1.

| | |
|---|---|
| Record Nr. | UNINA9910886092703321 |
| Autore | Foster Simon |
| Titolo | The Application of Formal Methods : Essays Dedicated to Jim Woodcock on the Occasion of His Retirement / / edited by Simon Foster, Augusto Sampaio |
| Pubbl/distr/stampa | Cham : , : Springer Nature Switzerland : , : Imprint : Springer, , 2024 |
| ISBN | 3-031-67114-7 |
| Edizione | [1st ed. 2024.] |
| Descrizione fisica | 1 online resource (0 pages) |
| Collana | Lecture Notes in Computer Science, , 1611-3349 ; ; 14900 |
| Altri autori (Persone) | SampaioAugusto |
| Disciplina | 005.131 |
| Soggetti | Machine theory<br>Programming languages (Electronic computers)<br>Software engineering<br>Formal Languages and Automata Theory<br>Programming Language<br>Software Engineering |
| Lingua di pubblicazione | Inglese |
| Formato | Materiale a stampa |
| Livello bibliografico | Monografia |
| Nota di contenuto | Intro -- Preface -- Dedications -- Contents -- Denotational and Algebraic Semantics for the SMrCaIT Calculus Based on UTP -- 1 Background -- 2 Introduction -- 3 Backgrounds -- 3.1 Mobility Models -- 3.2 Scope Extrusion -- 3.3 The SMrCaIT Calculus -- 3.4 Guided Choice -- 4 Denotational Semantics -- 4.1 Semantic Model -- 4.2 Healthiness Conditions -- 4.3 Denotational Semantics of Basic Commands -- 4.4 Parallel Composition -- 5 Algebraic Semantics -- 5.1 Algebraic Laws of Basic Commands -- 5.2 Algebraic Laws of Parallel Composition -- 5.3 Algebraic Laws of Channel Restriction and Scope Extrusion -- 6 Conclusion and Future Work -- A Further Explanation of SMrCaIT -- References -- Specifying Fault-Tolerant Mixed-Criticality Scheduling -- 1 Introduction -- 2 Background -- 2.1 Rely-Guarantee ``thinking'' -- 2.2 Notes on Notation -- 3 Formalising Planning and Scheduling -- 4 Specifying the Fixed Priority Scheduler -- 4.1 Tasks Define Job Types -- 4.2 Job Arrival Assumptions -- 4.3 Assigning Priorities During Planning -- 4.4 State of Run-Time Model -- 4.5 Time vs Computer Clocks -- 4.6 Scheduler Class and Methods -- |

4.7 The FP Planning Theorem -- 5 Comparison Between the Current and Previous Models -- 6 Conclusions -- References -- Clarifying Assumptions -- 1 Introduction -- 1.1 Verification of Sequential Programs -- 1.2 Development Methods -- 1.3 Concurrency -- 1.4 Structure of Chapter -- 2 Reasoning About Interference on States -- 2.1 Background to Rely-Guarantee Research -- 2.2 Verification (Post Facto) of concurrent programs -- 2.3 Recovering Compositionality -- 2.4 Data Abstraction/Reification -- 2.5 Ghost Variables -- 2.6 Using a Fiction of Atomicity -- 3 Developments of ``rely-guarantee thinking'' -- 3.1 Assumptions About External Components -- 3.2 Semantics and Tool Support -- 3.3 Related Research -- 3.4 Tackling Synchronisation.

