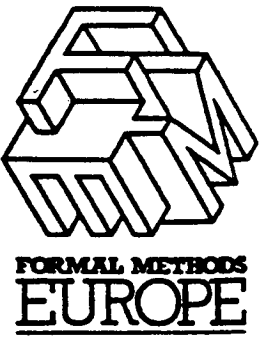

FACS Europe

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

Series I Vol. 3, No. 2, Autumn 1996 ISSN 1361-3103

1 Editorial

As you can see, the style of the newsletter has changed a little for this issue. This is part of moving to fully electronic production, and will hopefully ensure a much smoother operation, with fewer delays in publication. The next issue will be the Xmas Workshop special edition, and will be with you, by grace of the usual gremlins, by the end of January.

Since the last newsletter we have had two more workshops, hard on each others' heels in September. They were quite different events, with FAHCI serving a specialized interdisciplinary community, and FMNorth providing a very welcome opportunity for FM folk in the Northern reaches of these islands to meet and catch up with each others' concerns. FM North was the

brainchild of the Dept of Computer Science at the University of Bradford-we could do with more ideas as good as that one! Both will run again (see below).

The abstracts of all the papers are (FAHCI) or will be soon (FMNorth) available (free) on the Springer website, *http://www.springer.co.uk*

To access the full papers you need passwords which are derived from Springer's printed booklet of abstracts. Please pass any user feedback on this arrangement to John Cooke, who is currently reviewing our workshop publication experiences via eWiC over the past year.

From FME, this issue features the Formal Methods Applications Database. I hope this snapshot encourages you to explore the website in detail-I certainly found it was an interesting browsing ground.

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 FACS Events

2.1 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. All FACS members should have had full details and a booking form in the mail by now; if you need more information, please contact Liz Bromley, Centre for HCI Design, City University, Northampton Square, London, EC1V OHB. Tel: 0171-477-8427. Fax: 0171-477-8859. E-mail: E.M.Bromley@city.ac.uk

2.2 Partial Theories Workshop, April 1997

This is planned as an informal workshop to bring together several research communities which use and combine theories which are intended to describe parts or aspects of an application or domain. Design pattern frameworks, viewpoints, and configuration control are examples of areas where this arises naturally. If there is enough interest, and interesting convergence, then we

hope to use this workshop as a springboard for a larger event in 1998. Full details will be available as soon as practicable via the FACS website, and in the next newsletter issue.

If you are interested in taking part, please contact one of the organisers:
John Boarder, *100064.1533@CompuServe.COM*
Alan Wills, *alan@trireme.com*
Ann Wrightson, *a.m.wrightson@hud.ac.uk*

2.3 FM North 1997

FM North will run again in 1997, this time as a joint workshop of BCS-FACS and Bradford University Dept of Computer Science. The date is yet to be set precisely, but will be in July to avoid clashing with other events, eg FME. Watch this space for further details in the Spring. In the meantime, if you have any enquiries, contact Andy Evans, a.s.evans@comp.brad.ac.uk.

2.4 BCS-FACS Xmas Workshop 1997

This will happen, somewhere in London, at the usual time. Details will hopefully be available in the next newsletter.

2.5 8th Refinement Workshop

Planned for 1998, but no details as yet.

2.6 FAHCI

FAHCI is planned to be reincarnated in September 1998, and possibly to run as a regular event every two years.

2.7 FACS Anniversary?

FACS will be 20 years old on 30 Nov 1997 (if you count the semi-official inaugural meeting) or 16 May 1998 (if you take the first meeting date we were 'official' with respect to the BCS). If you have any ideas for a suitable celebration, please let us know.

3 FME events

3.1 FMEIndSem - Formal Methods Europe Industrial Seminars

FMEIndSem, a project sponsored by the European Commission under the ESSI (European Systems and Software Initiative - Software Best Practice), is producing a series of seminars across Europe aimed at giving software developers an insight into the reality of formal methods applied in industry.

The seminars are aimed at managers, senior technical staff and software development specialists, especially those with little or no prior experience of formal methods. No technical background is required. Speakers include experienced formal methods practitioners from Formal Methods Europe as well as specially invited guests.

