A reasoned booklist for computer science amateurs and beginners.

Total: 106 books.

Ordering of class: EX -> A -> B -> C -> D -> E, “+” and “++” means probably useful in some specific areas.

Artificial Inteilligence [11]

Artificial Intelligence - A Modern Approach 3rd Edition , Class: A , Beginner-friendly

- Part I: Artificial Intelligence - Sets the stage for the following sections by viewing AI systems as intelligent agents that can decide what actions to take and when to take them.
- Part II: Problem-solving - Focuses on methods for deciding what action to take when needing to think several steps ahead such as playing a game of chess.
- Part III: Knowledge, reasoning, and planning - Discusses ways to represent knowledge about the intelligent agents' environment and how to reason logically with that knowledge.
- Part IV: Uncertain knowledge and reasoning - This section is analogous to Parts III, but deals with reasoning and decision-making in the presence of uncertainty in the environment.
- Part V: Learning - Describes ways for generating knowledge required by the decision-making components and introduces a new component: the artificial neural network
- Part VI: Communicating, perceiving, and acting - Concentrates on ways an intelligent agent can perceive its environment whether by touch or vision.
- Part VII: Conclusions - Considers the past and future of AI by discussing what AI really is and why it has succeeded to some degree. Also, the views of those philosophers who believe that AI can never succeed are given discussion.

Artificial Intelligence - Neural Networks Algorithms, Applications, And Programming Techniques , Class: C

- Adaline and Madaline
- Backpropagation
- The BAM and the Hopfield Memory
- Simulated Annealing: Networks discussed include the Boltzmann completion and input-output networks
- The Counterpropagation Network
- Self-Organizing Maps: includes the Kohonen topology-preserving map and the feature-map classifier
- Adaptive Resonance Theory: Networks discussed include both ART1 and ART2
- Spatiotemporal Pattern Classification: discusses Hecht-Nielsen's spatiotemporal network
- The Neocognitron

Foundations of Data Science , Class: A , Beginner-friendly

- High-Dimensional Space, Best-Fit Subspaces and Singular Value Decomposition (SVD), Random Graphs, Random Walks and Markov Chains, Learning and VC-dimension, Algorithms for Massive Data Problems, Clustering, Topic Models, Hidden Markov Process, Graphical Models, and Belief Propagation, Rankings, Hare System for Voting, Compressed Sensing and Space Vectors.

Intelligent Systems for Engineers and Scientists 2rd Edition - Adrian A. Hopgood , Class: C

Machine Learning A Probabilistic Perspective , Class: B+

Paradigms of Artificial Intelligence Programming - Case Studies in Common LISP , Class: B+ , For Beginers

- Introduction to Common Lisp, The General Problem Solver, ELIZA: Dialog with a Machine, Building Software Tools, STUDENT: Solving Algebra Word Problems, Symbolic Mathematics: A Simplification Program, Efficiency Issues, Low Level Efficiency Issues, Logic Programming, Compiling Logic Programs , Object-Oriented Programming, Knowledge Representation and Reasoning, Advanced AI Programs, Symbolic Mathematics with Canonical Form, Expert Systems, Line-Diagram Labeling by Constraint Satisfaction, Search and the Game of Othello, Introduction to Natural Language, Unification Grammars, A Grammar of English

Pattern Recognition and Machine Learning , Class: A

Prolog and Natural Language Analysis , Class: B

Reinforcement Learning, An Introduction , Class: B+ , Beginner-friendly

The Elemtes of Statistical Learning , Class: B+

The Handbook of Applied Expert Systems - Jay Liebowitz , Class: C

Automated Theorem Proof [9]

Automated Theorem Proving , Class: D

Certified Programming with Dependent Types , Class: A

Interactive Theorem Proving and Program Development Coq'Art: The Calculus of Inductive Constructions , Class: A++ , Beginner-friendly

- Aim to give a practical understanding of the Coq system and its underlying theory, the Calculus of Inductive Constructions.

