FACS

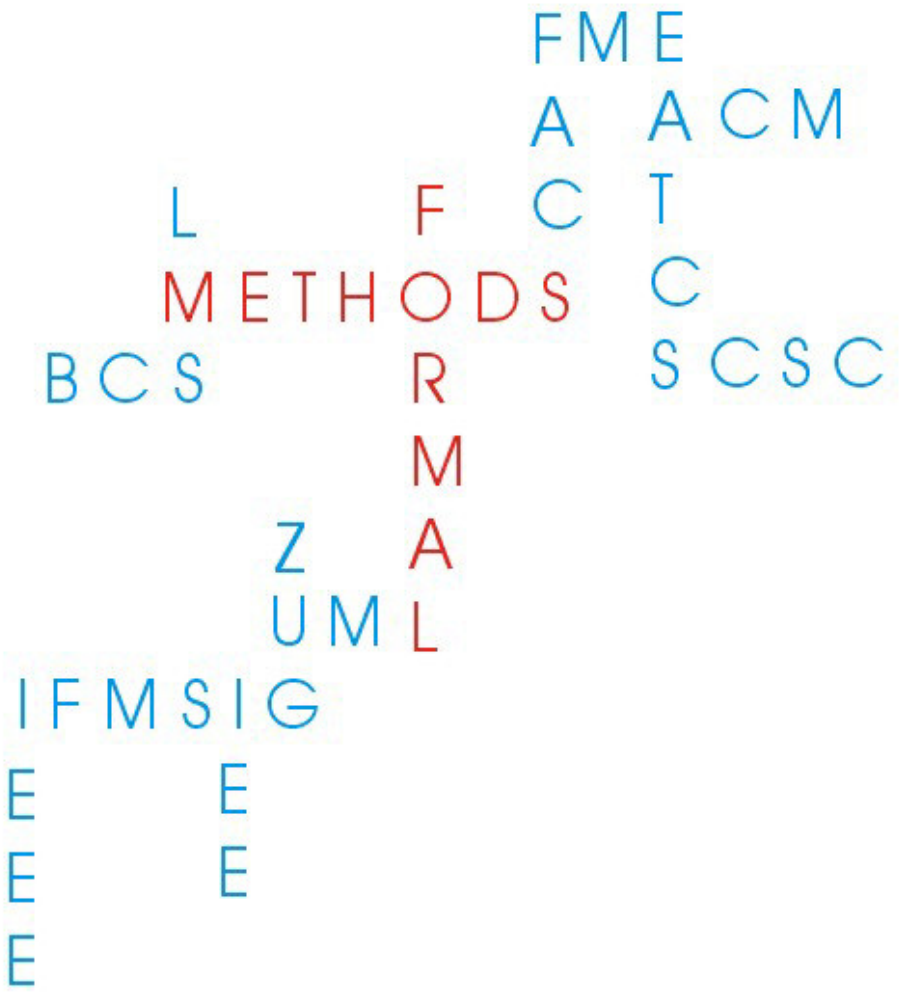
A

C

T

S

Issue 2004-3
November 2004



BCS FACS

The Newsletter of the Formal Aspects of Computing Science (FACS) Specialist Group

ISSN 0950-1231

About *FACS FACTS*

FACS FACTS [ISSN: 0950-1231] is the newsletter of the BCS Specialist Group on Formal Aspects of Computing Science (FACS). *FACS FACTS* is distributed in electronic form to all FACS members.

As from 2005, *FACS FACTS* will be published four times a year: **March**, **June**, **September** and **December**. Submissions are always welcome. Please see the advert on page 22 for further details or visit the newsletter area of the FACS website: http://www.bcs-facs.org/newsletter .

Back issues of *FACS FACTS* are available to download from:

http://www.bcs-facs.org/newsletter/facsfactsarchive.html .

The *FACS FACTS* Team

Contributors to this Issue

Jean-Raymond Abrial, Alvaro Arenas, Juan Bicarregui, Paul Boca, Eerke Boiten, Jonathan Bowen, Pat Browne, Andrew Butterfield, Judith Carlton, David Crocker, Issam Damaj, José Fiadeiro, Chris George, John Fitzgerald, Martin Henson, Neal Harman, Rob Hierons, Adrian Hilton, Mark Josephs, Kalpesh Kapoor, Petros Kefalas, Dexter Kozen, Shayoing Liu, Islam Ahmed El-Maddah, Savi Maharaj, Greg Michaelson, Teresa Numerico, Charles Rattray, Steve Reeves, Alexander Romanovsky, Anthony Seda, Carron Shankland, Helen Treharne

Contents

Editorial 4
Conference and Workshop Reports 5
SEEFM03: 1st South East European Workshop in Formal Methods, Thessaloniki, Greece, 20 November 2003 5
AMAST 04: 10th International Conference on Algebraic Methodology and Software Technology, Stirling, 12–16 July 2004 7
MPC 04: 7th International Conference on the Mathematics of Program Construction, Stirling, 12–14 July 2004 11
MFCSIT 2004: 3rd Irish Conference on the Mathematical Foundations of Computer Science and Information Technology, Ireland, 22–23 July 2004 13
FM in NZ 14
FMICS 2004: 9th International Workshop on Formal Methods for Industrial Critical Systems, Linz, Austria, 20–21 September 2004 15
The Nature of Proof, The Royal Society, 18–19 October 2004 16
FORTEST Workshop on Model-Based Testing IBM Hursley Laboratories, 21 October 2004 19
RefineNet Workshop, University of York, 7– 8 September 2004 23
25 Years of CSP 25
The first day: Theoretical issues of CSP 25
The panel discussion: History and future challenges of CSP 27
The conference dinner: Reminiscences of CSP's influence 29
The second day: Practical applications of CSP 29
Conclusion 30
Acknowledgements 31
References 31
Perfect Developer: What it is and what it does 32
An example: Specifying and refining a queue 33
Verifying security properties of the Mondex abstract world 36
Obtaining Perfect Developer 39
References 40
News from FACS 41
FACS Website gets Honourable Mention 41
FACS Committee Meeting, London, 8 July 2004 42
RODIN – Rigorous Open Development Environment for Complex Systems 43
LFSL '04 Summer School – a student's perspective 46
The right school for me! 46
The presentations 47
Social activities 47
Student Profile 47
News from FME 49
Adding Informal Analysis Ingredients to the Development Recipe 50
Workshop and Conference Announcements 55
Book Announcements 57
PhD Theses in Formal Methods 59
Jobs 65
FACS Committee 66
Formal Methods Coffee Time 68
And Finally 70

Editorial

Jonathan Bowen, BCS-FACS Chair

Welcome to a bumper edition of the *FACS FACTS* Newsletter. It is good to see the excellent response to the request for items in the newly invigorated newsletter. We have many reports on conferences and meetings, as well as information on tools, projects, PhD theses, etc. The size partially reflects the limited number of newsletters this year. In 2005, we hope to make *FACS FACTS* a more regular publication again with four issues per year.

Our major event in 2004 has been the highly successful CSP 25 Conference held at London South Bank University in July, celebrating 25 (or more!) years of Tony Hoare's *Communicating Sequential Processes* formalism, approximately coinciding with the 25th anniversary of FACS too. Indeed, the original version of CSP was presented by Tony Hoare at an early FACS event. Overall, this was an extremely enjoyable occasion that was very worthwhile from both a scientific and social perspective, with all the main CSP pioneers (in particular, Tony Hoare, Steve Brookes and Bill Roscoe) actively participating. Many thanks are due to Ali Abdallah, the FACS Events Coordinator, for organizing this conference almost single-handedly. In addition, Ali Nasrat Haidar and Michelle Hammond helped with local organization before and during the event. There is a full report included in this issue of *FACS FACTS* and the published proceedings will be available in due course.

Recently the FACS Committee met for an Away Day on Saturday 23 October to discuss the state and future of FACS. We welcomed Rick Thomas, Professor of Mathematics and Computer Science at the University of Leicester, as a new member of the Committee. His role will be to liaise with the London Mathematical Society. For a good part of the day, we considered the scope of FACS, including the collaborative development of a new "mission statement" under the expert facilitation of our treasurer, Jawed Siddiqi. It was felt that currently we did not reflect all the areas that we wished to cover, ranging from theoretical computer science to the practical application of formal methods. A proposed new draft mission statement reflects this wider remit. A fuller report of the Away Day is planned in the next *FACS FACTS* newsletter.

In December we have two meetings planned. The first, on 2 December, will be a follow-on meeting from one in 2001 on *Program Verification and Semantics*. Further information can be found on page 48. The second is partly a result of the desire of FACS Committee members to revive the traditional BCS-FACS Christmas Meeting and partly due to the Chair joining the Grand Challenge 6 Steering Committee on *Dependable Systems Evolution*, chaired by Tony Hoare. A day meeting on *The Verified Software Repository* is planned for 21 December at the new BCS offices near Covent Garden in central London. More information will be issued on the FACS electronic mailing list shortly.

I hope you enjoy this issue of *FACS FACTS* and encourage you to actively participate with contributions in 2005. I also hope that you can attend one of the December meetings. Do bring your colleagues along too!

Conference and Workshop Reports

SEEFM03: 1st South East European Workshop in Formal Methods, Thessaloniki, Greece, 20 November 2003

Petros Kefalas

The 1st South-East European Workshop in Formal Methods took place in Thessaloniki, Greece, on 20 November 2003. SEEFM03 was organized by SEERC and CITY College and hosted at the STEIN Building Auditorium. The event was sponsored by SINGULAR Software.

The workshop attracted participation from scientists, academics and researchers from Institutes and Universities all over Europe. More specifically, out of the 15 papers accepted for presentation, 5 of them were from Greece, 2 from UK, and 1 from Netherlands, New Zealand, Turkey, Romania, Finland, Russia, Slovenia, Singapore. For 13 of the papers, at least one author was related to South-East Europe.

Registration was free. There were 42 people registered for the workshop. The nationalities of the participants were as follows: Greece 19, Romania 5, Turkey 4, FYRoM 4, UK 3, Bulgaria 2, Serbia & Montenegro 1, Slovenia 1, Russia 1, and others.

The event commenced with welcome speeches from Dr. P. Ketikidis, Vice-Principal of CITY College and Dr. D. Stone, Director of SEERC. Both expressed their satisfaction with the number of participants and their willingness to find ways of promoting and organizing similar workshops in the future, as well as establishing links between academics in this research area in South-East European states.

Prof. M. Holcombe from the University of Sheffield was the invited speaker. He focused on ways that Formal Methods could contribute to agile methodologies in software development and presented his experience with

experimenting on agile formal methods on real-life projects undertaken by students at Sheffield.

Three sessions followed. Session A was on agile methodologies and testing. Session B and Session C were on theoretical foundations, verification and tools. All papers were very interesting and raised stimulating discussions among the participants. Each presentation lasted for 20 minutes allowing 5 minutes for questions and 15 minutes for discussion at the end of each session.

Online Resources

SEEFM03 website
http://www.city.academic.gr/seefm03/

SEEFM Website
http://www.seefm.info

Online access to proceedings
http://skyblue.csd.auth.gr/~bci1/SEEFM03/Index.html

In between the last two sessions, there was a panel discussion, which also involved questioning from the participants, chaired by Ms. A. Sotiriadou, on the general topic of "Trends in Formal Methods and Opportunities for South-East Europe". The panel consisted of Prof. F. Belli (Turkey), Prof. D. Kleftouris (Greece), Dr. M. Gheorghe (Romania) and Prof. M. Holcombe (UK). The outcomes of this discussion are summarised below:

- Formal methods are still a hot research area but there is still no evidence of wide acceptance.
- Industry needs to be convinced about the importance of using Formal Methods.
- Participants from the industry should be invited to express their opinions.
- Formal methods might be suitable for integration with agile methodologies for software development.
- Education on formal methods should be provided to students who will identify the future needs of the industry.
- The "human" factors should be taken seriously into account when using Formal Methods.
- The particular characteristics of people in SEE could be identified and due to their strong mathematical background, there might be a chance there to use these people in disseminating the practice on Formal Methods.
- The workshop provided a good chance to establish strong links between academics in SEE countries and those links should be exploited.

As a satellite workshop of the 1st Balkan Conference in Informatics, the proceedings of the workshop were included in the CD produced for BCI and reproduced in hard copies for the presenters of the papers. The CD is available on request from the SEEFM secretariat:

seefm03_secretariat@city.academic.gr.

The workshop concluded with a dinner in a Greek traditional taverna offered to all the presenters from the SEE countries.

Conclusions

- The workshop was the first attempt to bring people from SEE together, based on their common interest in Formal Methods.
- All participants expressed the opinion that the workshop is to be continued on an annual basis.
- It is thought that the workshop could be hosted in Thessaloniki a couple of times before it is opened to other interested parties.
- A small, flexible Steering Committee should be formed.
- A web page containing all links and email addresses of people in SEE should be constructed as a means to disseminate information and practices. The page should initially contain information collected during the workshop, e.g. people, institutions, research groups, interests, projects, courses etc.
- Common projects should be sought between various institutions in SEE with common research interests in Formal Methods.
- Every effort should be made to involve people from industry in the next workshop.
- Contacts should be made to seek publication of selected papers in an international journal.
- Discussion groups could be formed in order to maintain contact over the Internet at least once a month for those interested in participating in open discussions. The technological infrastructure for e-discussions will be investigated.
- The structure of the workshop might change to include more invited speakers and a second day that will be mainly devoted to panel discussions and focused discussions on specific work presented during the first day.

AMAST 04: 10th International Conference on Algebraic Methodology and Software Technology, Stirling, 12–16 July 2004

Charles Rattray, Savi Maharaj, Carron Shankland

The meeting of AMAST 04 at the University of Stirling was a great success: in total 71 delegates attended the conference. The meeting was co-located with MPC (The 7th International Conference on Mathematics of Program Construction), ARTS (the 6th AMAST Real Time Workshop), and CMPP (the 4th International Workshop on Constructive Methods for Parallel Programming). A report on MPC appears on page 11 of this issue of *FACS FACTS*.

AMAST itself began on Monday 12 July, with an opening invited talk by Muffy Calder from the University of Glasgow on “Abstraction for Safety, Induction for Liveness”. The remainder of the technical and invited programme was spread over 5 days. The programme was varied, with the invited talks representing the AMAST focus on developing software on a firm mathematical basis. The virtues of this approach have been envisioned as being capable of providing software that is (a) correct, and the correctness can be proved mathematically,

From left to right: Savi Maharaj (AMAST co-organizer), Carron Shankland (AMAST co-organizer), JJ Meyer (speaker), Roland Backhouse (speaker), Michel Bidoit (speaker), Bart Jacobs (speaker), Mike Johnson (AMAST Steering Committee Chair), Charles Rattray (AMAST 04 Chair)

(b) safe, so that it can be used in the implementation of critical systems, (c) portable, i.e., independent of computing platforms and language generations, and (d) evolutionary, i.e., it is self-adaptable and evolves with the problem domain. The AMAST proceedings are available as volume 3116 in the Lecture Notes in Computer Science series published by Springer. We are currently preparing a special issue of selected papers for publication in the Elsevier journal *Theoretical Computer Science*.

Representing software engineering was Don Batory from the University of Texas at Austin with “A Science of Software Design”, drawing together elements of software engineering and algebraic manipulation. A more high level formal approach was shown by Michel Bidoit, of the Specification and Verification Laboratory (LSV) of the French National Centre for Scientific Research (CNRS), with “Glass Box and Black Box Views of State-Based System Specifications” and Roland Backhouse of the University of Nottingham with “Algebraic Approaches to Problem Generalisation”. Bart Jacobs from the Radboud University, Nijmegen talked about a very practical application of formal methods in online voting in the Netherlands “Counting Votes with Formal Methods”. Finally, JJ Meyer, from Utrecht University, brought us up to date with evolutionary

Online Resources

AMAST website
http://www.amast.org

AMAST 2004 Website
http://www.cs.stir.ac.uk/events/amast2004/

AMAST 2004 proceedings
http://www.springeronline.com/sgw/cda/frontpage/0,11855,3-40109-22-32107332-0,00.html

ARTS 2004 Website
http://www.cs.le.ac.uk/events/ARTS2004/

computing approaches with his talk “Agent-Oriented Programming: where do we stand?”.