There will be three tours, each consisting of three or four seminars in different countries. In each tour the seminars will be delivered by the same speakers. These collectively are: John Fitzgerald (University of Newcastle), Andrew Butterfield (K&M Technologies), Nico Plat (Cap Volmac), Jan Storbank Pedersen (CRI), Peter Gorm Larsen (IFAD), Jan Ekman (Logikkonsult), Hans-Martin Hoercher (Deutsche System-Technik), Maddelena Cinnella (SSI), Eric Delalonde (Cap Sesa), Peter Lucas (Graz University of Technology). The dates of the tours are:

```
Tour 1
Dublin, Ireland            15 May 1997
London, England            23 May 1997
Utrecht, The Netherlands   29 May 1997

Tour 2
Garching, Germany       17 February 1997
Birkerod, Denmark       18 February 1997
Stockholm, Sweden       19 February 1997
Helsinki, Finland       20 February 1997

Tour 3
Rome, Italy      7 April 1997
Vienna, Austria  9 April 1997
Paris, France    11 April 1997
```

For further details see *http://www.ifad.dk/projects/fmeindsem.html*

3.2 FME '97

Formal Methods: Their Industrial Applications and Strengthened Foundations, Graz, Austria, 15-19 September 1997. See insert in last issue for full details, or FME website.

4 (A few) Other Events

4.1 SAFECOMP'97

The 16th International Conference on Computer Safety, Reliability and Security, University of York, September 8th-10th, 1997

An annual event reviewing the state of the art, experiences and new trends in the areas of computer safety, reliability and security. The conference focuses on critical computer applications, and is also a platform for technology transfer between academia, industry and research institutions.

For more information see *http://www.cs.york.ac.uk/safecomp-97*

4.2 FMPPTA'97

International Workshop on formal methods for parallel programming; theory and applications. This workshop is being held on one day (April 1, 1997) of the 11th International Parallel Processing Symposium IPPS'97; there is no separate registration for the workshop. More information from *http://www.loria.fr/conferences/FMPPTA97*

4.3 Lfm97: Fourth NASA LaRC Formal Methods Workshop

Hampton, Virginia, 10-12 Sept 1997. The purpose of this workshop is to bring together leading formal methods researchers and practicing engineers in an environment in which each group can learn from the other. The three previous workshops have been limited to invited presentations, but the 1997 workshop will include a small number of submitted papers. Further information from *http://atb-www.larc.nasa.gov/Lfm97/.*

4.4 Report on ENCRESS 1996

This ENCRESS (European Network of Clubs for REliability and Safety of Software) conference took place in Paris on 13th and 14th June 1996. Its predecessor had been in Bruges in September 1995 and was combined with the regular UK CSR - Centre for Software Reliability - event. This second

ENCRESS conference was arranged to coincide with and be at the same site as the Software Testing 96 conference and exhibition. The French software engineering club, SEE, organised the local arrangements.

The attendance was considerably lower than at ENCRESS 95, but this was certainly at least partly because for the first time the conference was detached from the CSR event. This was a reasonable step to take; there had been a feeling at ENCRESS 95 that it was too dominated by UK delegates and speakers. To detach ENCRESS 96 from the UK event was a remedy for this, and was also a necessary step in the "coming of age" of ENCRESS.

Peter Barrett welcomed delegates and gave a brief report on the ENCRESS network of clubs. The network is alive and growing. Clubs exist in ten European countries and more are being sought. New features are the newsletter, issue 4 of which was distributed to delegates, "application groups" and a WWW server. Five application groups have been set up. ENCRESS 97 is to take place in Athens and plans are well in hand.

National clubs are active, holding their own events and activities. For example, the Swedish and Danish clubs held a joint meeting to discuss medical devices. Seventeen ENCRESS and closely related events within twelve calendar months are listed in the ENCRESS Newsletter. Five application groups have been formed. These are:

1. Safety Applications of Computer Based Systems for the Process Industry
2. Software Controlled Medical Devices
3. User Needs
4. Fault Tolerance Techniques
5. Risk and Hazard Analysis

The presentations started with a tutorial: "Application of Formal Methods in the Development and Validation of Railway Control Systems" by Allessandro Fantechi, of the Universita di Firenze, Italy. The speaker gave a condensed version of a twenty-hour course. He was clearly very well informed and had many good points to make. In the second half of the two and a half hour tutorial, he gave an account of two applications of formal methods to railway control systems. These were the use of B for the subway speed control system for the Paris Metro (SACEM - see Some Recent Applications in this newsletter), and the use of CCS and temporal logic in the

Ansaldo Interlocking System (B, CCS and temporal logic are all examples of formal methods).

There were ten presentations in all, covering safety management, software process improvement, hazard analysis, metrics and testing as well as formal methods. A great deal of audience interaction took place; by the end of the conference, most of the delegates had had discussions with nearly all the other delegates present. ENCRESS 97 promises to be an interesting event.

(Thanks to Tim Denvir for this report, and for the information on FME below.)

