1. Record Nr. UNISA996490353403316

Titolo	Mathematics of program construction : 14th International Conference, MPC 2022, Tbilisi, Georgia, September 26-28, 2022, proceedings / / edited by Ekaterina Komendantskaya
Pubbl/distr/stampa	Cham, Switzerland : , : Springer, , [2022] ©2022
ISBN	3-031-16912-3
Descrizione fisica	1 online resource (281 pages)
Collana	Lecture Notes in Computer Science ; ; v.13544
Disciplina	371.39445
Soggetti	Computer programming Computer science - Mathematics
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Includes index.
Nota di contenuto	Intro -- Preface -- Organization -- Abstracts of Invited Talks -- Picking Your Way Through Pascal's Triangle -- The Semifree Monad -- Lens Theoretic Foundations for Learning: From Semantics to Verification -- Contents -- Breadth-First Traversal via Staging -- 1 Introduction -- 2 Applicative Functors -- 2.1 Applicative Traversal -- 2.2 Trees -- 3 Shape, Contents, Relabelling -- 4 Fusing Traversals via Staged Computation -- 4.1 The Repmin Problem -- 4.2 Fusing Traversals -- 4.3 Day Convolution -- 4.4 Repmin in Two Phases -- 5 Multiple Phases -- 5.1 Free Applicatives -- 5.2 Two Phases, More or Less -- 5.3 The `Sort-Tree' Problem -- 6 Breadth-first Traversal in Stages -- 7 Discussion -- A Fusion of traversals -- A.1 Length-indexed vectors -- A.2 Size-indexed trees -- A.3 Make functions -- A.4 Representation Theorem -- A.5 Commutativity -- References -- Subtyping Without Reduction -- 1 Introduction -- 2 Example: Even Numbers -- 3 Path Types -- 4 Higher-Inductive Evenness -- 5 Higher-Inductive Recursive Even Numbers -- 6 Reflection -- 7 Example: Ordered Finite Sets -- 8 IF Formalisation -- 8.1 Indexed Containers -- 8.2 Propositional Monads -- 8.3 Free Propositional Monad -- 8.4 Example -- 9 IR Formalisation -- 9.1 Fibers -- 9.2 Free Subtype Extension -- 9.3 Decode Is an Embedding -- 9.4 Equivalence Between IF and IR Approaches -- 9.5 Example -- 10 Generalising Our Technique

-- 11 Related Approaches -- 12 Conclusion -- References -- Calculating Datastructures -- 1 Introduction -- 2 One-sided Flexible Arrays -- 3 Peano Numerals -- 3.1 Number Type -- 3.2 Index Type -- 4 Functions as Datastructures -- 5 Lists Also Known as Vectors -- 6 Leibniz Numerals -- 6.1 Number Type -- 6.2 Index Type -- 7 Functions as Datastructures, Revisited -- 8 One-two Trees -- 9 Braun Trees -- 9.1 Index Type, Revisited -- 9.2 Trie Type, Revisited -- 10 Random-Access Lists.
11 Related Work -- 12 Conclusion -- A Deriving Operations -- References -- Flexibly Graded Monads and Graded Algebras -- 1 Introduction -- 2 Graded Monads -- 2.1 Examples -- 3 Locally Graded Categories -- 4 Flexibly and Rigidly Graded Monads -- 4.1 Eilenberg-Moore and Kleisli -- 4.2 E-actegories -- 5 Rigidly Graded Monads from Flexibly Graded Monads -- 6 Flexibly Graded Monads from Rigidly Graded Monads -- 6.1 Constructing Extensions -- 7 Related Work -- 8 Conclusions -- References -- Folding over Neural Networks -- 1 Introduction -- 2 Background -- 3 Fully Connected Networks -- 3.1 Forward Propagation as a Catamorphism -- 3.2 Back Propagation as an Anamorphism -- 3.3 Training Neural Networks with Metamorphisms -- 4 Neural Networks à la Carte -- 4.1 Free Monads and Coproducts -- 4.2 Training Neural Networks with Free Monads -- 5 Training with Just Folds -- 5.1 Back Propagation as a Fold -- 5.2 Training as a Single Fold -- 5.3 Example: Training a Fully Connected Network -- 6 Summary -- 6.1 Related Work -- References -- Towards a Practical Library for Monadic Equational Reasoning in Coq -- 1 Introduction -- 2 An Extensible Implementation of Monad Interfaces -- 2.1 Hierarchy-Builder in a nutshell -- 2.2 Functors and Natural Transformations -- 2.3 Formalization of Monads -- 2.4 Extending the Hierarchy with New Monad Interfaces -- 2.5 Monad Transformers Using Hierarchy-Builder -- 3 Difficulties with the Termination of Monadic Functions -- 3.1 Background: Standard Coq Tooling to Prove Termination -- 3.2 Limitations of Coq Standard Tooling to Prove Termination -- 4 Add Dependent Types to Return Types for Formal Proofs -- 4.1 Add Dependent Types to Called Functions to Prove Termination -- 4.2 Add Dependent Types with a Dependently-Typed Assertion -- 5 A Complete Formalization of Quicksort Derivation -- 5.1 Formal Properties of Nondeterministic Permutations.
5.2 Program Refinement -- 5.3 A Complete Formalization of Functional Quicksort -- 5.4 A Complete Formalization of In-Place Quicksort -- 6 Related Work -- 7 Conclusion -- References -- Semantic Preservation for a Type Directed Translation Scheme of Featherweight Go -- 1 Introduction -- 2 Overview -- 3 Featherweight Go -- 4 Type Directed Translation -- 4.1 Target Language -- 4.2 Translation -- 5 Semantic Preservation -- 6 Related Work -- 7 Conclusion -- References -- Streams of Approximations, Equivalence of Recursive Effectful Programs -- 1 Introduction -- 2 Recursion Streams -- 2.1 Approximations -- 2.2 Relating Streams -- 3 Higher-Order Programs -- 3.1 Algebraic Effect Operations -- 3.2 Call-by-Push-Value -- 4 Effectful Behaviour -- 4.1 Improvement Orders -- 5 Full Program Denotations and Equivalence -- 5.1 The Subtlety of Approximation -- 5.2 The Substitution Lemma -- 5.3 Operational Semantics and Soundness -- 6 Discussion and Conclusion -- References -- Fantastic Morphisms and Wherepg to Find Them -- 1 Introduction -- 1.1 Diversification -- 1.2 Unification -- 1.3 Overview -- 2 Datatypes and Fixed Points -- 3 Fundamental Recursion Schemes -- 3.1 Catamorphisms -- 3.2 Anamorphisms -- 3.3 Hylomorphisms -- 4 Accumulations -- 5 Mutual Recursion -- 5.1 Mutumorphisms -- 5.2 Dual of Mutumorphisms -- 6 Primitive (Co)Recursion -- 6.1

Paramorphisms -- 6.2 Apomorphisms -- 6.3 Zygomorphisms -- 7 Course-of-Value (Co)Recursion -- 7.1 Histomorphisms -- 7.2 Dynamorphisms -- 7.3 Futumorphisms -- 8 Monadic Structural Recursion -- 8.1 Monadic Catamorphism -- 8.2 More Monadic Recursion Schemes -- 9 Structural Recursion on gadts -- 10 Equational Reasoning with Recursion Schemes -- 11 Closing Remarks and Further Reading -- References -- Author Index.