FORMAL METHODS EUROPE

FACS Europe

BCS FACS

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

Series I Vol. 2, No. 2, Autumn 1995 ISSN 1361-3103

Contents

Guest Editorial: A View from the Bridge

John Cooke

At an impromptu committee meeting last year I was persuaded by colleagues to stand once again as FACS chairman. [I'm sorry but until I have four wooden legs, and risking being politically incorrect, I shall never regard myself as a chair - even though I am accustomed to being sat upon]. Eventually, along came the AGM and, in the absence of other 'volunteers' I took over the reins from Tim. Perhaps it is inappropriate to talk of reins, FACS now runs very smoothly but we can always use extra helpers.

1 The Journal

In her first FACS Europe editorial, Ann made a side-ways comment about the FACS Journal. As the main link man between FACS proper and the journal, I think it proper that I should comment and try to explain where we are going. The journal is now in its 7th year; nominally a volume now consists of 6 issues of 120pp with extra pages available via ftp (these contain larger versions of papers, with detailed proofs etc.). In the early days we received, and published, many papers that were 'theoretical' rather than 'formal'. I think that our launch coincided with TCS moving to a publication rate of in excess of a volume per month, so we probably attracted some of their overspill.

Such papers were perhaps not ideal but, like any journal, we can't publish papers we don't receive and attracting submissions is far from easy until you are established. Today I think that the journal is well-respected and is a recognised vehicle for the publishing of research in formal methods.

Much of what we publish is very technical, but in an attempt (a) to cope with the challenges emerging from the move towards electronic publishing - driven primarily by the ACM - and (b) to make papers more accessible to a wider audience, we are now actively encouraging smaller, more focussed papers (backed up by larger papers on ftp).

2 What is 'formal methods'?

It is not theory per se, but neither is it "informal methods with a sprinkling of mathematical notation".

At a recent workshop on the "Teaching of Formal Methods" to which I was invited, I was amazed at the almost total lack of what I would regard as formal methods (of software construction). Whilst it was true that there was a lot of Z etc. in evidence, nobody considered what you would do with a specification. Don't get me wrong, what was being discussed is very important - but it wasn't formal methods, it was more akin to Requirements Engineering, a necessary prerequisite to formal methods.

The raison d'etre for using formalisms is so that we can reason about processes and artifacts. Whether they be implicit or explicit, we have proofs rather than hand-waving and hacking. True, we need to make our work more accessible to the software engineers who should be using it. This may be achieved by encapsulating "theorems" into methodologies, or by continually striving for better, more natural, notations. I think we are getting there slowly. The key seems to be in abstraction and simplification. All too often specifications are overly complex and include elements of (implementation) design, or are structured, not so as to aid comprehension but, so as to resemble programs.

Doubtless I shall get comments about this!

3 Your Input

Whether or not you wish to take issue with the statements given above, we hope you will contribute to FACS Europe. In the past FACS have tried to keep track of members interests by means of questionnaires etc.. Another way in which we might reflect members interests is for you to contribute, to tell us what you are doing. How about sending us brief details (title, journal ref., abstract) of your recent work, or even the work of others that you feel may be of interest to members. This should involve very little extra work yet allow us to reflect members current interests. It may also contribute to reaching a clearer agreement on what formal methods, and hence what FACS, is all about.

FME Update

Tim Denvir

1 FME '94

The second FME symposium took place in Barcelona, hosted by the Escola d'Enginyeria La Salle at the University of Ramon Llull, during 24 to 28 October 1994. It followed a previous symposium held in Odense in April 1993. Like its predecessor, FME '94 was a most successful and enjoyable event. Barcelona, the capital city of Catalonia, lying between the Collserola mountains and the Mediterranean, was an excellent location for the symposium. It is a major commercial and industrial centre, noted for its Gothic town and the architecture of Gaudi, whose style remains arrestingly radical after 70 or more years. I and many other delegates were brought face to face with the realisation that Catalonia is a very individual country with its own Catalan language, considerably different from Spanish, and its own culture and national character.

The symposium itself was divided into two chronological parts: a Tutorial following two tracks during the first two days and the Conference for the last three days. Nine tutorials were given, and those attending the tutorial programme were provided with substantial bound notes supplementing the lectures, which were two to three hours in length. The tutorials covered:

Formal Methods for Reactive Systems - Claus Lewerents and Thomas Lindner
Running a Formal Methods Project - John Nicholls
The Rôle of Formal Specifications in Software Test - Jan Peleska and Hans-Martin Hörcher
The RAISE Development Method - Anne Haxthausen
Data Refinement in VDM: a different approach - Derek Andrews
IMPS, an Interactive Mathematical Proof System - Joshua Guttman
Refinement: Tools and Applications - David Jackson
Action Semantics - Peter Mosses
EVES - Dan Craigen and Mark Saaltink

A particular feature of the conference was the inclusion of seven reports on the industrial usage of formal methods. The applications covered ranged over a security management system, British Rail's signalling rules, a comparative study of formal and informal specifications of a secure system component (a gateway), a network protocol, a virtual memory scheme, a verified compiler, a graphics kernel system and others. The paper "Seven More Myths of Formal Methods" by Jonathan Bowen and Michael Hinchey gave a very comprehensive collection of references to other industrial uses. With this substantial documented collection of industrial applications, no-one can any longer sensibly claim that formal methods are not being used in industry.

