1. Record Nr. UNISA996475771003316

Titolo	NASA formal methods : 14th international symposium, NFM 2022, Pasadena, CA, USA, May 24-27, 2022, proceedings / / edited by Jyotirmoy V. Deshmukh, Klaus Havelund, and Ivan Perez
Pubbl/distr/stampa	Cham, Switzerland : , : Springer, , [2022] ©2022
ISBN	3-031-06773-8
Descrizione fisica	1 online resource (846 pages)
Collana	Lecture Notes in Computer Science ; ; v.13260
Disciplina	004.0151
Soggetti	Formal methods (Computer science)
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Includes index.
Nota di contenuto	Intro -- Preface -- Organization -- Abstracts of Invited Tutorials -- Total Functional Programming in Idris: A Tutorial -- The Lean 4 Theorem Prover and Programming Language: A Tutorial -- Formally Reasoning about Distributed Systems using P -- Contents -- Invited Keynotes -- Formal Methods for Trusted Space Autonomy: Boon or Bane? -- 1 Introduction -- 2 Past Verification and Validation of Autonomy Flight Software -- 2.1 Remote Agent Experiment -- 2.2 Autonomous Sciencecraft on Earth Observing One -- 2.3 WATCH/SPOTTER on Mars Exploration Rovers -- 2.4 AEGIS on MER, MSL, and M2020 -- 2.5 MSL FSW -- 2.6 Intelligent Payload Experiment (IPEX) -- 3 Current Validation of Autonomy Software: Onboard Planner for M2020 -- 4 Discussion of Competing Verification and Validation Methods -- 4.1 Limitations of Testing and Informal Methods -- 4.2 Limitations of Formal Methods -- 5 Conclusions -- References -- An Essence of Domain Engineering -- 1 Introduction -- 1.1 What Is a Domain? -- 1.2 Structure of Paper -- 2 Philosophy: What Must be in any Domain Description? -- 2.1 The Search -- 2.2 Sørlander's Findings -- 2.3 The Basis -- 3 Elements of Domain Science and Engineering -- 3.1 Phenomena, Entities, Endurants and Perdurants -- 3.2 Endurants -- 3.3 Transcendental Deduction -- 3.4 Perdurants -- 3.5 The Domain Analysis and Description Process -- 4 An Example Domain Description -- 4.1 Endurants -- 4.2 Perdurants -- 5 Relevance to Aeronautics and

Space -- 5.1 But First -- 5.2 Air Traffic Control, ATC -- 5.3 An Aeronautics and Space Domain -- 6 Conclusion -- References -- Concept Design Moves -- 1 Introduction: Codifying Design Expertise -- 1.1 Designers Bring Prior Knowledge -- 1.2 Standard Solutions and Moves -- 1.3 Design vs. Engineering -- 2 Concept Structuring -- 2.1 Concept Independence -- 2.2 Concept Synchronization -- 3 Design Moves: Mechanical Analogues.