The ARTS workshop programme ran as part of AMAST, on Monday 12 July. The ARTS invited speaker was Jan Friso Groote from Eindhoven University of Technology, who gave a very interesting talk “Process Analysis Tools for the Next Generation: the muCRL toolkit”. More details about ARTS can be obtained from Dr Irek Ulidowski of the University of Leicester.

A highlight of the week, from a social point of view (!), was the Thursday afternoon excursion and conference dinner. After lunch at the University, we boarded buses which took us into the Trossachs, an area of outstanding natural beauty. Our helpful guides told us lots of interesting facts on the way (and some things which we thought were decidedly tall tales). Dinner was at the Winnock Hotel in Drymen. Entertainment was provided by members of the Callander Pipe Band who turned up to play in the main square in Drymen half way through our meal. We went out to enjoy the music and dancing, and managed to get back inside before the midges became too voracious.

The organizers are grateful to BCS-FACS for providing two best paper prizes. The best overall paper was awarded to Bernhard Möller (pictured left receiving prize) and Georg Struth for their paper “Modal Kleene Algebra and Partial Correctness”. There was also a best student paper prize, which was awarded to Sun Meng (pictured right receiving prize) for his contribution to the paper “On Refinement of Generic State-based Software Components”, written with Luís S. Barbosa. Both prizes consist of a year’s membership of FACS and a year’s subscription to the *Formal Aspects of Computing* journal.

The organizers would also like to thank Stirling Council for providing a civic reception on the opening night of AMAST. All our visitors were captivated by the young pipers who met us at the entrance to the Municipal Buildings. We would also like to thank the Edinburgh Mathematical Society and the London Mathematical Society for financial support for two of the speakers, and the Engineering and Physical Science Research Council for their support of UK based PhD students. Thirteen students were able to attend AMAST for only

the cost of their own travel. This was an excellent opportunity for them to meet members of their community from all over the world.

List of accepted papers:

Title	Authors
On Guard: Producing Run-Time Checks from Integrity Constraints	Michael Benedikt, Glenn Bruns
Behavioural Types and Component Adaptation	Antonio Brogi, Carlos Canal, Ernesto Pimentel
Towards Correspondence Carrying Specifications	Marius C Bujorianu, Eerke A Boiten
Formalizing and Proving Semantic Relations between Specifications by Reflection	Manuel Clavel, Narcissi Martí-Oliet, Miguel Palomino
Model-Checking Systems with Unbounded Variables without Abstraction	Magali Contensin, Laurence Pierre
A Generic Software Safety Document Generator	Ewen Denney, Ram Prasad Venkatesan
Linear Temporal Logic and Z Refinement	John Derrick, Graeme Smith
Formal JVM Code Analysis in JavaFAN	Azadeh Farzan, José Meseguer, Grigor Rosu
Verifiying a Sliding Window Protocol in mCRL	Wan Fokkink, Jan Friso Groote, Jun Pang, Bahareh Badban, Jaco van de Pol
State Space Reduction for Process Algebra Specifications	Hubert Garavel, Wendelin Serwe
A Hybrid Logic of Knowledge Supporting Topological Reasoning	B Heinemann
A Language for Configuring Multi-level Specifications	Gillian Hill, Steven Vickers
Flexible Proof Reuse for Software Verification	Chris Hunter, Peter Robinson, Paul Strooper
Deductive Verification of Distributed Groupware Systems	Abdessamad Imine, Pascal Molli, Gérald Oster , Michaël Rusinowitch
Formal Verification of a Commercial Smart Card Applet with Multiple Tools	Bart Jacobs, Claude Marché, Nicole Rauch
Abstracting Call-Stacks for Interprocedural Verification of Imperative Programs	Bertrand Jeunnet, Wendelin Serwe
On Refinement of Mobile UML State Machines	Alexander Knapp, Stephan Merz, Martin Wirsing
Verifying Invariants of Component-based Systems through Refinement	Olga Kouchnarenko, Arnaud Lanoix
Modelling Concurrent Interactions	Juliana Küster-Filipe

Proof Support for RAISE by a Reuse Approach based on Institutions	Morten P Lindegaard, Anne E Haxthausen
Separate Compositional Analysis of Class-based Object-Oriented Languages	Francesco Logozzo
Abstract Domains for Property Checking Driven Analysis of Temporal Properties	Damien Massé
Modular Rewriting Semantics of Programming Languages	José Meseguer, Cristiano Braga
Modal Kleene Algebra and Partial Correctness	Bernhard Möller, Georg Struth
Modularity and the Rule of Adaptation	Cees Pierik, Frank S de Boer
Modal Abstractions in mCRL	Jaco van de Pol, Miguel Valero Espada
Semantics of Plan Revision in Intelligent Agents	M. Birna van Riemsdijk, John-Jules Charles Meyer, Frank S. de Boer
Generic Exception Handling and the Java Monad	Lutz Schröder, Till Mossakowski
Expressing Iterative Properties Logically in a Symbolic Setting	Carron Shankland, Jeremy Bryans, Lionel Morel
Extending Separation Logic with Fixpoints and Postponed Substitution	Élodie-Jane Sims
A Formally Verified Calculus for Full JavaCard	Kurt Stenzel
On Refinement of Generic State-based Software Components	Sun Meng, Luís S Barbosa
Techniques for Executing and Reasoning About Specification Diagrams	Prasanna Thati, Carolyn Talcott, Gul Agha
Formalising Graphical Behaviour Descriptions	Kenneth J Turner
Model-Checking Distributed Real-Time Systems with States, Events, and Multiple Fairness Assumptions	Farn Wang

MPC 04: 7th International Conference on the Mathematics of Program Construction, Stirling, 12–14 July 2004

Dexter Kozen

The 7th International Conference on the Mathematics of Program Construction (MPC 2004) was held in the beautiful surroundings of the campus of University of Stirling, Scotland, 12–14 July 2004.

The MPC series aims to promote the development of mathematical principles and techniques that are demonstrably useful in the process of constructing computer programs, whether implemented in hardware or software. The focus is on techniques that combine precision with conciseness, enabling programs to be constructed by formal calculation. Within this theme, the scope of the series is very diverse, including programming methodology, program specification and transformation, programming paradigms, programming calculi, and programming language semantics.

There were 33 registered participants for MPC as well as several other attendees from three co-located conferences and workshops: the 10th International Conference on Algebraic Methodology and Software Technology (AMAST 2004), the 6th AMAST Workshop on Real-Time Systems (ARTS 2004), and the 4th International Workshop on Constructive Methods for Parallel Programming (CMPP 2004). (A report on AMAST can be found on page 7 of this issue of *FACS FACTS*.) The conference participants were treated to two and a half days of technical talks of very high quality in the beautiful surroundings of the University of Stirling campus, as well as a lovely banquet hosted by the restaurant Chambo in nearby Bridge of Allan.

The proceedings of the conference have appeared as volume 3125 in the Lecture Notes in Computer Science (LNCS) series published by Springer. This volume contains 19 outstanding papers selected for presentation by the program committee from 37 submissions, as well as the abstract of one invited talk: "Extended Static Checking for Java" by Greg Nelson of HP Labs in Palo Alto, California.

This MPC conference was the seventh in a series. The previous six were held in 1989 in Twente, The Netherlands; in 1992 in Oxford, UK; in 1995 in Kloster Irsee, Germany; in 1998 in Marstrand near Göteborg, Sweden; in 2000 in Ponte de Lima, Portugal; and in 2002 in Dagstuhl, Germany. The proceedings of these conferences are available as LNCS 375, 669, 947, 1422, 1837, and 2386, respectively.

The conference was sponsored by Cornell University, the University of Stirling, and the Formal Aspects of Computing Science (FACS) Specialist Group. The conference organizers wish to take this opportunity to express their gratitude to these organizations for their generous support.

The FACS Specialist Group contributed a best paper award in the form of a membership and journal subscription for the author(s) of the best paper as judged by the program committee. The award was presented to Clare E. Martin and Sharon A. Curtis of Oxford Brookes University (pictured left with Dexter Kozen) and Ingrid Rewitzky of the University of Cape Town for their paper "Modelling Nondeterminism".

MFCSIT 2004: 3rd Irish Conference on the Mathematical Foundations of Computer Science and Information Technology, Ireland, 22–23 July 2004

Anthony Seda

The 3rd Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT 2004) took place in Trinity College Dublin (TCD) on 22–23 July, 2004. Once again, we were fortunate in having invited speakers of the highest calibre who gave talks as follows:

- "Information is Physical, but Physics is Logical" by Samson Abramsky of Oxford University,
- "Mathematics for Software Engineers" by David Parnas of University of Limerick,
- "Topological analysis of refinement" by Michael Huth of Imperial College, and
- "Using Multi-Agent Systems to Represent Uncertainty" by Joseph Halpern of Cornell University.

Online Resources

MFCSIT 2004 Website
http://www.cs.tcd.ie/MFCSIT2004/

MFCSIT 2002 Website
http://grobner.nuigalway.ie/MFCSIT2002/

MFCSIT 2000 Website
http://euclid.ucc.ie/pages/staff/seda/htdocs/uccconf.html

Submitted talks were given by:

- Georg Essl of Media Lab Europe: "Computation of Wavefronts on a Disk I: Numerical Experiments";
- John Power of LFCS, Edinburgh: "Discrete Lawvere Theories";
- Colm O hEigeartaigh and Mike Scott of Dublin City University: "A Comparison of Point Counting Methods for Hyperelliptic Curves over Prime Fields and Fields of Characteristic 2";
- Andrew Butterfield and Jim Woodcock of TCD and University of Kent: "Formal Models of CSP-like Hardware";
- Sharon Flynn and Dick Hamlet of NUI Galway and Portland State University: "Composition of Imperfect Formal Specifications for Component-based Software";
- Anthony Seda and Máire Lane of NUI, Cork and BCRI: "On the Integration of Connectionist and Logic-based Systems";
- Alessandra Di Pierro and Herbert Wiklicky of University of Pisa and Imperial College: "Operator Algebras and the Operational Semantics of Probabilistic Languages";
- Michael B. Smyth and R. Tsaur of Imperial College: "Convenient Categories of Geometric Graphs";
- S. Romaguera, E. A. Sánchez-Pérez and O. Valero: "The Dual Complexity Space as the Dual of a Normed Cone";
- Homeeira Pajoohesh and Michel Schellekens of NUI, Cork: "Binary trees equiped with semivaluations";
- Xiang Feng and Michael B. Smyth: "Partial Matroid approach to Geometric Computations";

- Micheal O Heigheartaigh of NUI, Dublin: “r-Chains in Graphs: Applications in Counting Hamiltonian Tours”; and
- Paul Harrington, Chee K. Yap and Colm O Dunlaing of Trinity College, Dublin: “Efficient Voronoi Diagram Construction for Convex Sites in Three Dimensions”.

The high quality of all these talks and the relaxed atmosphere in TCD ensured a scientifically valuable and enjoyable meeting.

The conference proceedings will again appear as a volume in ENTCS, Elsevier's series *Electronic Notes in Theoretical Computer Science*. More information can be found at www.cs.tcd.ie/MFCSIT2004. ■

FM in NZ

Steve Reeves

It would be fair to say that formal methods in New Zealand are in a very good state. In a country where almost all research funds go on killing possums (they eat all our native trees and were an import from Australia in the days when the plan to introduce a species for sport or food was not thought completely mad, not to mention cruel) or making sheep and cows as “productive” as possible or developing a new version of the Chinese gooseberry (now better known as the kiwi fruit), that we in the small FM community have won any funds for research is remarkable. However, we are thriving.

Recently, sitting in my room on a gloomy Hamilton morning I realized that we had the makings of a fine one-day conference (which sort of fits into a series of meetings we have had sporadically over the last ten years – some very up-to-date information is available at www.mcs.vuw.ac.nz/research/NZFPDC/) for next-to-no cost. “FM in NZ” was born! What follows is essentially the introductory remarks from the notes for that meeting. These, as well as abstracts etc., can also be seen at

www.cs.waikato.ac.nz/~stever/FMinNZ/index.htm.

“FM in NZ” was the result of several happy coincidences. Martin Henson (University of Essex) was making one of his (approximately) annual visits to Steve Reeves and the rest of the FM group at Waikato, so we invited Lindsay Groves and Ray Nickson (Victoria University of Wellington) up to visit. Though Ray Nickson was not able to come at short notice, Lindsay Groves was, so we booked him in for a departmental seminar.

Then, out of the blue, an email from Jing Sun at Auckland (who had visited us before) said that Jin Song Dong (his PhD supervisor) was going to be visiting him from Singapore. Many of us know Jin Song from various conferences

(and Mark Utting and Petra Malik here at Waikato are working with him on the CZT project) so, of course, we invited Jing and Jin Song to the day too.

At the time we happened also to have Ali Pouyan from Birjand University in Iran and Rohit Bansal from IIT in New Delhi in India visiting, so this was a truly multi-national day and too good a chance to miss getting everyone together, which we managed to do at extremely short notice.

Participants

Rohit Bansal, IIT, India
Judy Bowen, University of Waikato
Jin Song Dong, National University of Singapore
Lindsay Groves, Victoria University of Wellington
John Hamer, University of Auckland
Martin Henson, University of Essex, U.K.
Petra Malik, University of Waikato
Robi Malik, University of Waikato
Ali Pouyan, University of Birjand, Iran
Greg Reeve, University of Waikato
Steve Reeves, University of Waikato
Jing Sun, University of Auckland
David Streader, University of Waikato
Mark Utting, University of Waikato

Online Resources

FM in NZ talks
www.cs.waikato.ac.nz/~stever/FMinNZ/index.htm

Photos from the event
http://www.cs.waikato.ac.nz/~stever/FMinNZ/FM%20in%20NZ%202004.html

NZ Formal program development colloquium
http://www.mcs.vuw.ac.nz/research/NZFPDC/

A final comment: we're always pleased to receive guests, so please invite yourself!

FMICS 2004: 9th International Workshop on Formal Methods for Industrial Critical Systems, Linz, Austria, 20–21 September 2004

Juan Bicarregui, Andrew Butterfield and Alvaro Arenas

The 9th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 04) was held in Linz, Austria, during 20–21 September, as a co-located event of the 19th IEEE Conference on Automated Software Engineering. The workshop series promotes the use of formal methods for industrial applications by supporting research in this area and by serving as a forum for the exchange of ideas between researchers and practitioners, in both industry and academia. This workshop, organized by CCLRC Rutherford Appleton Laboratory, was attended by 35 participants from academia and industry from 16 countries.

The two keynote speakers gave interesting and stimulating presentations. Jeremy Dick from Telelogic spoke on linking formal methods to formal requirements describing how existing tools for supporting requirements traceability could be adapted to work with formal specification and refinement

documents. Cedric Fournet of Microsoft Research spoke on the verification of the security of XML-based web-services and described how the “applied” pi-calculus was used to analyse the safety of security policies, work that has contributed to recent revisions of Microsoft code. FMICS would like to thank both invited speakers for their relevant and highly informative contributions to the success of the workshop.

Seventeen submitted papers were presented with authors from 17 countries spanning formal methodologies as diverse as Statecharts, model checking, mixed intuitionistic logic and Boolean equation systems; and applications ranging from operating systems, network services, communications protocols and middleware behaviour, to flight guidance.