Two stimulating invited talks were given by Ralph Back "From Action Systems to Modular Systems" (R.J. Back and K. Sere) and Babak Dehbonei "Formal Methods in the Railway Signalling Industry" (B. Babak and F. Mejia). The latter talk recounted the SACEM project, an automated train protection system developed using B for the Paris metro system, and so was in fact a further report on industrial usage.

Thirty two papers were accepted and presented, out of some 150 which were submitted. These largely fell into the topics of refinement, proof, formal process modelling, formal aspects of object orientation, semantics of programming languages, RAISE, railway applications (a growing application area for formal methods), aspects of time, and synthesis with other methods. There was also a poster session in which short talks on recent work were given.

Another important feature of the symposium was the tools exhibition which ran in parallel with the other events throughout. Tools supporting algebraic specifications, proof, RAISE, CSP, HOL, VDM, B, Action Semantics, VDM++, Z, SDL, process algebra, and functional languages were demonstrated. A booklet of abstracts on the tools was provided.

Finally, but not least, our hosts arranged an excellent conference banquet of typical Catalan cuisine at the Palau Abadal in Barcelona.

The proceedings of FME'94 are published by Springer-Verlag as LNCS 873.

The next FME Symposium will be held at Oxford University from 18 to 22 March 1996.

2 Tools Info

Details of the tools exhibited at FME '94, and other tools, can be obtained by anonymous ftp to the following sites:

```
directory pub/misc, file fm_tools_db at site ecs.soton.ac.uk

directory pub/fm_tools, file fm_tools_db at site chopwell.newcastle.ac.uk

at site ftp.informatik.tu-muenchen.de/local/lehrstuhl/nipkow/fm_tools_db

directory pub/clt/FMTDB/, file fm_tools_db at site sail.stanford.edu

in /ftp/pub/formal_methods/fm_tools_db at site ftp.cs.waikato.ac.nz

at site ftp://chopwell.ncl.ac.uk/pub/fm_tools/fm_tools_db
```

First Annual ENCRESS (European Network of Clubs for REliability and Safety of Software) conference

Tim Denvir

Bruges, 12 - 15 September 1995

This conference was part of a Dissemination Action funded by ESSI, the Software Best Practice initiative of the European Commission. The conference was also the twelfth annual workshop of CSR (Centre for Software Reliability), the U.K. club in the network. Since CSR is the founding member of the network, there was, unsurprisingly, a strong British presence, with nearly half the delegates and half the papers from the UK. However, with six new non-U.K. members of the network, this excessive British influence may soon diminish.

I did not attend the first day of the event. This included a half-day tutorial from John Rushby of SRI on "The Rôle of Formal Methods in the Certification of Critical Systems". I heard good reports on this from other delegates and the paper he provided for the proceedings went to some 60 pages and struck me as being a very thorough contribution. Its main headings are:

1. The rationale for Formal Methods
2. Issues and Choices in Formal Methods
3. Formal Methods and Certification

Another well-received contribution was a keynote address from Martyn Thomas of Praxis. The title of his topic was "Safety Cases for Software Based Systems" and he dwelt largely on the theme of engineering judgment. One of his theses was that numbers (i.e. measurements) should inform judgment but not replace it.

There are many aspects to safety critical software: development and design technology, measurement of features of the software and its development process, safety regulation issues, process improvement and allied topics (capability maturity models, ISO 9000, TickIT, etc.), fault tree analysis, simulation, and even legal issues. It is not too surprising that formal methods did not feature strongly; but it was the topic of the principal tutorial lecture and the topic of two other papers.

Jonathan Draper of GEC-Marconi reported on an ESSI Application Experiment (MIST) which is half-way through its duration. B is being applied to a subset of an avionics application - about 3,000 lines of embedded Ada code out of a 15,000 line system - in a parallel development. He described how they had used the refinement and implementation principles of B, the automatic and interactive proof tools and the configuration tool which automatically regenerates the necessary proof obligations. Predefined metrics are to be used to make comparisons between the two developments. After 8 months they had written, proved and analysed the abstract specification. Reviews had exposed 9 errors, proofs had exposed 6, and animation 2. The errors not found by the proof techniques consisted of "missing functionality", i.e. features which had been accidentally omitted. This was a most encouraging paper on practical use of B.

