1.	Record Nr.	UNINA9910861096203321
	Autore	Gibbons Jeremy
	Titolo	Functional and Logic Programming : 17th International Symposium, FLOPS 2024, Kumamoto, Japan, May 15–17, 2024, Proceedings / / edited by Jeremy Gibbons, Dale Miller
	Pubbl/distr/stampa	Singapore : , : Springer Nature Singapore : , : Imprint : Springer, , 2024
	ISBN	981-9723-00-0
	Edizione	[1st ed. 2024.]
	Descrizione fisica	1 online resource (336 pages)
	Collana	Lecture Notes in Computer Science, , 1611-3349 ; ; 14659
	Altri autori (Persone)	MillerDale
	Disciplina	005.1
	Soggetti	Software engineering Artificial intelligence Programming languages (Electronic computers) Computer programming Computer science Computer systems Software Engineering Artificial Intelligence Programming Language Programming Techniques Computer Science Logic and Foundations of Programming Computer System Implementation
	Lingua di pubblicazione	Inglese
	Formato	Materiale a stampa
	Livello bibliografico	Monografia
	Nota di contenuto	Intro -- Preface -- Organization -- Abstracts of Invited Talks -- Verse: A New Functional Logic Language -- Continuations from Three Angles -- Verification of Refactoring in Answer Set Programming -- Contents -- Extended Abstract -- Algebraic Connection Between Logic Programming and Machine Learning (Extended Abstract) -- 1 Introduction -- 2 Linear Algebraic Approaches to Logic Programming -- 3 Differentiable Approaches to Logic Programming -- References -- Rewriting -- ACGtk: A Toolkit for Developing and Running Abstract Categorial Grammars -- 1 Introduction -- 2 Abstract Categorial Grammars -- 3 Properties -- 3.1 Expressive Power -- 3.2 ACG

Composition -- 3.3 Parsing with Abstract Categorial Grammars and Datalog Reduction -- 4 ACGtk -- 5 Notable Implementation Features -- 5.1 Datalog Evaluation, Tabular Parsing and Shared Forest -- 5.2 Magic Set Rewriting -- 6 Related Work -- 7 Conclusion -- A Typing Rules -- References -- Term Evaluation Systems with Refinements: First-Order, Second-Order, and Contextual Improvement -- 1 Introduction -- 1.1 Examples of TES and TERS -- 2 Preliminaries -- 3 First-Order Term Evaluation and Refinement Systems -- 4 Critical Pair Analysis for Local Coherence -- 5 Second-Order Term Evaluation and Refinement Systems -- 6 Second-Order Critical Pair Analysis for Local Coherence -- 7 Related Work -- 8 Conclusion and Future Work -- A Omitted Proofs -- A.1 Proofs for Sect. 3 and Sect. 4 -- A.2 The TERS Nats -- A.3 On Linearity Conditions -- A.4 Proofs for Sect. 5 and Sect. 6 -- A.5 The TERS CBV"115 and Hndl -- B Critical Pair Analysis of Hndl by Our Prototype Analyzer -- B.1 Definition of TERS Hndl -- B.2 Local Coherence Check -- References -- A Complete Dependency Pair Framework for Almost-Sure Innermost Termination of Probabilistic Term Rewriting -- 1 Introduction -- 2 The DP Framework.