4 Further Abstractions -- 4.1 Algebraic Presentation -- 4.2 Implementation Patterns -- 5 Conclusions -- References -- PCSP# Denotational Semantics with an Application in Sports Analytics -- 1 Introduction -- 2 The Probabilistic CSP# Language (PCSP#) -- 3 The Denotational Semantics for PCSP# -- 3.1 Semantic Model -- 3.2 Semantics of Processes -- 4 Sports Analytics Using Probabilistic Model Checking -- 4.1 Overview -- 4.2 The Proposed Approach -- 4.3 Data Mining -- 4.4 Modeling Tennis in MDP -- 4.5 Implementation in PCSP# -- 4.6 Strategy Recommendations -- 4.7 Evaluation -- 5 Related Work -- 5.1 Integration and Evolution of Formal Specification Languages in System Modeling and Verification -- 5.2 Sports Analytics in Racket Sports -- 6 Conclusion -- References -- Towards the Composition of Digital Twins -- 1 Dedication -- 2 Introduction -- 3 Background -- 3.1 Cyber-Physical Systems, Digital Twins and DT-Enabled Systems -- 3.2 System of Systems -- 4 Composing DT-Enabled Systems -- 4.1 A Fleet of Digital Twins -- 4.2 A Hierarchy of Digital Twins -- 4.3 Collaboration Between Digital Twins -- 5 Realising Composition of DT-Enabled Systems -- 5.1 Building DTs from Reusable Components -- 5.2 The Digital Twin as a Service Platform -- 5.3 Support for a Fleet and Hierarchy of Digital Twins -- 5.4 Support for Collaboration Between Digital Twins -- 6 The Flex Cell Use Case -- 7 Concluding Remarks -- References -- Formal Modelling of Peercoin and Proof-of-Stake Protocols -- 1 Introduction -- 2 Peercoin -- 2.1 Peercoin Protocol -- 3 Formal Modelling -- 3.1 A Brief Introduction to PRISM -- 3.2 Modelling the Protocol -- 3.3 Desirable Properties -- 4 Results and Analysis -- 5 Conclusions -- References -- On the Unification of Conformance Notions -- 1 Introduction -- 1.1 Background and Objectives -- 1.2 Main Contributions -- 1.3 Overview.

2 Finite State Machines and Conformance Relations -- 2.1 Definition of Finite State Machines -- 2.2 FSM Conformance Relations -- 2.3 Critical Assessment of FSM Conformance Relations -- 2.4 Unified Equivalence and Reduction Concepts for FSMs -- 2.5 Option-Preserving Reduction - A Novel Conformance Relation -- 3 A UTP Theory for FSMs -- 3.1 FSMs as UTP Processes -- 3.2 Universal Quasi-Equivalence and Universal Reduction in UTP -- 4 Conclusion -- References -- Abstracting and Verifying Decentralised Systems in CSP -- 1 Introduction -- 2 Background -- 2.1 CSP -- 2.2 FDR -- 3 Stochastic Reasoning -- 3.1 Judging Impossibility -- 3.2 Stochastic Proof -- 4 Consensus Machine -- 5 The Handover Protocol -- 5.1 Sequential Model -- 5.2 Distributed Model -- 6 Szymanski's Algorithm -- 7 Related Work -- 8 Conclusions -- References -- Towards an Algebra for Unifying Theories of Concurrent Programming (UTCP) -- 1 Introduction -- 1.1 Introduction to UTCP -- 1.2 Structure of This Chapter -- 2 Background -- 2.1 UTP -- 2.2 UTCP -- 2.3 Formal Definition of CKA -- 3 Methodology -- 3.1 Composing Atomic Actions -- 3.2 W(…): The Wash Cycle -- 3.3 Atomic Action Flows -- 3.4 Composite Flows -- 3.5 Atomic Control Merge -- 4 Law Proofs -- 4.1 Proving Rearrangments -- 4.2 Proving Collapses --

4.3 Laws Involving miracle -- 4.4 Laws Involving Refinement -- 5 Related Work -- 6 Conclusions -- 6.1 Future Work -- References -- Static Race Detection for Periodic Real-Time Programs with IPCP Locks -- 1 Introduction -- 2 Overview -- 3 Periodic Programs with IPCP Locks -- 4 Data Races -- 5 Some Properties of IPCP Programs -- 6 Computing Response Time -- 7 Rules of Disjointness -- 7.1 Race Detection Algorithm -- 8 Experimental Evaluation -- 8.1 Implementation -- 8.2 Benchmarks -- 8.3 Comparison -- 8.4 Rule Coverage -- 9 Related Work -- 10 Conclusion -- References.

