Formal Methods Specification And Verification Guidebook

Among the most important problems confronting computer science is that of developing a paradigm appropriate to the discipline. Proponents of formal methods - such as John McCarthy, C.A.R. Hoare, and Edgar Dijkstra - have advanced the position that computing is a mathematical activity and that computer science should model itself after mathematics. Opponents of formal methods - by contrast, suggest that programming is the activity which is fundamental to computer science and that there are important differences that distinguish it from mathematics, which therefore cannot provide a suitable paradigm. Disagreement over the place of formal methods in computer science has recently arisen in the form of renewed interest in the nature and capacity of program verification as a method for establishing the reliability of software systems. A paper that appeared in Communications of the ACM entitled, `Program Verification: The Very Idea', by James H. Fetzer triggered an extended debate that has been discussed in several journals and that has endured for several years, engaging the interest of computer scientists (both theoretical and applied) and of other thinkers from a wide range of backgrounds who want to understand computer science as a domain of inquiry. The editors of this collection have brought together many of the most interesting and important studies that contribute to answering questions about the nature and the limits of computer science. These include early papers advocating the mathematical paradigm by McCarthy, Naur, R. Floyd, and Hoare (in Part I), others that elaborate the paradigm by Hoare, Meyer, Naur, and Scherlis and Scott (in Part II), challenges, limits and alternatives explored by C. Floyd, Smith, Blum, and Naur (in Part III), and recent work focusing on formal verification by DeMillo, Lipton, and Perlis, Fetzer, Cohn, and Colburn (in Part IV). It provides essential resources for further study. This volume will appeal to scientists, philosophers, and laypersons who want to understand the theoretical foundations of computer science and be appropriately positioned to evaluate the scope and limits of the discipline.

Formal Methods Specification and Verification Guidebook for Software and Computer Systems. Volume 1; Planning and Technology InsertionCreatespace Independent Publishing Platform

Real-time systems need to react to certain input stimuli within given time bounds. For example, an airbag in a car has to unfold within 300 milliseconds in a crash. There are many embedded safety-critical applications and each requires real-time specification techniques. This text introduces three of these techniques, based on logic and automata: duration calculus, timed automata, and PLC-automata. The techniques are brought together to form a seamless design flow, from real-time requirements specified in the duration calculus; via designs specified by PLC-automata; and into source code for hardware platforms of embedded systems. The syntax, semantics, and proof methods of the specification techniques are introduced; their most important properties are established; and real-life examples illustrate their use. Detailed case studies and exercises conclude each chapter. Ideal for students of real-time systems or embedded systems, this text will also be of great interest to researchers and professionals in transportation and automation.

Model checking is a powerful approach for the formal verification of software. It automatically provides complete proofs of correctness, or explains, via counter-examples, why a system is not correct. Here, the author provides a well written and basic introduction to the new technique. The first part describes in simple terms the theoretical basis of model checking: transition systems as a formal model of systems, temporal logic as a formal language for behavioral properties, and model-checking algorithms. The second part explains how to write rich and structured temporal logic specifications in practice, while the third part surveys some of the major model checkers available.

Specification and Verification of Systolic Arrays

Formal Methods Specification and Verification Guidebook for Software and Computer Systems. Volume 1; Planning and Technology Insertion

Tools for Formal Specification, Verification, and Validation of Requirements

Specifying and Programming the Steam Boiler Control

International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004. Revised Lectures

B is one of the few formal methods which has robust, commercially-available tool support for the entire development lifecycle from specification through to code generation. This volume provides a comprehensive introduction to the B Abstract Machine Notation, and to how it can be used to support formal specification and development of high integrity systems. A strong emphasis is placed on the use of B in the context of existing software development methods, including object-oriented analysis and design. The text includes a large number of worked examples, graduated exercises in B AMN specification and development (all of which have been class-tested), two extended case studies of the development process, and an appendix of proof techniques suitable for B. Based on material which has been used to teach B at postgraduate and undergraduate level, this volume will provide invaluable reading a wide range of people, including students, project technical managers and workers, and researchers with an interest in methods integration and B semantics.