3 Probabilistic Annotated Dependency Pairs -- 4 The ADP Framework -- 5 Transforming ADPs -- 6 Conclusion and Evaluation -- References -- Algebra -- Tabulation with Zippers -- 1 Introduction -- 2 Zipper Tabulation of Functions on Trees -- 2.1 Efficient Zipper-Based Attribute Grammars -- 3 Zip with Zipper -- 3.1 zipWithZipper -- 3.2 Tying the Knot -- 3.3 A Generic Zipper Instance of zipWithZipper -- 3.4 A Generic Table -- 4 The Expression Language Revisited -- 5 Performance -- 6 Related Work -- 7 Conclusions -- References -- Declarative Pearl: Rigged Contracts -- 1 Introduction -- 2 The Contract Library -- 2.1 Acquisition Date and Expiry Date -- 2.2 Discount Bonds -- 2.3 Composing Contracts -- 3 Instant Semiring, Just Add Expired -- 3.1 Definition of a Semiring -- 3.2 The Contract Semiring -- 3.3 Beyond Semirings: Groups -- 4 Denotational Semantics: Expiry Date and Worth -- 4.1 A Contract's Expiry Date -- 4.2 The Bottom Line -- 4.3 Comparison with the Original -- 4.4 Executable Semantics -- 5 Conclusion -- References -- Applications -- System Description: DeepLLM, Casting Dialog Threads into Logic Programs -- 1 Introduction -- 2 System Architecture -- 3 Interactors -- 4 Recursors -- 4.1 Synthesizing the Logic Program on Recursive Descent -- 4.2 The Unfolder -- 4.3 The AndOrExplorer -- 4.4 The SVO Relation Extractor -- 4.5 The Logic Programming Connection -- 5 Refiners -- 5.1 The Embeddings Vector Store -- 5.2 Filtering with Semantic Distance to Ground-Truth Facts -- 5.3 Refining Decisions with LLM-Based Oracles -- 5.4 Toward Trustable Generative AI -- 6 The Model Builder: A Propositional Horn Clause Satisfiability Solver -- 7 Applications -- 8 Discussion on Limitations and Variations on the Theme -- 8.1 Limitations -- 8.2 Future Work Directions -- 9 Related Work -- 10 Conclusion -- References.
A Constraint-Based Mathematical Modeling Library in Prolog with Answer Constraint Semantics -- 1 Introduction -- 2 Motivating Example -- 2.1 List Recursion Versus Array Iteration in the N-Queens Problem -- 2.2 Breaking Symmetries in the N-Queens Problem -- 2.3 Performance Figures -- 3 Comparison to MiniZinc Modeling Language -- 3.1 Types -- 3.2 Programming Search -- 3.3 Answer Constraint Semantics in Fourier's Example -- 3.4 Model Debuging and Visualization in a Unique IDE -- 4 Modeling Library -- 4.1 Subscripted Variables with Library arrays.pl -- 4.2 Bounded Quantification Meta-predicates in quantifiers.pl -- 4.3 Interface to Constraint Solvers with Library clp.pl -- 4.4 Proposal for a Second-Level of Normalization of ISO-Prolog -- 5 Conclusion -- References -- Grants4Companies:

Applying Declarative Methods for Recommending and Reasoning About Business Grants in the Austrian Public Administration (System Description) -- 1 Introduction -- 2 Development History -- 3 Grants4Companies Overview -- 4 Representation and Evaluation of the Grant Conditions -- 4.1 Representation of the Grants -- 4.2 Evaluation of the Grants -- 5 PoC: Extensions and Interfaces -- 5.1 Symbolic Reasoning over Grants -- 5.2 Interface Between Lisp and Prolog -- 6 Conclusion and Outlook -- References -- Program Analysis -- Inferring Non-failure Conditions for Declarative Programs -- 1 Introduction -- 2 Functional Logic Programming and Curry -- 3 Call Types and Abstract Types -- 4 In/Out Types -- 5 Inference and Checking of Call Types -- 5.1 Initial Call Types -- 5.2 Call Type Checking -- 5.3 Iterated Call Type Checking -- 5.4 Extensions -- 6 Evaluation -- 7 Related Work -- 8 Conclusions -- References -- Being Lazy When It Counts -- 1 Introduction -- 2 Presentation of Constant-Time Reference Counting -- 2.1 Eager and Lazy Reference Counting -- 2.2 Allocation Size.