The second paper whose topic was the application of formal methods was delivered by Ronald Tol from the University of Groningen, on "ARTIE - A Proven Correct Architecture for Safety Critical Applications". The hardware and real-time OS which he described supports HI-PEARL, an extension of the real-time programming language PEARL. The paper described how the scheduling algorithm for a railway crossing system was proved to meet its timing requirements. This paper was very clearly delivered and one of the three I rated best of all those I heard.

Bruges is a beautiful city in which to hold a conference (or indeed, to do anything else), full of extremely well preserved old buildings and with many canals within the city. The conference itself was held in the Sofitel hotel which was formerly a 17th century monastery. Unfortunately, once inside the building, no trace of its antiquity is perceptible; the interior could be that of any modern four star hotel.

The organisers hope to publish the proceedings through Springer Verlag.

I attended the conference at the request of the European Commission's ESSI (Software Best Practice) initiative, and I am grateful to them for permission to publish this report.

The Role of formal specifications in Software Testing

Hans-Martin Hoercher

Report of tutorial at ESEC'95

7 people attended this 1-day seminar. The subject matter was:

1. Introduction into Formal Software Engineering, esp. Z
2. The use of formal specifications for sound test case selection
3. How to completely automate test result evaluation using formal specs

The audience was from industry and academia on nearly equal parts. The main expectiations of them was

- give additional argumentation/topics when teaching formal methods (academia)
- to learn about alternatives for traditional specification techniques, their limitations and their possibilities, also, but not only, related to SW testing.

The response was very positive. There were constructive discussions, on the technical level as well as on corresponding organizational aspects (eg applying FM in the SW development cycle).

Some of the audience had heared about FM before, others did not. The introduction into Z seemed to suit all of them, because detailed discussions on specification styles also came up.

The overall judgements afterwards were very positive. It seems that everybody learned, what he expected to.

A Brief Overview of Gödel

Pat Hill
School of Computer Studies, University of Leeds, LS2 9JT, UK
email: hill@scs.leeds.ac.uk, www: agora.leeds.ac.uk/hill/

Gödel is a computer language based on logic with functionality and expressiveness similar to Prolog, but with greatly improved declarative semantics. Facilities provided by Gödel include types, meta-programming, control annotations, modules, and input/output. To give the flavour of programming in Gödel, we present an example program about the European Union.

```
MODULE     EuropeanUnion.

IMPORT     Lists, Sets.

BASE       Region.
CONSTANT   EU, France, Britain, Wales, England, WestYorkshire, Leeds : Region.

PREDICATE Component : Region * Region.
Component(EU, France).
Component(EU, Britain).
Component(Britain, Wales).
Component(Britain, England).
Component(England, WestYorkshire).
Component(WestYorkshire, Leeds).

PREDICATE IsIn : zPz : Region * Region.
DELAY u IsIn v UNTIL NONVAR(u).
x IsIn y <- Component(y,x).
x IsIn y <- Component(z,x) & z IsIn y.

PREDICATE Universe : Region.
Universe(u) <- ALL [r] (Component(_,r) -> r IsIn u).
```

This program has a main module called `EuropeanUnion`, beginning with the keyword `MODULE` and its name. The declarations beginning with the keywords `BASE`, `CONSTANT`, and `PREDICATE` define the language of the program. There can be similar language declarations for type constructors, functions, and propositions. The `zPz` in the declaration for `IsIn` indicates that this is an infix binary predicate. Note that (unlike Prolog) names of symbols such as constants are denoted by identifiers beginning with an upper case letter whereas names of variables are denoted by identifiers beginning with a lower case letter. The declaration beginning with the keyword `DELAY` is a control declaration and will be explained later.

Gödel is a strongly typed language. This means that the language declarations should fully define the language of a program and the statements must be written in this language. The type system is based on many-sorted logic with parametric polymorphism. For instance, there is a system module `Lists` in Gödel for list processing which provides a type constructor `List` of arity 1. `Lists` also provides a constant `Nil` and a function `Cons`, whose declarations are:

```
CONSTANT   Nil : List(a).
FUNCTION   Cons : a * List(a) -> List(a).
```

The usual `[...]` and `|` syntax for lists is allowed in Gödel and is syntactic sugar for the `Nil` and `Cons` notation. Thus the list

```
Cons(England,Cons(France,Cons(Wales,Nil)))
```

can be written more conveniently as

```
[England,France,Wales]
```

A collection of useful list processing predicates, including `Append`, is defined in `Lists`. These predicates together with the above syntax for lists are available for use in any module containing an `IMPORT` module declaration for `Lists`. In general, if a module imports from another module, it imports *all* the symbols exported by the other module.