Formal methods play an important part in the development as well as testing stages of software and hardware systems. A significant and often overlooked part of the process is the development of specifications and correctness requirements for the system under test. Traditionally, English has been used as the specification language, which has resulted in verbose and difficult to use specification documents that are usually abandoned during product development. This research focuses on investigating the use of Live Sequence Charts (LSCs), a graphical and intuitive language directly suited for expressing communication behaviors of a system as the specification language for a system under test. The research presents two methods for using LSCs as a specification language: first, by translating LSCs to temporal logic, and second, by translating LSCs to an automaton structure that is directly suited for formal verification of systems. The research first presents the translation for each method and further, identifies the pros and cons for each verification method.

As the complexity of embedded computer-controlled systems increases, the present industrial practice for their development gives cause for concern, especially for safety-critical applications where human lives are at stake. The use of software in such systems has increased enormously in the last decade. Formal methods, based on firm mathematical foundations, provide one means to help with reducing the risk of introducing errors during specification and development. There is currently much interest in both academic and industrial circles concerning the issues involved, but the techniques still need further investigation and promulgation to make their widespread use a reality. This book presents results of research into techniques to aid the formal verification of mixed hardware/software systems. Aspects of system specification and verification from requirements down to the underlying hardware are addressed, with particular regard to real-time issues. The work presented is largely based around the Occam programming language and Transputer microprocessor paradigm. The HOL theorem prover, based on higher order logic, has mainly been used in the application of machine-checked proofs. The book describes research work undertaken on the collaborative UK DTI/SERC-funded Information Engineering Dictorate Safemos project. The partners were Inmos Ltd., Cambridge SRI, the Oxford University Computing Laboratory and the University of Cambridge Computer Laboratory, who investigated the problems of formally verifying embedded systems. The most important results of the project are presented in the form

of a series of interrelated chapters by project members and associated personnel. In addition, overviews of two other ventures with similar objectives are included as appendices. The material in this book is intended for computing science researchers and advanced industrial practitioners interested in the application of formal methods to real-time safety-critical systems at all levels of abstraction from requirements to hardware. In addition, material of a more general nature is presented, which may be of interest to managers in charge of projects applying formal methods, especially for safety-critical-systems, and others who are considering their use.

This is the final report prepared under SBIR Phase II Contract N00014-95-C-0131 from the Office of Naval Research. It reports on the development of a set of software tools for specification and verification of distributed real time systems using formal methods. The task of this SBIR Phase II effort was to create a commercial-strength CASE toolset suitable for handling real-life verification problems. A preceding Phase I contract was concerned with development of the approach to the problem and design of a prototype environment 14. Forthcoming Phase III work will demonstrate the utility of the toolset to potential customers and establish it as a commercial off- the-shelf (COTS) specification and verification product. It is important to note that commercialization work, which is the task of Phase III efforts, has already begun during the current Phase II period (see Section IV). The toolset has been given the name PARAGON, which stands for 'Process-Algebraic Real-time Analysis with Graphics-Oriented Notation'. The toolset is indented to be used by designers of real-time systems for early detection of errors. The mathematical complexity of formal specification and verification has been hidden from the end users as much as possible. To achieve this, the specification language uses notions used by designers in their work as primitives. This provides for concise specifications, readable even by a non-specialist.