2.3 Implementation in Koka -- 2.4 Soundness and Garbage-Free Guarantee -- 2.5 Eagerly-Deallocating-Allocation Effect -- 3 Preliminary Experiments -- 3.1 Experimental Setup -- 3.2 Performance and Memory Usage -- 3.3 Latency Measurements -- 4 Related Work and Conclusion -- 4.1 Related Work -- 4.2 Conclusion -- A Formalization -- A.1 Syntax -- A.2 Reference Koka Semantics -- A.3 New CTRC Semantics -- A.4 Metatheory -- B CTRC Allocator Source Code -- References -- Metaprogramming -- MetaOCaml: Ten Years Later -- 1 Introduction -- 2 Introduction to Staging and MetaOCaml -- 3 Implementing MetaOCaml -- 3.1 Type-Checking Staged Programs -- 3.2 Optimized Translation of Brackets and Escapes -- 4 Let-Insertion -- 5 Cross-Stage Persistence -- 6 Related Work -- 7 Conclusions -- References -- An ML-Style Module System for Cross-Stage Type Abstraction in Multi-stage Programming -- 1 Introduction -- 1.1 Multi-stage Programming -- 1.2 Point at Issue: Modularity -- 2 Motivating Examples -- 3 Related Work and Our Approach -- 4 Source Language -- 5 Target Language and Its Type Safety -- 6 Elaboration and Its Soundness -- 7 Extension with Cross-Stage Persistence -- 8 Future Work and Provisional Implementation -- 9 Conclusion -- A An Example Elaboration Involving CSP -- B Complete Definitions -- C Proofs for Target Type Safety -- D Proofs for Soundness of Elaboration -- References -- Rhyme: A Data-Centric Multi-paradigm Query Language Based on Functional Logic Metaprogramming -- 1 Introduction -- 2 Rhyme Front End -- 2.1 JSON API -- 2.2 Pipe API -- 2.3 Visualization API -- 3 Rhyme Backend -- 3.1 AST -- 3.2 Structure of the Rhyme IR -- 3.3 Code Generation -- 4 Related Work -- 5 Conclusion -- References -- Proofs -- Language-parameterized Proofs for Functional Languages with Subtyping -- 1 Introduction -- 2 Operational Semantics and Lang-n-Prove (Review).
3 Declarative Subtyping with Lang-n-Prove -- 3.1 Inversion Subtyping Lemmas -- 3.2 Canonical Form Lemmas -- 3.3 Progress Theorem -- 3.4 Inversion Typing Lemmas -- 3.5 Type Preservation Theorem -- 4 Algorithmic Subtyping -- 4.1 Subtyping and Typing Soundness -- 4.2 Subtyping Completeness -- 4.3 Typing Completeness -- 5 Evaluation and Limitations of Our Work -- 6 Related Work -- 7 Conclusion -- A Examples of Inversion Typing Lemmas -- B Examples of Generated Proofs -- C Inversion Typing Lemmas for Errors -- D Examples of Algorithmic Typing Rules -- References -- System Description: A Theorem-Prover for Subregular Systems: The Language Toolkit and Its Interpreter, Plebby -- 1 Introduction -- 2 Generalized Regular Expressions -- 2.1 Factors and Symbol Sets -- 2.2 Booleans, Concatenation, and Iteration -- 2.3 Subsequences and Shuffle Ideals --

2.4 Tiers and Neutral Letters -- 2.5 Brzozowski Derivatives and Quotients -- 2.6 Summary -- 3 Constraint Analysis -- 4 Algebra and Complexity Analysis -- 4.1 Some Varieties with Shortcuts -- 4.2 Some Other Well-Studied Classes -- 4.3 Summary -- 5 Conclusion -- References -- Author Index.

Sommario/riassunto	This book constitutes the proceedings of the 17th International Symposium on Functional and Logic Programming, FLOPS 2024, held in Kumamoto, Japan, in May 2024. The 15 papers presented in this volume were carefully reviewed and selected from 28 submissions. The scope includes all aspects of the design, semantics, theory, applications, implementations, and teaching of declarative programming. FLOPS speci cally aims to promote cross-fertilization between theory and practice and among di erent styles of declarative programming.