1. Record Nr. UNINA9910483818103321

Titolo	Programming Languages and Systems : 12th Asian Symposium, APLAS 2014, Singapore, Singapore, November 17-19, 2014, Proceedings / / edited by Jacques Garrigue
Pubbl/distr/stampa	Cham : , : Springer International Publishing : , : Imprint : Springer, , 2014
ISBN	3-319-12736-5
Edizione	[1st ed. 2014.]
Descrizione fisica	1 online resource (XVIII, 490 p. 117 illus.)
Collana	Programming and Software Engineering ; ; 8858
Disciplina	005.13
Soggetti	Programming languages (Electronic computers) Software engineering Computer logic Mathematical logic Programming Languages, Compilers, Interpreters Software Engineering Logics and Meanings of Programs Mathematical Logic and Formal Languages
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Includes Index.
Nota di contenuto	Intro -- Preface -- Organization -- What Is the Essence of Bidirectional Programming? -- Incremental Adoption of Static-Typing -- NetKAT - A Formal System for the Verification of Networks -- Table of Contents -- Invited Presentation -- NetKAT - A Formal System for the Verification of Networks -- 1 Introduction -- 1.1 Software-Defined Networking -- 1.2 NetKAT -- 2 NetKATBasics -- 2.1 Kleene Algebra (KA) -- 2.2 Kleene Algebra with Tests (KAT) -- 2.3 NetKAT -- 2.4 Semantics -- 3 Examples -- 3.1 Encoding Network Topology -- 3.2 Switch Policies -- 3.3 Reachability -- 3.4 All-Pairs Reachability -- 3.5 Waypointing -- 3.6 Forwarding Loops -- 3.7 Other Applications -- 4 Soundness and Completeness -- 5 NetKAT Coalgebra and a Decision Procedure -- 5.1 NetKAT Coalgebra -- 5.2 The Brzozowski Derivative -- 5.3 Matrix Representation -- 5.4 Kleene's Theorem for NetKAT -- 6 Implementation -- 6.1 Optimizations -- 7 Related Work -- 8

Conclusion -- References -- Regular Papers -- Optimized Compilation of Multiset Rewriting with Comprehensions -- 1 Introduction -- 2 A Motivating Example -- 3 Syntax and Notations -- 4 Operational Semantics of CHRcp -- 4.1 Semantics of Matching and Rule Body Execution -- 4.2 Operational Semantics -- 5 Compiling CHRcp Rules -- 5.1 Introducing CHRcp Join Ordering -- 5.2 Bootstrapping for Active Comprehension Head Constraints -- 5.3 Uniqueness Enforcement -- 6 Building Join Orderings -- 7 Executing Join Orderings -- 8 Correctness of CHRcp Abstract Matching Machine -- 9 Prototype and Preliminary Empirical Results -- 10 RelatedWork -- 11 Conclusion and Future Works -- References -- Logic Programming and Logarithmic Space -- 1 Introduction -- 1.1 Geometry of Interaction and Logic Programming -- 1.2 Unification and Complexity -- 2 The Unification Semiring -- 2.1 Flows and Wirings -- 2.2 The Balanced Semiring -- 2.3 The Computation Graph.
2.4 Tensor Product and Other Semirings -- 3 Words and Observations -- 3.1 Representation of Words -- 3.2 Observations -- 4 Logarithmic Space -- 4.1 Completeness: Observations as Pointer Machines -- 4.2 Soundness of Observations -- 5 Conclusion -- References -- Automatic Memory Management Based on Program Transformation Using Ownership -- 1 Introduction -- 2 Suenaga-Kobayashi Type System -- 2.1 Language -- 2.2 Type System -- 3 Program Transformation -- 3.1 Casts -- 3.2 Constraints -- 3.3 Algorithm -- 3.4 Soundness and Completeness -- 3.5 Extension -- 4 Related Work -- 5 Conclusion -- References -- The Essence of Ruby -- 1 Introduction -- 2 Overview of Ruby and Our Strategies -- 3 The Essential Core of Ruby -- 3.1 The Core Object Calculus -- 3.2 The Core Control Calculus -- 3.3 The Core Ruby Calculus -- 4 Extension to the Core Calculi -- 5 The Ruby Calculus -- 6 Elaborating Ruby to the Ruby Calculus -- 7 Conformity Evaluation -- 8 Related Works -- 9 Conclusions -- References -- Types for Flexible Objects -- 1 Introduction -- 1.1 Key Features of TinyBang -- 2 Overview -- 2.1 Language Features for Flexible Objects -- 2.2 Self-awareness and Resealable Objects -- 2.3 Flexible Object Operations -- 3 Formalization -- 3.1 A-Translation -- 3.2 Operational Semantics -- 3.3 Type System -- 4 Related Work -- 5 Conclusions -- References -- A Translation of Intersection and Union Types for the -Calculus -- 1 Introduction -- 2 Intersection and Union Types for the -Calculus -- 2.1 The -Calculus -- 2.2 An Intersection and Union Type System for the -Calculus -- 2.3 The Type System of van Bakel, Barbanera and de'Liguoro -- 2.4 A Translation of Intersection and Union Types -- 3 Intersection and Union Types for the -Calculus -- 3.1 The -Calculus -- 3.2 An Intersection and Union Type System for the -Calculus -- 3.3 Translating into .
3.4 Characterisation of Strongly Normalising Terms -- 4 Conclusion -- References -- A Formalized Proof of Strong Normalization for Guarded Recursive Types -- 1 Introduction -- 2 Guarded Recursive Types and Their Semantics -- 3 Formalized Syntax -- 3.1 Types Represented Coinductively -- 3.2 Well-Typed Terms -- 3.3 Type Equality -- 3.4 Examples -- 4 Reduction -- 5 Strong Normalization -- 6 Soundness -- 7 Conclusions -- References -- Functional Pearl: Nearest Shelters in Manhattan -- 1 Specification -- 2 Looking Toward the Northeast -- 3 A Divide-and-Conquer Approach -- 3.1 Finding the Nearest Shelter in a List Homomorphism -- 3.2 Sweeping -- 3.3 Complexity Analysis -- 4 A Thinning Approach -- 4.1 Minimum, Thinning, and Filtering -- 4.2 Thinning the Set of Shelters -- 4.3 A Splay Tree Representation -- 4.4 Complexity Analysis -- 5 Conclusion -- References -- A Proof of Lemma 1 -- A Flexible Language for Policies -- 1 Introduction -- 2