Integrated Formal Methods

Modular Specification and Verification of Object-Oriented Programs

Formal Engineering for Industrial Software Development

4th International Symposium on Leveraging Applications, ISoLA 2010, Heraklion, Crete, Greece, October 18-21, 2010, Proceedings

Software Tools for Formal Specification and Verification of Distributed Real-Time Systems

Systems and Software Verification

This volume contains the conference proceedings of the 4th International S- posium on Leveraging Applications of Formal Methods, Veri?cation and Vali- tion, ISoLA 2010, which was held in Greece (Heraklion, Crete) October 18–21, 2010, and sponsored by EASST. Following the tradition of its forerunners in 2004, 2006, and 2008 in Cyprus and Chalchidiki, and the ISoLA Workshops in Greenbelt (USA) in 2005, in Poitiers (France) in 2007, and in Potsdam (Germany) in 2009, ISoLA 2010 p- vided a forum for developers, users, and researchers to discuss issues related to the adoption and use of rigorous tools and methods for the speci?cation, ana- sis, veri?cation, certi?cation, construction, testing, and maintenance of systems from the point of view of their di?erent application domains. Thus, the ISoLA series of events serves the purpose of bridging the gap between designers and developers of rigorous tools, and users in engineering and in other disciplines, and to foster and exploit synergetic relationships among scientists, engineers, software developers, decision makers, and other critical thinkers in companies and organizations. In particular, by providing a venue for the discussion of c- mon problems, requirements, algorithms, methodologies, and practices, ISoLA aims at supporting researchers in their quest to improve the utility, reliability, ?exibility, and e?ciency of tools for building systems, and users in their search for adequate solutions to their problems.

Formal methods is the term used to describe the specification and verification of software and software systems using mathematical logic. Various methodologies have been developed and incorporated into software tools. An important subclass is distributed systems. There are many books that look at particular methodologies for such systems, e.g. CSP, process algebra. This book offers a more balanced introduction for graduate students that describes the various approaches, their strengths and weaknesses, and when they are best used. Milner's CCS and its operational semantics are introduced, together with notions of behavioural equivalence based on bisimulation techniques and with variants of Hennessy-Milner modal logics. Later in the book, the presented theories are extended to take timing issues into account. The book has arisen from various courses taught in Iceland and Denmark and is designed to give students a broad introduction to the area, with exercises throughout.

Through fundamental contributions from leading researchers, this volume describes the use of formal modeling methods in the areas of requirements, design and validation. The self-contained chapters provide readers with rich background information and a diverse breadth of specialist material.

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.

Model-Checking Techniques and Tools

Understanding Formal Methods

Formal Methods for Communication Protocol Specification and Verification

A Literature Survey on Formal Methods of Specification and Verification of Hardware Designs

Program Verification

Formal Methods of Program Verification and Specification

This book constitutes the refereed proceedings of the 13th International Conference on Formal Engineering Methods, ICFEM 2011, held in Durham, UK, October 2011. The 40 revised

full papers together with 3 invited talks presented were carefully reviewed and selected from 103 submissions. The papers address all current issues in formal methods and their applications in software engineering. They are organized in topical sections on formal models; model checking and probability; specification and development; security; formal verification; cyber physical systems; event-B; verification, analysis and testing; refinement; as well as theorem proving and rewriting.

This extensively revised and updated new edition of Specification of Software Systems builds upon the original focus on software specification with added emphasis on the practice of formal methods for specification and verification activities for different types of software systems and at different stages of developing software systems. Topics and features: provides a wide coverage of formal specification techniques and a clear writing style, supported by end-of-chapter bibliographic notes for further reading; presents a logical structure, with sections devoted to specification fundamentals, basics of formalism, logic, set theory and relations, property-oriented specification methods, and model-based specification techniques; contains end-of-chapter exercises and numerous case studies, with potential course outlines suggested in the Preface; covers Object-Z, B-Method, and Calculus of Communicating Systems; offers material that can be taught with tool-supported laboratory projects.

This book constitutes the refereed proceedings of the 10th International Conference on Formal Engineering Methods, ICFEM 2008, held in Kitakyushu-City, Japan, October 2008. The 20 revised full papers together with 3 invited talks presented were carefully reviewed and selected from 62 submissions. The papers address all current issues in formal methods and their applications in software engineering. They are organized in topical sections on specification and verification; testing; verification; model checking and analysis; tools; application of formal methods; semantics.

