1. Record Nr. UNINA9910633937403321

Titolo	Static analysis : 29th International Symposium, SAS 2022, Auckland, New Zealand, December 5-7, 2022, proceedings / / editors Gagandeep Singh and Urban Caterina
Pubbl/distr/stampa	Cham, Switzerland : , : Springer, , [2022] ©2022
ISBN	3-031-22308-X
Descrizione fisica	1 online resource (482 pages)
Collana	Lecture notes in computer science ; ; Volume 13790
Disciplina	005.1
Soggetti	Computer programming Programming languages (Electronic computers)
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Nota di bibliografia	Includes bibliographical references and index.
Nota di contenuto	Intro -- Preface -- Organization -- Contents -- Invited Talks -- Specification-Guided Reinforcement Learning -- 1 Introduction -- 2 Reinforcement Learning from Logical Specifications -- 3 Algorithms -- 4 Theoretical Guarantees -- References -- Towards Efficient Reasoning of Quantum Programs -- 1 First Section -- References -- Regular Papers -- Solving Invariant Generation for Unsolvable Loops -- 1 Introduction -- 2 Preliminaries -- 3 From Loops to Recurrences -- 4 Defective Variables -- 5 Synthesising Invariants -- 5.1 Solution Space of Invariants for Unsolvable Operators -- 6 Adjusting Defective Variables for Unsolvable Operators in Probabilistic Programs -- 7 Applications of Unsolvable Operators Towards Invariant Synthesis -- 8 Experiments -- 9 Conclusion -- References -- Principles of Staged Static+Dynamic Partial Analysis -- 1 Introduction -- 2 Background: The PYE Framework -- 2.1 Conditional Values -- 2.2 Evaluation of Conditional Values -- 3 Partial Evaluation and Futamura Projections -- 3.1 Partial Evaluation -- 3.2 First Futamura Projection -- 3.3 Second Futamura Projection -- 3.4 Third Futamura Projection -- 4 Staged Partial Analysis -- 4.1 Partial Analysis -- 4.2 First AM Projection -- 4.3 Second AM Projection -- 4.4 Third AM Projection -- 4.5 Correctness, Precision, and Efficiency of Staging -- 5 Specializers for Partial-Result Evaluation -- 5.1 A Grammar for Conditional Values -- 5.2