5 Electronic Information Sources

5.1 FME Information Resources

This project, sponsored by the European Commission under ESSI - European Systems and Software Initiative, Software Best Practice - is developing on-line databases of Formal Methods tools, applications, frequently asked questions. The databases are being expanded during the course of the project.

http://www.cs.tcd.ie/FME/

5.2 FMEGuides

This project, sponsored by the European Commission under ESSI - European Systems and Software Initiative, Software Best Practice - is developing multimedia management guide books accessible through servers. It is intended to enable managers to read success-stories about the use of formal methods. The material will include videos, magazine articles, and images and video sequences usable by TV producers.

http://demain.cgs.fr/formal/

5.3 Formal Methods Applications Database

In the context of the project FMEInfRes (Formal Methods Europe Information Resources, partially funded by the EC under the ESSI programme, project number 21375)a database has been set up with descriptions of industrial applications of formal methods. Much of the experience gained with that industrial use is not available to the outside world. Such experience can be very useful in avoiding the pitfalls one may encounter when starting

to use formal methods in a 'serious way. The database gives concise descriptions of - either successful or unsuccessful - applications of formal methods/ specification languages that will allow users to assess:

- whether or not formal methods/specification languages are being used for large, industrial applications;
- what the difficulties/advantages are of applying formal methods/ specification languages on a large scale in a domain of interest;
- for which application domains formal methods/specification languages are being used and which formal methods/specification languages are actually being used.

The database is now available and can be accessed through the FME Website.

If you know of any applications of formal methods/specification languages worth considering for inclusion in the database (i.e. you have been involved yourself or you know someone who was involved with such an application) then it would be appreciated if you would provide the following information and send it to Nico Plat, the database administrator, at the address below. The information required is:

Name: Name of the application or a one-line description of it.

Developed by: Main organisation(s) involved with the development of the application.

Formal method or specification language: Name which formal method or which formal was used for the development.

Tools used: Name of the tool used (if any).

Domain: A short characterisation of the application domain in which the formal method/specification language has been applied, e.g."medical systems", "railway systems" or "Air-traffic control (ATC) systems".

Period: Period during which the application has been or will be developed. This is relevant information because as time passes, the time during which something has been developed can be used as an indication of the state of the art that has been applied.

Size: A rough indication of the "size" of the application, in terms of lines of specification/ source code of the implementation, or in terms of human resources, eg the number of man-years involved.

Short description: A short (maximum 15 lines) description of the application and the way formal methods/ specification languages were applied on the application, e.g. did you apply formal proofs? Which mode of working did you use, i.e. separate specification teams/implementation teams, did you consult a formal method "guru", did you use formal methods/specification languages in conjunction with other software engineering techniques (e.g. Yourdon). Etc.

Conclusions: Any (subjective) observations on the application of formal methods/specification languages for your application. Do you feel that application of formal methods/specification languages for the case you describe was successful? If so/not, why/not? Etc.

Relevant publications: Adequate references to relevant publications, if any.

Contact: Name + address details (postal address, phone, fax, e-mail) of a contact person who may be approached if a person wants to have more information about a case.

URLs: URLs to any web pages which are relevant to the application described.

Further remarks: Any further remarks which you think may be relevant.

An electronic copy of the form can be obtained by sending an e-mail message containing only the line: send vdm-forum fm-appl-db-form.txt to mailbase@mailbase.ac.uk, and the form will be send back to you. Alternatively, you can pick it up at the URL above or by anonymous ftp from ftp.ifad.dk, directory pub/vdm.

Please send your contributions to (e-mail preferred): Nico Plat, Cap Volmac, P.O. Box 2575, 3500 GN Utrecht, The Netherlands. Fax: +31-30-2522234. E-mail: Nico.Plat@ACM.org (please use the format given in the database form as above).

5.4 EPSRC/LMS MATHFIT programme

MATHFIT is MATHs For IT, a successor to the LOGFIT programme many of you may remember.

Mathfit has recently started, and runs for 3 years. Its aim is to encourage interdisciplinary research broadening the use of maths in IT, through project grants, visiting fellowships, and workshops and summer schools. The priority areas which have been identified are:

- Algorithms and Structures
- AI, in cluding learning, neural computation, planning and reasoning
- Complex, communicating and concurrent systems
- Computer graphics, robotics and vision
- Principles of programming languages, including semantics and language design
- Networks, telecommunications and information security

More details are available from
http://www.epsrc.ac.uk/progs/area/it_cs/mfitcall.htm

5.5 Foresight: High Integrity Real-Time Software Working Party

This working party has prepared a report, which is available electronically. It has quite a lot to say about formal aspects, especially in suggesting areas where new approaches are needed. No big surprises, but an interesting read all the same. Here are a few extracts to give the flavour:

Timing requirements: "There has been considerable formal work...it is not clear that further theoretical work of this kind is required.....

needto

integrate such notations and forms of analysis into the mainstream design methods...The move towards more effective scheduling is intrisically linked to improvements in capturing temporal requirements and design decisions that have temporal implications."

