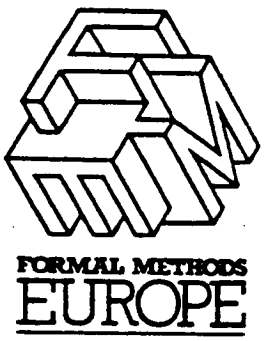

FACS Europe

The Newsletter of the BCS Formal Aspects of Computing Science Special Interest Group and Formal Methods Europe.

Series I Vol. 3, No. 1, Summer 1996 ISSN 1361-3103

Contents

Editorial 2
CS Research and the 'Theory Vertical', AGM Talk 7
VDM Column—Examples Repository 10
Functional Programming Column 14
Book Reviews 16
Northern Formal Methods Workshop 22

1 Editorial

I'm writing this soon after a very successful and enjoyable Refinement Workshop (there will be a report in the next issue); congratulations to He Jifeng, and a good omen for our two other events coming up - FAHCI in Sheffield, and FM North in Bradford, one at each end of September. We also have advance notice of FME '97 (see below).

This year has been pretty good for events; in a bid to keep this up, the FACS committee are currently having a close look at how to run FACS events, and what to plan for future ones. If you have any suggestions or comments, then please email/speak to John Cooke so that they can be taken into account in the deliberations. Events currently being planned include a B workshop, and a workshop on roles for formal models of systems and processes in large-scale systems development. If you want to contribute to the development of either of these events, please contact Kevin Lano (B workshop) or me (large-scale development).

Looking a little further back, the FACS AGM duly happened, and we had a thought-provoking talk from Roy Crole, which he has written up for this issue. Subscriptions have been revised for FACS, and for EATCS - see FACS web pages for full information.

Contributions to the Newsletter on any relevant topic are welcome. Please send them electronically if possible, in (plain) LaTeX or TeX form, or postscript, if you can; next best is plain ASCII. Otherwise please send A4 copy fit to reproduce by fast photocopying (i.e. no paste-ups), with 300dpi laserprint or equivalent a minimum standard.

FACS/FME Newsletter
c/o Ann M Wrightson
School of Computing and Mathematics
University of Huddersfield
Queensgate
Huddersfield
HD1 3DH
UK

Contributions express the opinions of contributors, not of FACS, FME or any other organization with which they are associated (unless they say otherwise!). Letters are welcome and should be sent to the Editor.

Advertisements are welcome, as full or half page printed ads, or as inserts (i.e. loose sheets or booklets mailed with the Newsletter). Advertisements and inserts will only be accepted where they are clearly of specific interest to the FACS/FME community. Please contact the editor for current rates and due dates for copy.

2 Events

2.1 The 13th BRITISH COLLOQUIUM FOR THEORETICAL COMPUTER SCIENCE

University of Sheffield, from Monday March 24th to Wednesday March 26th, 1997.

The meeting will take place in Halifax Hall, a former steel baron's mansion and University Hall of Residence which has been newly refurbished, and which is located in pleasant surroundings (once referred to by John Betjeman as the "prettiest suburb in England") some 10 minutes walk from the main campus and 20 minutes from the city centre. Local organisers are Mike Holcombe (m.holcombe@dcs.shef.ac.uk) and Matt Fairtlough (matt@dcs.shef.ac.uk).

See also: *http://www.dcs.shef.ac.uk/ wmlh/BCTCS13.html*

2.2 FME'97: Formal Methods: Their Industrial Application and Strengthened Foundations.

The Technical University of Graz will host the fourth FME Symposium from 15 to 19 September 1997. This will be the successor of six previous VDM and FME symposia, and again will bring together users, researchers and developers of precise mathematical methods for software development. Contact Peter Lucas, lucas@ist.tu-graz.ac.at, for details - at time of going to press, we hadn't managed to print their call for papers, but if Roger managed it at Loughborough it will be there as an insert... (Who said postscript was a common language?)

2.3 BCS-FACS Xmas Workshop 1996

This is a joint event with the BCS Requirements Engineering SIG, focussing on challenges and synergies currently felt between Formal Methods and Requirements Engineering. There really will be full details in the *next* newsletter issue (sorry it didn't make this one (Ed)); for now please note that 16-17 December, at City University, will be the place to be for Xmas 96.

2.4 FAHCI Preliminary Programme

FAHCI, a BCS-FACS workshop on Formal Aspects of the Human Computer Interface, is to be held on 10th-12th September, 1996, at Sheffield Hallam University. Here is the preliminary programme; if you have not yet decided to go, maybe this will convince you...

2.4.1 Invited speakers

Prof. Michael Harrison (HCI Group, University of York) and Prof. Keith Stenning (HCRC, Edinburgh University)

2.4.2 Panels

"Informality versus Formality" and "HCI Challenges" to be confirmed

2.4.3 Interface Architectures and Control

"A Modelling the MVC and PAC architectures with Object-Z" by A. Hussey and D. Carrington (University of Queensland, Australia)

"On the Composition of Interactor Specifications" by P. Markopoulus, J. Rowson and P. Johnson (Queen Mary and Westfield College, UK)

"Context Sensitive User Interfaces" by J. Cressiac-Campos and F. Mario Martins (University of Minho, Portugal)

2.4.4 Helping Inform the Designer

"Dialogue Graphs - A Formal and Visual Specification Technique for Dialogue Modelling" by E. Schlungbaum and T. Elwert (University of Rostock, Germany)

"A Design Life-Cycle for the Formal Design of User Interfaces" P. Palanque and R. Bastide (Toulouse University, France)

"Verification of Properties of Human Computer Dialogues with an Infinite Number of States" by M. Mezzanotte and F. Paterno (CNUCE - CNR, Pisa, Italy)

2.4.5 Processes and Pitfalls

"Formally Assessing Software Modifiability" by C. Roast and J. Siddiqi (Sheffield Hallam University, UK)

"A Modal Model of Versions" by A. Dix, T. Rodden and I. Sommerville (University of Huddersfield and Lancaster University, UK)

"A Formal Basis for Modelling Process and Task Managements Aspects of Human-Computer Interface Design" by P. A. Lindsay (University of Queensland, Australia)

2.4.6 Short Papers

"Using Temporal Logic in the Specification of Reactive and Interactive Systems" by R. J. Butterworth and D. J. Cooke (Loughborough University of Technology, UK)

"Employing Queueing Modelling in the Domain of Intelligent User Interfaces" by C. Karagiannidis, A. Koumpis, C. Stehanidis and A. Georgiou (University of Macedonia, Greece)

"An Analysis of Behaviour in Interactive Auditory Field" by T. Ebina and S. Igi (Ministry of Telecommunciations, Japan)

3 Electronic Information Sources

Any selection of electronic resources in a space bounded by newsletter pages is a small one. I hope these are useful—more will follow in future issues. If you have a favourite not yet included, then please email me its details, plus a paragraph or so of description. (Please don't just send me a list of URLs!) I will moderate contributions for relevance to the FACS/FME community.