Gödel allows the use of arbitrary formulas in the bodies of statements and goals. Conjunction is denoted by `&`, disjunction by `\/`, negation by `~`, left implication by `<-`, right implication by `->`, and equivalence by `<->`. The universal and existential quantifiers are denoted by `ALL` and `SOME`. The definition of `Universe` in the `EuropeanUnion` program illustrates this. Allowing for arbitrary formulas in the bodies of statements and goals considerably enhances Gödel's expressiveness and facilitates executable specifications. For example, below is a specification of a relation between a list and a list with the same elements in reverse order.

```
ReverseList(x1,x2) <-
    Length(x1,l) & Length(x2,l) &
    ALL[i]
      (1 =< i =< l ->
         SOME [a](Suffix(x1,i,[a|_]) & Suffix(x2,l-i+1,[a|_]))).
```

For this to execute, just a value for one of the arguments `x1` or `x2` has to be input so that `Reverse` will run in either direction. Thus such a specification is not only executable but also the "what if" form of question can be asked of it.

The connectives and quantifiers introduced so far provide an expressive language in which to write goals and statements. However, a programmer that needs to use a formula of the form

(*Condition* & *Formula1*) \/ (~*Condition* & *Formula2*).

will not want the *Condition* test to be computed twice. For this reason, Gödel has ***conditionals*** with specialised procedural semantics to avoid this inefficiency. As an example of a conditional and its use, consider the following definition of the predicate `Max`.

```
Max(x,y,z) <- IF x =< y THEN z = y ELSE z = x.
```

The meaning of this is that `Max(x,y,z)` is true if either `x =< y` and `z = y` or `x > y` and `z = x`. Procedurally, as soon as `x =< y` is ground, this inequality is tested. If the call to `x =< y` succeeds, then `z` is bound to the value of `y`. If the call fails, then `z` is bound to the value of `x`.

Equality and disequality predicates are built into the Gödel system and have the following dꞏclarations

```
PREDICATE =  : zPz : a * a.
PREDICATE ~= : zPz : a * a.
```

Equality and disequality predicates can be used in any module without being imported. Note that the axioms for equality and disequality are also built-in and the = and ~= predicates cannot be defined by the user. Although, for most types, equality is assumed by the Gödel system to be syntactic identity, special equality theories for certain system types are supported. There are three arithmetic modules in Gödel: `Integers`, `Rationals`, and `Floats` that allow infinite precision integer arithmetic, infinite precision rational arithmetic, and floating-point number computations, respectively. Each of these modules supports an appropriate equality theory. For instance, the module `Integers` provides an equality theory for integers. To illustrate how this equality theory for the integers may be used, suppose a module which imports `Integers` defines a predicate `P`:

```
PREDICATE P : Integer * Integer.
P(x,x).
```

Then the goal

```
<- P(3 + 4, (17 Div 6) + 5).
```

succeeds.

The arithmetic modules not only provide the actual numbers, their operators, and equality theory, but also relations such as < and >. Systems of (not necessarily linear) constraints which involve integer and rational expressions can be solved (although expressions that can be computed will depend on the implementation being used). For example, with the **Integers** module loaded, the goals

```
<- 32*4 >= (130 Mod 4).
<- x + 43 = 73 + (34 Mod 4).
<- 0 =< x =< 10  &  0 < y =< 10  &  35*x + 33*y =< 34.
<- 0 < x =< 10  &  x ~= 2  &  x^2 < 12.
```

will succeed with the answer `x = 32` for the second goal, answer `x = 0` and `y = 1` for the third goal, and answers `x = 1` and `x = 3` for the fourth goal.

Gödel supports a modular approach to programming. In the Gödel module system, part of a module (the local part) is hidden and not visible by the importing modules. This not only hides implementation details but also provides support for abstract data types. The imports relation declared by the **IMPORT** declarations is transitive. For example, The **EuropeanUnion** program imports the system modules **Lists** and **Sets**. As **Lists** (and **Sets**) import the **Integers** module, queries to the **EuropeanUnion** program can use all the facilities provided by the **Integers**. There are 20 system modules. These support arithmetic, input/output, meta-programming, and a number of common data types such as lists, strings, sets, and tables.