Architectures and partitioning: ...software tools for system capture and formal specification based on graphical interfaces (CAD packages)...the application of formal methods to distributed asynchronous systems.

Alternative V&V Methods: ...formal proofs of correctness have only made limited inroads into V&V...regarded as expensive, difficult and unproved on large scale projects...(Formal proofs are typically made on the high level source code and provide no analysis of the actual low level target code.

System Integration and Reuse: ...The problem is how to devise software architectues that minimise the dependence of one componenet on the integrity of another...define architectures for large systems that are 'defensive'...specify... key properties that would have to be proven to assure the decoupling.

The full report, with email contact for comments, is available from
http://www.npl.co.uk/npl/collaboration/partners/foresight/index.html

5.6 Formal Methods Group at Bradford University

Formal Methods Group [Dr. A.N.Clark, Mr. A.S.Evans, Dr. K.P.Coplan, Dr. D.R.W.Holton, Dr. L.M.Lai, Prof. I.S.Torsun, Dr. P.Watson]

This recently expanded group includes 7 academic staff, one of whom is a post-doctoral research fellow, and most of whom are recent appointments. Other members of the department who do not consider themselves to be full members of the group have also worked in the Formal Methods field. The group's activities are concerned with basic and applied research into the specification and verification of hardware and particularly software.

Current areas of particular interest to the group are:

- *Application of Z to the specification and development of concurrent and real-time systems*: this has resulted in a set of sound techniques for applying Z to such systems, all based on the standard Z notation. Techniques for improving the refinement of Z specifications are also being investigated.

- *CSP- and CCS-like languages for concurrency*: a unified model for CSP-like languages with specification statements has been developed, and based on this model a refinement calculus has been developed which formalises program development for CSP-like languages.

- *Production Cell Controllers*: the use of formal methods for the analysis production cells. In particular, the benefits of stochastic process algebras to analyse functional and performance characteristics is being investigated.

- *Real-time specification*: We are very interested in the general advantages and disadvantages of current real-time notations. A number of these have been tested out on the Generalised Railway Crossing problem with interesting results.

- The application of temporal logics to *Multi-Agent Systems* and Computer Graphics.

The group also organises a yearly seminar series, which has attracted speakers from both home and abroad. We also co-organise with FACS an

international workshop on formal methods (The Northern Formal Methods Workshop).

Full details of our work can be found at *http://www.comp.brad.ac.uk.*

(Thanks to Andy Evans for this contribution - similar entries from other research groups are very welcome for future issues.)

5.7 Schwerpunktprogramm Deduktion

The Germanwide research programme "Deduktion" is funded by the "Deutschen Forschungsgemeinschaft" (DFG, roughly: German Research Foundation) and brings together almost all research groups engaged in the field of automated reasoning within Germany. The program started in 1992, and the current, final phase lasts from 1996 to 1998.

Up-to-date information is available from *http://www.uni-koblenz.de/ag-ki/Deduktion/*; this includes a list of participants, events, systems, the 'DFG-syntax' and a collection of theorem proving problems discussed in the 'Schwerpunkt'.

6 Theory and Practice of System Design 7th Refinement Workshop, Bath, UK, 3-5 July 1996

The 7th refinement workshop was small and quiet compared to some of its predecessors, but nevertheless interesting and enjoyable. It brought together a wide variety of interests and appraches, with a mixture of long-standing themes and new ventures. The social side was also pretty good: the American Museum, a privately owned collection of American art and artefacts, made an interesting venue for the first social event, with a museum tour followed by a tasty buffet on a pleasant garden terrace. Then our old friends Praxis made sure we weren't at a loose end on the second night, by generously providing a reception at the end of a lightning coach tour of Bath.

The 8th refinement workshop will happen in 1998; this much was decided at the closing session, though the exact date and venue are yet to be arranged.

6.1 Invited Lecture: C A R Hoare, The ProCoS Project

There is lots of information about this project available electronically from Oxford, via *http://www.comlab.ox.ac.uk/archive/procos.html.* Perhaps the

best thing to summarize here is the response to a question from R J Back concerning the project's industrial significance.

> We spent millions of ECU, and when it finished it came to an end ... there is still work going on. There are also a number of other projects concerned with correctness, not all of which have heard of ProCoS, for example there are 2 industry-led projects which are starting from scratch. Industry contacts generally didn't turn up to working group meetings, and the keen people moved into academic jobs ... though some people who worked on ProCoS moved into industry, eg into Danish railways. The effect is long term. It creates a climate for further development, which we can only see in hindsight.
>
> An important technical conclusion has been that there is no single solution to all problems. Different approaches are needed, and also to move between them; more people in the theoretical and tool-building community are realizing that contributions will be made from many different threads, with no one of them a 'best' approach.