Thanks to Dan Simpson, Roger Stone, and Philip Wadler for contributions.

3.1 COPAC, the new national OPAC

http://copac.ac.uk/copac/

COPAC, the new national OPAC, is based on the union catalogue database of the Consortium of University Research Libraries (CURL). The development has been funded by the Joint Information Systems Committee (JISC). COPAC currently gives access to the combined university library catalogues of Cambridge, Edinburgh, Glasgow, Leeds and Oxford. The catalogues of a further ten CURL member libraries will be added over the coming months. The service is still at an early stage of development, but provides search, retrieve, display and download facilities via two interfaces, text and WWW. Access to COPAC is free of charge.

Postscript versions of the User Guides for both interfaces are available for downloading at: *http://copac.ac.uk/copac/userguide/*

A WWW page giving more information about COPAC is at URL *http://copac.ac.uk/copacinfo/*

3.2 Functional programming FAQ

http://www.cs.nott.ac.uk/Department/Staff/mpj/faq.html

Mark Jones maintains this excellent functional programming FAQ. It answers questions ranging from basic ('Are there any books about functional programming?') to not-so-basic ('What is a monad?'). It includes links to sites describing the following languages: ASpecT, Caml, Clean, Erlang, FP, Gofer, Haskell, Hope, Hugs, Id, IFP, J, Miranda(TM), ML, NESL, OPAL, Oz, Scheme, and Sisal. It also has links to bibliographies, meetings, active research groups, and other resources.

3.3 The functional programming newsgroup

news:comp.lang.functional

This group is unmoderated, and is a mixed bag. The best thing about it is its FAQ, cited above. Otherwise, much dross with a few gems. Discussion largely consists of novice questions, sometimes answered ineptly by other novices, sometimes answered superbly by experts. There is the occasional flamewar, and, rarely, a novel insight. Post here if you have a question not answered by the FAQ; otherwise, it may be better to give it a miss.

3.4 Functional programming conference list

http://www.dcs.qmw.ac.uk/research/theory/department/FP_conferences/conf.html

The Queen Mary and Westfield CS department run this list. They allow anyone to make an entry via a form, and as a result some entires are untimely or unrelated to the topic. The list of meetings in the FAQ, cited above, may be a better bet.

3.5 The functional programming archive

http://www.lpac.ac.uk/SEL-HPC/Articles/FuncArchive.html

Jon Hill maintains this handy archive. Researchers can register articles, which are listed with full bibliographic information, keywords, and links to the paper proper. Papers are grouped by primary and secondary topics, so one can browse for related papers. Researchers can also register themselves: the list of home pages has 200 entries, including all the leading lights of the field.

3.6 The Journal of Functional Programming

http://www.dcs.gla.ac.uk/jfp/

The JFP web site contains data on how to subscribe or submit, and a complete list of every paper published in JFP. The entries are currently being linked into the Hypertext Bibliography Project, see below. The JFP bibliography entries are incomplete as of this writing (no citations yet, and abstracts only for the last three years), but give it time.

3.7 The Hypertext Bibliography Project

http://theory.lcs.mit.edu/ dmjones/hbp/

Originated by Albert Meyer and David Jones for *Information and Computation*, this remarkable database has expanded to include a number of titles. It has a scope far beyond mere functional programming, but indexed titles include the *International Conference on Functional Programming* and the *Journal of Functional Programming*. Full entries contain title and abstract of the paper, plus a list of all papers in the database that this one cites or (here's the kicker) that cite this one. All papers and authors are cross-linked in true hyper-text fashion. Currently only a few titles contain complete entries, but as time passes this should grow into an indispensable reference.

3.8 Real-world applications of functional programming

http://www.dcs.gla.ac.uk/fp/realworld/

This is a list of programs written primarily to get a task done, rather than just to experiment with functional languages. Some applications are incestuous (compilers for Haskell and SML), some are industrial wonders (Erlang is used to build phone switches), some are in expected domains (theorem provers and natural language processors), and others are refreshingly out in left field (the MC-SYM tool for molecular biologists, the Cherry chess processor, the Pittsburgh map and restaurant data base). Initiated by myself and Andy Gill, the list is now maintained by Jonathon Hogg. It's main drawback is that it is woefully incomplete and short. Please add your contribution!

3.9 The Petri Net Web Site

http://www.daimi.aau.dk/PetriNets

This site contains a lot of information on Petri Nets, a bibliography, examples, info on the DesignCPN tool, links to other tool lists, links to research groups, info on the Aarhus CPN group and all the usual web stuff.

3.10 The Petri Net Mailing List

PetriNets@daimi.aau.dk

A moderated mailing list. Often used to ask questions, announce information available and get advice. All messages get a good response and it is read by most major workers in the field.

3.11 Irish Formal Methods Special Interest Group

http://www.cs.tcd.ie/www/jgllgher/ifmsig/index.html

A new-ish focal point for Formal Methods, with a distinctly Irish flavour. It is a collection point for links relevent to formal methods and theoretical computer science in Ireland as well as being a notice board for the activities of the SIG.

3.12 WISDEN project

http://cs-fm.lboro.ac.uk/wisden and *http://shu.ac.uk/schools/cms/wisden/wisden.htm*

No, not cricket - WISDEN stands for Wide-ranging Integrated Software Design Education Network, and includes a solid element of formal aspects for software engineering education.

3.13 CTI/Formal Methods & Discrete Maths

http://www.ulst.ac.uk/misc/cticomp/subject/formeth.html

Another raft of teaching stuff, this time under the 'computers in teaching initiative' umbrella.

3.14 FACS WWW Home page

http://cs-fm.lboro.ac.uk/facs/

We couldn't miss this one out—boring-but-useful, with current subscription rates, events, etc etc.

3.15 WWW Virtual Library: Formal Methods

http://www.comlab.ox.ac.uk/archive/formal-methods.html

Another boring-but-(very)-useful page—the page for Formal Methods within the 'Virtual Library' Web subject index. Provides links to a good selection of relevant sites and resources for formal aspects. This is a good place to link your own pages to - it would be good to see more European links.

The Virtual Library is also an interesting resource in itself, with stuff of interest to FACS/FME folk also in subjects 'Software Engineering' and 'Safety-Critical Systems' .