The **Sets** system module is based on the standard mathematical concept of a set but can only handle finite sets. As the order and duplication of elements are irrelevant in a set, a special equality theory is supported. For instance, with the **EuropeanUnion** program, the queries:

```
<- {England, France} = {France, England, France}.
<- x = {y : y IsIn Britain & ~Component(WestYorkshire, y)}.
```

will succeed with the answer for the second query:

```
 x = {Wales, England, WestYorkshire}.
```

Gödel has a flexible computation rule that can be restricted by means of control declarations. For example, the **DELAY** declaration in the **EuropeanUnion** program ensures that an atom of the form *s* `IsIn` *t* is not selected if *t* is a variable so that any call to `IsIn` must terminate with success, failure, or flounder. (Floundering occurs if every literal in the current goal is delayed.) In addition, there is a pruning operator which has two main forms: the *one solution commit* that indicates that once a solution for the query is found, then ignore any other possible solutions; and the *bar commit* which is similar to the commit of the concurrent logic programming languages.

Considerable emphasis is placed on the metalogical facilities for meta-programming; that is, analysis, transformation, compilation, verification, debugging, and so on of other programs called object programs. These facilities consist of special system modules as well as certain utilities that create a representation of the object programs that is suitable for manipulation by the meta-programs.

The clean separation of the control annotations from the logic and the expressive syntax makes Gödel particularly suitable for executable specifications. The declarative nature of the language makes it a good teaching language; enables advanced software engineering tools such as declarative debuggers and compiler generators; and offers substantial scope for parallelisation. Gödel has already been used in a number of research projects. These include work on self-applicable partial evaluators, declarative debuggers, program synthesis, program algebras, set processing, and data parallel implementations of logic programs.

The book "The Gödel Programming Language" by Pat Hill and John Lloyd, published by MIT Press describes this language. An implementation is available and can be obtained by anonymous ftp from the directory `goedel` at: `ftp.cs.bris.ac.uk`. More information about Gödel can be obtained from the web page: `http://www.cs.bris.ac.uk/~bowers/goedel.html`.

FACS-E Recent books column

Cliff Jones

August 27, 1995

I agreed to produce listings of books which relate to the purpose of this newsletter. Authors should send references in BibTeX format to `cbj@cs.man.ac.uk`; we try to pick up some citations without authors intervention so you can also check via WWW `http://www.dit.upm.es/~cdk/fac.html` to see whether your reference has been noted.

References

[AS94] S Abiteboul and E Shamir, editors. *Automata, Languages and Programming*, volume 820 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[BA92] M Ben-Ari. *Mathematical Logic for Computer Science*. Prentice Hall International, 1992.

[Bab91] R L Baber. *Error-Free Software:Know-how and Know-why of Program Correctness*. Wiley, 1991.

[BC93] M. Bidoit and C. Choppy, editors. *Recent Trends in Data Type Specification*, volume 655 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[BC94] Daniel P Bovet and Pierliugi Crescenzi. *Introduction to the Theory of Complexity*. Prentice-Hall International, 1994.

[BH94a] J P Bowen and J A Hall, editors. *Z User Workshop, Cambridge 1994*. Workshops in Computing. Springer-Verlag, 1994.

[BH94b] J. P. Bowen and J. A. Hall, editors. *Z User Workshop, Cambridge 1994*, Workshops in Computing. Springer-Verlag, 1994.

[BKL+91] M. Bidoit, H.-J. Kreowski, P. Lescanne, F. Orejas, and D. Sannella, editors. *Algebraic System Specification and Development: A Survey and Annotated Bibliography*, volume 501 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[BMM+94] S Brookes, M Main, A Melton, M Mislove, and D Schmidt, editors. *Mathematical Foundations of Programming Semantics*, volume 802 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[BN93] J. P. Bowen and J. E. Nicholls, editors. *Z User Workshop, London 1992*, Workshops in Computing. Springer-Verlag, 1993.

[BN94] H Barendregt and T Nipkow, editors. *Types for Proofs and Programs*, volume 806 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Bun94] A Bundy, editor. *Automated Deduction-CADE-12*, volume 814 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Cas94] Chris Casey. *A Programming Approach to Formal Methods*. McGraw Hill, 1994.

[Coo93] Martin Cooke. *Modelling Auditory Processing and Organisation*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1993.

[Cra91] I. Craig. *The Formal Specification of Advanced AI Architectures*. AI Series. Ellis Horwood, September 1991.

[CS90] D. Craigen and K. Summerskill, editors. *Formal Methods for Trustworthy Computer Systems (FM89)*, Workshops in Computing. Springer-Verlag, 1990.

[Dah94] Ole-Johan Dahl. *Verifiable Programming*. Prentice-Hall International, 1994.

[Dav93] Jim Davies. *Specification and Proof in Real-Time CSP*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1993.

[de 95] E O de Brock. *Foundations of Semantic Databases*. Prentice-Hall International, 1995.

[Dil94] D L Dill. *Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Dix91] A. J. Dix. *Formal Methods for Interactive Systems*. Computers and People Series. Academic Press, 1991.