Formal Techniques in Real-Time and Fault-Tolerant Systems focuses on the state of the art in formal specification, development and verification of fault-tolerant computing systems. The term `fault-tolerance' refers to a system having properties which enable it to deliver its specified function despite (certain) faults of its subsystem. Fault-tolerance is achieved by adding extra hardware and/or software which corrects the effects of faults. In this sense, a system can be called fault-tolerant if it can be proved that the resulting (extended) system under some model of reliability meets the reliability requirements. The main theme of Formal Techniques in Real-Time and Fault-Tolerant Systems can be formulated as follows: how do the specification, development and verification of conventional and fault-tolerant systems differ? How do the notations, methodology and tools used in design and development of fault-tolerant and conventional systems differ? Formal Techniques in Real-Time and Fault-Tolerant Systems is divided into two parts. The chapters in Part One set the stage for what follows by defining the basic notions and practices of the field of design and specification of fault-tolerant systems. The chapters in Part Two represent the `how-to' section, containing examples of the use of formal methods in specification and development of fault-tolerant systems. The book serves as an excellent reference for researchers in both academia and industry, and may be used as a text for advanced courses on the subject.

Formal Methods: State of the Art and New Directions

15th International Conference, IFM 2019, Bergen, Norway, December 2–6, 2019, Proceedings

Formal Specification and Verification of Real-Time Sequential Control Systems

Real-Time Systems

Assessing Formal Methods for the Specification and Verification of Electronic Voting Protocols

Formal Specification and Verification of Microprocessor Systems

The Formal Methods Specification and Verification Guidebook for Software and Computer Systems describes a set of techniques called Formal Methods (FM), and outlines their use in the specification and verification of computer systems and software. Development of increasingly complex systems has created a need for improved specification and verification techniques. NASA's Safety and Mission Quality Office has supported the investigation of techniques such as FM, which are now an accepted method for enhancing the quality of aerospace applications. The guidebook provides information for managers and practitioners who are interested in integrating FM into an existing systems development process. Information includes technical and administrative considerations that must be addressed when establishing the use of FM on a specific project. The guidebook is intended to aid decision makers in the successful application of FM to the development of high-quality systems at reasonable cost. This is the first volume of a planned two-volume set. The current volume focuses on administrative and planning considerations for the successful application of FM. Unspecified Center...

As computer technology is used to control critical systems to an increasing degree, it is vital that the methods for developing and understanding these systems are substantially improved. The mathematical and scientific

foundations currently used are extremely limited which means that their correctness and reliability cannot be ensured to an acceptable level. Systems engineering needs to become a fully fledged scientific discipline and formal methods, which are characterised by their firm mathematical foundations, are playing a vital role in achieving this transition. This volume is based on the proceedings of the Formal Methods Workshop (FM91), held in Drymen, Scotland, 24-27 September 1991. This was the second workshop sponsored by the Canadian and US governments to address the role of formal methods in the development of digital systems. Traditionally, formal methods have evolved in isolation from more conventional approaches, and one of the aims of this workshop was to emphasise the benefits of integrating the two areas. The workshop concentrated on the themes of quality assurance, design methods and mathematical modelling techniques. Particular emphasis was given to safety and security applications. Among the topics covered in this volume are: what is a formal method?; social research on formal methods; current quality assurance methods and formal methods; a pragmatic approach to validation; integrating methods in practice; composition of descriptions; and topics in large program formal development. Formal Methods in Systems Engineering provides an overview of many of the major approaches to formal methods and the benefits which can result from them. It is relevant to academic and industrial researchers, industrial practitioners and government workers with an interest in certification.

This book, with the CD-ROM included, is the documentation of a unique collaborative effort in evaluating formal methods for usage under industrial constraints: the major techniques for formally supported specification, design, and verification of large programs and complex systems are applied to a non-trivial and non-academic problem which is typical for industrial informal requirements specifications. The 21 papers included in the book, together with an introduction and competition report, were selected from 33 candidate solutions. This book comes with a CD-ROM containing, besides the printed papers, executable code, full definitions of all parts of the specifications, and detailed descriptions of foundational matters where appropriate.