And any lessons for future projects? "Once you have a DPhil student working on a notation you can't change it!"

6.2 Models for Configuration Management of Refinement Calculus Developments

Kelvin J Ross, University of Queensland. kjross@cs.uq.oz.au

This paper described a way of supporting configuration control of formal developments, by making use of the formal structure of the development to drive the configuration control structure. The formal connexions in a refinement calculus development help in devising appropriate configuration control by making it clear where connexions are needed, but also put an added burden on configuration control, because of the formal reasoning required.

The model presented is intended to help with development of automated support for refinement calculus developments, eg for change impact analysis and traceability. This work is part of a wider research programme on fine-grained configuration control at the University of Queensland - see Peter Lindsay's paper in FAHCI, September 1996, for another aspect of this work.

6.3 Separating Algorithm and Implementation in Refinement of Parallel Program Specifications

Denis Roegel, CRIN-CNRS and INRIA-Lorraine. roegel@loria.fr

This paper outlines a method for separating algorithms and data in refinement towards parallel programs, called *task separation*. Starting from a specification in TLA^+, the first stage introduces an algorithm, but without constraining data structures; then a second stage goes the rest of the way. The intermediate state is represented in an object-oriented way. The method is illustrated using block decomposition in matrix multiplication.

6.4 Development of Concurrent Systems in B AMN

K Lano, Imperial College, and J Dick, B-Core. kcl@doc.ic.ac.uk

The lack of support for concurrency in B has been a major criticism from industrial users. This paper outlines techniques for concurrent specifications using extensions to the B AMN language, using linear temporal logic, and Ada-style task definitions. The extension is then demonstrated using the 'production cell' example.

6.5 On Compositionality in Refining Concurrent Systems

Q W Xu, UN University, Macau. qxu@iist.unu.edu

Compositionality in three styles of refinement is investigated here. Refining complete systems is not at all compositional; there is a middle level where the refinement of modules is followed by testing their compatibility using an interference freedom test. Last, there is a more novel method where the concept of rely-guarantee is used to allow compositional refinement. These methods are suited to different situations; not surprisingly, one conclusion drawn is that the compositional method is most suitable when the systems are loosely coupled, and becomes ineffective when the interactions between processes are complex.

6.6 Invited Lecture: R J Back, Interpreting Nondeterminism in the Refinement Calculus

Paper by R J Back and J von Wright, Åbo Akademi University. backrj@abo.fi

This lecture was a wide-ranging discussion of 'angelic' and 'demonic' nondeterminism, using a simple programming language and its predicate transformer semantics as a vehicle for the discussion.

6.7 Probabilistic Predicate Transformers

Carroll Morgan, Oxford University PRG, carroll@comlab.ox.ac.uk

Initially billed as 'Proof rules for probabilistic loops', later amended to 'An introduction to probabilistic predicate transformers', this was an introduction to the use of probabilistic predicates and their transformers. Probabilistic predicate transformers provide a semantics for imperative programming languages which can describe sequential randomized algorithms; the semantics makes it possible in principle to prove their correctness and efficiency, that is, to calculate the probability of establishing a desired result and the expected number of steps needed to do so.

This is joint work with Karen Seidel and Annabelle McIver from Oxford PRG.

6.8 Calculational Derivation of Algorithms on Tree-based Pointer Structures

Michael Butler, University of Southampton. M.J.Butler@ecs.soton.ac.uk

Pointer structures present characteristic problems for refinement. This paper builds on Möller's approach to linked lists, which generalizes to tree structures, but with considerable overheads. These are partly caused by side-effects, and partly by changing from an applicative style of specification (natural for an abstract tree type) to an imperative style (natural for algorithms on pointer structures).

This paper describes a way of overcoming these difficulties by devising abstract representations of commonly used pointer manipulations on trees, and providing calculational-style refinement rules for these manipulations.

6.9 Procedures in the Refinement Calculus: A New Approach?

Lindsay Groves, Victoria University of Wellington, NZ.
lindsay@comp.vuw.ac.nz

This approach to handling procedures takes 'separation of concerns' further into program design, so that algorithm design is firmly separated from the arrangement of code into packages. Refinement concentrates on showing that the original problem can be solved constructively, deferring questions about how these results are combined to obtain a program. One stated conclusion is that 'in designing a formal development method, we must consider carefully the practical implications of the theorems we prove.' True, true!