[dS93] H.C.M. de Swart. *LOGIC: Mathematics, Language, Computer Science and Philosophy. Volume I: Logic: Mathematics, Language and Philosophy*. Verlag Peter Lang, 1993.

[dS94] H.C.M. de Swart. *LOGIC, Volume II: Logic and Computer Science*. Verlag Peter Lang, 1994.

[Edm92] D. Edmond. *Information Modeling: Specification and Implementation*. Prentice Hall, 1992.

[EJOR91] H. Ehrig, K.-P. Jantke, F. Orejas, and H. Reichel, editors. *Recent Trends in Data Type Specification*, volume 534 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[EO95] H. Ehrig and F. Orejas, editors. *Recent Trends in Data Type Specification*, volume 785 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[FH92] Norman E Fenton and Gillian A Hill. *Construction and Analysis of Complex Systems: A Mathematical and Logical approach*. McGraw Hill, UK, 1992.

[FJ92] L.M.G. Feijs and H.B.M. Jonkers. *Formal Specification and Design*. Cambridge University Press, 1992.

[GO94] D M Gabbay and H J Ohlbach, editors. *Temporal Logic*, volume 827 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Gol93] Leslie Ann Goldberg. *Efficient Algorithms for Listing Combinatorial Structures*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1993.

[Gor94] Andrew D Gordon. *Functional Programming and Input/Output*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1994.

[Gro95] The RAISE Method Group. *The RAISE Development Method*. BCS Practitioner Series. Prentice Hall, 1995. ISBN 0-13-752700-4.

[Hah94] Reiner Hahnle. *Automated Deduction in Multiple-Valued Logics*, volume 10 of *International Series of Monographs on Computer Science*. Oxford University Press, 1994.

[HB95] M. G. Hinchey and J. P. Bowen, editors. *Applications of Formal Methods*. Prentice-Hall International, 1995. ISBN 0-13-366949-1.

[HG92] C A R Hoare and M J C Gordon. *Mechanised Reasoning and Hardware Design*. Prentice-Hall International, 1992.

[HJ95] Michael G Hinchey and Stephen A Jarvis. *Concurrent Systems: Formal Development in CSP*. McGraw Hill, 1995.

[Hun94] W A Jr Hunt. *A Verified Microprocessor*, volume 795 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[HWW95] Hoon Hong, Dongming Wang, and Franz Winkler, editors. *Algebraic Approaches to Geometric Reasoning*. J. C. Baltzer AG Scientific Publishing Company, 1995.

[Imp91] M. Imperato. *An Introduction to Z*. Chartwell-Bratt, 1991.

[JGS94] Neil D Jones, Carsten K Gomard, and Peter Sestoft. *Partial Evaluation and Automatic Program Generation*. Prentice-Hall International, 1994.

[Jon94] Mark P Jones. *Qualified Types: Theory and Practice*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1994.

[KMR94] J Karhumäki, H Maurer, and G Rozenberg, editors. *Results and Trends in Theoretical Computer Science*, volume 812 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[KR94] Haim Kilov and James Ross, editors. *Information modeling: an object-oriented approach*. Prentice-Hall, 1994. ISBN 0-13-083033-X.

[KST93] J. Koebler, U. Schoening, and J. Toran. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser, 1993.

[Lal94] Rene Lalement. *Computation as Logic*. Prentice-Hall International, 1994.

[Lig91] D. Lightfoot. *Formal Specification using Z*. Macmillan, 1991.

[LV93] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov complexity and Its Applications*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1993. ISBN 0-387-94053-7/ISBN 3-540-94053-7.

[McC92] Francis Gregory McCabe. *Logic and Objects*. Prentice-Hall International, 1992.

[Mil94] G.J. Milne. *The Formal Specification and Verification of Digital Systems*. McGraw-Hill, 1994.

[Mor94] Carroll Morgan. *Programming from Specifications*. Prentice-Hall International, second edition, 1994.

[MP93] M. A. McMorran and S. Powell. *Z Guide for Beginners*. Blackwell Scientific, 1993.

[MV93] S. Mauw and G.J. Veltink, editors. *Algebraic specification of communication protocols*. Cambridge Tracts in Theoretical Computer Science 36. Cambridge University Press, 1993.

[MV94] C. C. Morgan and T. Vickers. *On the Refinement Calculus*. Formal Approaches to Computing and Information Technology series (FACET). Springer-Verlag, 1994.

[Nic90] J. E. Nicholls, editor. *Z User Workshop, Oxford 1989*, Workshops in Computing. Springer-Verlag, 1990.

[Nic91] J. E. Nicholls, editor. *Z User Workshop, Oxford 1990*, Workshops in Computing. Springer-Verlag, 1991.