The best paper award supported by the European Association of Software Science and Technology (EASST) was awarded to Martin Fränzle and Christian Herde for their paper on proof engines for bounded model checking of hybrid systems.

Online Resources

FMICS 2004 Website
http://www.fmics04.cclrc.ac.uk/

FMICS 2003 Programme
http://www.inrialpes.fr/vasy/fmics/workshop-8/program.html

FMICS 2002 Draft Proceedings
http://www.inrialpes.fr/vasy/fmics/workshop-7/proceedings.pdf

Other papers presented included Object-Oriented concepts identification from formal B specifications; an Abstract Interpretation Toolkit for µCRL; Early Verification and Validation of Critical Systems; and Model Checking Flight Guidance Systems: from Synchrony to Asynchrony, among others.

The proceedings of the workshop are published as a technical report of the Johannes Kepler University and will appear in *Electronic Notes in Theoretical Computer Science*. Selected papers will be invited for publication in a special issue of *Formal Methods in System Design*

The organizers wish to thank FME and the i-Trust Working Group for sponsorship for the invited speakers. Participants enjoyed a good Austrian dinner courtesy of ERCIM.

The Nature of Proof, The Royal Society, 18–19 October 2004

Judith Carlton

During 18–19 October 2004, the Royal Society in London was the venue for a discussion meeting on “the nature of mathematical proof”. The discussion was based upon the idea that the “increasing use of computers both within mathematics and to automate mathematical reasoning has raised new questions about the nature of mathematical proof”, acknowledging that there are several different viewpoints.

The meeting was organized by:

- Professor Alan Bundy (University of Edinburgh)
- Professor Donald MacKenzie (University of Edinburgh)
- Sir Michael Atiyah, OM FRS (University of Edinburgh and Trinity College, Cambridge)
- Professor Angus MacIntyre, FRS (Queen Mary, University of London)

The first presentation of the meeting was by Donald MacKenzie on **"Computing and the cultures of proving"**. He provided a historical and sociological perspective and discussed what exactly constitutes a proof. Interestingly, there has been some litigation that hinges on the definition adopted.

MacKenzie asked if there is a difference between a "formal proof" and a "rigorous argument", and he answered that the latter provides a lower level of design assurance, illustrating with this diagram:

	Formal Proof	**Rigorous Argument**
Mechanized	Mainstream automated theorem proving	"Hard" artifical intelligence
Not Mechanized	Early logicism; Dijkstra's calculational proofs **	Ordinary mathematics; IBM "Cleanroom"

** MacKenzie said that there is an argument saying that humans just do not have enough self-discipline to operate in this quadrant

In the question time that followed, Dr Rod Chapman (Praxis) asked why many in the audience had laughed after Donald Mackenzie revealed that the source of the definitions he had presented for formal proof and rigorous argument was the Ministry of Defence rather than some renowned mathematician.

Dr Richard Lipton (Georgia Institute of Technology, Atlanta, USA) spoke on **"Social processes and mathematical proof in Mathematics & Computing: A quarter-century perspective"**.

Twenty-five years ago, he and others wrote a paper on how mathematics is a social process, arguing in particular: "Real proofs are tested and checked by a complex social process. One of the consequences of our position is that it is unlikely that real computer systems can or will ever be proved correct. The core of the argument is a careful examination of the difference between formal proofs and real proofs".

Lipton made the following points that I understood, but I'm afraid I've missed something:

- The social mechanism for the acceptance of a proof requires that the statement proved be simple and interesting (after Hilbert: explain to the first person on the street)

- This acceptance mechanism includes discussion of the proof with colleagues, presenting it, publication, being read, and being used for other proofs (this is critically important)
- Changes in computer technology have made this argument even more relevant today. For instance, Windows XP has of the order 100,000,000 lines of code. How can we hope to prove something of this size correct, given the social mechanisms involved in acceptance of proof?

Lipton finished his talk with a clip from a Monty Python film that illustrated the difficulty of providing a precise specification.

Professor Dr Henk Barendregt (Radboud University Nijmegen, The Netherlands) spoke on **"The challenge of computer mathematics"**.

"Progress in the foundations of mathematics has made it possible to formulate all thinkable mathematical concepts, algorithms and proofs in one language and in an impeccable way … The challenge is to make the systems more mathematician-friendly, by building libraries and tools. The eventual goal is to help humans to learn, develop, communicate, referee and apply mathematics".

I found that this was an extremely interesting and informative talk. It is too detailed to reproduce readily here, but I hope it may be the subject of a later article.

The final talk of the day was given by Professor Alan Bundy, who asked **"What is a proof?"**. He revealed that the logic-based tradition is largely a 20th century invention; earlier proofs had a different nature.

Bundy showed that errors in well-known proofs can remain undetected for many years, and successfully made the point that trying to prove something about undefined concepts can lead to error. He observed that, although it is quite common to find errors in historical proofs, such errors can usually be patched up quite easily. He also had an interesting card trick.

Once again, my notes are too detailed to reproduce here, but I would like to make this theme the subject of a future article.

The first day closed with a ninety-minute panel discussion on **"Formal versus rigorous proof for verification"**, chaired by Donald MacKenzie. The panellists were Rod Chapman, Professor Cliff Jones (University of Newcastle), Richard Lipton, and Professor Ursula Martin (Queen Mary, University of London).

The second day of this meeting was expected to have a heavier emphasis on mathematics, so, having no claim to be a mathematician, I did not attend. I am very glad to have been present on the first day, but there were many unanswered questions at the end of it. I await the Proceedings with interest. ■

FORTEST Workshop on Model-Based Testing IBM Hursley Laboratories, 21 October 2004

Fortest

Jonathan Bowen and Rob Hierons

On 21 October 2004, the EPSRC FORTEST Network on *Formal Methods and Testing* organized a workshop on model-based testing at the IBM Hursley Laboratories near Winchester. At the start of the workshop, Paul Gibson of IBM welcomed delegates and emphasized the potential importance of model-based testing to industry. He noted that however beautiful the theory, it must also work in practice and in particular it must be scalable.

Stuart Reid of Cranfield University gave a helpful general introduction to model-based testing, originally designed for presentation to industry, setting the general scene for the workshop well. Christopher Robinson-Mallett (Potsdam, Germany) gave a detailed technical presentation of work with Tilo Müche (Technische Universität Braunschweig, Germany) entitled "Automated Test Generation for State-based Specifications using UPPAAL", as previously presented at the TESTCOM Conference. Vlad Rusa (IRISA/INRIA, France) also presented technical work with Henré Marchard and Thiery Jéron on verification and symbolic test generation for safety properties.

Online Resources

FORTEST website
http://www.fortest.org.uk

Model-based testing
http://www.model-based-testing.org

Presentations given at FORTEST meetings
http://www.fortest.org.uk/index.shtml?documents

In the afternoon, Lee Onn Mak and Yongyan Zheng of the University of Surrey (a member of the FORTEST Network) gave a joint talk entitled "Test Execution and Generation for Digital Business Ecosystem". The work is funded as part of the European Information Society Digital Business Ecosystem (DBE) project. A Digital Business Ecosystem is an evolutionary self-organizing system aimed at creating a digital software environment for small organizations. Kirill Bogdanov (a lecturer at the University of Sheffield and an active FORTEST member) then presented "Agile Testing of an Interactive System using X-machines: A case study".

Paul Baker from the Motorola Laboratories near Basingstoke presented an industrial view of "Test Generation towards TTCN-3 Model-Driven Development (MDD)". TTCN-3 is a *Testing and Test Control Notation* (see www.ttcn-3.org). In combination with UML 2.0 and presentation formats it provides a sound basis for UML 2.0 testing profiles. It has been promoted for a number of years but acceptance is still difficult.

Mark Harman (who has just moved from Brunel University to take up a chair at King's College London) gave an excellent and motivational talk on "Transformation for Testability Improvement". By using a combination of global fitness and local fitness, some spectacular improvements in testing

speed have been obtained. Technical information is available in a recent journal paper (*IEEE Transactions on Software Engineering*, 30(1):3–16, 2004).

The workshop ended with a discussion session chaired by Prof. Rob Hierons, leader of the FORTEST Network, from Brunel University. The following is a summary of the issues raised and discussed:

Languages. We need languages that are formal and trendy. How do we make formal languages trendy? People are often interested in using, and developing expertise in, trendy languages and approaches as they see this as a way of enhancing their career prospects (see, for example, UML). Languages have to be sufficiently expressive but it is often useful if they are executable. Naturally, these two factors can clash.

Tools and case studies. We need convincing tools and case studies to support the use of model-based testing and to demonstrate its value. We cannot expect model-based testing to be widely used without tools to support it. We also cannot expect managers to invest in model-based testing without a significant amount of evidence regarding its effectiveness. Without such evidence, managers will see a guaranteed cost – training staff and buying tools and then the cost of modelling – with no guarantee of a return on this investment.

Design for test. Testing is cheaper and more effective if our software is designed to be testable. Since software testing is so expensive, design for test guidelines have the potential to lead to significant improvements in the development process. This is accepted practice in hardware development but not in software development.

Processes for model-based testing. How should we integrate model-based testing into the development process? Practitioners need guidelines. Maybe we need a maturity model for model-based testing?

Required models. What kinds of model do we need for model-based testing, or what types of information do we need? In model-based testing the model is used to drive and support testing. This is a rather different purpose to traditional specifications and design models and thus we may well require different types of models. We also have to be clear what we are modelling and how this will contribute to testing.

How can we reason about quality on the basis of testing? Having tested our system, we would like to know what this tells us about the system (beyond the fact that it contains certain traces). If we have a test hypothesis or a fault domain then we might be able to make further assertions about system behaviour. Possibly we could verify such test hypotheses or provide confidence in them?

Partial models. Models are often partial but many techniques assume that the models are complete. We need additional techniques that allow us to test (and reason) on the basis of partial models.

Semantic adequacy criteria. White-box test criteria typically insist that elements of the structure of the code are covered. They are thus syntactic in their nature and it is difficult to relate this sort of coverage to test effectiveness. It is thus desirable to have alternative criteria that are based on the semantics of the code, rather than syntax.

Prediction. Can we predict, for example, where failures are likely to occur and which test techniques will be effective? Can we do this on the basis of metrics gathered earlier in the development process?

Fault tolerance. How can we make systems fault tolerant without incurring a significant increase in costs? In addition, by building in fault tolerance it is possible to reduce testability since fault tolerance aims to mask faults (of course, in the extreme we mask all failures and have a correct system). There is thus a potential trade-off between fault tolerance and testability.

Diagnostics. If we observe failures how can we trace these back to faults? This is a difficult problem and it might be possible to choose tests that simplify the problem?

Education. Of developers, testers and users!

Finally, thank you to Clive Stewart of IBM who ably undertook the local organization for the workshop. The magnificent and beautiful surroundings at Hursley of the country house and landscaped gardens, saved by IBM in the 1950s, added to the atmosphere.

IBM Hursley Park

For further online information on the EPSRC FORTEST Network, see www.fortest.org.uk. For general information on model-based testing, see www.model-based-testing.org.

FACS FACTS Issue 2005-1

Call For Submissions

Deadline **11 February 2005**

We welcome contributions for the next issue of *FACS FACTS*, in particular:

- Letters to the Editor
- Conference reports
- Reports on funded projects and initiatives
- Calls for papers
- Workshop announcements
- Seminar announcements
- Formal methods websites of interest
- Abstracts of PhD theses in the formal methods area
- Formal methods anecdotes
- Formal methods activities around the world
- Formal methods success stories
- News from formal methods-related organizations
- Experiences of using formal methods tools
- Novel applications of formal methods
- Technical articles
- Tutorials
- Book announcements
- Book reviews
- Adverts for upcoming conferences
- Job adverts
- Puzzles and light-hearted items

Please send your submissions in plain text or Microsoft Word format to Paul Boca [Paul.Boca@virgin.net], the Newsletter Editor, by **11 February 2005**.

If you would like to be an official *FACS FACTS* reporter or a guest columnist, please contact the Editor.

RefineNet Workshop, University of York, 7 – 8 September 2004

Adrian Hilton, RefineNet Reporter for *FACS FACTS*

RefineNet is an EPSRC-funded collaboration of UK university research departments and firms from industry. The third network meeting was held at the University of York on 7 – 8 September, and examined existing real-world systems where modern refinement techniques could be applied. The definition of a real-world problem, coined by Susan Stepney during the workshop, was when "you can't change the specification if it becomes too hard!"

The main case study for the workshop was the publicly available specification, refinement and proof of the Mondex Purse [1].

Tuesday's session started with Susan Stepney (University of York) describing an application of "Specification Mutation" to the Purse CSP specification; model-checking mutated specifications against the Purse security properties yielded interesting information about the redundancy and abstraction present in the original specification. Jim Woodcock and Steve King (both from the University of York) then described a joint project: Woodcock attempted to recast and restructure the Mondex proofs so that they could be mechanised with a high degree of automation in the Z/Eves tool, and King applied ProofPower-Z to automate the existing specification. Several issues with effective use of the tools arose, and are detailed in the online workshop notes.

Online Resources

York meeting notes
http://www.refinenet.org.uk/york_notes.html

RefineNet website
http://www.refinenet.org.uk

Refinement workshop (call for papers)
http://www.refinenet.org.uk/cfp_rw.html

Wednesday could reasonably be described as "technically intense", with John Derrick (University of Sheffield) starting by identifying cases where refinement fails: an infinite buffer and an arbitrary integer adding machine which are impossible to implement exactly in the real world. Richard Banach (University of Manchester) and Mike Poppleton (University of Southampton) then led a detailed tutorial on Retrenchment, a generalisation of refinement that enables developers to reason about cases like this. The key difference is a "concedes" clause which enables the developer to specify that "A refines B except for a particular aspect C". John Derrick presented some ideas about approximations of refinement, including distance metrics for measuring how well observations of specification models agree.

David Crocker (Escher Technologies) gave a quick presentation on feeding the Purse abstract model and two security properties into *Perfect Developer* and trying a proof. (An article on *Perfect Developer* can be found on page 32 of this Issue.) It took about two hours to input the model, and *Perfect Developer* nailed the 23 proof obligations in a matter of minutes.

John Clark (University of York) finished off the afternoon by looking at refinement's relation to security and described various finalisation attacks on secure systems.

Notes of this meeting are online at

http://www.refinenet.org.uk/york_notes.html.

The next meeting will be in January, with location and exact date to be determined. The theme will be Foundations of refinement.

The RefineNet website is at www.refinenet.org.uk and includes a list of current members, details of past meetings, a schedule of future meetings and contact details. Enquiries are welcomed: enquiries@refinenet.org.uk

References

1. Stepney, S., Cooper, D. and Woodcock, J., “An Electronic Purse”. Technical Monograph PRG-126, Oxford University Computing Laboratory (July 2000).

DIRC BCS FACS

BCS-FACS Christmas meeting on

The Verified Software Repository

21 December 2004

BCS London Offices
The Davidson Building
5 Southampton Street
London WC2E 7HA

Meeting to be organized by Dr. Juan Bicarregui, Prof. Jonathan Bowen and Prof. Jim Woodcock, with advice and support from the Grand Challenge 6 Steering Committee Chair, Tony Hoare.

