so-called unified field, and thus range-domain and domain-range phenomena will correspond to the so-called strong and weak fields.

It is therefore proposed that the Group should concentrate its activities firstly on the focus provided in the scenario presented in the first part of this analysis, and where it can be agreed, on understanding the universal model outlined in the second part.