3.16 Hypatia Electronic Library

http://hypatia.dcs.qmw.ac.uk

Possibly the most comprehensive single resource of computer science and pure mathematics on the Web - Paul Taylor, now at QMW, has surpassed his former efforts at Imperial. Contains bibliographies, a directory of researchers and research groups, and links to oodles of papers on sites in a number of countries. You need to know what you're looking for - by author, or by research group, since no other searches are provided. There is also a route provided to link in your own resources (ftp-able papers) automatically - once in, they are automatically kept up to date on Hypatia, for deletions as well as additions.

The Computing Science Research Strategy Group and the

Theory Vertical

Roy L. Crole

July 25, 1996

In the first half of this article there is a brief description of the structure of the Research Strategy Group (RSG) and the RSG's aims and objectives, and a report on the ways in which RSG is providing a forum for the discussion and development of Computing Science. The second half is about the RSG's concept of a *Theory Vertical*, and outlines some recent research in programming semantics.

We begin with a short history of the Research Strategy Group. The original idea was, in part, to provide a framework for the formulation and discussion of EPSRC directed programmes. Such a framework would be established and coordinated by a small sub-committee of the Conference of Professors and Heads of Computing (CPHC). An initial membership of eight was chosen by the CPHC, and seven more members were selected based on nominations and reports from the research community at large. The current membership consists of active researchers in Computing Science, both academic and industrial.

The first meeting took place in Glasgow, during February 1996. At this initial meeting, the RSG aimed to establish what its precise role should be, though providing a forum through which EPSRC directed programmes could be formulated was considered a central policy. The unifying concept was that the group should help to coordinate, in an impartial way, the planning, debate, and organisation of U.K. Computing Research in general. The RSG would aim to represent and advocate all views on Computing Science research, from all areas of the community, and would act as a "centralised" partner to other research organisations.

The current aims of the group are:

- to stimulate and participate in a debate about the nature and research agenda of Computing Science, within both academia and industry. In particular the group will provide a forum for discussion, for example by the use of electronic mailing lists;
- to provide a platform for the development of *Research Acorns* which are initial sketches of research programmes. Once fully grown they can be put forward to funding bodies such as the EPSRC;
- to provide workshops dedicated to cross-communication of ideas—RSG can actively target individuals;
- to encourage and support researchers, especially those who are isolated or at an early stage in their career;
- to help improve the interface between the user community (broadly conceived) and computing science researchers; and
- to help improve the public image of computing science research, for example by establishing an annual public lecture.

We held a Research Strategy Workshop on 16th July 1996 in Bristol at which many of the current *Research Acorns* were discussed. Details will be posted on the Web in due course; indeed if you would like any other details about what the RSG is currently providing for the research community, then please see our home-page at `http://www.dcs.gla.ac.uk/uk-cs-research`. One of the primary uses of our Web site is to provide an electronic discussion forum for all aspects of Computing Science research. You can:

- join the uk-cs-research mailing list. This list is used only for announcements to alert you to fuller information available elsewhere. You are guaranteed not to be saturated in email by this list. Mail sent to the list is discarded;
- join the uk-cs-research-open mailing list. This mailing list is an un-moderated forum for open discussion; mail sent to it is automatically broadcast to everyone;
- read the Web pages for the Research Acorns, and join their mailing lists; and
- submit a Research Acorn of your own to Malcolm Atkinson (mpa@dcs.gla.ac.uk).

The second part of this article moves from a general discussion of the RSG to the rather more specific topic of how the group can help to promote the use of ideas from Theoretical Computer Science throughout other areas of computing. From our discussions about the nature of Computing Science research, the ways in which certain areas are developing, and the possibilities of encouraging greater cross-disciplinary communication, we arrived at a matrix picture of Computer Science:

	Semantics	Networks	etc
Biological Computing	⋮	⋮	⋮
Functional Programming	⋮	⋮	⋮
Imperative Programming	⋮	⋮	⋮
Data Mining	⋮	⋮	⋮
etc	⋮	⋮	⋮

Thus we have, for example, a *Theory Vertical(s)* arising from potential applications to other *Horizontal* subject topics. Of course, this model is very much simplified, but it has been useful in general discussions about the nature and direction of Computing Research.

Perhaps one of the most significant ways in which theory can have an impact on apparently disparate subjects is through the development of both *hierarchical* and *unifying* theories of computation. The following table illustrates what we intend by the former

Computing	**Physics**	**Mathematics**
Implementation	Atomic level	Set Theory
⋮	⋮	⋮
Abstract Models	Macro level	Category Theory

and the modern notation for specifying the operational semantics of programming languages from the level of global executions to implementation models is a nice example of a hierarchical theory.

We shall finish this article by giving a few details of recent progress in *unifying* programming language semantics. In particular, this research concerns the unification of *programming paradigms*. Some examples are functional programming, concurrent programming, imperative programming, object-oriented programming, and so on. One way of envisaging the *theory vertical* for these topics is to see if there is a framework in which the semantics of languages associated with these paradigms can be presented uniformly. Research by a number of people over the past few years suggests that the notion of bisimulation, which is most commonly associated with formalisms for describing concurrency, can uniformly express the operational semantics for the kinds of languages we have mentioned. If P_1 and P_2 are two processes, and α is an action which a process can perform, then

writing $\alpha.P_1 \mid P_2 \xrightarrow{\alpha} P_1 \mid P_2$ expresses one possible action which the parallel process $\alpha.P_1 \mid P_2$ can perform. In general, we can say that the notation $P \xrightarrow{\alpha} P'$ has the intended meaning that a process (or program) P can undergo an action α and become a process (or program) P'. Of course, those familiar with elementary concurrency will know that there is a very rigorous theory of coinduction which supports this notation—and in fact we can shift the essence of this theory to other programming paradigms.

The key idea here is that a similar notation (of labelled transition systems) can be used to express the operational semantics of other programming paradigms. For example, $\lambda x.x*3 \xrightarrow{@2} 2*3$ expresses that the functional program $\lambda x.x*3$ can undergo the action of being applied to 2, resulting in the expression $2*3$. And $\texttt{read} \xrightarrow{?n} \underline{n}$ expresses that the command `read` can undergo the action of reading in the integer n and returning $\underline{n}$. Using labelled transition systems, we can then formulate a precise notion of program equivalence: programs P_1 and P_2 are *bisimilar*, denoted $P_1 \simeq P_2$, just in case

$$\forall a, P_1, P_2.(P_1 \xrightarrow{\alpha} P_1' \implies P_2 \xrightarrow{\alpha} P_2' \wedge P_1' \simeq P_2') \wedge (P_2 \xrightarrow{\alpha} P_2' \implies P_1 \xrightarrow{\alpha} P_1' \wedge P_2' \simeq P_1').$$

Informally, two programs are bisimilar if and only if any action observed of one program can always be matched by the other, and the resulting programs are themselves bisimilar.