Meeting supported by DIRC and BCS-FACS

25 Years of CSP

Teresa Numerico & Jonathan Bowen

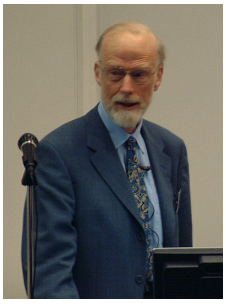
At London South Bank University on 7 – 8 July 2004, in the new Keyworth Centre, a conference was held to celebrate the 25th anniversary of the introduction of Tony Hoare's programming language CSP (Communicating Sequential Processes) [1]. The language, designed to facilitate synchronized communication between parallel processes, opened a new perspective for the development of efficient and consistent parallel processing techniques. After 26 years from that seminal article, the organizers of the conference (Ali E. Abdallah, Cliff Jones and Jeff Sanders) had the aim to focus experts' attention on the history of CSP, on the present state of the art in formal methods techniques for parallel and distributed systems, together with future strategies for improving their presence in various suitable applications. The event was organized with the invaluable support of the BCS-FACS (British Computer Society *Formal Aspects of Computing Science*) Specialist Group, which was particularly significant in this case, since Tony Hoare presented CSP at one of the first meetings of the BCS-FACS group, in 1978, demonstrating the longevity of both!

A welcome reception was held in the evening before the conference at the top floor of the new Keyworth Centre, with excellent views over London. At the start of the conference itself, Prof. Jonathan Bowen, Chair of BCS-FACS, and Prof. Deian Hopkin, Vice Chancellor of London South Bank University, welcomed delegates. There were around a hundred attendees in all, from the UK and abroad, and from academia and industry.

The first day: Theoretical issues of CSP

The conference examined the impact of the CSP on many different areas, from semantics to logic, from parallel programming language creation, to applications in various fields such as information security, web services, concurrent hardware circuits.

The invited speakers' background ranged from academia to industry, and from all over Europe. Such a variety of perspectives provided a great opportunity for dialogue and interaction between theoretical and practical approaches in the application of formal methods.



Tony Hoare, a scientist who embodied both aspects of formal methods, holding leading positions within industry and the academic world, was the keynote speaker at the conference. His contribution “Simulation and refinement: A unification” was seminal and innovative as usual. He explored the possibility of unification through similarities between the two most relevant theories of concurrency, creating a potential reconciliation between theories based on Robin Milner’s Calculus of Communicating Systems (CCS) [2] and the theories that have flourished from CSP. Exploiting the coincidence between trace refinement and similarity in classical deterministic automata, Hoare succeeded in extending a deterministic theory with nondeterminism using the reduction (commitment) relation that is itself a simulation. The *barbed* calculus is a familiar version of CCS that requires no change to the previous calculus in term of axioms, definition or theorem. He showed that the calculus allows one to work with deadlocks and divergence of processes as if they were terminal events in ordinary traces. By this means it is possible to reconcile similarity (which is a notion of CCS) and refinement (that belongs to barbed calculus), which was a long-term research goal. Moreover, according to Hoare, this achievement is likely to produce a lot of practical benefits. His work on the unification of theories was pursued, technically refined and simplified by the talk of He Jifeng “Linking theories of concurrency”, aiming at providing a link between model-based languages such as CSP, Z and transition system based languages, such as CCS, ACP.

The next section of the conference was dedicated to the semantics of CSP. Stephen Brookes and A.W. Roscoe, two of the fathers of CSP, who worked with Hoare in Oxford from the very beginning [3], introduced the debate about new solutions for the interpretation of the programming language. Brookes’ talk “Retracing the semantics of CSP” proposed a rethinking of the classical notions of semantics for CSP such as communication traces, failure sets, divergent traces, incorporating them into a new framework that allows a unified account of shared memory parallelism, asynchronous and synchronous communication. Such a framework permits a weak form of fairness that is functional to build models for compositional reasoning about liveness properties, safety properties and deadlock. The denotational semantics framework proposed avoids the model dependence from the hardware assumptions regarding the granularity of the actions. Roscoe’s talk “Seeing beyond divergence” focused on the possibility of an operational semantics that, provided with the suitable definition of a fixed point, allowed one to see something beyond the first divergence in CSP calculus. This new insight about processes within the divergence presented new analysis tools that were unthinkable in the context of a conventional denotational fixed-point for the same calculus.

Always relying on a theoretical approach but being more directly connected with its practical issues, Mark Josephs gave a talk on “Models for data flow sequential processes”. The idea of his contribution was the construction of various models of nondeterministic data flow based on a variant of CSP [4], a process algebra for data flow, with the aim of defining operators that are

convenient for constructing processes that could take place in the various models. In the same area of asynchronous circuits and systems, Ad Peeters, who works for Philips, presented in his talk “Handshake technology and CSP” the first practical solution for asynchronous data flow circuits, completely based on CSP: Handshake. The system can be used in smart card circuits, wireless applications and in-vehicle networking. CSP is used in all the different features of the application: programming language, circuit formalism, implementation of components, and it is the first large-scale commercial exploitation of self-timed technology.

The last section of the first day dealt with transactions and various CSP applications in this field. Michael Butler with his talk “Towards a process algebra for long-running transactions” developed a process algebra that copes with long-running business transactions, in which it is not possible to arrest the process without compensation and there are multiple agents involved in the process. Jonathan Lawrence, from IBM, proposed some techniques that may be used to model procedural design in CSP, in order for the program to be verified by the FDR model-checking tool, describing a case study of the IBM software service chosen to implement and to exemplify his ideas.

The panel discussion: History and future challenges of CSP

Panel: Jeff Sanders, Bill Roscoe, Stephen Brookes, Tony Hoare and Willem-Paul de Roever

At the end of the first day there was a panel discussion on the history and the future of formal methods in practical applications. Answering the questions of the chairman, Jeff Sanders, and of the audience, the Dutch computer scientist Prof. Dr. Willem-Paul de Roever, and some of the “Oxford golden boys of formal methods”, Brookes, Hoare and Roscoe, discussed the role of CSP-related methods in programming during the 1980s and the challenges for their future implementation in software writing strategies and/or code checking procedures. They all acknowledged that it has always been difficult to convey the importance of CSP with respect to language design, to people who are not experts in the field. Brookes introduced the issue that though formal methods had a strong impact on language design, sometimes it is very difficult to succeed in explaining their role to the industrial community.

Roscoe tried to analyse the lack of trust that characterized formal methods, remarking that the fault could be traced to the excess of success of the methods during the 1980s that produced a consequent opposition during subsequent years. According to Tony Hoare, with his complete experience in the field and duel roles in academia and industry – now holding a leading position as a consultant for Microsoft – the key element is the clear and effective separation of the roles of the scientists from the role of the entrepreneurs. In fact he stressed that it was often true that scientific discoveries were implemented successfully only much later than their origination. Scientists therefore have to be especially critical in their advisory roles about the feasibility of the implementation of their latest discoveries, in order to avoid the risk of undermining their reputations and the trustworthiness of their suggestions.

Hoare noted that Bill Gates has affirmed that he ignores the issue of when exactly Microsoft would be able to exploit the work that is done in its research laboratories. There are a lot of different research projects that are funded within Microsoft, but at the same time they are not prepared to support an official company position on formal methods and their use within various areas of software development. In particular, Hoare reminded the audience that there is much research being done regarding behavioural specifications and descriptions of the correct dynamic procedures of components, and also model checking strategies with special attention to relationships between programs and libraries.

De Roever claimed, instead, that people in industry were not keen on acknowledging that programs are based on CSP; the real challenge, therefore, should be to sell products that incorporate formal methods without mentioning them. He also complained about the lack of self-marketing capabilities of the English scientific community.

With regard to the failure of some of the products built using formal methods, Roscoe suggested that the mistake was not in formal methods research but in the attitude of thinking sequentially displayed by the programmers at present. In the future, everybody will get used to parallelism and will be ready to organize the parallel interaction of processes, and it will definitely happen in the next 25 years, according to him.

Hoare, underlining that it was not likely that a unique notation could be used to solve every problem, drew the conclusion of the panel discussion. His position implied that it was very important to maintain the separation between the “real world” and the “scientific world” avoiding taking into account scientists that recommend their own scientific ideas instead of the good ones, no matter who introduced them. The key challenge, for him, was not the use of CSP in all projects against all other systems, but the choice of the right solution for each problem, even if it were not based on CSP.

The conference dinner: Reminiscences of CSP's influence

The CSP 25 conference dinner was held in the Tower Restaurant at London South Bank University, run by the National School of Bakery. Prof. David May, FRS, previously of Inmos and now at the University of Bristol, gave an interesting and extended personal reminiscence on the influence of CSP, especially in the development of the Transputer processor and the associated Occam programming language, based on CSP.

The second day: Practical applications of CSP

During the second day, the conference was mainly concentrated on the role of CSP in practical solutions. Peter Welch presented an application that used the language Occam-π to deal with mobile processes activities relative to the location and monitoring of the presence of local agents in wireless environments. This language combines process and channel mobility (of the π-calculus) with the discipline and safety of Occam, including the semantics of CSP. The application is based on the idea that it is likely and, even necessary, to make the concurrent processes easier and more natural than they used to be, simulating the organization of complex systems in nature.

Jeff Magee presented an overview of five years use of a tool-supported approach to the design of concurrent Java programs, using a modelling notation based on CSP. He concluded that a clear CSP-based model with tool support can be attractive both for students and practitioners.

The second section of the final day was dedicated to the links of CSP with various other theories. Carroll Morgan connected probabilistic action systems and probabilistic CSP. However, as he clearly pointed out, there are still a lot of open questions that await an answer in this area, such as the role and the definition of compositionality. Mike Reed in his talk "Order, topology, and recursion induction in CSP" developed a general theory of recursion induction based on the Scott topology of the maximal elements in a domain, obtaining the solution of some open questions about the topology of the set of maximal elements in a domain.

Jan Peleska from the University of Bremen and Verified Systems International GmbH presented some “real world” projects, such as the fault-tolerant computers operating in the International Space Station, hardware in-the-loop tests for the novel Airbus A380 aircraft, and the test for the conformance to the European Train Control System. He demonstrated the benefits that all the projects gained through using formal techniques, such as language design or semantics and tool support. He emphasized the importance of hybrid control systems and of executable formal specifications, arguing that in order for formal methods to be applicable it is necessary to create new specification formalisms, capable of distinguishing between implementation and other properties of CSP.

One of the areas in which formal methods have been usefully employed is information security. Two papers addressing this area were to be presented, one by Peter Ryan and the other by Sadie Creese. Peter Ryan was unable to attend CSP 25, for personal reasons, but his paper did appear in the proceedings. The paper concentrated on modelling non-interference, in order to define a clear concept for absence of information. It gave an overview of the applications of process algebra to these problems, arguing that the absence of information can be characterized in terms of process equivalence, an important but subtle concept. Creese presented different services offered by QinetiQ, specialising in military and security problems, in the development of high integrity systems.

The last part of the conference was dedicated to program checking techniques based on formal methods. Program checking seems to be one of the applied research areas in which formal methods, particularly CSP, have been most successful. Michael Goldsmith introduced the FDR refinement-checking tool, a commercial product of Formal Systems Europe, available free of charge for academic purposes. The checking technique relies upon the similarities between operational and denotational semantics for CSP. However using standard operational semantics calculations, it is inevitable for bottlenecks in the tool’s performance to occur, so he compiled an optimised form of the inference system that helps guarantee efficient performance, especially if code is reused for different purposes. The last talk was more theoretical; Ranko Lazic proved the decidability and undecidability of different programs that are data-independent with respect to the number of arrays and the type of variables that are stored in them.

Conclusion

Overall, the conference was extremely dynamic and lively; participants were involved in exchanges of ideas, questions and discussions on a variety of subjects related to CSP in a wide context. During breaks there were PhD poster presentations, testifying that the field is still active, promising and fertile. The road of formal methods has not always been direct and smooth, but it is very likely that in 25 years time the community will meet again to envisage future and past achievements both in the scientific community and in the wider industrial arena.

Tony Hoare (front row) and a number of his "followers" at the CSP 25 conference. 2nd row: Jonathan Bowen, Ali Abdallah, Wayne Luk, Mark Josephs, Bill Roscoe, Stephen Brookes, Cliff Jones

Acknowledgements

Teresa Numerico is currently a Visiting Research Fellow at London South Bank University, funded by the Leverhulme Trust.

References

1. Hoare, C.A.R. "Communicating Sequential Processes", *Communications of the ACM*, 21 (8):666–677, (1978).
2. Milner, R. *A Calculus for Communicating Systems*, Springer-Verlag, Lecture Notes in Computer Science, volume 92 (1980).
3. Brookes, S.D., Hoare, C.A.R. and Roscoe, A.W. "A theory of Communicating Sequential Processes", *Journal of the ACM*, 31(3):560–599 (1984).
4. Hoare, C.A.R. *Communicating Sequential Processes*, Prentice Hall International Series in Computer Science (1985).

Other Groups of interest to FACS Members

Z User Group
http://www.zuser.org

London Mathematical Society
http://www.lms.ac.uk

BCS
Advanced Programming Specialist Group
http://www.bcs.org.uk/siggroup/advprog

Formal Methods Europe
http://www.fmeurope.org

SCSC
Safety Critical Systems Club
http://www.safety-club.org.uk

Perfect Developer: What it is and what it does

David Crocker & Judith Carlton

Perfect Developer (also known as the *Escher Tool*) is a formal tool aimed at software development but with applications in the formal specification of other sorts of system. It is built around a notation for expressing state-based specifications and optionally refining them to a form resembling a program in an imperative programming language. In this sense, it is rather like the B method, or the combination of VDM-SL and VDM-IL. However, being a relatively recent entry to the field, it is designed around two technologies that matured long after VDM and B were designed:

- Object-oriented (O-O) and component-based design
- Automated reasoning

We based *Perfect Developer* around object-oriented design because that is the dominant paradigm used in industry today, but we recognise that not all problems benefit from an O-O approach. Furthermore, some features of O-O design are not yet accepted as safe by the developers of safety-critical systems. So while *Perfect Developer* does require use of two of the foundations of O-O design (abstraction and encapsulation), use of polymorphism and dynamic binding is discretionary and the use of objects obeying reference semantics is discouraged.

Thus, *Perfect Developer* is based on the paradigm of classes that encapsulate data and methods that operate on that data, in the same way that B is based on the paradigm of abstract machines.

Another advantage of using the O-O paradigm is that *Perfect Developer* can import UML models to generate skeleton specifications, on which detailed semantics can be hung. It can also generate ready-to-compile code in C++ or Java, which can be interfaced to graphical user interfaces or to other components written in those languages.

In general, just the process of writing a formal specification is likely to improve the quality of a program written from it. Clearly, actually proving that the specification is consistent, and that it's correctly implemented, is of greater value. From the commercial point of view, though, producing proofs by hand is far too time-consuming. Even if an interactive theorem prover assists the user in constructing the proofs, the process is nowhere near fast enough for widespread commercial use.

The second major technology underlying *Perfect Developer* is automated reasoning, and this helps solve the difficulties with commercial productivity.

Automated reasoning technology has advanced in leaps and bounds during the last decade – particularly in the field of first-order theorem proving. We therefore designed the notation of *Perfect Developer* to give rise to verification conditions (or *proof obligations*) that are overwhelmingly first-order. Then we built a theorem prover, optimising it to handle real-life verification conditions, rather than the abstract mathematical theorems for which academic provers are designed. The result is that the tool is able to discharge more than 95% of valid verification conditions without user intervention in typical commercial applications – one real-life system has recently reached 99.89%.

We tried to make the notation of *Perfect Developer* easy for software developers to learn, including those unused to formal methods or mathematical notation. Users who are already familiar with VDM or B should find it even easier. Those who are more accustomed to Z need to get used to separating pre- and post-conditions. Further information about *Perfect Developer* can be found in [1], [2], [3] and [4].

An example: Specifying and refining a queue

Listing 1 shows a small example in which a bounded queue is specified as a *Perfect* class called *Queue* **of** *X*.

The generic parameter *X* represents the type of element that will be stored. The **abstract** section of the class declares the abstract model of the data held by the queue, which in this case comprises a sequence of elements *b* and a fixed bound *maxlen*. The number of elements in *b* at any time cannot exceed *maxlen,* and we declare this property as an **invariant** of the class. The unary # operator, applied to a sequence, yields its length.