6.10 A Tool for Developing Correct Programs by Refinement

D Carrington, I Hayes, R Nickson, G Watson and J Walsh, University of Queensland. {davec,ianh,nickson,gwat,jim}@cs.uq.edu.au

The Program Refinement Tool (PRT) is a refinement tool providing theorem-prover assistance with applying refinement rules, and for proof obligations; an explicit proof paradigm supporting contexts in refinement and proof; extensible theories; and a user interface supporting reuse between program derivations. The paper details the perceived requirements fulfilled by PRT, and compares it with other tools supporting refinement.

6.11 Invited Lecture: B Moszkowski, Using Temporal Fixpoints to Compositionally Reason about Liveness.

Ben.Moszkowski@ncl.ac.uk

This lecture discussed the use of assumptions and committments [closely related to rely/guarantee conditions...but for some reason needing a different name...] in Interval Temporal Logic, and what you need to add to handle liveness and safety together. The resulting analysis is shown in detail for several examples, including absence of deadlock, and mutual exclusion, using an extension of Owicki & Gries' proof-outlines to present the reasoning.

6.12 Experiences in Applying the Formal VSE Development Method in Industry

F Koob, M Ullmann, S Wittmann, Bundesamt für Sicherheit in der Informatik, Bonn. vse@bsi.de

The VSE (Verification Support Environment) is a formal specification and verification tool. Its specification language, VSE-SL, uses abstract data types and abstract state transitions; the tool is designed to allow integrated use of formal and semi-formal development techniques. The paper describes VSE in terms of its underlying principles, development method, the tool architecture, and its use in eight industrial pilot projects concerned with safety and/or security (Sicherheit).

6.13 Design and Verification of a Coherent Shared Memory

He J, Oxford University, A McIsaac and G Barrett, SGS-Thomson Microelectronics. jifeng@comlab.ox.ac.uk,mcisaac@bristol.st.com

This work arose out of the architectural design of a family of commercial 64-bit microprocessors. When load and store memory instructions are de-

fined in a model containing main memory and caches, two questions arise, which are addressed in this paper using formal models for specification and refinement. The questions both concern maintaining coherence of the caches; first, what extra machine instructions are required to make it possible, and second, what requirements on the software using these instructions will ensure that the caches do actually keep coherent.

6.14 Specification of the Dynamic Channel Selection (DCS) in DECT

N A Lobo, Nokia Mobile Phones, Surrey. noel.lobo@nmp.nokia.com

GSM and DECT are two methods for managing physical channels within a single communication link, used for linking the fixed and portable parts of mobile and cordless telephones respectively. This paper elucidates the relationship between the two by specifying the DECT algorithm as a refinement of a formal specification of GSM.

6.15 Invited Lecture: Zhou Chaochen, Chopping a Point

Paper by Zhou C, M R Hansen, TU Denmark. {zcc,mrh}@it.dtu.dk

This paper introduces super-dense computation into duration calculus. The point of this is to provide a way of treating a combination of very different time granularities, such as fast computation combined with slow sensor activity. A point in large-scale time maps to several points, or an interval, in fine-scale time. The dense points are ordered, though without duration. With this concept, the paper defines a real-time semantics for an OCCAM-like language.

6.16 On Using Syntactic Action Refinement to Derive Compositionally a Timed Efficient Implementation

A Dekdouk and A Schaff, CRIN/INRIA. dekdouk@loria.fr

This paper presents a durational process algebra incorporating syntactic action refinement. The language presented allows for non-instantaneous actions, thus allowing naturally refinement of actions by behaviour that contains explicit positive delays. A timed syntactic action refinement is given which is based on timed syntactic substitution, and is correct with respect to semantic refinement in a transition system model. The motivation is to support formal refinement into a timed efficient implementation for a specification.

6.17 A Real-time Refinement Calculus that Changes only Time

M Utting and C Fidge, University of Queensland. {marku,cjf}@cs.uq.edu.au

Introduced as 'How to implement Magic', this paper presents a way of using the refinement calculus for developing real-time programs from requirements expressed as possible traces. These trace-based specification statements and target language construct constrain the traces of system variables, rather than updating them destructively like the usual state-machine model. The resulting calculus allows refinement from formal specification with hard real-time requirements, to high-level language constructs with precise timing constraints.

7 User Interfaces for Theorem Provers 1996

Thanks to Stuart Aitken, stuart@dcs.gla.ac.uk, for this report.

The second international workshop on user interface design for theorem provers took place in York on the 19th of July 1996. This series of workshops, which began in Glasgow in 1995, aims to provide a forum for the exchange of ideas and research on the analysis and design of user interfaces for theorem proving assistants (TPAs). The 1996 workshop was organised by Nicholas Merriam, Michael Harrison and Andy Dearden of the Department of Computing Science, University of York.