One consequence of the definition of bisimilarity is that in many situations this notion of equivalence corresponds to the notion of observational equivalence. We say that programs P_1 and P_2 are *observationally equivalent* if they produce the same output $O = O_{\mathcal{C}}$ when appearing as subprograms of any larger piece of code $\mathcal{C}$. This is potentially useful as one may often be interested in such equivalences: informally, observational equivalence is the universal interchangeability of two pieces of code. We write $P_1 \approx P_2$ to mean that P_1 and P_2 are observationally equivalent, and this holds just in case

$$\forall \text{ code } \mathcal{C}, \text{ outputs } O. \quad \mathcal{C}[P_1] \Downarrow O \iff \mathcal{C}[P_2] \Downarrow O.$$

It is usually quite difficult to establish when this happens, due to the universal quantification over all code fragments $\mathcal{C}$. However, one can often show that the relations $\simeq$ and $\approx$ coincide, which is useful as it is often much easier to prove that two programs are bisimilar than to prove directly that they are observationally equivalent. If you would like more details of these ideas, and a list of references, please see the Leicester **Mathematics and Computer Science Technical Report** *1996/5*, which is a preprint of the paper *Relating Operational and Denotational Semantics* by Roy L. Crole and Andrew D. Gordon. This work originally appeared in the proceedings of Computer Science Logic 1994, Kazimierz, Poland.

Dr. Roy L. Crole

Department of Mathematics and Computer Science,
University of Leicester,
University Road,
LEICESTER,
LE1 7RH,
United Kingdom.

email: r.crole@mcs.le.ac.uk
http://www.mcs.le.ac.uk/~rcrole

VDM Examples Repository

Peter Gorm Larsen
IFAD
Forskerparken 10
DK-5230 Odense M
Denmark

July 24, 1996

Below is a list of examples which can be freely obtained. All the volunteers which have submitted these examples so far have also made the source text available (those who have submitted each of the examples given below are mentioned as well). Thus, if you are interested in investigating the examples further you can play around with the source (they all come as a gzip'ed tar files; if retrieved with FTP use "gunzip -c file.tar.gz — tar -xvf -"; if retrieved with Mosaic, Mosaic will automatically gunzip it and you should save it as file.tar and un-tar by "tar -xvf file.tar") and if you have access to the IFAD VDM-SL Toolbox[1] you can actually analyse the source texts further. All the postscript versions of the documents have been automatically generated from the source texts which have been submitted unless the providers have explicitly produced the postscript output. Those who have submitted plain ASCII specifications have been included in a simple skeleton. Thus, some places the line breaks could be improved.

- Specification of an ammunition control system. (Paul Mukherjee)
  The Source file[2] and a postscript[3] version can be obtained (12 pp).

  This specification describes the safety requirements involved in adding and removing explosives at an explosives storage site. The specification is based on United Kingdom Ministry of Defence regulations concerning safe storage of explosives, which in turn are based on UN regulations. Details of the specification may be found in:

  P. Mukherjee and V. Stavridou, "The Formal Specification of Safety Requirements for the Storage of Explosives", technical report DITC 185/91, National Physical Laboratory, 1991.

  P. Mukherjee and V. Stavridou, "The Formal Specification of Safety Requirements for Storing Explosives", Formal Aspects of Computing, 5(4):299-336, 1993.

- Railway Interlocking Systems. (Kirsten Mark Hansen)
  The source files[4] and postscript[5] version can be obtained (32pp).

  This specification presents a VDM model of a real-life railway interlocking system. The safety requirements, which the interlocking system should fulfil are specified using VDM-SL, and are validated by executing the specification using the VDM-SL Toolbox. An interpretation of the safety requirements is proposed, and the interpretation is validated to be acceptable. The model development illustrates how concepts may be captured and validated for a non-trivial system. Further publications may be found in:

  K.M. Hansen, Formalising Railway Interlocking Systems, Nordic Seminar on Dependable Computing Systems, Department of Computer Science, Technical University of Denmark, August 1994, pp. 83–94.

[1] http://www.ifad.dk/products/toolbox.html
[2] ftp://ftp.ifad.dk/pub/vdm/examples/acs.avdm.tar.gz
[3] ftp://ftp.ifad.dk/pub/vdm/examples/acs.ps.gz
[4] ftp://ftp.ifad.dk/pub/vdm/examples/rail.vdm.tar.gz
[5] ftp://ftp.ifad.dk/pub/vdm/examples/rail.vdm.ps.gz

- Formal Semantics of Data Flow Diagrams. (Peter Gorm Larsen)
  The Source file[6] and a postscript[7] version can be obtained (48pp).

  Data Flow Diagrams are used in Structured Analysis and are based on an abstract model for data flow transformations. This specification is a transformation from an abstract syntax representation of a data flow diagram into an abstract syntax representation of a VDM specification. Details can be found in:

  P.G. Larsen, Nico Plat, and Hans Toetenel, "A Formal Semantics of Data Flow Diagrams"[8], Formal Aspects of Computing, 1994, December.

- Specification of the Single Transferable Vote (STV) algorithm. (Paul Mukherjee)
  The Source file[9] and a postscript[10] version can be obtained (15pp).

  STV is a scheme for performing elections, as advocated by the Electoral Reform Society. This specification formalises the English language requirements and has been tested by animating the specification. Details may be found in:

  P. Mukherjee and B.A. Wichmann, "STV: A Case Study of the Use of VDM", Technical Report DITC 219/93, National Physical Laboratory, 1993.

  P. Mukherjee and B.A. Wichmann, "Single Transferable Vote" A Case Study of the Use of VDM-SL", Proc. The Mathematics of Dependable Systems, ed. C.J. Mitchell, Oxford University Press 1994.

- Specification of the MAA standard. (Graeme Parkin)
  The Source file[11] and a postscript[12] version can be obtained (9pp).

  The Message Authenticator Algorithm (MAA) standard is used in the area of data security in banking and the scope of the standard is authentication. More details can be found in:

  G.I. Parkin and G O'Neill, "Specification of the MAA standard in VDM", In S. Prehn and W.J. Toetenel (eds), "VDM'91: Formal Software Development Methods", Springer-Verlag, October 1991.

- The Specification of a Binary Relational Database System (NDB) (Rich Bradford) The Source file[13] and a postscript[14] report including the validation plan of the specification can be obtained (37pp).

  The Non-programmer database system (NDB) is a nicely engineered binary relational database system invented by Norman Winterbottom of IBM. The formal Specification of NDB was originally undertaken by Anne Walshe, who has subsequently documented the specification and its refinement.