Formal methods are mathematically-based techniques, often supported by reasoning tools, that can offer a rigorous and effective way to model, design and analyze computer systems. The purpose of this study is to evaluate international industrial experience in using formal methods. The cases selected are representative of industrial-grade projects and span a variety of application domains. The study had three main objectives: · To better inform deliberations within industry and government on standards and regulations; · To provide an authoritative record on the practical experience of formal methods to date; and À To suggest areas where future research and technology development are needed. This study was undertaken by three experts in formal methods and software engineering: Dan Craigen of ORA Canada, Susan Gerhart of Applied Formal Methods, and Ted Ralston of Ralston Research Associates. Robin Bloomfield of Adelard was involved with the Darlington Nuclear Generating Station Shutdown System case. Support for this study was provided by organizations in Canada and the United States. The Atomic Energy Control Board of Canada (AECB) provided support for Dan Craigen and for the technical editing provided by Karen Summerskill. The U.S. Naval Research Laboratories (NRL), Washington, DC, provided support for all three authors. The U.S. National Institute of Standards and Technology (NIST) provided support for Ted Ralston.

With Isabelle/HOL

Leveraging Applications of Formal Methods, Verification, and Validation

Formal Specification and Automatic Verification

FORMAL METHODS OF PROGRAM VERIFICATION AND SPECIFICATION

Verification of Reactive Systems

Fundamental Issues in Computer Science

This is an excellent introduction to formal methods which will bring anyone who needs to know about this important topic up to speed. It is comprehensive, giving the reader all the information needed to explore the field of formal methods in more detail. It offers: a guide to the mathematics required; comprehensive but easy-to-understand introductions to various methods; a run-down of how formal methods can help to develop high-quality systems that come in on time, within budget, and according to requirements.

This book constitutes the refereed proceedings of the 15th International Conference on Formal Engineering Methods, ICFEM 2013, held in Queenstown, New Zealand, in October/November 2013. The 28 revised full papers together with 2 keynote speeches presented were carefully reviewed and selected from 88 submissions. The topics covered are abstraction and refinement, formal specification and modeling, program analysis, software verification, formal methods for software safety, security, reliability and dependability, tool development, integration and experiments involving verified systems, formal methods used in certifying products under international standards, and formal model-based development and code generation.

This book constitutes the refereed proceedings of the 4th International Conference on Formal Engineering methods, ICFEM 2002, held in Shanghai, China, in October 2002. The 43 revised full papers and 16 revised short papers presented together with 5 invited contributions were carefully reviewed and selected from a total of 108 submissions. The papers are organized in topical sections on component engineering and software architecture, method integration, specification techniques and languages, tools and environments, refinement, applications, validation and verification, UML, and semantics.

Increasingly numerous and complex communication protocols are being employed in distributed systems and computer networks of all types. This Note describes some of the more formal techniques that are being developed to facilitate design of correct protocols. Our major conclusion is that it is vital to specify the services provided by a protocol layer in addition to specifying the cooperating protocol entities which make up the layer. We develop service specifications of several representative protocols by using formal techniques from software engineering such as abstract machines and buffer histories. A survey of protocol verification methods and a bibliography indexed by key phrases are also provided. (Author).

Towards Verified Systems
4th International Conference on Formal Engineering Methods, ICFEM 2002, Shanghai, China, October 21-25, 2002, Proceedings
Using Live Sequence Chart Specifications for Formal Verification of Systems
Formal Methods in Computer Science
Integrating Formal Specification and Verification Methods in Software Development
13th International Conference on Formal Engineering Methods, ICFEM 2011, Durham, UK, October 26-28, 2011. Proceedings

This book constitutes the refereed proceedings of the 15th International Conference on Integrated Formal Methods, IFM 2019, held in Bergen, Norway, in December 2019. The 25 full papers and 3 short papers were carefully reviewed and selected from 95 submissions. The papers cover a broad spectrum of topics: from language design to verification and analysis techniques, to supporting tools and their integration into software engineering practice including both theoretical approaches and practical implementations. Also included are the extended abstracts of 6 "journal-first" papers.