The interface section contains the declarations of operations available to users of a *Queue*. In this example, we declare query functions *empty* and *full*, together with operations *add* and *remove.* We also declare a constructor **build** for creating an empty queue.

The symbol ^= used in the function declarations means “is defined as”. The keyword **pre** introduces a precondition, while **post** declares a schema postcondition.

In *Perfect*, a postcondition either implicitly or explicitly includes a frame, thereby defining not only how the final values of changed variables relate to the initial conditions, but also requiring that other variables remain unchanged. For example, the assignment-like postcondition *b! = b.append(x)* is actually short for **change** *b* **satisfy** *b’ = b.append(x)*. This states that the only variable affected is *b,* and that its final value *b’* must be equal to *b.append(x)*. The

Listing 1: Specification of a bounded queue

```
final class Queue of X ^=
abstract
  var b: seq of X,       // the queue data
      maxLen: nat > 0;   // maximum items in the queue

  invariant #b <= maxLen;

interface
  function empty: bool    // test if the queue is empty
    ^= #b = 0;

  schema !add(x: X)       // add an element to the end of the queue
    pre ~full
    post b!= b.append(x);

  function full: bool      // test if the queue is full
    ^= #b = maxLen;

  schema !remove(x!: out X)  // remove the head element
    pre ~empty
    post x! = b.head,
         b! = b.tail;

  build{!maxLen: nat, dummy: X}  // build an empty queue
    pre maxLen ~= 0
    post b! = seq of X{};

  ghost operator =(arg);  // we do not evaluate equality at run-
time

  // Verify that after adding an element, a queue is not empty
  property (x: X)
    pre ~full
    assert ~(self after it!add(x)).empty;

  // Verify that if we add an element to an empty queue,
  // the next element we remove will be the one we added
  ghost schema !addToEmptyThenRemove(e: X, r!: out X)
    pre empty
    post !add(e) then !remove(r!)
    assert r' = e;

end;
```

append function is a predefined method of class **seq of** *X*. It yields a new sequence, comprising the original with the parameter appended.

In order to improve confidence in the specification, we can also declare behavioural properties that we expect to hold. In this example, we have declared some expected behaviour by declaring a **property** and a **ghost schema**. The property declaration asserts that immediately after calling the *add* method of a queue, the empty function should return *false*. The ghost schema describes the scenario of adding an element to an empty queue and

Listing 2: Implementation of the queue using a ring buffer

```
final class Queue of X ^=
abstract
  var b: seq of X,       // the queue data
      maxLen: nat > 0;   // maximum items in the queue

  invariant #b <= maxLen;

internal
  var ring: seq of X,    // implement internally as a ring buffer
      hd, tl: nat;       // indices of the first and last elements

  invariant #ring = maxLen + 1,
        hd < #ring,
        tl < #ring;

  function b ^=          // retrieve function for variable 'b'
    (  [tl >= hd]:  ring.take(tl).drop(hd),
       [tl < hd]:   ring.drop(hd) ++ ring.take(tl)
    );
interface
  function empty: bool  // test if the queue is empty
    ^= #b = 0
    via
      value hd = tl
    end;

  schema !add(x: X)      // add an element to the end of the queue
    pre ~full
    post b! = b.append(x)
    via
      ring[tl]! = x,  tl! = (tl + 1)%(#ring)
    end;

  function full: bool   // test if the queue is full
    ^= #b = maxLen
    via
      value (tl + 1)%(#ring) = hd
    end;

  schema !remove(x!: out X)  // remove the head element
    pre ~empty
    post x! = b.head, b! = b.tail
    via
      x! = ring[hd],  hd! = (hd + 1)%(#ring)
    end;

  build{!maxLen: nat, dummy: X}  // build an empty queue
    pre maxLen ~= 0
    post b! = seq of X{}
    via
      ring! = seq of X{dummy}.rep(maxLen + 1),
      hd! = 0, tl! = 0
    end;

  // Include property and ghost schemas here as before…

end;
```

then removing an element, and asserts that the element removed should be equal to the element added.

When asked to verify this specification, *Perfect Developer* generates and proves 16 verification conditions, assuring us that the specification is well-formed and consistent and that it exhibits the expected behaviour.

Although the specification in Listing 1 can be used to generate code directly, in practice it is more efficient to implement a bounded queue using a ring buffer. Listing 2 shows the same specification with refinement from the abstract model to an array *ring* together with head and tail indices *hd* and *tl*. The data refinement is declared in the *internal* section, together with the invariants that the sequence *ring* has fixed length and the two index variables are in range. By redeclaring the original abstract sequence *b* as a retrieve function, we indicate that it is not a stored variable in the implementation and we describe the value of *b* that is represented by any combination of values of *ring*, *hd* and *tl* that satisfy the invariant. In defining the retrieve function, we use a conditional expression, which has the form *([guard1]: expression1, [guard2]: expression2)* and has the meaning "if *guard1* then *expression1* else if *guard2* then *expression2*". The member function *take(n)* of class **seq of** *X* returns the first *n* elements of the sequence, while *drop(n)* returns all but the first *n* elements. The operator ++ applied to sequences denotes concatenation.

Alongside this data refinement, the specifications of the public operations are refined to implementations in the **via**…**end** blocks. The implementations declare how the corresponding specifications should be implemented as operations on the ring buffer and associated head and tail variables.

The refinement of the specification to a ring buffer implementation causes *Perfect Developer* to generate and prove an additional 34 verification conditions. Taken together, these show that the implementation is well-formed and that it faithfully implements the original specification.

Verifying security properties of the Mondex abstract world

At the recent Refinement Workshop (see report on page 23), proof of the Z specification of the Mondex electronic purse [5] was discussed. As an exercise, a reformulation of the top level of this specification provided by Jim Woodcock was translated into *Perfect* and proved automatically.

Listing 3 shows a revised version of this translation, in which we have tried to mirror the Z original more closely.

Listing 3: Abstract specification of Mondex electronic purse

```
// Declare a type for identifying purses
class NAME ^= tag;    // this creates a new abstract type called NAME

// Class to represent a Mondex purse
class AbPurse ^=
abstract
  var balance, lost: nat;
interface
  function balance, lost;    // this makes 'balance' and 'lost'
                             // readable from outside the class

  // Schema to represent an amount being lost from the purse
  schema !lose(amt: nat)
    pre amt <= balance
    post balance!- amt, lost!+ amt;

  // Schema to represent an amount being removed from the balance
  schema !remove(amt: nat)
    pre amt <= balance
    post balance!- amt;

  // Schema to represent an amount being added to the balance
  schema !add(amt: nat)
    post balance!+ amt;

  // Constructor
  build{}
    post balance! = 0, lost! = 0;
end;

// Class to represent details of a proposed transfer between purses
class TransferDetails ^=
abstract
  var frm, to: NAME,         // the 'from' and 'to' purses
      val: nat;              // the amount of the transfer
interface
  function frm, to, val;

  // Constructor
  build{!frm, !to: NAME, !val: nat};
end;

// Class to represent the abstract Mondex world
class AbWorld ^=
abstract
  var AbAuthPurse: map of (NAME -> AbPurse); // the authorised purses

  // Get the total balance of all authorised purses
  function totalAbBalance: int
    ^= + over (for x::AbAuthPurse.ranb yield x.balance);

  // Get the total lost from all authorised purses
  function totalAbLost: int
    ^= + over (for x::AbAuthPurse.ranb yield x.lost);

  // Determine whether a purse name is authentic
  function Authentic(id: NAME): bool
    ^= id in AbAuthPurse;
```

Listing 3 continued

```
  // Determine whether the 'from' purse in a proposed transfer
  // has sufficient funds
  function SufficientFundsProperty(details: TransferDetails): bool
    pre Authentic(details.frm)
        ^= details.val <= AbAuthPurse[details.frm].balance;

  // Schema to represent a transfer attempt that is ignored
  schema !AbIgnore(details: TransferDetails)
    post pass;

  // Schema to represent a transfer attempt that results
  // in the amount being lost
  schema !AbTransferLost(details: TransferDetails)
    pre Authentic(details.frm),
      Authentic(details.to),
      details.frm ~= details.to,
      SufficientFundsProperty(details)
    post change AbAuthPurse
      satisfy AbAuthPurse' =
        AbAuthPurse.replace(details.frm -> AbAuthPurse[details.frm]
                                           after
it!lose(details.val));

  // Schema to represent a successful transfer
  schema !AbTransferOkay(details: TransferDetails)
    pre Authentic(details.frm),
      Authentic(details.to),
      details.frm ~= details.to,
      SufficientFundsProperty(details)
    post change AbAuthPurse
      satisfy AbAuthPurse' =
        AbAuthPurse.replace(details.frm -> AbAuthPurse[details.frm]
                                           after
it!remove(details.val))
                   .replace(details.to -> AbAuthPurse[details.to]
                                           after it!add(details.val));
interface

  // Schema to represent a transfer attempt that may be successful,
  // ignored, or result in the amount concerned being lost
  opaque schema !AbTransfer(details: TransferDetails)
    post
     ( opaque  // use nondeterministic guarded choice here to mimic
               // the Z schema disjunction operator
       [  Authentic(details.frm)
        & Authentic(details.to)
        & details.frm ~= details.to
        & SufficientFundsProperty(details)
       ]:
         !AbTransferLost(details),
       [  Authentic(details.frm)
        & Authentic(details.to)
        & details.frm ~= details.to
        & SufficientFundsProperty(details)
       ]:
         !AbTransferOkay(details),
       [true]:
         !AbIgnore(details)
     )
```

Listing 3 continued

```
// declare security properties
    assert        self'.totalAbBalance        <=        totalAbBalance,
                  // no value created (Z: NoValueCreation)
            self'.totalAbBalance + self'.totalAbLost
                = totalAbBalance + totalAbLost;
                  // all value accounted (Z: AllValueAccounted)

  // Constructor
  build{!AbAuthPurse: map of (NAME -> AbPurse)};
end;
```

We declare classes to represent the contents of a purse, the details of a transfer, and the abstract world itself. As in the Z version, the collection of authorized purses is represented as a mapping from the names of purses to their contents. There are three possible outcomes of attempting a transfer between purses: the transfer may succeed, or be ignored, or the amount may be lost. We have declared separate schemas *AbTransferOkay*, *AbIgnore* and *AbTransferLost* in the abstract world to represent each of these.

A transfer attempt is represented by schema *AbTransfer* and its outcome is a nondeterministic choice between the other three schemas. The Z specification uses the schema disjunction operator to express this choice. Since in *Perfect* it is necessary to respect the schema preconditions, we use a conditional postcondition to select which of the three schemas may be invoked. The fact that *AbTransfer* is intentionally nondeterministic is flagged by declaring it **opaque.** We again use the keyword **opaque** within the conditional postcondition, to indicate nondeterministic choice between those schemas whose guards are true, rather than deterministically choosing the first one whose guard is satisfied. The two security properties are expressed as post-assertions attached to schema *AbTransfer*.

Of the 30 verification conditions generated and proved by *Perfect Developer* for this example, two represent the security properties; the remainder are precondition checks and domain checks.

Obtaining Perfect Developer

Perfect Developer is free to evaluate, including use in small-scale student projects. Over 20 universities are using the tool in one way or another and six have purchased licences for classroom teaching or for research. For more information, please email info@eschertech.com.

References

1. *Perfect Developer Language Reference Manual*. Available at http://www.eschertech.com/product_documentation/Language%20Reference/language_reference.pdf or online in HTML format via the *Support* section of www.eschertech.com.
2. *Proof Obligations Generated by Perfect Developer*. Available at http://www.eschertech.com/product_documentation/ProofObligations.pdf.
3. Crocker, D. Safe Object-Oriented Software: "The Verified Design-by-Contract Paradigm". *Proceedings of the Twelfth Safety-Critical Systems Symposium* (ed. F. Redmill & T. Anderson), pages 19–41, Springer-Verlag, London, 2004.
4. Crocker, D. and Carlton, J. "A High Productivity Tool for Formally Verified Software Development". To be published in the *International Journal on Software Tools for Technology Transfer* (Special Section on FME2003).
5. Stepney, S., Cooper, D. and Woodcock, J. "An Electronic Purse". Technical Monograph PRG-126, Oxford University Computing Laboratory (July 2000).

Call For Papers

Photo by Graeme Peacock, used with permission

www.csr.ncl.ac.uk/fm05

Important Dates

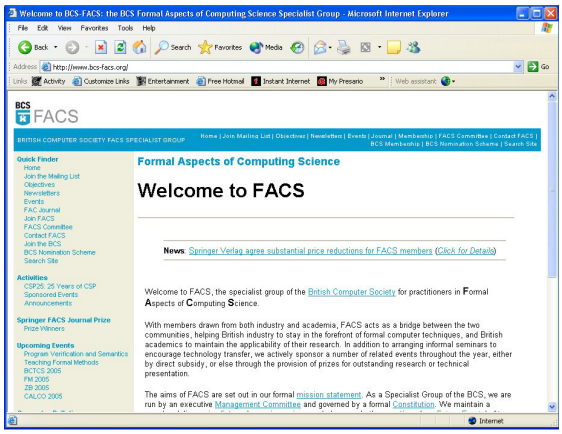
24 January 2005	Paper submission deadline
07 March 2005	Workshop & Tutorial proposals
09 April 2005	Decisions on papers
02 May 2005	Final versions of papers due
09 May 2005	Tools Exhibition & Demonstration proposals

News from FACS

FACS Website gets Honourable Mention

Mike Stannett, FACS Webmaster

BCS-FACS has received an Honourable Mention in the annual competition run by the BCS for the best Specialist Group web site. The commendation rewards the hard work put into the site (accessible via http://www.bcs-facs.org and http://www.bcs-facs.org.uk) by the Publications team (Mike Stannett, Paul Boca and Jonathan Bowen) over the last year, who have overseen the introduction of a number of key innovations. For the first time, members can pay their subscriptions using our online PayPal account, while at the same time inspecting the latest items in our regularly updated News Ticker, or browsing the site with our dedicated Google-powered search engine. The whole site has been redesigned to meet the BCS' high standards, with the BCS corporate layout implemented through a careful application of Cascading Style Sheets (CSS). Every single page on the site has been individually validated against the World Wide Web Consortium's official specification of HTML.



Nonetheless, we see this only as a first step in our long-term program of improvements. We are already constructing a members-only section of the site, and will be adding a number of interactive pages to help users get the most from the site. Conference and seminar organizers will be able to add details of their events automatically, and members will be able to have details of relevant events, news and services emailed to them directly. Eventually it will be possible to submit articles and news items for *FACS FACTS* online, and we are hoping to accept membership renewals online in 2005 too. Here's to the continuing year-on-year improvement of all things FACS!

Paid-up BCS-FACS members receive a generous **25%** discount on books published by Springer and **20%** discount on the Requirements Engineering Journal. If you would like to take advantage of these discounts, please contact Springer directly on:

journalslondon@springer-sbm.com

FACS Committee Meeting, London, 8 July 2004

Paul Boca, FACS Newsletter Editor & Membership Secretary

The FACS committee had an informal breakfast meeting (at 8am!) on 8 July 2004 at the Union Jack Club, London. Most of the committee members were attending CSP 25, so it was an ideal opportunity to catch up on progress since the AGM (see Issue 2004-2 for a report on the AGM). The FACS committee invited Christiane (Chris) Notarmarco from Springer (pictured above, left with Margaret West and John Cooke) to attend the meeting. This was a good opportunity to discuss ways in which Springer and FACS could work together, and to keep Springer informed about FACS activities.