Conditional-Value Evaluators and Specialization -- 5.3 Running the Partial-Result Evaluators -- 6 Directions and Connections -- 6.1 Runtime Features: Challenges and Possibilities -- 6.2 Drawing Newer Connections -- 7 Related Work -- 7.1 Partial Evaluation -- 7.2 Partial Program Analysis -- 7.3 Other Applications of Partial Evaluation -- 7.4 Staged Analysis -- 8 Conclusion -- References -- SecWasm: Information Flow Control for WebAssembly -- 1 Introduction. 2 Background on Wasm -- 2.1 Basics -- 2.2 Structured Control Flow -- 2.3 Linear Memory -- 3 Challenges and Design Choices -- 3.1 Attacker Model -- 3.2 Unstructured Linear Memory -- 3.3 Structured Control Flow -- 3.4 A-Equivalences -- 3.5 Big-Step Semantics -- 4 SecWasm -- 4.1 Syntax -- 4.2 Semantics -- 4.3 Security Type System -- 5 Security Properties -- 6 Discussion -- 7 Related Work -- 8 Conclusions -- References -- Lifting Numeric Relational Domains to Algebraic Data Types -- 1 Introduction -- 2 Syntax and Semantics -- 2.1 Algebraic Types and Values -- 2.2 A Language with Algebraic Data Types -- 2.3 Running Example -- 3 Extending Numeric Domains to Algebraic Types -- 3.1 Background: Numeric Abstract Domains -- 3.2 Extended Variables -- 3.3 Numeric Domains over Extended Variables -- 3.4 Constructor Constraints -- 3.5 Structural Equalities -- 3.6 Bringing Everything Together: Product Domain and Disjunctive Completion -- 4 A Collecting Semantics of Relations -- 4.1 Relational Collecting Semantics -- 4.2 Leveraging Relations in Space to Express Relations in Time -- 5 Analysis -- 5.1 Analysis Result for the !doticks! Function -- 5.2 Intra-procedural Analysis -- 5.3 Analysis of Function Calls -- 6 Implementation, Experimental Results and Complexity -- 7 Related Work -- 8 Conclusion and Future Work -- References -- Automated Synthesis of Asynchronizations -- 1 Introduction -- 2 Asynchronous Programs -- 3 Synthesizing Asynchronous Programs -- 4 Enumerating Sound Asynchronizations -- 4.1 Enumeration Algorithm -- 5 Computing Maximal Asynchronizations -- 5.1 Data Race Ordering -- 5.2 Repairing Data Races -- 5.3 A Procedure for Computing Maximal Asynchronizations -- 6 Asymptotic Complexity of Asynchronization Synthesis -- 7 Asynchronization Synthesis Using Data-Flow Analysis -- 8 Experimental Evaluation -- 9 Related Work -- 10 Conclusion -- References.
Case Study on Verification-Witness Validators: Where We Are and Where We Go -- 1 Introduction -- 2 Evaluation -- 3 Suggestions for Advances in Witness Validation -- 4 Conclusion -- References -- Deciding Program Properties via Complete Abstractions on Bounded Domains -- 1 Introduction -- 2 Background -- 2.1 Notation -- 2.2 Abstract Interpretation -- 2.3 Programs -- 2.4 Conditions for Completeness of Guards -- 3 Bounded Domains -- 4 Program Termination -- 4.1 Deciding Program Termination -- 4.2 Exploiting Boolean Abstractions -- 5 Program Equivalence -- 5.1 Backward Computation -- 5.2 Select Normal Form -- 5.3 Normal Form Scaling in Combined Domains -- 5.4 Deciding Program Equivalence -- 6 Conclusions -- References -- Invariant Inference with Provable Complexity from the Monotone Theory -- 1 Introduction -- 2 Preliminaries -- 3 Background: The Monotone Theory -- 3.1 Least b-Monotone Overapproximations -- 3.2 Monotone Hull -- 4 Super-Efficient Monotonization -- 5 Efficient Inference of CDNF Invariants -- 5.1 Background: Interpolation with the Fence Condition -- 5.2 CDNF Inference with the Fence Condition -- 6 Efficient Implementation of Abstract Interpretation -- 6.1 Background: Abstract Interpretation in the Monotone Theory -- 6.2 Complexity Upper Bound -- 7 Related Work -- 8 Conclusion -- References -- Efficient Modular SMT-Based Model Checking of Pointer Programs -- 1 Introduction -- 2 Related Work -- 3 Motivating Example -- 4 Static