[Nic92] J. E. Nicholls, editor. *Z User Workshop, York 1991*, Workshops in Computing. Springer-Verlag, 1992.

[NM94] A Nerode and Y V Matoyasevich, editors. *Logical Foundations of Computer Science*, volume 813 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[NN92a] F. Nielson and H. R. Nielson. *Two-Level Functional Languages*. Cambridge University Press, 1992.

[NN92b] H. R. Nielson and F. Nielson. *Semantics with Applications: A Formal Introduction*. Wiley, 1992.

[NS91] A. Norcliffe and G. Slater. *Mathematics for Software Construction*. Series in Mathematics and its Applications. Ellis Horwood, 1991.

[Pau94] I C Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Pfe94] F Pfenning, editor. *Logic Programming and Automated Reasoning*, volume 822 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Plü90] Lutz Plümer. *Termination Proofs for Logic Programs*, volume 446 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, 1990.

[Pop94] S Popkorn. *First steps in Modal Logic*. Cambridge University Press, 1994.

[PST91] Ben Potter, Jane Sinclair, and David Till. *An Introduction to Formal Specification and Z*. Prentice Hall International, 1991.

[PW95] Jochen Pfalzgraf and Dongming Wang, editors. *Automated Practical Reasoning: Algebraic Approaches*. Springer-Verlag, 1995.

[Rat94] B. Ratcliff. *Introducing Specification Using Z: A Practical Case Study Approach*. International Series in Software Engineering. McGraw-Hill, 1994.

[Ros94] A. W. Roscoe, editor. *A Classical Mind: Essays in Honour of C. A. R. Hoare*. Prentice-Hall, 1994.

[Roz94] Grzeorz Rozenberg. *Cornerstones of Undecidability*. Prentice-Hall International, 1994.

[SBC92] S. Stepney, R. Barden, and D. Cooper, editors. *Object Orientation in Z*. Workshops in Computing. Springer-Verlag, 1992.

[She95] Deri Sheppard. *An Introduction to Formal Specification with Z and VDM*. McGraw Hill, 1995.

[Ste93] S. Stepney. *High Integrity Compilation: A Case Study*. Prentice Hall, 1993.

[Ste94] Graham A Stephen, editor. *STRING SEARCHING ALGORITHMS*. World Scientific Publishing, 1994. ISBN 981-02-1829-X.

[Sze95] A. Szepietowski. *Turing Machines with Sublogarithmic Space*, volume 843 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[TC91] M M Tanik and E S Chan. *Fundamentals of Computing for Software Engineers*. Van Nostrand Reinhold, 1991.

[Ten91] R. D. Tennent. *Semantics of Programming Languages*. International Series in Computer Science. Prentice-Hall International, 1991.

[Ten94] R D Tennent. *Semantics of Programming Languages*. Prentice-Hall International, 1994.

[The92] The RAISE Language Group. *The RAISE Specification Language*. BCS Practitioner Series. Prentice Hall International, 1992.

[Til94] D Till, editor. *6th Refinement Workshop*. Workshops in Computing. Springer-Verlag, 1994.

[TM94] J G Turner and T L McCluskey. *The Construction of Formla Specifications: An Introduction to the Model-based and Algebraic Approaches*. McGraw Hill, 1994.

[Wat94] David A Watt. *Programming Language Syntax and Semantics*. Prentice-Hall International, 1994.

[Wu94] Wen-tsün Wu. *Mechanical Theorem Proving in Geometries: Basic Principles*. Springer-Verlag, 1994.

International Workshop on User Interface Design for Theorem Proving Systems

Nicholas A. Merriam

1 Bringing Together Interest Groups

On the 18th July 1995 in Glasgow, a workshop on Interface Design for Theorem Proving Systems took place. This was organised by Phil Gray, Tom Melham, Muffy Thomas, members of the Department of Computing Science at the Univerisity of Glasgow, and joint investigators with Stuart Aitken in the EPSRC sponsored User Interface Design for Theorem Provers project.

Traditionally, user iterface designers and researchers have not seen theorem proving as an interesting domain for particular treatment. Likewise, architects of theorem proving assistants (TPAs) have been more concerned with functionality than usability or aesthetics. The task of the workshop was to bring together researchers in the fields of human-computer interaction (HCI) and theorem proving and to foster cross-over research.

2 Presentations

The following is a list of the talks given together with an extremely brief description of the content. The order here is the same as the order of presentations at the workshop.

Assessing Theorem Proving Assistants: Concepts and Criteria
Nicholas A. Merriam, Andrew M. Dearden, Michael D. Harrison
Three activities were suggested as being key in making theorem proving viable: the *reuse* of earlier proof work, the *planning* of future proofs and *reflection*[1] to analyse on the state of a proof in progress.

