1. Record Nr. UNINA9910822121603321

| | |
|---|---|
| Autore | Harper Robert <1957-> |
| Titolo | Practical foundations for programming languages / / Robert Harper [[electronic resource]] |
| Pubbl/distr/stampa | Cambridge : , : Cambridge University Press, , 2013 |
| ISBN | 1-107-23730-0<br>1-107-25486-8<br>1-107-30175-0<br>1-107-30684-1<br>1-107-30904-2<br>1-107-31459-3<br>1-139-34213-4 |
| Descrizione fisica | 1 online resource (xviii, 471 pages) : digital, PDF file(s) |
| Classificazione | COM051010 |
| Disciplina | 005.13 |
| Soggetti | Programming languages (Electronic computers) |
| Lingua di pubblicazione | Inglese |
| Formato | Materiale a stampa |
| Livello bibliografico | Monografia |
| Note generali | Title from publisher's bibliographic system (viewed on 05 Oct 2015). |
| Nota di bibliografia | Includes bibliographical references and index. |
| Nota di contenuto | Machine generated contents note: Part I. Judgments and Rules: 1. Inductive definitions; 2. Hypothetical judgments; 3. Syntactic objects; 4. Generic judgments; Part II. Levels of Syntax: 5. Concrete syntax; 6. Abstract syntax; Part III. Statics and Dynamics: 7. Statics; 8. Dynamics; 9. Type safety; 10. Evaluation dynamics; Part IV. Function Types: 11. Function definitions and values; 12. Godel's system T; 13. Plotkin's PCF; Part V. Finite Data Types: 14. Product types; 15. Sum patterns; 16. Pattern matching; 17. Generic programming; Part VI. Infinite Data Types: 18. Inductive and co-inductive types; 19. Recursive types; Part VII. Dynamic Types: 20. The untyped 1-calculus; 21. Dynamic typing; 22. Hybrid typing; Part VIII. Variable Types: 23. Girard's system F; 24. Abstract types; 25. Constructors and kinds; 26. Indexed families of types; Part IX. Subtyping: 27. Subtyping; 28. Singleton and dependent kinds; Part X. Classes and Methods: 29. Dynamic dispatch; 30. Inheritance; Part XI. Control Effects: 31. Control stacks; 32. Exceptions; 33. Continuations; Part XII. Types and Propositions: 34. Constructive logic; 35. Classical logic; Part XIII. Symbols: 36. Symbols; 37. Fluid |

binding; 38. Dynamic classification; Part XIV. Storage Effects: 39. Modernized algol; 40. Mutable data structures; Part XV. Laziness: 41. Lazy evaluation; 42. Polarization; Part XVI. Parallelism: 43. Nested parallelism; 44. Futures and speculation; Part XVII. Concurrency: 45. Process calculus; 46. Current algol; 47. Distributed algol; Part XVIII. Modularity: 48. Separate compilation and linking; 49. Basic modules; 50. Parameterized modules; Part XIX. Equivalence: 51. Equational reasoning for T; 52. Equational reasoning for PCF; 53. Parametricity.

Sommario/riassunto

Types are the central organizing principle of the theory of programming languages. In this innovative book, Professor Robert Harper offers a fresh perspective on the fundamentals of these languages through the use of type theory. Whereas most textbooks on the subject emphasize taxonomy, Harper instead emphasizes genetics, examining the building blocks from which all programming languages are constructed. Language features are manifestations of type structure. The syntax of a language is governed by the constructs that define its types, and its semantics is determined by the interactions among those constructs. The soundness of a language design - the absence of ill-defined programs - follows naturally. Professor Harper's presentation is simultaneously rigorous and intuitive, relying on elementary mathematics. The framework he outlines scales easily to a rich variety of language concepts and is directly applicable to their implementation. The result is a lucid introduction to programming theory that is both accessible and practical.