A summary of points discussed is given below:

- FACS Committee: Rick Thomas was co-opted on to the FACS Committee. Rick will work with Mike Stannett on building stronger links with the London Mathematical Society (LMS). Rick is Chair of the LMS Computer Science Committee.
- Events: Ali Abdallah mentioned that there will be a workshop on Formal Aspects of Security and Trust (FAST) at FM 2005 in July 2005. This workshop, which is being organized by Peter Ryan (University of Newcastle), may have some FACS involvement.
- Publicity: Paul Boca reported back on the success of the best-paper prizes and listed the conferences where prizes would be awarded. Details of the prize-winners can be found on the FACS website at http://www.bcs-facs.org/winners.html. Springer kindly offered to sponsor the journal subscription part of 4 prizes each year. To reflect this, the prize will now be known as the *Springer FACS Journal Prize*. If you are organizing a conference and would like to offer a best-paper prize, please contact Prof. Jonathan Bowen, the FACS Chair, on Jonathan.Bowen@lsbu.ac.uk.
- Member Discounts: some ideas for further discounts for FACS members were suggested by Chris from Springer. These are still under investigation. When we have some definite news, we will let you know.
- *Online First* and *Open Choice*: Chris explained that *Online First*, the system where papers accepted in the FAC journal are available online within two weeks for download, has been a success. Further details at: http://www2.springeronline.com/sgw/cda/frontpage/0,,5-113-2-99044-0,00.html.
  A new initiative called *Open Choice* has recently started. Authors can pay a fee to allow their journal article to be made available to the public. For further details, please visit: http://www2.springeronline.com/sgw/cda/frontpage/0,,1-40359-0-0-0,00.html. ■

RODIN – Rigorous Open Development Environment for Complex Systems

Alexander Romanovsky

Project

RODIN – Rigorous Open Development Environment for Complex Systems (Project Number: IST 2004-511599)

Partners

- University of Newcastle upon Tyne, UK (Coordinator);
- Åbo Akademi, Turku, Finland;
- ClearSy System Engineering, France;
- NOKIA Corporation, Finland;
- Praxis Critical Systems Ltd, UK;
- VT Engine Controls Ltd, UK;
- Federal Institute of Technology, Zürich, Switzerland;
- University of Southampton, UK.

Starting date

September 2004. Duration: 36 months

Project Coordinator

Alexander Romanovsky
School of Computing Science,
University of Newcastle upon Tyne,
Newcastle upon Tyne,
NE1 7RU, UK

The overall objective of RODIN is the creation of a methodology and supporting open tool platform for the cost effective rigorous development of dependable complex software systems and services.

The project focuses on tackling complexity

- caused by the environment in which the software is to operate;
- which comes from poorly conceived architectural structure.

Mastering complexity requires design techniques that support clear thinking and rigorous validation and verification. Formal design methods do so. Coping with complexity also requires architectures that are tolerant of faults and unpredictable changes in environment. This is addressed by fault tolerance design techniques dealing with faults in the system environment, faults of the individual component, component mismatches, as well as errors affecting interacting components.

The project will develop a unified methodology combining formal methods with fault tolerance design principles by using a systems approach, where both software and environment are modelled together. We will tackle complex architectures: the systems approach will support the construction of appropriate abstractions and provide techniques for their structured refinement and decomposition.

To maximise cost effectiveness, the methods and platform will support reuse of existing software. We will thus extend existing formal methods with generic mechanisms to support component reuse and composition.

Tool support for construction, manipulation and analysis of models is crucial and we will concentrate on a comprehensive tool platform which is openly available and openly extendable. The methods and platform will be validated and assessed through industrial case studies.

The novel aspects of this proposal are the pursuit of a systems approach, the combination of formal methods with fault tolerance techniques, the development of formal methods support for component reuse and composition and the provision of an open and extensible tools platform for formal development. In particular, we believe that the open tools platform will have a significant impact on future research in formal method tools and will encourage greater industrial uptake; it has the potential to set a European standard for industrial formal method tools.

RODIN results will be:

- A collection of reusable development templates (models, architectures, proofs, components, etc.) produced by the case studies.
- A set of guidelines on a systems approach to the rigorous development of complex systems, including design abstractions for fault tolerance and guidelines on model mapping, architectural design and model decomposition.
- An open tool kernel supporting extensibility of the underlying formalism and integration of tool plug-ins.
- A collection of plug-in tools for model construction, model simulation, model checking, verification, testing and code generation.

Extensive preparatory work preceded the beginning of the project. Funded RODIN work started on 1 September 2004. The kick-off meeting of all the project participants was held in Newcastle on 4–6 October.

More information can be found at: http://rodin.cs.ncl.ac.uk

4th International Conference of B and Z Users

13-15 April 2005, University of Surrey, Guildford, UK

Integration of model-based specification methods
Requirements engineering using formal methods
Industrial applications using Z and B
Theoretical issues in formal development
Z and B extensions and tools

Invited Speakers

Carroll Morgan, UNSW, Australia
Cliff Jones, Newcastle, UK
Frédéric Badeau, ClearSy, France

Supported by the APCB and the Z User Group

LFSL ’04 Summer School – a student's perspective

Pat Browne (Edited and extended by Martin Henson)

The PhD Summer School "Logics of Formal Software Specification Languages", LFSL ’04, was held between June 6 and 19, 2004 at the beautiful Congress Centre “Academia”, in Stara Lesna, at the foot of the wonderful High Tatras mountains in Slovakia. This event was the inspired idea of Prof. Dines Bjørner – in part celebrating the membership of the accession countries into an expanded EU. In fact, among the ten presenters and forty-six students, there were an astonishing twenty-six countries represented – we celebrated the birthday of one student by singing in over twenty languages!

Two very early morning starts bracketed a challenging and rewarding fortnight of serious academic work, balanced with a lighter social side. These two aspects were not always separate and distinct: I recall a very pleasant alfresco lunch in Bratislava with Prof. Martin Henson from the UK and his student Besnik Kajtazi from Kosovo. During the meal Martin explained the details of comma categories with occasional asides on the quality of the food and wine, or was it the other way around?

The right school for me!

I am a second year part-time student on a PhD programme run by the School of Computing at the Dublin Institute of Technology (DIT), Ireland. My supervisor is Prof. Mike Jackson from the University of Central England in Birmingham. This was my very first summer school and I was a novice in the details of formal specifications. After some discussion with my supervisor we selected LFSL ‘04. The two things that attracted us were that the specific topics important to my own studies and the fact that the presentations were specifically aimed at PhD students. In hindsight our decision was correct, and LFSL ‘04 provided me with a much-needed exposure to formal specifications.

The main theme of the conference addressed my research area very well. I am looking for ways to compose (or join) various specifications, some of which are based on different logics. My research aim is to construct a unified spatial-temporal model for Geographic Information Systems (GIS). The wide spectrum languages CafeOBJ and CASL were of most relevance to my PhD studies. Several approaches to composition were presented, including the structuring specification mechanism in CASL and presentation push-outs in CafeOBJ.

The presentations

The summer school focused on formal methods of system development using a range of methods and tools. The model-based approaches were Z and B, while the algebraic approaches included CASL, CafeOBJ and RAISE. Reactive, distributed, and real-time systems were covered by TLA+, ASM, and Duration Calculus. Beautiful and comprehensive tutorials were presented for each of these approaches.

The wide spectrum languages CafeOBJ and CASL were of most relevance to my own studies and it was a delight to hear people of the calibre of Razvan Diaconescu and Till Mossakowski describe the theoretical and practical aspects of there respective approaches.

Obviously all the other presentations and presenters were also of the same high standard, however these two hit the nail on the head with respect to my own research. Another highlight for me was Prof. Dines Bjørner's presentation on the ontology of rail systems. This I found very similar in spirit to my own research. Essentially we are both taking complex real world domains and trying to formalize them with a view to automating many of their processes.

The organization was excellent. Prof. Dines Bjørner, Prof. Martin Henson, Prof. Branislav Rovan, Dr. Dusan Guller and Mr. Martin Penicka provided the appropriate mix of topics. Prof. Henson kept the show moving ensuring the lectures ran as timetabled and provided ample time for questions. He made it all look so easy, but I am sure that there was a lot of work required to get things to run so smoothly.

Social activities

Several trips were organized that allowed us to enjoy the beautiful mountains, castles and architecture of Slovakia. There was a tremendous sense of friendship generated over the two weeks. Meeting people from so many countries with such diverse backgrounds was a rich and rewarding experience.

Student Profile

The student profile varied. Some of us (myself included) were novices. Others were well into their studies, seeking detailed knowledge of a specific formal approach. Several attendees had completed their PhDs and were

interested in topics for further research or in sharing knowledge with fellow researchers. Some were just toying with the idea of studying for a PhD and were looking for an appropriate topic. I felt that LFSL '04 addressed the diverse needs of all groups of students.

All in all, this was a very enjoyable and educational experience. When and where is the next one?!

A summary of the summer school, and several photographs, can be found at: http://cswww.essex.ac.uk/staff/henson/sssl/

The School was generously sponsored and supported by CoLogNET; Microsoft Research; the United Nations University; the Slovak Society for Computer Science; the Technical University of Denmark; the University of Essex UK and Comenius University, Slovakia.

BCS FACS

Program Verification and Semantics: Further Work

Science Museum
London

2 December 2004
2pm start

An event covering early work in program verification and semantics was held in 2001. Professor Jonathan Bowen, Professor Cliff Jones and Dr Teresa Numerico are organizing a follow up event at the Science Museum on 2 December 2004 starting at 2pm. The following pioneers of the field will be speaking:

- **Prof. John C. Reynolds** (Carnegie Mellon University, USA)
- **Prof. Gordon Plotkin** (University of Edinburgh, UK)
- **Prof. Cliff Jones** (University of Newcastle upon Tyne, UK)

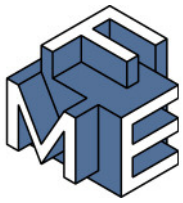
The contributions will range from formal presentations to personal reminiscences.

The Computer Conservation Society (CCS) and Formal Aspects of Computing Science (FACS) Specialist Groups are both supporting the event. For further details, please visit

http://vmoc.museophile.org/pvs04/

News from FME

John Fitzgerald, FME Chair



Two forthcoming symposia were very much on the minds of FME members at the recent Newcastle meeting. Our First International Symposium on the teaching of formal methods is being held at Ghent in November. I am using it as an opportunity to give my students a lecture or two off and counting it as professional development. See the FME website www.fmeurope.org for further details.

The other big event is FM'05, to be held in Newcastle on 18 – 22 July 2005. The core is the symposium, held on 20 – 22 July. The submission deadline for papers is **24 January**: details via www.csr.ncl.ac.uk/fm05. Remember that papers on practice are welcomed on par with those on theory.

Workshops and co-located events include Formal Aspects of Security and Trust, and Calculemus 2005. Worth a special mention is the RODIN Workshop (see report on RODIN on page 43 of this Issue of *FACS FACTS*) on the development of a platform for the integration of formal methods tools. Tutorials planned so far include tools (Spark, ClawZ and Perfect Developer), and wider ranging topics including formal aspects of software architecture, categories in software engineering and asynchronous communications mechanisms. For both workshops and tutorials, new proposals are more than welcome: see www.csr.ncl.ac.uk/fm05 for details. ■

Call for Discipline Hopping Awards at the Mathematical Sciences and Computer Science Interface

EPSRC

Discipline Hopping Awards provide short-term support to pump prime new collaborations between Mathematical Scientists and Computer Scientists, with the aim of fostering long-term interaction. This scheme allows researchers who have a track record in their own field in the mathematical sciences to apply for funding to investigate and develop ideas, skills and collaborations in the areas of computer science. Alternatively, Computer Science researchers can apply for funding to develop ideas, skills and collaborations in the area of mathematical science.

The deadline for this year's applications is 14 December 2004

For more details, please visit the EPSRC website

Adding Informal Analysis Ingredients to the Development Recipe

Islam A M El-Maddah

Formal methods, including B, VDM and Z, have gained a considerable level of trust among reactive systems' developers. Although they provide a systematic way of constructing software applications, these formal methods lack a common language that can be easily spoken and understood by both the client and service provider. Moreover, the cost and effort required to formally develop and test entire applications is too expensive for the average client. This suggests some informal analysis ingredients need to be added to the development recipe to complement formal methods. Thus, there will be a better understanding and logical presence of the client to ensure his/her agreement in the early development stages. This will be achieved as well, without disturbing the formal treatment supported by the software specification engineer. The informal analysis may not stop at the requirements analysis stages, but survive further than these stages to the specification checks and tests.

In order to deliver a successful piece of software, the software application should undergo a number of stages, termed development lifecycle steps. The main concept beyond splitting the development into a number of ordered steps is to organize and distribute the human effort. The effort will be split into small but manageable activities, each of which, in turn, will be applied to the software application to translate the user needs from one form into another. The overall lifecycle will be completed when the client's initial needs are mapped to a correct and acceptable executable implementation. A development team of people, with different specialities, should share the responsibility of these development activities. This team usually includes the client, stakeholders, system analysts, designers, implementers and testers.

A quick look at the targeted product (a software program) reveals that it consists of a sequence of deterministic formal steps performing some specific task under specific conditions. This formal description of the final product

Call for Papers

CALCO 2005
1st Conference on Algebra and Coalgebra in Computer Science

3 – 6 September 2005

University of Wales Swansea, UK

Submission deadline: 21/31 Jan 2005

http://www.cs.swan.ac.uk/calco/

prepares us to vote for formal methods to play the main role on the development stage. They should be employed to specify the software application at the early stages. For that reason, it is now the responsibility of the specification team to formalise the client's needs and hence decide on what should be refined into a full implementation.

Formal methods have achieved a witnessed success, especially in some applications that cannot tolerate errors, such as safety-critical systems [1]. This may be due to the well-defined path they represent. Formal methods simply construct a well-defined road that can be followed to obtain an implementation that conforms to the initial specification set by the developer (usually a software engineer). This road leads directly from the formal specifications to an implementation through a number of refinement, verification, validation, and possibly correction stages. This may explain the success VDM, B, and Z have achieved in industry. Actually, software engineers and scientists trust formal methods because they are a hybrid of mathematics and logic. Thus, it is ensured that their initial specifications will be systemically refined, proved and deduced.

But wait a minute, who said that the specification team knows what to develop? How to compromise between the safety, productivity and security aspects of the intended application? What decision to make when knowing that, for example, one of the client's initial needs cannot be realized?

No one is more aware of these issues than the client himself/herself (especially managers who know the relative importance of each aspect of the intended application). This makes it difficult to rely only on formal specifications because they are difficult for the average clients to understand and observe (tested, proved and changed into a client-readable form). This makes the cost of employing formal methods for developing the entire application from the beginning of the development unaffordable. Again, the problem is initially informal from the client's perspective; he/she needs some application that satisfies some requirements. Is it not irrational to start the development using formal specifications full of proofs and advanced theories about temporal logic? This makes the client's life hard, particularly when tracing his/her needs within the formal specifications or even agreeing with them.

Do we not agree that it is unacceptable to deliver an implementation written in a high-level programming language such as Java or C++ without providing

detailed documentation? Then, what about formal specifications? Will the client be satisfied at such an early development stage with, for example, a number of B machines, or Z schemas, even with their test results?

This leaves us with the informal analysis as an initial target at the preliminary development stages. Informal analysis should be employed in such situations to complement formal methods in order to achieve success in developing agreed and correct software applications. For example, writing down an assignment statement (abstract state change) in B without explanatory comments may result in a lack of traceability because it separates the specification from the client's original needs. On the other hand, having a tangled description without a corresponding formal construct may result in having an ambiguous requirement that has many possible interpretations, some of which will be rejected by the client later on when the requirements are eventually tested (if it is still possible to trace the implementation of these requirements back to the client's needs, that is).

