Summary Proceedings of the Fifth New Zealand Formal Program Development Colloquium

Neil Leslie (Editor)
Institute of Information and Mathematical Sciences
Massey University at Albany
Private Bag 102-904
North Shore Mail Centre
Auckland
`N.Leslie@massey.ac.nz`

22 January 1999

Introduction

This document is a summary of the Proceedings of the Fifth New Zealand Formal Program Development Colloquium (NZFPDC), held at Massey University's Albany Campus in January 1999. For technical and copyright reasons the full proceedings will not be made available on the WWW. The full proceedings are published as Proceedings of the Fifth New Zealand Formal Program Development Colloquium, Leslie (Ed.), IIMS Technical Report 99-1 ISSN 1174-8273.

The first two NZFPDC meetings were both informal workshops. The third and fourth meetings, held under the umbrella title of Formal Methods Pacific, were more formal, and we felt that we should return the NZFPDC to its roots and, once again, hold a more informal workshop.

The NZFPDC was established to encourage and promote work in New Zealand in the areas of "formal methods", "programming foundations" and "formal software engineering". It is therefore a pleasure to see contributions in this volume from half the New Zealand Universities, and from colleagues in Australia and the UK. However, much work remains to be done to promote the benefits of the use formal methods for program development, both within academia and within industry. Those of us who work in this field surely do so not merely because it is fun but because we believe that formal methods are useful. Demonstrating the utility of formal methods and encouraging their use in industry remains the major challenge. It is a challenge we must accept.

Doug Goldson, Lindsay Groves, Paddy Krishnan, Ray Nickson, Steve Reeves and Mark Utting all assisted the Colloquium in various ways. The Colloquium has also benefited from the support, both moral and financial, of the Institute of Information and Mathematical Sciences (IIMS). Jeff Hunter, Kay Rowbottom and Diane Allen are to be thanked for their assistance.

This volume is also the first issue in the IIMS Technical Report Series. IIMS reports will be issued in two series: IIMS Technical Reports and IIMS Technical Notes. These series aim to provide timely publication of research by members of the Institute. Both series will be available by following the links from the Institute's website: `http://www.massey.ac.nz/~wwiims/`.

While these Proceedings mark the commencement of a new report series and the continuation of the NZFPDC, they also mark, for me personally, a conclusion, as I will soon be leaving Massey to take up a post at Victoria University of Wellington. I would like to thank my colleagues in Computer Science at Albany: Martin Johnson, Peter Kay and Chris Scogings; for being such a pleasure to work with. I am sure that the group at Albany will continue to grow and develop, and I hope that Massey University will recognise the quality and achievements of the staff here.

Neil Leslie

Abstracts of papers presented

Experiments with Model Checking for μ-Calculus in specification and verification project REAL

E.V. Bodin[*], V.E. Kozura[*], N.V. Shilov[*+]

[*]Institute of Informatics Systems of SD RAS,
6 Lavrent'ev av., Novosibirsk, 630090
Russia
{bodin,kozura,shilov}@iis.nsk.su
[+]School of Computing Science, University of Technology, Sydney,
P.O. Box 123, Broadway, NSW 2007
Australia
shilov@socs.uts.edu.au

Abstract

Combined real-time specification language for distributed systems (executable specifications) and their properties (logical specifications) is a kernel of specification and verification project REAL. This language has three levels: **Elementary**, **Basic** and **Conceptual**. **Elementary** and **Basic** levels have formal syntax and structural operational semantics, the **Conceptual** level is a general paradigm for REAL and has informal semantics. Executable specifications are syntactically similar and ideologically based on CCITT standard Specification and Design Language SDL-88, logical specifications are based on a combination of dynamic logic and branching temporal logic. An original model of real-time (multiple clocks) is incorporated in the **Conceptual** and **Basic** levels, but **Elementary** level is time-free. Problem-oriented approach to verification of distributed systems properties, presented by executable and, respectively, logical specifications, is a current research topic. The paper presents some experience with problem-oriented approach to verification of time-free **Elementary** specifications. The approach consists in (1)classification of properties and systems with respect to their syntactical structure, (2)formulate and validation of problem-oriented high-level proof principles, (3)design of proof-outlines in terms of proof principles, and (4)checking correctness of proof-outlines with help of global model-checking for finite state systems of moderate size. This model-checker is based on Faster Model Checking Algorithm for μ-Calculus — a program logic with fixed points. Since reliable model-checking for μ-Calculus and propositional variants of High Order Logics of programs is an essential part of REAL project then we also discuss our approach to validation of model-checkers. This approach consists in Hoare-style verification of prototypes and then an extensive testing of working releases. A verification example of a progress property for a parameterised distributed system is presented also.

An Example of Multiprogram Development

Doug Goldson

Institute of Information and Mathematical Sciences
Massey University at Albany
Private Bag 102-904
North Shore Mail Centre
Auckland
D.Goldson@massey.ac.nz

Abstract

Using an example due to W. Feijen, this presentation illustrates the use of the Gries-Owicki theory in proving the partial correctness of multiprograms. The example illustrates the use of heuristics in developing a multiprogram from a specification.