Little Engines of Proof , Class: D

Mathematical Components , Class: A

Proof Trick Small Inversions , Class: D

Software Foundations , Class: EX , Beginner-friendly

- Volume 1 Logical Foundations: serves as the entry-point to the series. It covers functional programming, basic concepts of logic, computer-assisted theorem proving,and Coq.
- Volume 2 Programming Language Foundations: surveys the theory of programming languages, including operational semantics, Hoare logic, and static type systems.
- Volume 3 Verified Functional Algorithms: shows how a variety of fundamental data structures can be mechanically verified.
- Volume 4 QuickChick: Property-Based Testing in Coq introduces tools and techniques for combining randomized property-based testing with formal specification and proof in the Coq ecosystem.

Structural Proof Theory , Class: B+

The Little Prover , Class: E

- Write a MiniKaren interpreter with Scheme.

Mathematics [17]

A Course In Mathematical Logic , Class: A+ , Beginner-friendly

- Chapters 1-3. These together constitute an elementary introduction to mathematical logic up to the Godel-Henkin completeness theorem.
- Chapter 4, Boolean Algebra. Chapter 5, model theory.
- Chapter 6 and 8 self-contained course on recursion theory.
- Chapter 7, an account of the limitative results about formal mathematical systems.
- Chapter 9, an introduction to first-order intuitionistic logic.
- Chapter 10, an axiomatic investigation of Zermelo-Fraenkel set theory.
- Chapter 11, an introduction to nonstandard analysis, which is an important method of applying model theory to mathematics.

A Mathematical Introduction to Logic, 2nd Edition , Class: B , Beginner-friendly

- Textbook for learning Mathematical Logic including propositional logic, first-order logic, Entscheidungsproblem and second-order logic.

Boolean Functions Theory, Algorithms, and Applications , Class: C+

- Fundamental concepts and applications, Boolean equations, Prime implicants and minimal DNFs, Duality theory.
- Special Classes: Quadratic functions, Horn functions, Orthogonal forms and shellability, Regular functions, Threshold functions, Read-once functions, Characterizations of special classes by functional equations.
- Generalizations: Partially defined Boolean functions, Pseudo-Boolean functions

Category Theory , Class: B++ , Beginner-friendly

Categories for the Working Mathematician, Saunders Mac Lane , Class: A++

Conceptual Mathematics, A First Introduction To Categories 2nd Ed , Class: B+

Discrete Mathematics and Its Applications , Class: B+ , Beginner-friendly

First-order logic , Class: E , Beginner-friendly

General Topology, John L. Kelley , Class: C , Beginner-friendly

Introduction to Axiomatic set theory, Takeuti G. Zaring W. , Class: B , Beginner-friendly

Introduction to Lattices and Order , Class: E , Beginner-friendly

Introduction to Linear Algebra , Class: C+ , Beginner-friendly

Logic for Applications 2nd Ed , Class: A , Beginner-friendly

- Propositional Logic, Predicate Logic, Prolog, Modal Logic, Intuitionistic Logic, Elements of Set Theory.

Mathematical Logic, Ebbinghaus , Class: C , Beginner-friendly

- Syntax and Semantics of First-Order Languages, A Sequent Calculus, The Completeness Theorem, The Lowenheim-Skolem Therem, The Compactness Theorem, The Scope of First-Order Logic