Informal analysis also includes graphical modelling of user needs/ requirements or possibly the specification of the proposed solution. It is appropriate to communicate some clients' mental models with drawings, such as data-flow diagrams, state-transition diagrams and and-or trees. These drawings usually break down the logical mental barrier set by the difficult dry specification formulae. But mind the GRAPH – do not get lost there! Graphical representation may lead to ambiguities and misunderstanding, if the drawings do not have a standard unified semantics (agreed by the client and the service provider) or if they are not packaged with animation tools.

The informal treatment should be regarded as a bridge that conveys the client's needs to formal methods domains. Furthermore, this is a bridge with two-way traffic. In one direction, the client's needs are being refined and formalised into formal specifications. Whereas, the other direction conveys the outcomes of checking the formal specifications against their origins in the client's needs domain. In fact, this bridge should ensure the traceability between the requirements and the formal specifications. In other words, it acts as an instantaneous interpreter between two negotiators, who speak two different languages (the client speaking in his/her informal needs language and the software specification engineer speaking in a formal methods language).

Hence, one should target some semi-formal model that enables giving an informal and abstract start-off, to begin the development. Afterwards, the model should gradually move towards a more formal and refined specification for the intended application. Thus, at the end, one can arrive at a fully formal and refined specification that can be used by a developer to complete the rest of the development lifecycle; whereas, these specifications are understood and agreed by the client. An alternative term, which is usually used for such an approach that embeds formal analysis for parts of the application and covers it with an informal treatment (normally tailored differently according to the targeted domains), is lightweight formal methods [5]. This makes early stages of the development closer to the client's perspective.

For example, in GOPCSD (goal oriented process control systems development) [2], we have employed goal, agent, component, variable and goal-model concepts (KAOS [4], TROPOS [3]) to build the so-called bridge. The software application will be developed starting from the client's needs, as a very abstract main goal, that will be refined (using some different types of refinement patterns) into a number of sub-goals. Each sub-goal carries more formal description than its parent goal. The requirements model in GOPCSD ends with terminal goals, each of which is a conditional assignment with an informal description of the terminal goal. Each client's need resides at one of the higher-level goals and is refined into a number of sub-goals that will eventually have a formal description. Thus, each client's need is traced into a number of terminal goals, which will form the formal specification.

A point to keep in mind is that informal analysis should not be confused with an unchecked or imprecise treatment. Similarly, having formal specifications does not ensure that the developed application will not suffer from ambiguity or imprecision or inconsistency. Thus, informal analysis should be augmented with a number of validation and check procedures, which ensure the client fully understands the meaning of what is written down in the requirements model. For example, reasoning about the why and how of the goals can be employed to serve as an early validation of the goal-model informal semantics.

So, after passing the bridge from the informal side to the formal side, can we aim for any other benefit from the informal analyses? Actually, Yes! It may not be a very good idea to throw away our informal analysis at this point, or think of its function as just a preliminary stage to produce a formal specification corresponding to the client's refined needs. The informal and formal forms should be better understood as two sides of a single coin. Some of the tests that are expensive or difficult to be applied on one side may be easily applied on the other side. For example, exhaustive model checking or reachability analyses are usually performed on the formal specification of the application. These analyses may be performed earlier on in the informal part of the application, with variant tests and checks, saving the effort required for proof obligation generation, for example in the B-Toolkit. These early checks, which are brought forward towards the requirements analysis stage should not be considered as premature testing; instead they should be considered as an issue of separation of concerns between requirements and specification activities.

On the other hand, the outcomes of formal methods check and verification activities are still essential, especially for the parts of the application in which informal specifications cannot be easily checked. However, any feedback coming out from such checks, and concerning changes in the client's needs, should be better "in-formalised" and passed back to the client seeking some correction to be applied at the informal side.

In conclusion, the battle of software development should not be left for the formal methods to fight alone. The informal analysis should be there ready to give a hand when formal analysis needs support. Thus formal methods and the informal analysis should be integrated together in a fashion that increases

the separation of concerns among the development team and decreases the cost and effort required to develop client-agreed and understandable applications.

This integration may be as clear as a single shipped formal method tool in which a number of informal front-end sub-tools are available and possibly tailored for different domains. Alternatively, informal analysis specialists should be encouraged to generate outputs compatible with different formal methods and make them accessible for shipment with formal methods packages.

References

1. Bowen, J. P. (2000), "The Ethics of Safety Critical Systems", *Communications of the ACM*, 43(4):91–97, April 2000.
2. El-Maddah, I. A. M. and Maibaum, T. S. E. (2004), "The GOPCSD Tool: An Integrated Development Environment for Process Control Requirements and Design", *FASE04, ETAPS*, Spain, 2004.
3. Mylopoulos, J. and Castro, J. (2000), "Tropos: A Framework for Requirements-Driven Software Development" In S. Brinkkemper and A. Solvberg (eds.), *Information Systems Engineering: State of the Art and Research Themes*, Lecture Notes in Computer Science, Springer-Verlag, pp. 261–273, June 2000.
4. van Lamsweerde, A., Dardenne, A., Delcourt, B. and Dubisy, F. (1991), "The KAOS Project: Knowledge acquisition in automated specifications of software", *Proceeding AAAI Spring Symposium series*, Track: "Design of composite systems", Stanford University, pp. 59–62, March 1991.
5. George, V. and Vaughn, R. (2003) "Application of Lightweight Formal Methods in Requirement Engineering 1", *CrossTalk, The Journal of Defense Software Engineering*, July 2003. URL: http://www.stsc.hill.af.mil/crosstalk/2003/01/George.html

BCS Advanced Programming Specialist Group Talks

Concepts, Techniques, and Models of Computer Programming
Peter Van Roy and Seif Haridi
9 December 2004

Communicating Mobile Processes
Peter Welch
25 January 2005

Both talks will be held at the new BCS London offices, central London

Please visit http://www.bcs.org.uk/siggroup/advprog/ for further details

Workshop and Conference Announcements

The following is a selection of the large number of calls for papers and conference announcements that have been received. More listings can be found on FME's events page linked from http://www.fmeurope.org and the EATCS events website linked from https://www.eatcs.org/ .

November 2004

Symposium on Teaching Formal Methods
18–19 November 2004
University of Ghent
http://www.intec.rug.ac.be/groupsites/formal/Sympos2004/Sympos2004.htm

TFP 2004: Fifth Symposium on Trends in Functional Programming,
25–26 November 2004
Ludwig-Maximilians University, Munich, Germany
http://www.tcs.ifi.lmu.de/~hwloidl/TFP04

December 2004

Program Verification and Semantics: Further Work
2 December 2004
Science Museum, London
http://vmoc.museophile.org/pvs04/

BCS-FACS Christmas Meeting: The Verified Software Repository
21 December 2004
BCS Offices, London
http://www.bcs-facs.org/events/xmas2004.html

January 2005

POPL2005: 32nd Annual Symposium on Principles of Programming Languages
12–14 January 2005
California
http://www.cs.princeton.edu/~dpw/popl/05/

PADL 2005: 7th International Symposium on Practical Aspects of Declarative Languages (Co-located with POPL 2005),
10–11 January 2005
California
http://www.unm.edu/~herme/padl05/

1st International Workshop on Abstract Interpretation of Object-oriented Languages (AIOOL'05)
21 January 2005
Paris, France
http://www.stix.polytechnique.fr/~logozzo/WEB/Aiool.html

March 2005

21st British Colloquium for Theoretical Computer Science
22–24 March 2005
University Of Nottingham
Submission: **7 February 2005**
http://www.cs.nott.ac.uk/~gmh/bctcs05.html

April 2005

ETAPS 2005
2–10 April 2005
University of Edinburgh
http://www.etaps05.inf.ed.ac.uk/

4th International Conference of B and Z Users
13–15 April 2005
University of Surrey
http://www.zb2005.org/

July 2005

FM2005
18–22 July 2005
University of Newcastle Upon Tyne
Submission: **24 January 2005**
http://www.csr.ncl.ac.uk/fm05/

August 2005

MFCS 2005
29 August – 2 September 2005
30th International Symposium on Mathematical Foundations of Computer Science, Gdansk, Poland
Submission: **15 March 2005**
http://www.mfcs.univ.gda.pl/

September 2005

CALCO 2005: 1st Conference on Algebra and Coalgebra in Computer Science
3–6 September 2005
University of Wales
Submission: **21/31 January 2005**
http://www.cs.swan.ac.uk/calco/

October 2005

ICTAC05: International Colloquium on Theoretical Aspects of Computing
17–21 October 2005
Hanoi, Vietnam
Submission: **25 May 2005**
http://www.iist.unu.edu/ictac05/

Book Announcements

Categories for Software Engineering

J L Fiadeiro
Springer, 2004,
ISBN 3-540-20909-3

http://www.fiadeiro.org/jose/CATBook/

This book provides a gentle introduction to category theory oriented to software engineering. Assuming only a minimum of mathematical background, this book explores the use of categorical constructions from the point of view of the methods and techniques that have been proposed for the engineering of complex software systems: object-oriented development, software architectures, logical and algebraic specification techniques, models of concurrency, *inter alia*. After two parts in which basic and more advanced categorical concepts and techniques are introduced, the book illustrates their application to the semantics of CommUnity – a language for the architectural design of interactive systems.

Written for: Advanced students, professionals, lecturers

Keywords: Agent-Oriented Software Engineering, Categories, Category Theory, CommUnity, Complex Systems, Component-Based Systems, Coordination Languages, Formal Methods, Object-Oriented Software, Service-Oriented Software Development, Software Architecture, Software Engineering, Software Specification, Systems Design, Systems Modelling, Systems Theory.

Formal Engineering for Industrial Software Development

Using the SOFL Method

Shaoying Liu
Springer, 2004
ISBN: 3-540-20602-7

In any serious engineering discipline, it would be unthinkable to construct a large system without having a precise notion of what is to be built and without verifying how the system is expected to function. Software engineering is no different in this respect. Formal methods involve the use of mathematical notation and calculus in software development; such methods are difficult to apply to large-scale systems with practical constraints (e.g., limited developer skills, time and budget restrictions, changing requirements). Here Liu claims that formal engineering methods may bridge this gap. He advocates the incorporation of mathematical notation into the software engineering process, thus substantially improving the rigor, comprehensibility and effectiveness of the methods commonly used in industry. This book provides an introduction to the SOFL (Structured Object-Oriented Formal Language) method that was designed and industry-tested by the author. Written in a style suitable for lecture courses or for use by professionals, there are numerous exercises and a significant real-world case study, so the readers are provided with all the knowledge and examples needed to successfully apply the method in their own projects.

Written for:

Researchers, Lecturers, Graduate Students, Professionals

Keywords:

Formal Engineering Methods, Formal Methods, Object-Oriented Development, SOFL, Software Development Process, Software Specification, VDM, Z

PhD Theses in Formal Methods

Title Synthesis of Parallel Algorithms for Field Programmable Gate Arrays, with Applications from Cryptography

Author Issam Damaj

Institution London South Bank University, UK

Supervisors Dr. Ali Abdallah, Prof. Mark Josephs

Examiners Prof. Wayne Luk, Dr Sylvia Jennings

Awarded August 2004

Abstract

Mapping parallel versions of algorithms onto hardware could enormously improve computational efficiency. Recent advances in the area of reconfigurable computing came in the form of *FPGAs* and their high-level *HDLs* such as *Handel-C*. In this thesis, we build on these recent technological advances by presenting, demonstrating and examining a systematic approach of behavioural synthesis. This system creates a functional specification of an algorithm without defining parallelism. Correspondingly, an efficient parallel implementation is derived in the form of *CSP* network of processes. Accordingly, we create efficient parallel implementations in *Handel-C*. The presented work included theory and practices about the suggested methodology. The examination through different studies led to a tuned realisation and facilitation of the development method. Future work includes extending the theoretical pool of rules for refinement, the investigation of automating the development processes, and the optimisation of the realisation for more economical implementations with higher throughput.

Keywords Reconfigurable Computing, Hardware Synthesis, FPGAs, Formal Methods, Cryptography, Parallel Processing.

Available from http://academics.idamaj.net

Roger Needham Lecture

Tuning Systems: From Composition to Performance

Jane Hilston, University of Edinburgh

8 December 2004, 6.30 for 7pm
The Royal Society, London

http://www.bcs.org/BCS/Awards/Events/needhamlecture/

Title Formal Verification of X-machine Models: Towards Formal Development of Computer-Based Systems

Author George Eleftherakis

Institution University of Sheffield, UK

Supervisors Eur Ing Dr Anthony J. Cowling, Dr Petros Kefalas

Examiners Prof. Howard Barringer and Prof. Mike Holcombe

Awarded January 2004

Abstract

With the wide use of computers over the last decades a need to create more robust and safe software appeared. That need was the leading force towards the use of mathematically based methods in the development life cycle. Model Checking is a formal verification technique, which determines whether a given property expressed as a temporal logic formula is satisfied by a system model, usually a finite state machine. Temporal logic has been demonstrated to be a powerful specification language suitable for describing properties of a system modelled as a finite state machine. Finite state machines lack the ability to model non-trivial data structures, in contrast to X-machines, which can model both the data and the control part of a system. It would be desirable to apply model-checking techniques to X-machines. However, with existing logic it is obscure how one can describe properties that refer to the memory data structure of an X-machine model.

This research work investigates the feasibility of model checking X-machine models and describes the syntax and the semantics of a new logic, namely XmCTL, which extends CTL with memory quantifiers, and proposes new XmCTL model checking algorithms thus facilitating verification of X-machine models. Also it proposes a new approach for communicating X-machine models that offers a methodology for building component based systems based on X-machine components that communicate, separating the modelling of the components and their communication, allowing a disciplined and modular development of large-scale systems. Finally all these together with the already existing techniques for the X-machines are integrated to a new proposed formal lightweight development methodology for building reliable computer systems, that will be suitable for adoption by developers of industrial products improving the confidence of using them by proving the equivalence of the verified model and the final product. Simple examples are used throughout this document as vehicles of study of all the new ideas, and a case study at the end is used to demonstrate all the proposed techniques, adopting the proposed formal development methodology. As a result of this work, sixteen papers have been published in international journals and conferences, some of them aiming to disseminate the ideas proposed in this thesis to researchers outside the formal methods community.

Keywords Formal methods, Formal Verification, X-machines, Communicating Machines

Available from

http://www.dcs.shef.ac.uk/research/phdtheses/Eleftherakis2003.pdf

Title Delay-Insensitive Processes – A Formal Approach to the Design of Asynchronous Circuits

Author Hemangee K. Kapoor

Institution London South Bank University, UK

Supervisors Prof. Mark B. Josephs, Prof. Jonathan P. Bowen

Examiners Prof. Alex V. Yakovlev, Dr. Martin E. Bush

Awarded July 2004

Abstract

With the proliferation of electronic devices in our day-to-day existence, the quality of the underlying circuits is becoming increasingly important. The devices are expected to run robustly under different operating conditions. Asynchronous circuits are promising as compared to synchronous approach, in achieving low power, low noise and high-speed circuits, which can be developed in a modular way. However, the absence of a global clock in these circuits comes at the cost of added concurrency. Therefore, it is important to have a better understanding of such highly concurrent systems in order to have confidence in the resultant devices.

A formalism known as delay-insensitive (DI) processes is used to reason about a special class of asynchronous circuits that make no assumptions about delays in any of its components or wires. The formalism is shown to be useful in verification of such circuits using existing verification tools. DI processes can be easily integrated into such tools and existing equivalence checking techniques applied to them, instead of starting from scratch. In particular, the application of the Concurrency Workbench (CWB) to the verification of DI processes is shown by modelling them in CCS and using MUST-testing for equivalence and refinement checking.

DI processes interact with their environment to form closed systems. A new restriction operator is defined to obtain the effective behaviour of a process in a given environment, which eases specification and facilitates implementation. This operator is also shown to be more general than the alternation operator defined previously by Mallon.