3.1 Split/Merge -- 3.2 Unify/Specialize -- 3.3 Tighten/Loosen -- 4 Concept Design Moves: Software Examples -- 4.1 Split: Emergence of a Concept in Keynote -- 4.2 Merge: The Yellkey URL Shortener -- 4.3 Unify: MITs Moira Service -- 4.4 Specialize: Three Similar Concepts in Lightroom -- 4.5 Tighten: Page Scheduling in Hugo -- 4.6 Loosen: Expert Control in ProCamera -- 5 Solving Problems with Design Moves -- 5.1 Aspect Ratio in Fujifilm Cameras -- 5.2 Message Filters in Apple Mail -- 5.3 Event Deletion in Calendars -- 5.4 Sticky Hands in Zoom -- 6 Discussion -- References -- Automating Program Transformation with Coccinelle -- 1 Introduction -- 2 Background -- 3 Coccinelle in a Nutshell, Illustrated by kzalloc -- 3.1 First Steps -- 3.2 A Refinement -- 3.3 A Second Refinement -- 4 A Second Example: of_node_put -- 4.1 The Problem -- 4.2 The Semantic Patch -- 4.3 Scaling Up -- 4.4 Impact -- 5 A Third Example: Inconsistent Atomicity Flags -- 5.1 The Problem -- 5.2 The Solution -- 6 Related Work -- 7 Conclusion -- References -- The Prusti Project: Formal Verification for Rust -- 1 Introduction -- 2 Prusti from a User's Perspective -- 2.1 (Almost) Zero-Cost Verification -- 2.2 Modular Verification of User-Specified Contracts -- 2.3 The Prusti Specification Language -- 2.4 Incremental Verification in Practice -- 3 Prusti from a Verification Expert's Perspective -- 3.1 Core Proofs in an Off-the-Shelf Separation Logic -- 3.2 Full Automation of Core Proofs for Type-Checked Rust -- 3.3 Incorporating Rich Functional Specifications -- 4 Prusti from a Tool Engineer's Perspective -- 4.1 Architecture and Design Overview -- 4.2 Specification Embedding -- 4.3 Compiler Interface -- 4.4 Encoding to Viper -- 5 Related Work -- 6 Conclusions and Future Work -- References -- Reachability Analysis for Cyber-Physical Systems: Are We There Yet? -- 1 Introduction.
2 Hybrid Systems and Reachability Analysis -- 2.1 Reachability Analysis -- 3 Set-Propagation Approaches -- 3.1 Linear Hybrid Systems -- 3.2 Nonlinear Hybrid Systems -- 4 Scaling up Reachability Analysis -- 5 Neural Network Controlled Systems -- 6 Conclusions -- References -- Regular Submissions -- Towards Better Test Coverage: Merging Unit Tests for Autonomous Systems -- 1 Introduction -- 2 Background -- 3 Problem Setup -- 4 Merging Unit Tests -- 4.1 Merging Test Specifications -- 4.2 Temporal Constraints on the Merged Test Specification -- 4.3 Receding Horizon Synthesis of Test Policy Filter -- 4.4 Searching for a Test Policy -- 5 Examples -- 5.1 Lane Change -- 6 Conclusion and Future Work -- 7 Appendix -- 7.1 Construction of the Partial Order -- 7.2 Live Lock -- 7.3 Example: Lane Change -- 7.4 Example: Unprotected Left Turn -- References -- Quantification of Battery Depletion Risk Made Efficient -- 1 Introduction -- 2 Battery Kinetics -- 3 Algorithms -- 3.1 Static Discretization -- 3.2 Adaptive Discretization -- 3.3 Percentile Propagation -- 4 Evaluation -- 5 Conclusion -- References -- Hierarchical Contract-Based Synthesis for Assurance Cases -- 1 Introduction -- 2 Hierarchical Contract Networks -- 2.1 Contract Networks and Library -- 2.2 Conditional Refinement and Hierarchical Contract Networks -- 3 Automatic Synthesis -- 3.1 Well-Formed Library -- 3.2 Synthesis Algorithm -- 4 Application: Assurance Cases -- 4.1 Assurance Case as a Hierarchical Contract Network -- 4.2 Case Study: Assurance Cases for Certification -- 5

Conclusion -- References -- Verified Probabilistic Policies for Deep Reinforcement Learning -- 1 Introduction -- 2 Background -- 3 Modelling and Abstraction of Reinforcement Learning -- 3.1 Modelling and Verification of Reinforcement Learning -- 3.2 Abstractions for Verification of Reinforcement Learning.