Probability theory and examples 4th Edtion , Class: C , Beginner-friendly

Proof Theory, Gaisi Takeuti, 2nd Ed , Class: B

- In Chapter 1, I have added Rasiowa-Sikorski's Completeness Theorem for intuitionistic predicate calculus using Heyting algebras. The relationship between Kripke semantics and Heyung valued models is discussed In the author's opinion, the present form Of the Completeness theorems for intuitiomsUc predicate calculus is much weaker than the counterpart for classical logic. The author believes that the investigation Of stronger completeness theorems for intuitionistic logic is a very attractive new area Of investigation.
- In Chapter 2, Wamer's theory on the Hardy class, Kirby-Paris' work on Goodstein sequences,Ketonen-Solovay-Quinsey's result on Paris-Harrington's theorem and weak form Of Friedman's work on Kruskal's theorem are discussed as applications Of Gentzen's consistency-proof
- In Chapter 4, I have made the material more readable and added a proof theoretic form of Borel determinacy. It is an interesting open problem to prove Borel determinacy by means Of cut elimination.
- In Chapter 5, I have simplified the accessibility proof of ordinal diagrams and included Arai's improvement Of my consistency-proof and cut elimination theorem using ordinal diagrams The theory Of quasi-ordinal diagrams are developed as a generalization Of ordinal diagrams

Recursion Theory for Metamathematics, Raymond M. Smullyan , Class: B

- Recursive Enumerability and Recursivity, Undecidability and Recursive Inseparability, Indexing, Generative Sets and Creative Systems, Double Generativity and Complete Effective Inseparability, Universal and Doubly Universal Systems, Shepherdson Revisited, Recursion Theorems, Symmetric and Double Recursion Theorems, Productivity and Double Productivity, Uniform Reducibility, Pseudo-Uniform Reducibility, Some Feeble Partial Functions, Uniform Godelization.

Operating System [4]

Computer Systems A Programmers Perspective 3rd Global Edition , Class: EX , Beginner-friendly

- Chapter 1: A Tour of Computer Systems.
- Chapter 2: Representing and Manipulating Information.
- Chapter 3: Machine-Level Representation of Programs.
- Chapter 4: Processor Architecture.
- Chapter 5: Optimizing Program Performance.
- Chapter 6: The Memory Hierarchy.
- Chapter 7: Linking.
- Chapter 8: Exceptional Control Flow.
- Chapter 9: Virtual Memory.
- Chapter 10: System-Level I/O.
- Chapter 11: Network Programming.
- Chapter 12: Concurrent Programming.

Modern Operating Systems 4th Edition , Class: B++ , Beginner-friendly

Operating System Concepts 7th Edition , Class: B , Beginner-friendly

Understanding The Linux Kernel.chm , Class: E

Program Analysis and Verification [2]

Principle of Model Checking , Class: A

Principles of Program Analysis , Class: A , Beginner-friendly

- Data Flow Analysis, Contraint Based Analysis, Abstract Interpretation, Type and Effect Systems, Worklist Algorithm, Iterating in Reverse Postorder, Interating Through Strong Components.

Programming and Languages [19]

A Little Java, A Few Patterns , Class: C

An Introduction To Programming In Emacs Lisp, 2nd Edition , Class: D , Beginner-friendly

- Textbook for learning Elisp.

Automate the Boring Stuff with Python , Class: E++ , Beginner-friendly

- Python Basics, Regular Expressions, File Manipulating, Document Processing, Image Processing, Mail Processing.

Functional Grammar In Prolog , Class: E

- Gives a detailed description of a computer program called ProfGlot, written in Prolog and using the theory of Functional Grammar in the version described in Dik (1989f).

How To Design Programs , Class: B , Beginner-friendly

Logic, Programming and Prolog , Class: D , Beginner-friendly

- Textbook for learning Prolog.

Object-Oriented Programming in Common LISP - A Programmers Guide to Clos , Class: E

On Lisp , Class: C

- Advanced Common Lisp techniques: Functions， Macros, Continuations, Prolog, Object-Oriented Lisp.

Practical Common Lisp , Class: C , Beginner-friendly

- Textbook Beginner-friendly： design programs with Common Lisp.

Programming in IDRIS , A Tutorial , Class: A+ , Beginner-friendly

- A quick tutorial for Idris beginners.

Programming Elixir 1.3 , Class: B , Beginner-friendly

- Textbook for learning Elixir and some advanced topics: Working with Multiple Processes, Macros and Code Evaluation, Linking Modules: Behaviours and Use, Protocols—Polymorphic Functions.

Real World OCaml , Class: C , Beginner-friendly

- Textbook Beginner-friendly： design programs with OCaml.