Building on the work of Josephs and Udding, the algebraic semantics of DI processes is investigated and transformations are automated using the term rewriting system Maude. A canonical form is defined and is further used for equivalence and refinement checking based on syntactic comparison.

The formalism is useful not only in verification, but also in the decomposition of certain forms of processes that have been found difficult for logic synthesis tool, such as Petrify, to handle. The decompositions introduce Wires and Fork elements that preserve the delay-insensitive behaviour of asynchronous controllers. The proposed heuristics are applied on benchmark examples to help the tool Petrify to synthesise area-efficient circuits rapidly.

Besides using the CWB, Maude and Petrify, the experiments reported here have involved tools developed in-house, namely, Furey's translation tool *di2pn*, verification tool *diana* and the authors translation tool *di2ccs*.

The thesis demonstrates that (i) DI processes can be verified by adopting existing verification tools and techniques; (ii) consideration of the environment of a process leads to simpler specifications; and (iii) decomposing specifications leads to area efficient implementations.

Keywords Asynchronous Circuits, Delay-Insensitive Processes, DI-Algebra, DISP, Formal Verification, Concurrent Systems, CCS, CWB, Restrictive Environments, *di2pn*, *diana*, Algebraic Semantics, Term Rewriting, Maude, Process Decomposition, Logic Synthesis, Petrify

Available from http://www.bcim.lsbu.ac.uk/ccsv/hemangeeThesis.pdf

Joining Other Societies and Groups

Formal Aspects of Computing Science Specialist Group
http://www.bcs-facs.org/forms/

London Mathematical Society
http://www.lms.ac.uk/contact/membership.html

Safety-Critical Systems Club
http://www.safety-club.org.uk/membership.php

Formal Methods Europe
http://www.fmeurope.org/fme/member.htm

European Association for Theoretical Computer Science
http://www.eatcs.org/howtojoin.html

Association for Computing Machinery
https://campus.acm.org/Public/QuickJoin/interim.cfm

IEEE Computer Society
http://www.computer.org/join/

Title Stability of Test Criteria and Fault Hierarchies in Software Testing

Author Kalpesh Kapoor

Institution London South Bank University, UK

Supervisors Prof. Jonathan P. Bowen, Prof. Mark B. Josephs

Examiners Dr. Martin Woodward, Prof. Nimal Nissanke

Awarded August 2004

Abstract

Software testing is an important activity to verify and validate a system. A test criterion is a set of rules that are used for assessing the system under test. The effectiveness of a test criterion is measured in terms of its ability to reveal faults. Two key issues in software testing are: (a) effectiveness of test criteria in detecting faults, and (b) minimisation of test effort. These issues are studied empirically and formally within the framework of fault domain and test hypothesis.

Typically, for a given test criterion, more than one test set may satisfy the criterion for a specification and implementation. A new notion of stability of test criteria is defined to assess the variation in effectiveness of test sets for a given test criterion. Experimental evaluation of stability is performed for various types of coverage such as condition coverage, decision condition coverage, full-predicate coverage, modified condition decision coverage and reinforced condition decision coverage.

Fault detection effectiveness is also studied using a formal framework, which is applied to identify the conditions for the detection of various fault classes. It is shown that the number of test cases in a test set that can detect all hypothesised faults in an implementation depends on the complexity of the specification.

One of the main difficulties with the fault-based testing approach is due to the large number of possible elements in a fault domain or that can be generated on the basis of a test hypothesis. To overcome this problem, various conditions to establish fault hierarchies that help to identify stronger faults are described. Here, the objective is to establish the relationship between faults such that detection of one fault guarantees the detection of another. The analysis of fault hierarchies is also shown to be useful in validating the coupling hypothesis, which states that if a test technique can detect an implementation with one fault, it can also detect the presence of multiple faults.

The results obtained from the empirical and formal analysis provide an insight into earlier contradictory results regarding the effectiveness of test criteria. The formal framework helps in the classification of specifications and implementations in order to evaluate the effort required for testing; and the concept of fault hierarchy is useful in reducing test effort.

Keywords Control-flow testing, DC, CC, DCC, FPC, MCDC, RCDC, Fault, Detection Effectiveness, Mutation Testing, Strength of Mutants

Available from http://www.cafm.lsbu.ac.uk/publications/kkapoor-thesis.pdf

Title Goal-Oriented Requirements Analysis for Process Control Systems

Author Islam Ahmed El-Maddah

Institution King's College London, UK

Supervisors Prof. Dr TSE Maibaum, Dr K Lano

Examiners Prof. Bashar Nuseibeh, Prof. Anthony Finkelstein

Awarded May 2004

Abstract

The research addresses the field of software development of process control applications. The B formal method has been demonstrated to provide effective support at the early design stages. It has been constructively used in building specifications of process control systems.

However, the formal method assumes correct, complete, consistent and user-agreed formal requirements; this is difficult to achieve without a potential interaction between the systems engineer, as the client, and the software engineer, as the contractor, which entails concern-interference between the two and a detailed awareness of formal methods from the systems engineer. This indicates an existing gap between the systems engineer's perspective and the formal specification level.

The research attempts to fill this gap and focuses on the early requirements stages, which have a direct impact on the success of implementing the executable programs; the research focuses on automating the requirements analysis stage. This automation is achieved by implementing an interactive software tool, GOPCSD.

We adopted the KAOS method to structure and analyse the requirements; this provides the GOPCSD tool with its goal-oriented nature that enables refining the high-level user needs into low-level operational goals.

However, unlike the general KAOS method, the GOPCSD method was implemented to support process control systems. This motivated us to specialise and extend the KAOS method and to construct a library of process control requirements where the details of the frequently-used components and the abstract high-level functions of the process control applications are stored to be reused in similar applications.

In particular, we identified new refinement patterns, which were shown to be helpful in constructing process control applications. Furthermore, we extended the checks the user can apply to the goal-models by adding completeness, animation, and reachability tests.

After modifying the requirements, the GOPCSD tool automatically generates a B specification, which can be further processed by a software engineer within the B toolkit environment.

Two case studies of a gas burner and production cell are presented to examine the GOPCSD method and assess its supporting tool. Finally, we compare the specifications of the GOPCSD tool with other related methods.

Keywords Formal Methods, B, Specification Generation, Goal Driven Requirements Analysis, Requirements Checks, Reusability, Validation.

Jobs

The RODIN European Project at ETH in Zurich is looking for a post-doc and/or engineer candidate. The project has just started and will continue until September 2007. The main goal of this project at ETH is to develop an open platform essentially devoted to offer a series of new tools for supporting Event-B development. The candidate must have a very good knowledge of B (and also possibly Event-B). He or she will be part of a development team. The work will essentially consists of developing tools (probably in Java and under Eclipse). The working language is English.

Send letter of interest and vitae to

Jean-Raymond Abrial [jabrial@inf.ethz.ch]

UNU-IIST is currently offering a few post doctoral positions in Macao. Specific research projects include the existing projects Formal Methods in Object-Oriented and Component-Based Development, Real-time Embedded Systems, Software Testing, and new projects in model-checking, in formal models for security, and in decision support systems for natural resources, in particular water.

See UNU-IIST's home page http://www.iist.unu.edu for details.

FACS Committee

Jonathan Bowen
FACS Chair
ZUG Liaison

Jawed Siddiqi
Treasurer

Roger Carsley
Secretary

Paul Boca
Membership Sec.
Newsletter Editor

John Cooke
FAC Journal
Liaison

Ali Abdallah
Events Coordinator

John Fitzgerald
FME Liaison
SCSC Liaison

Margaret West
BCS Liaison

Mike Stannett
Webmaster
LMS Liaison

Judith Carlton
Industrial Liaison

Kevin Lano
UML Liaison

Rick Thomas
LMS Liaison

FACS is always interested to hear from its members and keen to recruit additional Committee members. Presently we have vacancies for officers to handle publicity and help with fund raising, and to liaise with other specialist groups such as the Requirements Engineering group and the European Association for Theoretical Computer Science (EATCS). If you are interested in helping the Committee, please contact the FACS Chair, Professor Jonathan Bowen, at the contact points below:

BCS FACS
c/o Prof. Jonathan Bowen (Chair)
London South Bank University
Faculty of BCIM
Borough Road
London SE1 0AA
United Kingdom

T +44 (0)20 7815 7462
F +44 (0)20 7815 7793
E info@bcs-facs.org.uk
W www.bcs-facs.org

You can also contact the other Committee members via this email address.

Please feel free to discuss any ideas you have for FACS or voice any opinions openly on the FACS mailing list (FACS@jiscmail.ac.uk). You can also use this list to pose questions and to make contact with other members working in your area. Note: only FACS members can post to the list; archives are accessible to everyone at http://www.jiscmail.ac.uk/lists/facs.html .

Coming Soon in FACS FACTS....

Reports on:

Trends in Functional Programming

FACS Away Day

Symposium on Teaching Formal Methods

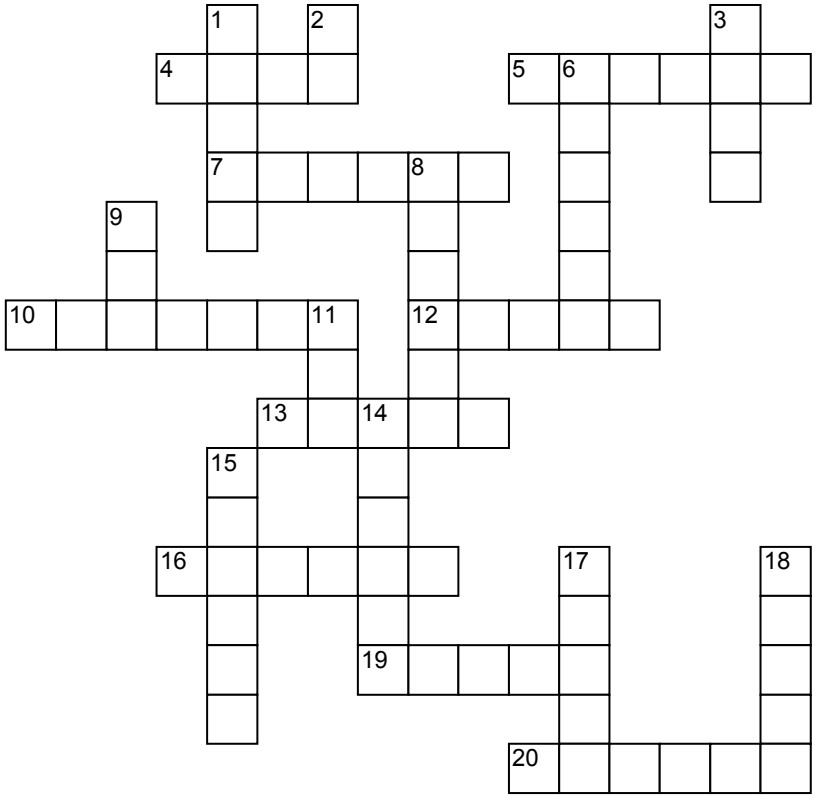
FACS Christmas Meeting

Next RefineNet Meeting

Program Verification and Semantics: Further Work

Formal Methods Coffee Time

Judith Carlton



Down

1 Representation of structure
2 A garage which sponsored the B method? (abbreviation)
3 Well-known for leaning, in part (geog.)
6 Dijkstra
8 Dutch graphic artist (1898-1972)
9 Emerged "precipitously" from IBM (abbreviation)
11 25 years of them now (abbreviation)
14 Jean-Raymond
15 Substantiate
17 A vestige left behind
18 Found on train

Across

4 Circle
5 The wall came down here a few years ago (geog.)
7 Parisian language?
10 Not an angelic choice
12 Tony
13 A bright radio engineer?
16 "a means of practice in a subject" from Bath
19 It runs over cliffs perhaps?
20 Procedure

FACS membership application/renewal (2005)

Title (Prof/Dr/Mr/Ms) _____ **First name** _____ **Last name**_____

Email address (required for options * below)_____

BCS membership No. (or sister society name + membership number)

_____

Address _____

_____

_____

Postcode _____ **Country** _____

I would like to take out **membership to FACS** at the following rate:

- ❑ £15 (Previous member of BCS-FACS now retired, unwaged or a student)
- ❑ £15 (Member of BCS or sister society with web/email access)*
- ❑ £30 (Non-member or member of BCS or sister society without web/email access)

In addition I would like to subscribe to **Volume 17 of the FAC journal** at the following rate:

- ❑ £46

For electronic only journal subscription*, please tick here ❑. No further discount given.

The total amount payable to BCS-FACS in **pounds sterling** is **£ 15 / 30 / 61 / 76** (delete as appropriate). I am paying by:

- ❑ Cheque made payable to **BCS-FACS** (**in pounds sterling**)
- ❑ Credit card via PayPal (instructions can be found on the BCS-FACS website)
- ❑ Direct transfer (in **pounds sterling**) to:

  Bank: Lloyds TSB Bank, Langham Place, London
  Sort Code: 30-94-87
  Account Number: 0173977
  Title of Account: BCS-FACS

If a receipt is required, please tick here ❑ and **enclose** a stamped self-addressed envelope.

Please send completed forms to:

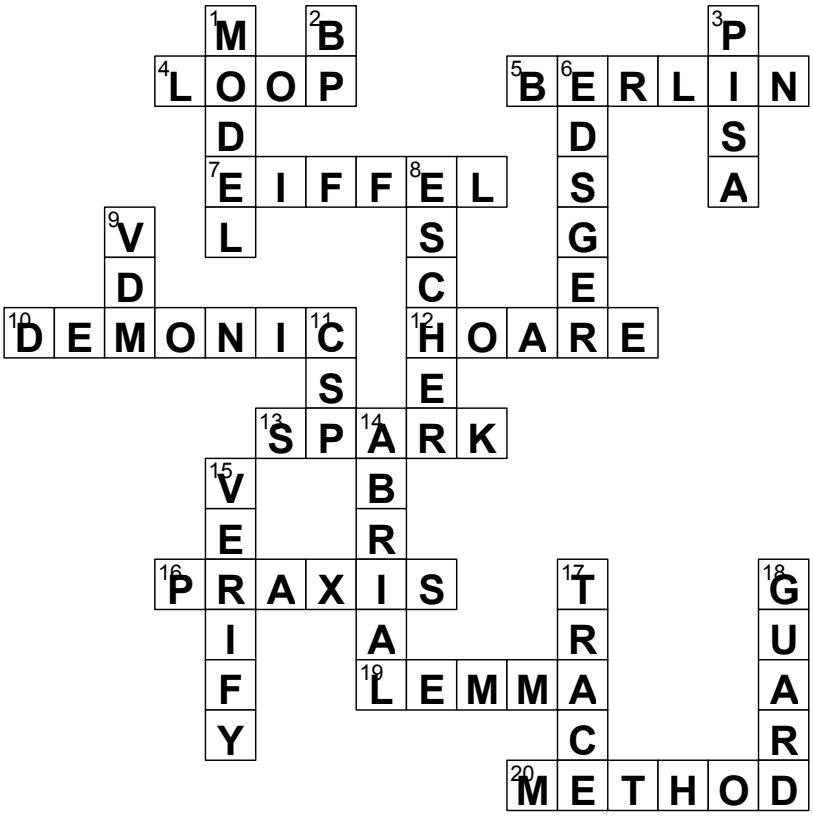
Dr Paul P Boca
PO BOX 32173
LONDON N4 4YP
UK

For FACS use only

Received by FACS	Date:	Initials:
Sent to Springer	Date:	Initials:
Actioned by Springer	Date:	Initials:

And Finally

Solution to Formal Methods Coffee Time:



Guess the caption competition

Prize:

One year's *free* subscription to BCS-FACS