4 Template-Based Abstraction of Neural Network Policies -- 4.1 Bounded Template Polyhedra -- 4.2 Constructing Policy Abstractions -- 4.3 Refinement of Abstract States -- 5 Experimental Evaluation -- 5.1 Experimental Setup -- 5.2 Experimental Results -- 6 Conclusion -- References -- NNLander-VeriF: A Neural Network Formal Verification Framework for Vision-Based Autonomous Aircraft Landing -- 1 Introduction -- 2 Problem Formulation -- 3 Framework -- 4 Neural Network Augmentation -- 5 Identifying the Allowable Control Actions Using Symbolic Abstractions -- 6 Numerical Example -- 7 Conclusion and Future Work -- References -- The Black-Box Simplex Architecture for Runtime Assurance of Autonomous CPS -- 1 Introduction -- 2 Black-Box Simplex -- 2.1 Formal Definition of Black-Box Simplex -- 2.2 Safety and Transparency Theorems -- 3 Case Studies -- 3.1 Multi-robot Coordination -- 3.2 Multi-aircraft Collision Avoidance -- 4 Related Work -- 5 Conclusions -- References -- Case Studies for Computing Density of Reachable States for Safe Autonomous Motion Planning -- 1 Introduction -- 2 Related Work -- 3 Problem Formulation -- 4 Technical Approaches -- 4.1 Data-driven Reachability and Density Estimation -- 4.2 Reach Set Probability Estimation -- 4.3 Motion Planning Based on Reachability Analysis -- 5 Experiments -- 5.1 Reachable States and Density Estimation -- 5.2 Online Planning via Reachable Set Density Estimation -- 5.3 Discussions -- 6 Conclusion -- A Car Model Dynamic and Controller Designs -- B Hovercraft Model Dynamic and Controller Designs -- C Nonlinear Programming for Controller Synthesize -- References -- Towards Refactoring FRETish Requirements -- 1 Introduction and Background -- 2 Refactoring Requirements -- 2.1 Analysis: Aircraft Engine Controller Requirements -- 2.2 Refactoring Requirements -- 3 Towards FRET-Supported Refactoring -- 4 Conclusion.
References -- Neural Network Compression of ACAS Xu Early Prototype Is Unsafe: Closed-Loop Verification Through Quantized State Backreachability -- 1 Introduction -- 2 Background and Problem Formulation -- 2.1 Collision Avoidance System Design -- 2.2 Assumptions and Plant Model -- 2.3 Reachability with AH-Polytopes -- 2.4 Safety Problem Formulation -- 3 Quantized State Backreachability -- 3.1 Partitioning the Unsafe States -- 3.2 Backreachability from Each Partition -- 3.3 Falsification of Original (Unquantized) System -- 4 Evaluation -- 4.1 Complete Proof of Safety Attempt -- 4.2 Proving Safety in More Limited Operating Conditions -- 4.3 Comparison with Other Approaches -- 5 Related Work -- 6 Conclusion -- References -- ZoPE: A Fast Optimizer for ReLU Networks with Low-Dimensional Inputs -- 1 Introduction -- 2 Background -- 3 Optimization Problems -- 4 Approach -- 4.1 Optimization with Branch and Bound -- 4.2 Split, UpperBound, LowerBound -- 4.3 Implementation -- 5 Experimental Results -- 5.1 ACAS Xu Benchmark -- 5.2 Optimizing Convex Functions -- 5.3 Maximum Distance Between Compressed and Original Networks -- 6 Conclusion -- A Appendix -- A.1 Maximum Distance Between Points in Two Hyperrectangles -- A.2 Verifier Configuration for the Collision Avoidance Benchmark -- References -- Permutation Invariance of Deep Neural Networks with ReLUs -- 1 Introduction -- 2 Preliminaries -- 3 Informal Overview -- 3.1 Running Example -- 4 Forward and Backward Propagation -- 4.1 Forward Propagation Using Tie Classes -- 4.2 Backward (Polytope) Propagation -- 4.3 Inclusion

Checking and Counterexample Propagation -- 4.4 Example (continued from Sect.3.1) -- 5 Experiments -- 6 Related Work -- 7 Conclusion -- References -- Configurable Benchmarks for C Model Checkers -- 1 Introduction -- 2 Tool and Code Generation -- 2.1 Code Generation. 3 Benchmarking the Open-Source Verifiers.