Real World Haskell , Class: C

- Textbook for learning Haskell.

Rust Essentials , Class: B , Beginner-friendly

- Textbook for learning Rust: Basic data structures and types, Functions and Control Structures, Structuring Data and Matching Patterns, Generalizing Code with Higher-order Functions and Parametrization, the functional and object-oriented features of Rust, Pointers and Memory Safety, Organizing Code and Macros, Concurrency and Parallelism, Programming at the Boundaries.
- Textbook Beginner-friendly： design programs with Scheme.

The Art of Prolog 2nd Edition , Class: D

The Little Schemer 4th Edition , Class: C+ , Beginner-friendly

- Textbook Beginner-friendly： design programs with Scheme.

The Reasoned Schemer , Class: C

- An implementation for a logic programming interpreter.

The Scheme Programming Language 4th Edition , Class: E

- Scheme reference book.

The Seasoned Schemer , Class: C , Beginner-friendly

- The squel of The Little Schemer.

Theoretical Computer Science [43]

Algorithms 4th Edition , Class: A , Beginner-friendly

- Data Abstraction, Bags, Queues, Stacks, Analysis of Algorithms.
- Algorithm for Sorting, Searching, Graphs and Strings.

An Introduction to Formal Languages and Automata 3rd Edition , Class: B+

An Introduction to the pi-Calculus , Class: C , Beginner-friendly

Basic Category Theory for Computer Scientists , Class: C , Beginner-friendly

Compilers, Principles, Techniques, and Tools, 2nd Edition , Class: C

- Compiler Structure
- Lexical Analysis (Including Regular Expressions And Finite Automata)
- Syntax Analysis (Including Context-Free Grammars, Ll Parsers, Bottom-Up Parsers, And Lr Parsers)
- Syntax-Directed Translation
- Type Checking (Including Type Conversions And Polymorphism)
- Run-Time Environment (Including Parameter Passing, Symbol Tables And Register Allocation)

- Code Generation (Including Intermediate Code Generation)
- Code Optimization
- Directed Translation
- New Data Flow Analyses
- Parallel Machines
- Garbage Collection

Compiling With Continuations , Class: D+

COMPUTABILITY An introduction to recursive function theory , Class: D , Beginner-friendly

Concepts Of Programming Languages, 11th Ed , Class: EX , Beginner-friendly

- Chapter 1 begins with a rationale for studying programming languages. It then discusses the criteria used for evaluating programming languages and language constructs. The primary influences on language design, common design trade-offs, and the basic approaches to implementation are also examined.

- Chapter 2 outlines the evolution of the languages that are discussed in this book. It provides the background necessary to understanding the practical and theoretical basis for contemporary language design. It also motivates further study of language design and evaluation.

- Chapter 3 describes the primary formal method for describing the syntax of programming language—BNF. This is followed by a description of attribute grammars, which describe both the syntax and static semantics of languages. The difficult task of semantic description is then explored, including brief introductions to the three most common methods: operational, denotational, and axiomatic semantics.

- Chapter 4 introduces lexical and syntax analysis. This chapter is targeted to those Computer Science departments that no longer require a compiler design course in their curricula.

- Chapters 5 through 14 describe in detail the design issues for the primary constructs of programming languages. In each case, the design choices for several example languages are presented and evaluated. Specifically, Chapter 5 covers the many characteristics of variables, Chapter 6 covers data types, and Chapter 7 explains expressions and assignment statements. Chapter 8 describes control statements, and Chapters 9 and 10 discuss subprograms and their implementation. Chapter 11 examines data abstraction facilities. Chapter 12 provides an in-depth discussion of language features that support object-oriented programming (inheritance and dynamic method binding), Chapter 13 discusses concurrent program units, and Chapter 14 is about exception handling, along with a brief discussion of event handling.