Suppl by Example -- 3 Suppl in Detail -- 3.1 Event Handlers -- 3.2 Predicates, Types, and Modes -- 4 Conflict Detection -- 5 Implementation -- 6 Related Work -- 7 Conclusion -- References -- A Method for Scalable and Precise Bug Finding Using Program Analysis and Model Checking -- 1 Introduction -- 2 Related Work -- 3 Illustrative Example -- 4 Model-Based Analysis -- 4.1 Specialised Abstraction -- 4.2 Example Revisited -- 4.3 Function Summaries and Interprocedural Support -- 5 Implementation -- 6 Experimental Results -- 6.1 Evaluation of Precision and Recall Against Benchmarks -- 6.2 Evaluation Using OpenJDK -- 6.3 Threats to Validity -- 7 Conclusion and Future Work -- References -- Model-Checking for Android Malware Detection -- 1 Introduction -- 2 Android Applications -- 3 Program Model -- 3.1 Pushdown Systems -- 3.2 Modeling Android Applications as PDSs -- 4 Android (Malicious) Behaviors Specifications. 4.1 The SCTPL Logic -- 4.2 The SLTPL Logic -- 4.3 SLTPL and SCTPL for Android Applications -- 4.4 Expressing Android (Malicious) Behaviors in SCTPL and SLTPL -- 5 Model-Checking Android Applications -- 5.1 Annotating the Program with encode Predicates -- 5.2 SCTPL and SLTPL Model-Checking for Android Applications -- 6 Experiments -- 6.1 Information-Leak Android Applications -- 6.2 Checking the OtherMalicious Behaviors -- 7 Related Work -- References -- Necessary and Sufficient Preconditions via Eager Abstraction -- 1 Introduction -- 2 Example -- 3 Preliminaries -- 4 EagerAbstraction -- 5 Experimental Results -- 6 Related Work -- 7 Conclusion -- References -- A Inference Rules -- Resource Protection Using Atomics -- 1 Introduction -- 2 Synchronisation in Java -- 3 Ownership Exchange via Atomics -- 3.1 Basic Rules -- 3.2 Synchronisation Protocol -- 3.3 Specifications of Atomics -- 3.4 Thread-Modular Contracts -- 4 Contracts of AtomicInteger -- 4.1 Specification Language -- 4.2 Predicates and Parameters -- 4.3 Specification -- 4.4 Verification -- 5 Related Work -- 6 Conclusion -- References -- Resource Analysis of Complex Programs with Cost Equations -- 1 Introduction -- 2 Cost Equations -- 3 Control-Flow Refinement of Cost Equations -- 3.1 Chain Refinement of an SCC -- 3.2 Forward and Backward Invariants -- 3.3 Terminating Non-termination -- 3.4 Propagating Refinements -- 4 Upper Bound Computation -- 4.1 Cost Structures -- 4.2 Example of Upper Bound Computation -- 4.3 Cost Structure of an Equation Application -- 4.4 Cost Structure of a Phase -- 4.5 Cost Structure of a Chain -- 5 Solving Cost Structures -- 6 Related Work and Experiments -- References -- Simple and Efficient Algorithms for Octagons -- 1 Introduction -- 2 Primer on the Octagon Domain -- 2.1 The Domain and Its Representation -- 2.2 Closure Algorithms on DBMs -- 2.3 Integer Closure.
2.4 Incremental Closure -- 3 Improved Incremental Strong Closure Algorithms -- 4 Simpler Proofs of Strong and Integer Closure -- 4.1 Integer Closure -- 5 Experiments -- 6 Discussion -- 7 Related Work -- 8 Conclusions -- References -- Compositional Entailment Checking for a Fragment of Separation Logic -- 1 Introduction -- 2 Separation Logic Fragment -- 3 Compositional Entailment Checking -- 3.1 Overview of the Reduction Procedure -- 3.2 Normalization -- 3.3 Selection of Spatial Atoms -- 3.4 Soundness and Completeness -- 3.5 Checking Entailments between a Formula and an Atom -- 4 Representing SL Graphs as Trees -- 5 Tree Automata Recognizing Tree Encodings of SL Graphs -- 6 Completeness and Complexity -- 7 Extensions -- 8 Implementation and Experimental Results -- 9 Related Work -- 10 Conclusion -- References -- Automatic Constrained Rewriting Induction towards Verifying Procedural Programs -- 1 Introduction -- 2 Preliminaries -- 2.1 Rewriting Constrained Terms -- 3 Transforming