The workshop had sessions on implementation issues, design principles for TPAs, and a session for position papers. There was also an opportunity for designers to demonstrate their interfaces and environments in the final session of the day.

The workshop proceedings include contributions from participants who presented papers and from those who gave system demonstrations. Copies are available from the organisers. (Copies of the papers can be accessed from `http://www.cs.york.ac.uk/ nam/uitp/proceedings.html`.) To be added to a mailing list created to discuss problems and issues in TPA interface design please send a request to: `uitp-request@dcs.gla.ac.uk`. It is intended that a third workshop in this field will take place in 1997, details to be announced on the mailing list.(Details will also be announced on the WWW site `http://www.dcs.gla.ac.uk/ stuart/ITP/uitp.html` which is a general site for TPA-interface related links.)

7.1 Implementing Interactive Proof

The use of structure editing and the question of the display of logical formulae was addressed by Bertot [3]. Bertot argued that formulae should not be displayed as trees, even though trees are the best internal representation in the prover interface code and are a possible means to communicate the structure of formulae. Structure editing should be combined with normal editing facilities as strict structure editing is too cumbersome. The design of CtCoq allows text areas to be used as annotations in a structured editing approach, where text can be freely entered into the annotation field, and checked at a later point in the interaction. Bertot also talked about proof by pointing and the generation of textual explanations from proofs.

Sufrin described Jape, and the underlying philosophy that the designer of a particular object logic should also responsible for the design of the presentation of the logic at the interface. The issue is then how to design a tactic language which permits the logician/designer to achieve both tasks. A key feature in the design of the interaction is the interpretation of gestures, i.e. the steps of inference performed by the prover that result from particular mouse clicks and menu selections. Sufrin explained the importance of a gesture context for the interpretation of user actions.

The IsaWin interface to Isabelle was described by Lüth. This interface uses SML/Tk to provide a principled functional approach to interface generation. The interface itself provides the user with an "assembly area" where objects denoting theorems and theories which are relevant to the proof can be collected, making use of a drag-and-drop metaphor.

7.2 Design Principles for Theorem Proving Assistants

Lowe began by noting that theorem provers are often developed in ad-hoc ways and are "never finished", perhaps because the desired level of competence is ill-defined. It was argued that provers should be more cooperative and a number of ways of achieving this in the Barnacle system were presented. These include providing explanations and appropriate proof presentations.

The importance of the surface detail of an interface was stressed by Bornat. The interface should look like a textbook, and only experts need to know the details of the implementation: these are further principles of the Jape proof editor. An interface should be declarative and not imperative, i.e. it should not require the user to calculate intermediate states. Bornat argued strongly against automation in proof assistants, and a discussion on

the merits of giving advice to the user took place (where advice might be offered as a result of some reasoning by the system).

The idea of proving as editing was presented by Hagiya. Proving a theorem can be viewed as writing a proof document and the structure of the proof can be seen as constraints on the proof document. The implementation of these ideas as an Emacs package was described, and two applications were presented.

7.3 Position Papers

Melham began by noting that much interface design effort addressed the tactic construction task, but that users actually engage in a wider range of activities while developing a theory in some domain. The proposals that there are definite phases of activity in theory development and that specific types of information might be required by users in each phase were explored. The design of an experiment to discover the frequency of phase changes and the use of information in each phase was described, and further debated in open discussion.

A formal approach to modelling interfaces was presented by Merriam who defined the concepts of complete and incomplete proofs in the Z notation. Merriam also described a less formal approach to understanding interface activity based on the action cycle theory of Norman. Both approaches were applied to the PVS prover.

The final talk was by Stevens who began with a critique of the capabilities of current automated and interactive provers and argued that a more robust approach was required. A number of ideas which might achieve this were presented, focussing on a scheme for recording the syntactic changes from theorem to theorem in a derivation.

7.4 Demonstrations

Valuable insights into the design and operation of a number of provers were gained during the demonstrations session. The algebra system Mathpad was demonstrated by Backhouse and the proof documentation and proof system Cadiz by Toyn.

The automated provers INKA and xClam which utilise proof planning were demonstrated by Hutter and Reid respectively. IPSA, a semi-automated prover using a proof plan as a display guide, was presented by Johansson. CtCoq, Jape, IsaWin and Hagiya's proof editor were demonstrated by their authors - given above.

References

[1] Stuart Aitken, Phil Gray, Tom Melham, and Muffy Thomas. Phases, Modes and Information Flow in Theory Development.

[2] Roland Backhouse and Richard Verhoeven. Mathpad: Tool Support for the Calculational Method.