- Chapters 15 and 16 describe two of the most important alternative programming paradigms: functional programming and logic programming. However, some of the data structures and control constructs of functional programming languages are discussed in Chapters 6 and 8. Chapter 15 presents an introduction to Scheme, including descriptions of some of its primitive functions, special forms, and functional forms, as well as some examples of simple functions written in Scheme. Brief introductions to ML, Haskell, and F# are given to illustrate some different directions in functional language design. Chapter 16 introduces logic programming and the logic programming language, Prolog.

Concepts, Techniques and Models of Computer Programming , Class: EX

- Introduction to Programming Concepts
- Declarative Computation Model
- Declarative Programming Techniques
- Declarative Concurrency
- Message-Passing Concurrency
- Explicit State
- Object-Oriented Programming
- Shared-State Concurrency
- Relational Programming
- Graphical User Interface Programming
- Distributed Programming
- Constraint Programming
- Language Semantics

Concrete Mathematics 2rd Edition , Class: A+ , Beginner-friendly

- The major topics treated in this book include sums, recurrences, elementary number theory, binomial coefficients, generating functions, discrete probability, and asymptotic methods. The emphasis is on manipulative technique rather than on existence theorems or combinatorial reasoning; the goal is for each reader to become as familiar with discrete operations (like the greatest-integer function and infinite summation) as a student of calculus is familiar with continuous operations (like the absolute-value function and infinite integration).

Concrete Semantics , Class: A

Essentials of Programming Languages , Class: B+ , Beginner-friendly

- EOPL Surveys the principles of programming languages from an operational perspective.It Starts with an interpreter in Scheme for a simple functional core language similar to the lambda calculus and then systematically adds constructs.

Foundations for Programming Languages, John.C..Mitchell , Class: EX , Beginner-friendly

- Written for advanced undergraduate and beginning graduate students, Foundations for Programming Languages uses a series of typed lambda calculi to study the axiomatic, operational, and denotational semantics of sequential programming languages. Later chapters are devoted to progressively more sophisticated type systems. Compared to other texts on the subject, Foundations for Programming Languages is distinguished primarily by its inclusion of material on universal algebra and algebraic data types, imperative languages and Floyd-Hoare logic, and advanced chapters on polymorphism and modules, subtyping and object-oriented concepts, and type inference.

Formal Reasionning About Programs , Class B+

Foundations of Computer Science, C Edition , Class: C

Hacker's Delight 2nd Edition , Class: D

Handbook of Process Algebra , Class: D

- Labelled transition systems and process graphs, Labelled transition systems, Process graphs, Embedding labelled transition systems in G, Equivalences relations and preorders on labelled transition

systems, Initial nondeterminism, Trace semantics, trace semantics, Failures semantics, Failure trace semantics, Ready trace semantics, Readiness semantics and possible-futures semantics, Simulation semantics, Ready simulation semantics, Reactive versus generative testing scenarios, 2-nested simulation semantics, Bisimulation semantics, Tree semantics, Possible worlds semantics, Deterministic and saturated processes, Complete axiomatizations, Criteria for selecting a semantics for particular applications, Distinguishing deadlock and successful termination

Homotopy Type Theory , Class: A

Introduction to Algorithms , Class: C

- Sorting, Order Statistics, Dynamic Programming, Greedy Algorithms, Amortized Analysis,
- Hash Table, Binary Search Trees, Red-Black Trees, B-Trees, Fibonacci Heaps, Van Emde Boas Trees, Data Structure For Disjoint Sets
- Graph Algorithms, Minimum Spanning Trees, Single-Source Shortest Path, All-Pairs Shortest Paths, Maximum Flow
- Nultithreaded Algorithms, Matrix Operations, Linear Programming, Number-Theoretic Algorithms, String Matching, Computational Geometry, Np-Completeness, Approximation Algorithms

Introduction To Automata Theory Languages, and Computation , Class: D , Beginner-Beginner-friendly

Introduction to Languages and The Theory of Computation 4th Edition , Class: C

Introduction to Process Algreba , Class: B++ , Beginner-friendly

- Algebra of Communicating Process, Recursion, Abstraction, Protocal Verifications.