Imperative Programs into LCTRSs -- 4 Rewriting Induction for LCTRSs -- 4.1 Restrictions -- 4.2 Rewriting Induction -- 4.3 Some Illustrative Examples -- 5 Lemma Generalization by Dropping Initializations -- 6 Implementation -- 7 Related Work -- 8 Conclusions -- References -- A ZDD-Based Efficient Higher-Order Model Checking Algorithm -- 1 Introduction -- 2 Preliminaries -- 2.1 Higher-Order Recursion Schemes and Co-trivial ATA Model Checking -- 2.2 Broadbent and Kobayashi's Algorithm -- 3 A ZDD-Based Algorithm -- 3.1 ZDD Types -- 3.2 Saturation Algorithm Using ZDD Types -- 3.3 Approximation of Control-Flow information -- 3.4 Fixed-Parameter Linear Time Algorithm -- 4 Experiments -- 4.1 Data Sets and Evaluation Environment -- 4.2 Experimental Results -- 5 Related Work -- 6 Conclusion -- References -- Inferring Grammatical Summaries of String Values -- 1 Introduction.
2 Preliminaries.

Sommario/riassunto

This book constitutes the refereed proceedings of the 12th Asian Symposium on Programming Languages and Systems, APLAS 2014, held in Singapore, Singapore in November 2014. The 20 regular papers presented together with the abstracts of 3 invited talks were carefully reviewed and selected from 57 submissions. The papers cover a variety of foundational and practical issues in programming languages and systems - ranging from foundational to practical issues. The papers focus on topics such as semantics, logics, foundational theory; design of languages, type systems and foundational calculi; domain-specific languages; compilers, interpreters, abstract machines; program derivation, synthesis and transformation; program analysis, verification, model-checking; logic, constraint, probabilistic and quantum programming; software security; concurrency and parallelism; as well as tools and environments for programming and implementation.