  NDB has been used as an example problem for *modular* specification in VDM-SL. However, the version available here is a "flat" specification. The postscript file includes a significant description of the validation of the specification using execution. Test coverage is not used though.

  Relevant publications are:

  A. Walshe, "NDB: The Formal Specification and Rigorous Design of a Single-User Database System", in C.B. Jones and R.C. Shaw (eds), "Case Studies in Systematic Software Development", Prentice Hall 1990, ISBN 0-13-116088-5

[6] ftp://ftp.ifad.dk/pub/vdm/examples/dfdexample.vdm.tar.gz
[7] ftp://ftp.ifad.dk/pub/vdm/examples/dfdexample.ps.gz
[8] ftp://ftp.ifad.dk/pub/papers/facs.ps.gz
[9] ftp://ftp.ifad.dk/pub/vdm/examples/stv.avdm.tar.gz
[10] ftp://ftp.ifad.dk/pub/vdm/examples/stv.ps.gz
[11] ftp://ftp.ifad.dk/pub/vdm/examples/maa.vdm.tar.gz
[12] ftp://ftp.ifad.dk/pub/vdm/examples/maa.ps.gz
[13] ftp://ftp.ifad.dk/pub/vdm/examples/ndb.avdm.gz
[14] ftp://ftp.ifad.dk/pub/vdm/examples/ndb.ps.gz

J.S. Fitzgerald and C.B. Jones, "Modularizing the Formal Description of a Database System", in D. Bjorner, C.A.R. Hoare and H. Langmaack (eds), VDM '90: VDM and Z – Formal Methods in Software Development, Springer-Verlag, LNCS 428, 1990

- Denotational Semantics of the programming language NewSpeak. (Paul Mukherjee)
  The Source file[15] and a postscript[16] version can be obtained (73pp).

  The programming language NewSpeak is a language designed specifically for use in safety-critical systems. It employs the notion of Orwellian programming – undesirable properties are avoided by restricting the syntax of the programming language. This is a formal semantics for the language in VDM-SL. Details of the language and its semantics:

  P. Mukherjee, "A Semantics for NewSpeak in VDM-SL". In T. Denvir, M. Naftalin, M. Bertran (eds), "FME '94: Industrial Benefit of Formal Methods", Springer-Verlag, October 1994, to appear.

  I.F. Currie, "NewSpeak - a reliable programming language". In C. Sennett (ed), "High-Integrity Software", Pitman 1989.

- Dynamic Semantics of mini-C (Niels K. Kirkegaard)
  The Source files[17]and a postscript[18] version can be obtained (92pp). Note that the source files here do not include the .sty file used by the IDERS project so some editing of the source text is needed if one wish to pretty-print the document directly.

  This specification describes the dynamic semantics of a language called mini-C. mini-C is dialect of the C programming language. mini-C is used as textual notation of the Hatley/Pirbhai notation in the Specification and Design Animator of the IDERS project[19].

  Further references:

  Alejandro Alonso, Luciano Baresi, Hanne Christensen, Marko Heikkinen, "IDERS: An Integrated Environment for the Development of Hard Real-Time Systems"[20]EUROMICRO Workshop on Real-Time Systems, Odense, Denmark, June 1995

- Static Semantics of mini-C (Niels K. Kirkegaard)
  The Source files[21]and a postscript[22] version can be obtained (103pp). Note that the source files here do not include the .sty file used by the IDERS project so some editing of the source text is needed if one wish to pretty-print the document directly.

  This specification describes the static semantic of a language called mini-C. mini-C is dialect of the C programming language. mini-C is used as textual notation of the Hatley/Pirbhai notation in the Specification and Design Animator of the IDERS project[23].

  Further references:

  Alejandro Alonso, Luciano Baresi, Hanne Christensen, Marko Heikkinen, "IDERS: An Integrated Environment for the Development of Hard Real-Time Systems"[24]EUROMICRO Workshop on Real-Time Systems, Odense, Denmark, June 1995

- Looseness Analysis Tool for a VDM-SL Subset. (Peter Gorm Larsen)
  The Source files[25] and a postscript[26] version can be obtained.

[15] ftp://ftp.ifad.dk/pub/vdm/examples/newspeak.avdm.tar.gz
[16] ftp://ftp.ifad.dk/pub/vdm/examples/newspeak.ps.gz
[17] ftp://ftp.ifad.dk/pub/vdm/examples/minicdyn.vdm.tar.gz
[18] ftp://ftp.ifad.dk/pub/vdm/examples/minicdyn.ps.gz
[19] http://www.ifad.dk/projects/iders.html
[20] ftp://ftp.ifad.dk/pub/papers/iders.ps.gz
[21] ftp://ftp.ifad.dk/pub/vdm/examples/minicstat.vdm.tar.gz
[22] ftp://ftp.ifad.dk/pub/vdm/examples/minicstat.ps.gz
[23] http://www.ifad.dk/projects/iders.html
[24] ftp://ftp.ifad.dk/pub/papers/iders.ps.gz
[25] ftp://ftp.ifad.dk/pub/vdm/examples/loose.vdm.tar.gz
[26] ftp://ftp.ifad.dk/pub/vdm/examples/loose.ps.gz

The specification language VDM-SL contains a notion of looseness. This is a specification of a looseness analysis tool which given a specification written in a subset of VDM-SL can determine which values a given VDM expression (using the definitions) may evaluate to in the different models for the specification. This illustrates how looseness (underdeterminedness) is combined with recursion in VDM-SL. Here the use of the test coverage facility of the IFAD VDM-SL Toolbox is illustrated. Details may be found in:

P.G. Larsen, "Evaluation of Underdetermined Explicit Definitions"[27]. In T. Denvir, M. Naftalin, M. Bertran (eds), "FME '94: Industrial Benefit of Formal Methods", Springer-Verlag, October 1994, to appear.

- A crosswords assistant. (Yves Ledru)
  The Source file[28] and a postscript[29] version can be obtained (10pp).

  This tutorial example is taken out of a VDM course given to the students of the Diplôme d'Etudes Supérieures Spécialisées en Génie Informatique (5th year) at the Université Joseph Fourier. This example uses the implicit style of specification of VDM-SL and thus may not be executed with the IFAD toolbox.

- Soccer Referee's book. (Yves Ledru)
  The Source file[30] and a postscript[31] version can be obtained (15pp).

  This tutorial example is taken out of a VDM course given to the students of the Diplôme d'Etudes Supérieures Spécialisées en Génie Informatique (5th year) at the Université Joseph Fourier. A first version uses the implicit style of specification of VDM-SL and thus may not be executed with the IFAD toolbox. An explicit version is given as an appendix.

- The specifications of the operations performed in a bar (Kevin Blackburn).
  The Source file[32] and a postscript[33] version can be obtained (6pp).