Lambda-Calculus and Combinators, an Introduction , Class: A

λ-calculus, Combinatory logic, Representing the computable functions, The undecidability theorem 63, The formal theories λβ and CLw, Extensionality in λ-calculus, Extensionality in combinatory logic, Simple typing, Church-style, Simple typing, Curry-style in CL, Simple typing, Curry-style in λ, Generalizations of typing, Models of CL, Models of λ-calculus, Scott's D ∞ and other models.

Language Implementation Patterns , Class: B++ , Beginner-friendly

- Getting Started with Parsing: We'll start out by looking at the overall architecture of language applications and then jump into the key language recognition (parsing) patterns.
- Analyzing Languages: To analyze DSLs and programming languages, we'll use parsers to build trees that represent language constructs in memory. By walking those trees, we can track and identify the various symbols (such as variables and functions) in the input. We can also compute expression result-type information (such as int and float). The patterns in this part of the book explain how to check whether an input stream makes sense.
- Building Interpreters: This part has four different interpreter patterns. The interpreters vary in terms of implementation difficulty and run-time efficiency.
- Translating and Generating Languages: In the final part, we will learn how to translate one language to another and how to generate text using the StringTemplate template engine. In the final chapter, we'll lay out the architecture of some interesting language applications to get you started building languages on your own.

Lectures on the Curry-Howard isomorphism-Elsevier Science , Class: A , Beginner-friendly

Let Over Lambda , Class: A+

- A book about programming macros in lisp. Unlike most programming books that only give them a cursory overview, this book is a series of tutorials and examples designed to get you programming sophisticated macros as efficiently and quickly as possible. Mastering macros is the final step to graduating from an intermediate lisp programmer to a lisp professional.

LISP In Small Pieces , Class: B++

- A book full of Scheme interpreters and compilers. This book is organized into two parts. The first takes off from the implementation of a naive Lisp interpreter and progresses toward the semantics of Scheme. The second part of the book goes in the other direction. Starting from denotational semantics and searching for efficiency, we'll broach the topic of fast interpretation (by pretreating static parts), and then we'll implement that preconditioning (by precompilation) for a byte-code compiler.

Logic in Computer Science, Modelling and Reasoning about Systems , Class: EX , Beginner-friendly

- Propositional Logic, Predicate Logic, Verification By Model Checking, Program Verification, Modal Logics And Agents, Binary Decision Diagram.

Logic Programming and Program Synthesis , Class: E

Mathematics for Computer Science 2013 , Class: B++

- Proofs: What is a Proof, The Well Ordering Principle, Mathematical Data Types, Logical Formulas, Induction, Recursive Data Types, Infinite Sets
- Structures: Number Theory, Directed graphs & Partial Orders, Communication Networks, Simple Graphs, Planner Graphs,
- Counting:Sums and Asymptotics, Cardinality Rules, Generating Functions Conditional Probability, Random Variables , Deviation from the Mean, Random Walks

Parsing techniques - a practical guide , Class: B

Practical Foundations for Programming Languages 2nd Edition , Class: D

- Types are the central organizing principle of the theory of programming languages. 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.
- The purpose of this book is to explain this remark. A variety of programming language features are analyzed in the unifying framework of type theory. A language feature is defined by its statics, the rules governing the use of the feature in a program, and its dynamics, the rules defining how programs using this feature are to be executed. The concept of safety emerges as the coherence of the statics and the dynamics of a language.

Programming Languages and Lambda Calculus , Class: D

Programming Languages Application and Interpretation , Class: D

- Write interpreters with Racket.

Structure and Interpretation of Computer Programs , Class: B , Beginner-friendly

- A textbook aiming to teach the principles of computer programming, such as abstraction in programming, metalinguistic abstraction, recursion, interpreters, and modular programming.