Analysis of Memory Footprints -- 5 Theory of Finite Maps -- 6 A CHC Encoding with Finite Maps -- 7 Experimental Evaluation -- 8 Conclusions -- References -- Property-Driven Code Obfuscations Reinterpreting Jones-Optimality in Abstract Interpretation -- 1 Introduction -- 2 Background -- 2.1 The Language `39`42`" 613A``45`47`"603AL and Control Flow Graphs -- 2.2 The Language Semantics -- 3 Symbolic Finite State Machines.
3.1 Finite State Machines -- 3.2 Finite State Transducers -- 3.3 Example: Parser as Symbolic Pushdown Automaton -- 4 Program (Re) Interpretation -- 4.1 First Phase: The Execution Sequence Extractor -- 4.2 Second Phase: The Semantic Interpretation -- 4.3 Interpreting Programs -- 4.4 Specializing Interpreters -- 5 Specializer (Dis) Optimality -- 5.1 Abstract Jones Optimality and Completeness -- 5.2 Distorting Interpreters -- 6 Obfuscation by Specializing Distorted Interpreters -- 7 Conclusions -- References -- Bootstrapping Library-Based Synthesis -- 1 Introduction -- 2 Overview -- 3 The Toshokan Framework -- 3.1 Libraries -- 3.2 The Library-Based Synthesis Problem -- 3.3 Inductive Synthesis with Angelic Libraries -- 3.4 Verifier and Logger -- 3.5 The Main Synthesis Algorithm -- 4 Angelic Inductive Synthesis -- 5 The Logger -- 6 Evaluation -- 7 Related Work -- 8 Conclusion -- References -- Boosting Robustness Verification of Semantic Feature Neighborhoods -- 1 Introduction -- 2 Preliminaries -- 3 Verification of Feature Neighborhoods: Motivation -- 4 Problem Definition: Time-Optimal Feature Verification -- 5 Prediction by Proof Velocity and Sensitivity -- 6 VeeP: A System for Time-Optimal Feature Verification -- 6.1 VeeP for Single Feature Neighborhoods -- 6.2 VeeP for Multi-feature Neighborhoods -- 7 Evaluation -- 8 Related Work -- 9 Conclusion -- References -- Fast and Incremental Computation of Weak Control Closure -- 1 Introduction -- 2 Background -- 3 Incremental Computation of WCC -- 3.1 Generating the Influencer Graph -- 3.2 An Incremental Algorithm to Compute WCC -- 3.3 Proof of Correctness -- 3.4 Worst-Case Time Complexity -- 4 Experimental Evaluation -- 5 Related Work -- 6 Conclusion and Future Works -- References -- Local Completeness Logic on Kleene Algebra with Tests -- 1 Introduction -- 2 Background on Kleene Algebra with Tests.
3 Local Completeness Logic in KAT -- 3.1 Program Properties in KAT -- 3.2 KAT Language -- 3.3 Kleene Abstractions -- 3.4 Local Completeness Logic on BdKAT -- 3.5 An Example of a Language-Theoretic KAT -- 3.6 Under-Approximation Logic -- 4 Incorrectness Logic in KAT -- 4.1 Relationship with Incorrectness Logic -- 5 Local Completeness Logic in TopKAT -- 5.1 Abstracting TopKATs -- 5.2 Local Completeness Logic on TopKAT -- 5.3 Relationship with Under-Approximation Logic -- 6 Conclusion -- References -- Semantic Foundations for Cost Analysis of Pipeline-Optimized Programs -- 1 Introduction -- 2 Processor Behavior on an Example -- 3 Concrete Small-Step Pipeline Semantics -- 4 Static Analysis -- 4.1 Instrumentation for a Numerical Analysis -- 4.2 Proof of Soundness -- 5 Implementation -- 6 Experiments -- 7 Related Work -- 8 Conclusion -- References -- Parameterized Recursive Refinement Types for Automated Program Verification -- 1 Introduction -- 2 Target Language -- 2.1 Syntax -- 2.2 Typing -- 2.3 Operational Semantics -- 3 A Parameterized Refinement Type System -- 3.1 Refinement Types -- 3.2 Typing -- 4 Inferring Parameterized Refinement Types -- 4.1 Step 2: Instantiation of Raw Types with F"0365F -- 4.2 Step 3: Reduction to CHC Solving -- 5 Implementation and Experiments -- 5.1 Implementation -- 5.2 Experiments and Results -- 6 Related Work -- 7 Conclusion -- A Details of Experiments -- References -- Adversarial Logic -- 1 Introduction -- 2 Motivation -- 3 Adversarial Logic -- 4

Reasoning with Adversarial Logic -- 4.1 Example 1: Trivial Case -- 4.2 Example 2: Oscillating Bit Protocol -- 4.3 Example 3: Equivalence Testing -- 5 Semantics -- 6 Alternative Presentation -- 6.1 Dynamic Logic -- 6.2 Information Systems -- 7 Related Work -- 8 Conclusion and Future Work -- References.

CLEVEREST: Accelerating CEGAR-based Neural Network Verification via Adversarial Attacks.