  This specification was produced during a VDM-SL course presented by Peter Gorm Larsen to ICL Enterprise Engineering. The modelling of bags was one of the exercises the attendees (including the author) was confronted with during the course. This specification is mainly intended for the purpose of illustrating how bags can be used.

- Modelling of Realms in VDM-SL (Peter Gorm Larsen).

  The Source files[34] and a postscript[35] version can be obtained (22pp).

  This document is simply an attempt to model the basic data structures and auxiliary functions necessary to represent realms. A geometric realm defined here is a planner graph over a finite resolution grid. This example have been partly tested and the test coverage information is displayed on the postscript version of the document. The script used for testing is included among the source files.

  Realms are used to represent geographical data. This document is based on:

  Realms: A Foundation for Spatial Data Types in Database Systems, Ralf Hartmut Güting and Marcus Schneider, Advances in Spatial Databases - Third International Symposium, SSD'93, Springer-Verlag, June 1993.

  Map Generalisation, Ngo Quoc Tao, UNU/IIST, Macau, Draft, January, 1996.

[27] ftp://ftp.ifad.dk/pub/papers/eval.ps.gz
[28] ftp://ftp.ifad.dk/pub/vdm/examples/crossword.vdm.gz
[29] ftp://ftp.ifad.dk/pub/vdm/examples/crossword.ps.gz
[30] ftp://ftp.ifad.dk/pub/vdm/examples/soccer.vdm.gz
[31] ftp://ftp.ifad.dk/pub/vdm/examples/soccer.ps.gz
[32] ftp://ftp.ifad.dk/pub/vdm/examples/bar.vdm.gz
[33] ftp://ftp.ifad.dk/pub/vdm/examples/bar.ps.gz
[34] ftp://ftp.ifad.dk/pub/vdm/examples/realm.vdm.tar.gz
[35] ftp://ftp.ifad.dk/pub/vdm/examples/realm.ps.gz

- A VDM Specification of the Steam-Boiler Problem (Yves Ledru and Marie-Laure Potet).

  The Source files[36] and a postscript version as: part 1[37] part 2[38] and part 3[39] can be obtained (60pp altogether).

  This is a VDM-SL solution to the Steam Boiler Control Specification Problem[40]. The seminar "Methods for Semantics and Specification" was held at Schloss Dagstuhl, Wadern, Germany, on June 5-9, 1995. The seminar took the form of a "competition" between different researchers who had been invited as representatives of their particular methods. This specification is split into a number of modules which are placed in different files.

- Mapping between EXPRESS representations (Marcel Verhoef)

  The Source file[41] and postscript[42] version can be obtained (5pp).

  The (building) industry in Europe currently uses the ISO-STEP standard to define information models with the aim to exchange data about those information models. The ISO-STEP standard contains the EXPRESS modelling language and several programming language bindings and an ASCII neutral format to implement interfaces to those models. Unfortunately, industry has not reached consensus on a particular information model, therefore multiple models exist. This raises the need to migrate instances from one model to another and vice-versa, commonly referred to as the "mapping". The aim of this exercise was to determine the applicability of VDM-SL with respect to these types of problems. For more details on the mapping issue down-loadable copies of two papers[43] resulting from this research is available. The example shows a simple VDM-SL abstract syntax representation of the ISO STEP part 21 physical file format and a transformation process for a particular set of abstract syntax instances. It implements a mapping between the relational model representation (rmrep) into a simple polynomial representation.

[36] ftp://ftp.ifad.dk/pub/vdm/examples/boiler.vdm.tar.gz
[37] http://www.informatik.uni-kiel.de/ procos/dag9523/potet1.ps.Z
[38] http://www.informatik.uni-kiel.de/ procos/dag9523/potet2.ps.Z
[39] http://www.informatik.uni-kiel.de/ procos/dag9523/potet3.ps.Z
[40] http://www.informatik.uni-kiel.de/ procos/dag9523/dag9523.html
[41] ftp://ftp.ifad.dk/pub/vdm/examples/express.vdm.gz
[42] ftp://ftp.ifad.dk/pub/vdm/examples/express.ps.gz
[43] http://dutcu15.tudelft.nl/ marcel/mapping/mapping.html

Functional Programming Column

Philip Wadler, University of Glasgow
wadler@dcs.glasgow.ac.uk
http://www.dcs.glasgow.ac.uk/~wadler

This month's functional programming column is short and sweet – an invitation to read or contribute to *The Journal Of Functional Programming.*

FACS readers are encouraged to submit to this column, as well as to JFP. Short papers on the following topics are especially welcome.

- Specifications written as functional programs.
- Proving properties of functional programs.
- Real-world applications of functional programming. (Please also submit these to our web page, `http://www.dcs.gla.ac.uk/fp/realworld.html`.)
- Tutorial summaries of relevant advances in FP.

Submissions are welcome at any time, electronically or by post.

Journal of Functional Programming: The state of play

Simon Peyton Jones

`http://www.dcs.gla.ac.uk/jfp`

We're delighted to announce the publication of a special issue of JFP 5(3) on applications of functional programming. Edited by Rinus Plasmeijer and Pieter Hartel, it contains six papers on applications ranging from photon transport to text retrieval, from spreadsheets to solid modelling.

The titles of the papers are given below. You can see their abstracts too on the above web page. You'll also now find titles and abstracts for the whole of Volumes 4 and 5, and titles for Volumes 1–3.

We'd like to take this opportunity to encourage you to submit papers to JFP. At one point we had so many submissions that we built up rather a long print queue, but Cambridge University Press have agreed to move from 4 to 6 issues a year (starting in 1996), and to make them big issues, which has effectively solved the print-queue problem.

We offer a 12-week turnaround to a decision about submitted papers, and the delay from final manuscript to publication is now dropping fast (our target is 3 months).

Please submit papers on any aspect of functional programming to any of the Editors:

Henk Barendregt	henk@cs.kun.nl
Paul Hudak	hudak@cs.yale.edu
John Hughes	rjmh@cs.chalmers.se
Simon Peyton Jones	simonpj@dcs.gla.ac.uk
Philip Wadler	wadler@dcs.gla.ac.uk

You can find full details of the journal's scope, editorial policy, Latex style files, and subscription information by following the web link above.