This book presents the revised versions of nine invited lectures presented by leading researchers at the fourth edition of the International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFT 2004, held in Bertinoro, Italy, September 2004. SFM 2004 is devoted to real-time systems. The lectures presented cover formal models and languages for the specification, modeling, analysis, and verification of time-critical systems, the expressiveness of such models and languages, as well as supporting tools and related applications in different domains. The book offers a unique and comprehensive state-of-the-art survey on real-time systems. Researchers and advanced students will appreciate the book as a valuable source of reference and a systematic guide to the use of formal methods for the specification, analysis, and verification of real-time systems.

Circuits and architectures have become more complex in terms of structure, interconnection topology, and data flow. Design correctness has become increasingly significant, as errors in design may result in strenuous debugging, or even in the repetition of a costly manufacturing process. Although circuit simulation has been used traditionally and widely as the technique for checking hardware and architectural designs, it does not guarantee the conformity of designs to specifications. Formal methods therefore become vital in guaranteeing the correctness of designs and have thus received a significant amount of attention in the CAD industry today. This book presents a formal method for specifying and verifying the correctness of systolic array designs. Such architectures are commonly found in the form of accelerators for digital signal, image, and video processing. These arrays can be quite complicated in topology and data flow. In the book, a formalism called STA is defined for these kinds of dynamic environments, with a survey of related techniques. A framework for specification and verification is established. Formal verification techniques to check the correctness of the systolic networks with respect to the algorithmic level specifications are explained. The book also presents a Prolog-based formal design verifier (named VSTA), developed to automate the verification process, as using a general purpose theorem prover is usually extremely time-consuming. Several application examples are included in the book to illustrate how formal techniques and the verifier can be used to automate proofs. Contents:Specification and Verification of Systolic Arrays — Definitions and Related WorkSystolic Temporal Arithmetic — A FormalismSpecification and Verification FrameworkSpecification and Verification of Systolic Arrays — Application ExamplesVSTA — A Special Purpose Formal Verifier for Systolic DesignsVerifying the Correctness of a Systolic Array for LU Decomposition Readership: Professors, researchers, postgraduate and graduate students in supercomputing parallel processing, formal verification and DSP architecture. Keywords:Systolic Arrays;Formal Verification;Formal Specification;Formal Methods;Hardware Verification;Digital Signal Processing Architecture;Systolic Temporal Arithmetic;Verifier;CAD Tools;Prolog

This book is a solid foundation of the most important formalisms used for specification and verification of reactive systems. In particular, the text presents all important results on m-calculus, w-automata, and temporal logics, shows the relationships between these formalisms and describes state-of-the-art verification procedures for them. It also discusses advantages and disadvantages of these formalisms, and shows up their strengths and weaknesses. Most results are given with detailed proofs, so that the presentation is almost self-contained. Includes all definitions without relying on other material Proves all theorems in detail Presents detailed algorithms in pseudo-code for verification as well as translations to other formalisms

Using the SOFL Method

Formal Methods and Digital Systems Validation for Airborne Systems
Formal Methods for the Design of Real-Time Systems
Formal Methods and Software Engineering
Industrial Applications of Formal Methods to Model, Design and Analyze Computer Systems
Formal Techniques in Real-Time and Fault-Tolerant Systems