User Interface Design for an Automated Pedagogic Tool
Jeremy Pitt
The MacKE program was described. This is an application with a graphic user interface (GUI) which supports theorem proving based on the **KE** calculus.

The User Interface of a Proof Assistant For Z Specifications
Ian Toyn
Cadiz is a set of tools for manipulating Z specifications, including facilities for proof construction. The iterface to Cadiz was described and comments were given on various design decisions.

Diagrams and Human Reasoning
C. A. Gurr
The use of diagrams in the representation of proofs was discussed. In particular the way the humans interpret such diagrams was considered with the intention of informing the design of diagrammatic languages.

User Interface Principles for Proof Editors
Richard Bornat, Bernard Sufrin

[1]Here reflection is *not* being used in the theorem proving sense, *i.e.* relating logics at two different levels.

Principles of user interface design as applied to TPAs were discussed with reference to the Jape TPA. Simplicity was particularly advocated and interface designers were urged only to add a feature after careful deliberation on its true value.

A Co-Operative Interface for Theorem Proving Using Proof Plans
Helen Lowe, Alan Bundy
Proof search in automatic theorem proving can be guided by proof planning. This planning activity must frequently be supplemented by user intervention, and such interaction is supported by the Barnacle interface, decribed in this talk.

Interactive First-Order Deduction with BDDs
Joachim Posegga, Klaus Schneider
It was observed that first-order binary decision diagrams (BDDs) have a graphical representation. An interface for an interactive first-order theorem prover was described which employs such a representation.

Interactive Proof Discovery: An Empirical Study of HOL Users
Stuart Aitken, Philip Gray, Tom Melham, Muffy Thomas
An experiment to determine the activities of HOL users was described. This revealed that due to various errors, user performance varied widely. The sources of these errors were identified and quantified.

The User Interface and the Proof Process
Brian Matthews
This talk focussed on the B-Toolkit and Mural, two TPAs. It was noted that the B-Toolkit provides automation at the expense of transparency, while in contrast, Mural's open style requires the user to perform virtually all the proof by hand.

Presenting Machine-Found Proofs
Xiaorong Huang
The difficulty of reading machine generated proofs was discussed. The PROVERB system was described, which attempts to present a proof using a higher level than the raw output from a theorem prover and generates "natural language".

A Generic Approach to Building User Interfaces for Theorem Provers
Laurent Théry
This talk addressed the issue of building a user interface suitable for use with a variety of proving engines. The appropriateness of machine representation of proof elements communicated between software components was highlighted, for example.

Implementing TkHolWorkbench — Lessons From Rapid Prototyping
Donald Syme
TkHolWorkbench provides a GUI for HOL, building on the platform of Tcl/Tk. The development of this software was described and considerations in its implementation were discussed.

3 Notes on the Discussions

Following the presentations, there was a discussion session, where the small amount of HCI community interest was lamented. Phil Gray pointed out that HCI practitioners have not traditionally focussed on single application domains. The difficulty of funding relevant projects was mentioned and a more coordinated proposal-writing strategy was suggested as a means to attract grants. There was general agreement about the need to better establish the actual activities which are involved in industrial strength theorem proving, which include re-specification and theory management.

The implemented interfaces described by speakers were demonstrated in the final official session of the day, and informal discussions carried on into the evening and the following day.

4 Further Information

Proceedings of the workshop, edited by Philip Gray, are published by the Department of Computer Science at the University of Glasgow.

Whilst not the first such workshop, the Glasgow meeting was seen as beginning a series of regular annual wōrkshops. The next will be held on the 19th of July 1996 in York. The organisers will be Michael Harrison, Andy Dearden and Nick Merriam at the Univeristy of York Department of Computer Science.

A mailing list has been created for related discussions and the dissemination of important information on user-interfaces for theorem provers: **uitp@dcs.gla.ac.uk**. To be added to this list, please mail a request to **uitp-request@dcs.gla.ac.uk**. Update information on next year's workshop will be posted to this list.

The program for the Glasgow workshop is available on the world-wide web. See the deparmental web site: **http://www.dcs.glasgow.ac.uk/**

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@lut.ac.uk

FACS Officers

Chair	John Cooke	D.J.Cooke@lut.ac.uk
Treasurer	Roger Stone	R.G.Stone@lut.ac.uk
Committee Secretary	Roger Carsley	roger@westminster.ac.uk
Membership Secretary	John Cooke	D.J.Cooke@lut.ac.uk
Newsletter Editor	Ann Wrightson	scomaw@zeus.hud.ac.uk
Liaison with BCS	Margaret West	mmwest@scs.leeds.ac.uk
Liaison with FME	Tim Denvir	timdenvir@cix.compulink.co.uk