JFP Special Issue 5(3) (July 1995) on State-of-the-art applications of pure functional programming

Editors: Pieter Hartel and Rinus Plasmeijer

- Comparing Id and Haskell in a Monte Carlo photon transport code, Jeffrey Hammes, Olaf Lubeck and Wim Bohm
- Funser: a functional server for textual information retrieval, Donald A Ziff, Stephen P Spackman and Keith Waclena
- Prototyping a parallel vision system in Standard ML, Greg Michaelson and Norman Scaife
- Implementing a functional spreadsheet in Clean Walter A.C.A.J.De Hoon, Luc M.W.J.Rutten and Marko C.J.D. Van Eekelen
- A polymorphic library for constructive solid geometry J.R. Davy and P.M. Dew
- Exploring the conformations of nucleic acids Marcel Turcotte, Guy Lapalme and Francois Major

Book Review

System Construction and Analysis: A Mathematical and Logical Framework (Norman Fenton and Gillian Hill), McGraw-Hill International Series in Software Engineering, 1993. Price: £19.95

K. Lano

This book is aimed towards the educational market, and presents a wide range of topics in the mathematical foundations of computing: set theory, propositional and predicate logic, algebras, category theory, the lambda calculus, process algebras, computational complexity, probability theory, coding theory, measurement theory and various styles of formal specification and refinement. Each area is built up from the basics, making few assumptions about the readers prior knowledge. The book contains sufficient material for a variety of courses, although the breadth of material means that more specialised topics such as computational complexity will need other textbooks as a supplement for a substantial course. In particular I would have liked to see more on Hoare logics and program verification beyond the short chapter included here.

The key strength of the book is its integration of this variety of topics, so that it would be ideal if used consistently through a series of courses – such as introductory logic and discrete mathematics; functional languages and program semantics; and formal specification and development. Much of the material is suitable for undergraduate courses, and has the advantage of not just teaching a particular formal specification language but stressing the general principles of specification and mathematical modelling.

It covers and contrasts a number of specification styles: algebraic, state-based and logical, giving background on their origin and areas of application. Z and VDM are introduced. An interesting innovation is the use of the lambda calculus to define and motivate process algebras, and the inclusion of measure theory and probability theory (the latter being used in the chapter on coding theory, which shows the relevance of combining traditional "formal methods" with other mathematical fields).

There are some, very occasional, minor typos, and some notations are a little unclear, but the general quality of the book is very high, with a good consistency of style for a book written by two authors. A set of worked solutions are also provided, in a separate booklet (of 100 pages) which additionally gives a teachers guide to the use of the material in various courses. The book isn't suitable for Arsenal fans unless they have a sense of humour however!

Book Review - an Informal Method

S.J. Goldsack

Object Oriented Technology for Real-Time Systems:
A Practical approach Using OMT and Fusion.

Maher Awad, Juha Kuusela and Jurgen Ziegler.
Prentice Hall International 1996
ISBN 0-13-227943-6

This book describes a methodology, which the authors call Octopus, to be used for the development of object oriented programs in the fields of real-time embedded and distributed systems. The authors are from the Nokia Corporation in Finland, and the method was developed initially for use within the company. They claim to have used it in the development of several of their company's products.

Octopus makes no pretence to promote the use of formal methods. Nevertheless I believe it can usefully be read carefully by anyone who is contributing to current efforts aimed at bringing the use of formal approaches to software development to the stage at which they present a real alternative to current practice, at least for object-oriented and real-time systems.

Over the past twenty years, structured methods of software analysis and design have been developed, starting mainly with Structured Analysis and Design from the Yourdon school. More recently methods aimed at supporting the construction of object oriented systems have been developed. OOA (object- oriented analysis)[1] developed from Structured Analysis (1985), Booch's method (1987)[2] and OMT, the Object Modelling Technique by Rumbaugh et al (1981)[3] are attempts to provide guide-lines for design and implementation. OMT has tended to become the popular industry standard in recent years.

Statecharts, an extension of abstract state machines, were developed by Harel as a specification technique for functional aspects of an object oriented system, and are incorporated into OMT and several other methods.

These approaches are all in some sense model oriented - the analysis develops a conceptual model of the problem world. Other approaches use a more behaviour-oriented approach, emphasising first the interactions between objects. Object Oriented Software Engineering, developed by Jacobson (1992), [4] is a well known example.

Fusion (Coleman et al. 1993) [5] and Syntropy (Cook and Daniels 1995)[6]

have developed methods which apply both these viewpoints in a complementary way. Fusion, in particular, applies a modelling approach to the analysis phase, but a more behaviour-oriented viewpoint in the design phase.

These methods are all, in the main, diagrammatic and structured, but do not attempt to give any formal definitions either of the behaviour of the objects and systems, nor (with the exception of Syntropy which does attempt this) even of the semantic meaning of the diagrams. Although it does not aim to present a formal approach, Fusion does offer the use of assertions and pre and post conditions to allow system analysts to make less ambiguous statements about the intended behaviour of the system.

Octopus is the most recent contribution to this collection of methods and techniques. As indicated by the subtitle of the book, it is in many ways an extension of OMT and Fusion, and uses the notations of OMT extensively. However, it presents a systematic approach which addresses the needs of Real-time embedded systems and multi-processor distributed systems, as well as the more familiar sequential systems.

Two main features distinguish the embedded real-time systems from others:

1. Software executes in an environment of interaction components which are considered external to the system. These may be themselves computer processors managing communicating devices, or may be hardware devices with digital interfaces to the system. It is therefore impossible to design the system without consideration of the external world. There are events, and actions triggered by events, to be taken into account.

2. Concurrency is an essential aspect of a system. Apart from the role of concurrency in modelling real world situations in which co-existing objects pursue their life cycles concurrently, it must also be appreciated that executions on processors consume time and during some action execution events in the external world may occur. It is essential to build concurrency into the design approach.

It is in the provision for these aspects that Octopus shows most originality.

The authors recommend system development following a life cycle which is familiar enough. There is, of course, provision for iteration over the following stages:

Requirements analysis

The approach uses a concept introduced by Jacobson of "Use Cases" to discover the main activities of each agent in the system and the actions in which they participate.

Special to Octopus is the presentation of the complete system as an aggregation of use cases using the notation of OMT. The method prescribes a standard

sheet on which the information on each use case should be recorded. This includes pre and post conditions for the actions, but in the examples shown these are often very informal. In my opinion most of these requirements should be recorded in terms of invariants, since in the worlds which are the target they are concerned with expressing conditions which are "forever" true.

The information contained in the use case sheets is supplemented by a "Context Diagram" which defines all the external world interactions to which the system is subject. This is a feature of early forms of Structured Analysis.

System Architecture and the Hardware Wrapper

Here the possible division of the system into subsystems is considered, but most crucially, the subsystem known as the *hardware wrapper* is introduced. This is the software lying between the physical hardware and the application software.

Subsystem Analysis

In this stage each subsystem is analysed to develop an object model using OMT class diagrams, a functional model using Operation sheets to define the effects of each operation in the system, and a dynamic model which establishes the events which occur in the software and its environment and specifies their effects using Harel style statecharts.