The Art of the Metaobject Protocol , Class: E

The Pi-Calculus A Theory of Mobile Processes, Class: B+

- Syntax of the Pi-calculus, Properties of the Transition System, Behavior Equivalence, Variations of the Pi-Calculus, Typed Pi-Calculus, Reasoning about Process Using Types

Programming Languages and Lambda Calculi , Class: C

The Art of Computer Programming , Class: A++

- Chapter 1 – Basic concepts. Algorithms, Mathematical, Preliminaries, MMIX, Some Fundamental Programming Techniques.
- Chapter 2 – Information Structures. Linear Lists, Trees, Multilinked Structures, Dynamic Storage,Allocation, History and Bibliography.
- Chapter 3 – Random Numbers. Generating Uniform Random Numbers, Statistical Tests, Other Types of Random Quantities, What Is a Random Sequence?
- Chapter 4 – Arithmetic. Positional Number Systems, Floating Point Arithmetic, Multiple Precision Arithmetic, Radix Conversion, Rational Arithmetic, Polynomial Arithmetic, Manipulation of Power Series.
- Chapter 5 – Sorting. Combinatorial Properties of Permutations, Internal sorting, Optimum Sorting, External Sorting.
- Chapter 6 – Searching. Sequential Searching, Searching by Comparison of Keys, Digital Searching, Hashing, Retrieval on Secondary Keys.
- Chapter 7 – Combinatorial Searching. Boolean Basics, Binary Decision Diagrams, Generating All Possibilities, Shortest paths, Graph algorithms, Network algorithms, Independence ,theory, Discrete dynamic programming, Branch-and-bound techniques, Herculean tasks (aka NP-hard problems), Near-optimization.
- Chapter 8 – Recursion
- Chapter 9 – Lexical scanning
- Chapter 10 – Parsing techniques
- Chapter X - The Theory of Context-free Languages
- Chapter Y - Compiler Techniques

The Lambda Calculus, It's syntax and Semantics , Class: A+

- Textbook covers almost all part of Lambda Calculus.

Theory of Formal Systems , Class: A

Type Theory and Functional Programming , Class: A+

- The first three chapters survey the three fields upon which type theory depends: logic, the λ-calculus and functional programming and constructive mathematics. The surveys are short, establishing terminology, notation and a general context for the discussion; pointers to the relevant literature and in

particular to more detailed introductions are provided. In the second chapter we discuss some issues in the λ-calculus and functional programming which suggest analogous questions in type theory.

- The fourth chapter forms the focus of the book. We give the formal system for type theory, developing examples of both programs and proofs as we go along. These tend to be short, illustrating the construct just introduced – chapter 6 contains many more examples.
- The system of type theory is complex, and in chapter which follows we explore a number of different aspects of the theory. We prove certain results about it (rather than using it) including the important facts that programs are terminating and that evaluation is deterministic. Other topics examined include the variety of equality relations in the system, the addition of types (or 'universes') of types and some more technical points.
- Much of our understanding of a complex formal system must derive from out using it. Chapter six covers a variety of examples and larger case studies. From the functional programming point of view, we choose to stress the differences between the system and more traditional languages. After a lengthy discussion of recursion, we look at the impact of the quantified types, especially in the light of the universes added above. We also take the opportunity to demonstrate how programs can be extracted from constructive proofs, and one way that imperative programs can be seen as arising. We conclude with a survey of examples in the relevant literature.
- As an aside it is worth saying that for any formal system, we can really only understand its precise details after attempting to implement it. The combination of symbolic and natural language used by mathematicians is surprisingly suggestive, yet ambiguous, and it is only the discipline of having to implement a system which makes us look at some aspects of it. In the case of T T , it was only through writing an implementation in the functional programming language Miranda 1 that the author came to understand the distinctive role of assumptions in T T , for instance.
- The system is expressive, as witnessed by the previous chapter, but are programs given in their most natural or efficient form? There is aost of proposals of how to augment the system, and we look at these in chapter 7. Crucial to them is the incorporation of a class of subset types, in which the witnessing information contained in a type like (∃x : A) . B(x) is suppressed. As well as describing the subset type, we lay out the arguments for its addition to type theory, and conclude that it is not as necessary as has been thought. Other proposals include quotient (or equivalence class) types, and ways in which general recursion can be added to the system without its losing its properties like termination. A particularly elegant proposal for the addition of co-inductive types, such as infinite streams, without losing these properties, is examined.
- Chapter eight examines the foundations of the system: how it compares with other systems for constructive mathematics,how models of it are formed and used and how certain of the rules, the closure rules, may be seen as being generated from the introduction rules, which state what are the canonical members of each type. We end the book with a survey of related systems, implemented or not, and some concluding remarks.