A Tour Through the Programming Choices: Semantics and Applications -- 1 Dedication -- 2 Introduction -- 3 Programming Choices -- 3.1 Nondeterministic Choice -- 3.2 Angelic Choice -- 3.3 Preferential Choice -- 3.4 Probabilistic Choice -- 4 Applications -- 4.1 Co-simulation for Cyber-Physical Systems -- 4.2 RoboChart -- 4.3 Random Walks -- 5 Conclusions -- References -- I Kaptured the System -- 1 Introduction -- 2 An Overview of System Kapture -- 2.1 The Purpose of System Kapture -- 2.2 Core Concepts -- 2.3 Underlying Semantics -- 3 Kapturing a System -- 3.1 An Iteration of the System Requirements -- 4 Analysis of System Requirements -- 4.1 Absence of Errors and Warnings -- 4.2 System Always Progresses and Rounds Complete in Bounded Time -- 4.3 Queries -- 4.4 No Violation of Emergent Requirements -- 5 Conclusions -- References -- Proving B with Atelier B -- 1 Introduction -- 2 Introduction to the B Method -- 3 Atelier B Proof Obligations -- 3.1 Characteristics and Constraints -- 3.2 Evolution -- 4 Atelier B Proof System -- 4.1 Automatic Proof -- 4.2 Extending the Prover -- 4.3 Validation -- 5 B Industrial Applications -- 6 Adaptation over Years -- 7 Conclusion and Perspectives -- References -- Semantics Formalisation - Modelling and Proving Strategies Using Event-B Versus Theories -- 1 Introduction -- 2 Background -- 2.1 Event-B -- 2.2 The Theory Plugin -- 2.3 SCXML -- 3 Approach 1. Using Event-B Contexts and Machines -- 3.1 Formalisation of the Untriggered Statechart Syntactic Elements -- 3.2 Formalisation of the Untriggered Statechart Semantics -- 3.3 Formalisation of the Run-to-Completion Syntactic Elements -- 3.4 Formalisation of the Run-to-Completion Semantics -- 3.5 Formalisation of Triggered Statecharts -- 4 Approach 2. Formalisation Using Theories -- 4.1 Algebraic Formalisation of Untriggered Statecharts.

4.2 Algebraic Formalisation of Run-to-Completion Schedule -- 4.3 Algebraic Formalisation of Triggered Statecharts -- 5 Proof Rules for the Algebraic Data Types -- 5.1 Expanding Operators' Definition -- 5.2 Rewrite Rules -- 5.3 Inference Rules -- 6 Related Work -- 7 Conclusion -- References -- Author Index.

Sommario/riassunto

This Festschrift, dedicated to Jim Woodcock, contains papers written by many of his closest collaborators. After a PhD on software verification at the University of Liverpool, Jim has combined a successful career in academia with outstanding industry research, in particular he has been a pioneer in applying mathematical modelling approaches in critical industries. At GEC's Hirst Research Centre he worked on a novel distributed telephone exchange and a service specification of a PABX exchange. In Oxford he collaborated with IBM Hursley Laboratories on modelling of the CICS transaction processing system, one of the most significant software systems ever. As part of the UK government's cybersecurity strategy, he used Z techniques to develop secure office automation systems and a secure version of UNIX. He worked with the Smith Institute and BR Research to verify the safety of railway signalling systems, approaches developed further in safety-critical control systems for the UK Nuclear Installation Inspectorate and British Energy.

He provided a technically complete theory of correctness for Z, verifying its soundness from first principles, and completed the verification of Mondex, a smartcard-based electronic cash system, the first application of a general theory of program correctness to an industrial product. He coordinated the experimental work of the Verified Software Initiative, an international grand challenge. More recently he extended the collection of standard Unifying Theories of Programming (UTP) with work on object orientation and hybrid systems. Currently he is working on a UTP theory of probabilistic programs with application to robotics. Jim has been a lecturer, research fellow, reader and professor at the University of Surrey, the University of Oxford, the University of Kent, and since 2004 the University of York, and he is a visiting professor at the Federal University of Pernambuco and Trinity College Dublin. He is a Fellow of the Royal Academy of Engineering, the British Computer Society, and the Formal Methods Europe association, and he was part of the team that won the Queen’s Award for Technological Achievement in 1992. He is the Editor-in-Chief of the ACM journal Formal Aspects of Computing, he has chaired major related academic conferences, and he has contributed to CCITT and Z ISO international standards. Throughout all these activities, Jim has been a guide and inspiration to colleagues and students, and collaborated successfully with researchers in the UK, Brazil, China, France, USA, Ireland, and Singapore. Many of these researchers show in their contributions to this volume the ongoing impact of his work.