It is in these stages that Octopus is most different from other techniques: the effects and interactions of events is studied using event scenarios and a simple calculus of "event significance"

Subsystem Design and Implementation

System design taking into account all the results of analysis, including the real-time aspects, proceeds through the use of further new techniques special to Octopus, including event threads (which may introduce real time). This leads to development of object groupings, process timings and priorities.

The book ends with two substantial case studies with real-time features, illustrating by example the use of all the techniques introduced in the book. These are:

1. A subscriber line tester for a telephone system
2. An automobile cruise control

The book is well written, and brings out the very large number of separate considerations which enter into the design of real-sized systems for this field. I have learned much in reading it, and appreciate that most of this large number of considerations which go into a realistic methodology for informal system

development must be relevant to any adequate formal method also. The development of such a formally based and complete methodology (supported by tools) is a formidable and daunting task.

References

[1] S. Shlaer and S.J. Mellor, *Object Lifecycles: Modelling the World in States.* Englewood Cliffs, NJ. Prentice Hall 1992.

[2] G. Booch. *Software Components with Ada.* Menlo Park Benjamin Cummings 1987.

[3] J.Rumbaugh, M. Blaha, W. Premerlani, F. Eddy and W. Lorensen. *Object-Oriented Modelling and Design.* Englewood Cliffs, NJ. Prentice Hall 1991.

[4] I. Jacobson, M. Christerson, P. Jonsson and G. Overgard. *Object-Oriented Software Engineering-A Use Case Driven Approach,* Reading Ma. Addison Wesley 1992.

[5] D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes and P. Jeremaes. *Object-Oriented Development - the Fusion Method,* Englewood Cliffs, NJ. Prentice Hall 1993

[6] S. Cook and J. Daniels. *Object Systems - Object-Oriented Modeling with Syntropy.* Englewood Cliffs, NJ. Prentice Hall 1994

Book Review

Proof in VDM: A Practitioners Guide (J. C. Bicarregui, J. S. Fitzgerald, P. A. Lindsay, R. Moore, B. Ritchie), FACIT Series, Springer-Verlag, 1994

K. Lano

July 25, 1996

This book represents the culmination of many years of work in the VDM community on defining a reasoning system for VDM specifications, and on developing tools (the Mural system) to support such reasoning. It explains the Logic of Partial Functions and the proof system step by step, building up from the basic theories of natural numbers, sets, sequences and maps, and applies this logic to reason about the soundness and validity of specifications, and refinement relationships between specifications. Its aim is to provide a practical guide to constructing proofs, from informal reasoning to rigorous and fully formal proofs.

A medium-sized case study of an air-traffic control system is given as an example of validation and refinement proofs, and the overall style of the book is as a tutorial aimed at advanced users, rather than being a report of recent research. The stress on completely formalised proofs throughout the book may nonetheless introduce feelings of panic in most people without PhDs in mathematical logic (and in quite a few people with, as well). More examples of rigorous reasoning applied to realistic problems, and discussion of the selective application of formal proof to critical obligations would be helpful in this respect.

The book represents a very clear and precise description of VDM and its underlying logic, and could be used as an advanced textbook on the language. Given the number of contributors to the book, the style is remarkably uniform and technical errors seem absent. It does not attempt to be a manual for the Mural tool, although the material here is consistent with the approach taken by Mural, and would be of considerable help for users of this tool. A complete listing of the proof rules and axioms are given, as are indexes for rules and technical terms.

The book does not address some of the more obscure and complex issues concerning VDM semantics and proof, in particular loose **let** expressions and general pattern matching. Instead these are identified as areas of ongoing research in Chapter 13 of the book. An excellent treatment of these aspects, together with a description of how the proof system of the book needs to be extended to accommodate them, is given in the thesis [1].

Program verification is omitted, as is mention of other variants of VDM, such as the Irish school. The former is important if a complete formal development process is to be supported in VDM.

The book could also be improved by including more comparisons with other forms of proof system and specification approaches, such as algebraic and term-rewriting based methods, or the Z proof system, but it is probably the best work of its kind that has been written to date, and represents an invaluable assistant for the serious user of VDM.

References

[1] P. Larsen, *Towards Proof Rules for the Full Standard VDM Specification Language*, PhD. Thesis, Department of Computer Science, Tech. Univ. of Denmark, Lyngby, Denmark, 1995.

Northern Formal Methods Workshop.

Call for Participation

ILKLEY, UK, 23-24 September 1996

Supported by BCS-FACS

In recent years formal methods have emerged as effective tools for the development complex systems where quality and reliability is of essential concern. Increasingly, formal methods are being applied to more diverse areas of software and hardware systems, as well as contributing to a greater theoretical understanding of modern computing practice.

This workshop brings leading figures in the research and application of formal methods and covers a range of current techniques. The focus of the workshop is on the use of formal methods for the development of concurrent, real-time and object-oriented systems. It aims to contribute to fundemental research in these areas and to provide a friendly and informal atmosphere in which participants can exchange and contribute new ideas and developments.

The Workshop will consist of invited talks by four leading figures in the use and application of formal methods and strong programme of contributed papers.

The workshop will be held at the Craiglands, a delightful country hotel on the edge of the beautiful Yorkshire Dales and near the famed Ilkely Moor.

It is expected that the proceedings will be published by Springer-Verlag.

Programme Review Committee

J. Armstrong (York)
A.Bryant (LMU)
A.N.Clark (Brad)
D.J. Duke (York)
R.W. Duke (Queensland)
A.S. Evans (Brad)
B. Fields (York)
M.J. Hinchey (New Jersey Institute of Tech)
D.R.W. Holton (Brad)
J.L. Jacob (York)
L.M.Lai (Brad)
K.Lano (Imperial)
N.A. Merriam (York)
P. Mukherjee (Leeds)
F.A.C. Polack (York)
L.T Semmens (LMU)
I.S.Torsun (Brad)

BCS FACS
Department of Computer Studies
Loughborough University of Technology
Loughborough, Leicestershire
LE11 3TU
UK
Tel: +44 1509 222676
Fax: +44 1509 211586
E-mail: FACS@lboro.ac.uk

FACS Officers

Chair	John Cooke	D.J.Cooke@lboro.ac.uk
Treasurer	Roger Stone	R.G.Stone@lboro.ac.uk
Committee Secretary	Roger Carsley	roger@westminster.ac.uk
Membership Secretary	John Cooke	D.J.Cooke@lboro.ac.uk
Newsletter Editor	Ann Wrightson	a.wrightson@.hud.ac.uk
Liaison with BCS	Margaret West	m.m.west@hud.ac.uk
Liaison with FME	Tim Denvir	timdenvir@cix.compulink.co.uk