[3] Janet Bertot and Yves Bertot. The CtCoq experience.

[4] Richard Bornat. Jape's quiet interface.

[5] Masami Hagiya and Hiroshi Kakuno. Proving as Editing.

[6] Dieter Hutter and Claus Sengler. The Graphical User Interface of INKA.

[7] Anna-Lena Johansson. Proof Planning, a Measurement for Effective Interactive Program Synthesis?

[8] Kolyang, C. Lüth, T. Meyer, and B. Wolff. Generating Graphical User Interfaces in a Functional Setting.

[9] Helen Lowe, Andrew Cumming, Michael Smyth, and Alison Varey. Lessons From Experience: Making Theorem Provers More Co-operative.

[10] Nicholas A. Merriam. Two Modelling Approaches Applied to User Interfaces to Theorem Proving Assistants.

[11] Gordon Reid. The Representation of the Rippling Heuristic in Proof Planning Using Coloured Annotations.

[12] Andrew Stevens. Toward Mechanised Revision of Proofs and Theories.

[13] Bernard Sufrin. User Interfaces for Generic Proof Assistants. Part I: Interpreting Gestures.

[14] Ian Toyn. Formal Reasoning in the Z Notation using CADiZ.

All of the above are published in the workshop proceedings, available from Nicholas Merriam, Department of Computer Science, University of York, at the cost £5 per copy.

8 TPHOLs'96

Thanks to Namhyun Hur, scomnh@zeus.hud.ac.uk, for this report.
This is a short summary of the proceedings of *The 9th International Conference on Theorem Proving in Higher Order Logics* (TPHOLs'96), held at

Turku, Finland in August 1996. The ever-growing numbers of attendants to the conference refelect the growing interests in theorem proving, especially using higher order logic. Started as the **HOL** system users meeting, it has now broadened its scope to include other hol-based theorem provers like Alf, Coq, Isabelle, LAMBDA, LEGO, Nuprl, and PVS.
The 27 papers in the proceedings can be roughly classfied by six categories as below. (Note that this is my personal classification.)

1. system merging and comparison (7 papers)
2. modelling and implementation of theories (8 papers)
3. system development (2 papers)
4. hardware verification (5 papers)
5. logic-related (3 papers)
6. others (2 papers)

Due to lack of spaces and knowledge we only describe few of the papers that are believed to be important or drew interests from me.

8.1 System merging and comparison

One paper particularly stands out in this category, *A mizar mode for HOL* by Harrison. What he did in this work is that he added the Mizar system's proof style to the existing HOL system's proof styles so that proof is now far more transparent and mathematical. This particular proof style closely resembles the everyday mathematician's style and thus gives more charm in theorem proving.

8.2 Modelling and implementation

Among the 8 papers in this category I'd like to describe the work by Dutertre, *Elements of Mathematical Analysis in PVS*. The motivation for this work, as one can guess, is from the need to model continuous domains like real numbers. Having reals and doing analysis in theorem provers is important. It expands the application areas into continuous systems like those in control industry.[1] But for serious applications to systems like aircraft flight control

[1]The elegant construction of real numbers by John Harrsion for HOL system is monumental in this respect.

system and spacecraft attitude control system we'd like to see more work to be done in this area.
For example, define a (3-dimensional) vector as a triple of reals and a (3 by 3) matrix as a triple of vectors. Imagine now you want to prove the associativity of matrix multiplication. This is really tedious from my personal experience. What one should do, I think, is to derive a general associativity theorem which enables us to rewrite products without parentheses.

8.3 System development

The paper by Konrad, *Function definition in Higher Order Logic*, explains how he built a function definition package for higher order logic based on the wellfounded recursion. This is an application of the celebrated recursion theorem from set theory to function deinition and thus is a valuable contribution to theorem proving.

8.4 Logic-related

The guest paper by Gordon, *Set Theory, Higher Order Logic or Both* identifies possible approaches for incorporating set-theoretical standardness and efficiency into the higher order logic theorem proving. This is a fundamental issue and thus is a long-term research topic. One has to understand both worlds to really appreciate the motivation. The two ways identified in the paper are (1) building set theory on higher order logic (2) building higher order logic on set theory. Between these the second approach is described as *still all fantasy.* Is anyone trying to encode higher order logic on top of Isabelle/ZF?
The proceedings are published by Springer, ed. von Wright, Grundy and Harrison, as LNCS 1125.

9 FACS Co-ordinates

9.1 FACS Central

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

Chairman	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	t-denvir@dircon.co.uk

Contributions to the Newsletter on any relevant topic are welcome. Please send them by email if possible, in LaTeX or plain text, to the Editor.

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