Software systems play an increasingly important role in modern societies. Smart cards for personal identi?cation, e-banking, software-controlled me- cal tools, airbags in cars, and autopilots for aircraft control are only some examples that illustrate how everyday life depends on the good behavior of software. Consequently, techniques and methods for the development of hi- quality, dependable software systems are a central research topic in computer science. A fundamental approach to this area is to use formal speci?cation and veri?cation. Speci?cation languages allow one to describe the crucial p- perties of software systems in an abstract, mathematically precise, and implementation-independent way. By formal veri?cation, one can then prove that an implementation really has the desired, speci?ed properties. Although this formal methods approach has been a research topic for more than 30 years, its practical success is still restricted to domains in which devel- ment costs are of minor importance. Two aspects are crucial to widen the application area of formal methods: – Formal speci?cation techniques have to be smoothly integrated into the software and program development process. – The techniques have to be applicable to reusable software components. This way, the quality gain can be exploited for more than one system, thereby justifying the higher development costs. Starting from these considerations, Peter Muller¨ has developed new te- niques for the formal speci?cation and veri?cation of object-oriented so- ware. The speci?cation techniques are declarative and implementati- independent. They can be used for object-oriented design and programming.

A variety of methods have been employed for specifying the behavior of sequential control systems that, e.g., can be implemented by programmable logic controllers (PLC's). In dealing with real-time issues, timed Petri nets and other popular formalisms provide limited capabilities with respect to both specification and analysis of behavior. At the same time, many of the formal methods that have been reported in the literature are not readily applicable to the area of sequential control. In this paper, we explore the application of a formal method for real-time systems based on hierarchical multi-state (HMS) machines, to the specification and verification of sequential control systems. Verification in this context goes beyond simulation and testing by attempting to provide mathematical proofs for correctness of behavior with respect to general safety properties. HMS machines, which were originally introduced in 1988, consist of parallel and hierarchical automata in which multiple states are true and multiple transitions can fire simultaneously. The temporal behavior of an HMS machine is defined in terms of an interval-based temporal logic. Examples of applications of HMS machines to sequential control problems and an overview of verification approaches are presented.

This textbook gives students a comprehensive introduction to formal methods and their application in software and hardware specification and verification. It has three parts: The first part introduces some fundamentals in formal methods, including set theory, functions, finite state machines, and regular expressions. The second part focuses on logi

Part I of this book is a practical introduction to working with the Isabelle proof assistant. It teaches you how to write functional programs and inductive definitions and how to prove properties about them in Isabelle’s structured proof language. Part II is an introduction to the semantics of imperative languages with an emphasis on applications like compilers and program analysers. The distinguishing feature is that all the mathematics has been formalised in Isabelle and much of it is executable. Part I focusses on the details of proofs in Isabelle; Part II can be read even without familiarity with Isabelle’s proof language, all proofs are described in detail but informally. The book teaches the reader the art of precise logical reasoning and the practical use of a proof assistant as a surgical tool for formal proofs about computer science artefacts. In this sense it represents a formal approach to computer science, not just semantics. The Isabelle formalisation, including the proofs and accompanying slides, are freely available online, and the book is suitable for graduate students, advanced undergraduate students, and researchers in theoretical computer science and logic.

The B Language and Method
15th International Conference on Formal EngineeringMethods, ICFEM 2013, Queenstown, New Zealand, October 29 - November 1, 2013, Proceedings
Formal Methods for Industrial Applications
Formal Methods in Systems Engineering
Specification of Software Systems
Concrete Semantics

Although formal methods for developing computer systems have been available for more than a decade, few have had significant impact in practice. A major barrier to their use is that software developers find formal methods difficult to understand and apply. One exception is a formal method called SCR for specifying computer system requirements which, due to its easy to use tabular notation and its demonstrated scalability, has already achieved some success in industry. Recently, a set of software tools, including a specification editor, a consistency checker, a simulator, and a verifier, has been developed to support the SCR method [9, 11, 5]. This paper describes recent enhancements to the SCR tools: a new dependency graph browser which displays the dependencies among the variables in the specification, an improved consistency checker which produces detailed feedback about detected errors, and an assertion checker which checks application properties during simulation. To illustrate the tool enhancements, a simple automobile cruise control system is presented and analyzed.

Formal Methods for Program Verification and Specification
A Guide to Practical Formal Development
Reactive Systems
10th International Conference on Formal Engineering Methods ICFEM 2008, Kitakyushu-City, Japan, October 27-31, 2008, Proceedings
Formal Methods and Algorithms
Modelling, Specification and Verification