Conjoining Derivations of Loops and Recursions in the Refinement Calculus

Lindsay Groves

School of Mathematical and Computing Sciences
Victoria University of Wellington
Wellington
lindsay@mcs.vuw.ac.nz

Abstract

We review the approach to adapting program derivations, based on properties of a program conjunction operator, presented in a previous paper (Lindsay Groves. "Adapting Program Derivations using Program Conjunction". Proceedings of IRW/FMP '98, Grundy, Swenke and Vickers (Eds), Springer, 1998). The law for distributing conjunction over loops is extended to provide a more general law for combining recursion blocks with common structure.

Improving Software using Requirements Formalisation

Lindsay Groves*, Ray Nickson*, Greg Reeve+, Steve Reeves+ and Mark Utting+

*School of Mathematical and Computing Sciences
Victoria University of Wellington
Wellington
{lindsay,nickson}@mcs.vuw.ac.nz
+Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton
{gregr,stever,marku}@cs.waikato.ac.nz

Abstract

We give a brief overview of a recently started, Public Good Science Fund (PGSF) funded research project, which is running under the auspices of the government's Foundation for Research, Science and Technology (FoRST or FRST). This project comes within the broad subject area of 'Formal Methods' and specifically concerns the use of formalization during the requirements and specification stages of typical software development projects.

Using Invariants in Program Refinement

Ian Hayes+ and Ray Nickson*

+Department of Computer Science and Electrical Engineering
The University of Queensland
Brisbane 4072
Australia
Ian.Hayes@csee.uq.edu.au
*School of Mathematical and Computing Sciences
Victoria University of Wellington
Wellington
nickson@mcs.vuw.ac.nz

Abstract

When developing an imperative program via refinement, one often uses a sequential composition in which an earlier command, C, sets up data structures to be used, but not modified, by later commands. Normally the postcondition, R, of C needs to be explicitly passed around during the refinement of the later commands, so that the data structures set up by C can be used in the refinement. The explicit passing of R is tedious, when in this case R is invariant over all the later commands. Local invariants, as devised by Morgan and Vickers, can be used to factor out such conditions. In this paper we present new refinement laws that make use of local invariants to solve the above problem, and their incorporation into a program refinement tool that supports local invariants as a form a context.

Using Z: a note on methodology

Martin C. Henson[*] and Steve Reeves[+]

[*]Department of Computer Science
University of Essex
England
hensm@essex.ac.uk
[+]Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton
stever@cs.waikato.ac.nz

Abstract

We show how a constructive version of the schema calculus for Z can be defined, paying attention to the differences that emerge between state and operation schemas. We also show some notational efficiencies and, further, show how these can be used to support the derivation of programs which are specified using promotion, one of the very useful, high-level specification structures that have been proposed in recent years. The reader is assumed to be familiar with, or have access to, our recent "New Foundations for Z", Proceedings of IRW/FMP '98, Grundy, Swenke and Vickers (Eds), Springer, 1998.

Trajectories in Timed Systems

Padmanabhan Krishnan

Department of Computer Science
University of Canterbury
PBag 4800
Christchurch

Abstract

In this article we present a synthesis technique for generating schedulers for real-time systems. The aim of the scheduler is to ensure (via restricting behaviour) that the real-time system satisfies a given specification. The real-time systems and the specification are described as Alur-Dill timed automata while the synthesised scheduler is a timed trajectory automaton. We also note a simple constraint that the specification has to satisfy for this technique to be useful.

Specification-based testing of interactive systems using Object-Z and CSP

Ian MacColl and David Carrington

Software Verification Research Centre
University of Queensland, Australia
{ianm,davec}@csee.uq.edu.au

Abstract

In this paper we apply our framework for specification-based testing of interactive systems to a semantic integration of Object-Z and CSP. Interactive systems can be analysed and developed in terms of functionality, presentation and behaviour with different notations appropriate to each aspect. Testing information can be derived from formal specifications of each of these aspects. In this paper we specify functionality and representation using Object-Z, and behaviour using CSP, and we derive testing information from formally specified views of a scrollbar.

From Ideal to Realisable Real-Time Specifications

Graeme Smith

Software Verification Research Centre
University of Queensland, Australia
smith@it.uq.edu.au

Abstract

Formally refining a real-time specification to an implementation is only possible when the specification allows for all physical limitations, and timing and signal errors inherent in the implementation. Allowing for such implementation-specific details in a top-level specification can, however, obscure the desired functionality and complicate analysis. Furthermore, such an approach assumes the specifier has an understanding of the physical limitations and errors of the implementation which may not yet have been developed. As an alternative, we propose introducing a notion of *realisation* into the formal development process. Realisation is and approach to specification development which allows errors and physical limitations to be introduced. It also allows properties of the new specification to be derived from those proved for the original.

Implementing the Z_c Logic in Ergo

Mark Utting and Steve Reeves

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton
{marku,stever}@cs.waikato.ac.nz

Abstract

Henson and Reeves have recently proposed a new logic, for the Z specification language and proved that it is sound. In this paper, we describe work we have been doing on implementing the logic in Ergo 5. Ergo 5 is the latest version of a series of interactive proof tools that have been designed and implemented at the Software Verification Research Centre (Brisbane, Australia) over the last ten years. We describe two alternative ways of handling the complex side-conditions of some inference rules and show how the multiple context sequents of Ergo 5 allow the logic to be modelled more naturally.