Types and Programming Languages , Class: A++

- Part I of the book discusses untyped systems. Basic concepts of abstract syntax, inductive definitions and proofs, inference rules, and operational semantics are introduced first in the setting of a very simple language of numbers and booleans, then repeated for the untyped lambda-calculus.
- Part II covers the simply typed lambda-calculus and a variety of basic language features such as products, sums, records, variants, references, and exceptions. A preliminary chapter on typed arithmetic expressions provides a gentle introduction to the key idea of type safety. An optional chapter develops a proof of normalization for the simply typed lambda-calculus using Tait's method.

- Part III addresses the fundamental mechanism of subtyping; it includes a detailed discussion of metatheory and two extended case studies.
- Part IV covers recursive types, in both the simple iso-recursive and the trickier equirecursive formulations. The second of the two chapters in this part develops the metatheory of a system with equi-recursive types and subtyping in the mathematical framework of coinduction.
- Part V takes up polymorphism, with chapters on ML-style type reconstruction, the more powerful impredicative polymorphism of System F, existential quantification and its connections with abstract data types, and the combination of polymorphism and subtyping in systems with bounded quantification.
- Part VI deals with type operators. One chapter covers basic concepts; the next develops System F ω and its metatheory; the next combines type operators and bounded quantification to yield System F ω <: ; the final chapter is a closing case study.

Others [1]

Quantum Computation and Quantum Information , Class: D++ , Beginner-friendly

- Part I provides a broad overview of the main ideas and results of the field of quantum computation and quantum information, and develops the background material in computer science, mathematics and physics necessary to understand quantum computation and quantum information in depth. Chapter 1 is an introductory chapter which outlines the historical development and fundamental concepts of the field, highlighting some important open problems along the way. The material has been structured so as to be accessible even without a background in computer science or physics. The background material needed for a more detailed understanding is developed in Chapters 2 and 3, which treat in depth the fundamental notions of quantum mechanics and computer science, respectively. You may elect to concentrate more or less heavily on different chapters of Part I, depending upon your background, returning later as necessary to fill any gaps in your knowledge of the fundamentals of quantum mechanics and computer science.
- Part II describes quantum computation in detail. Chapter 4 describes the fundamental elements needed to perform quantum computation, and presents many elementary operations which may be used to develop more sophisticated applications of quantum computation. Chapters 5 and 6 describe the quantum Fourier transform and the quantum search algorithm, the two fundamental quantum algorithms presently known. Chapter 5 also explains how the quantum Fourier transform may be used to solve the factoring and discrete logarithm problems, and the importance of these results to cryptography. Chapter 7 describes general design principles and criteria for good physical implementations of quantum computers, using as examples several realizations which have been successfully demonstrated in the laboratory.
- Part III is about quantum information: what it is, how information is represented and communicated using quantum states, and how to describe and deal with the corruption of quantum and classical information. Chapter 8 describes the properties of quantum noise which are needed to understand real-world quantum information processing, and the quantum operations formalism, a powerful mathematical tool for understanding quantum noise. Chapter 9 describes distance measures for quantum information which allow us to make quantitatively precise what it means to say that two items of quantum information are similar. Chapter 10 explains quantum error-correcting codes, which may be used to protect quantum computations against the effect of noise. An important result in this chapter is the threshold theorem, which shows that for realistic noise models, noise is in principle not a serious impediment to quantum computation. Chapter 11 introduces the fundamental information-theoretic

concept of entropy, explaining many properties of entropy in both classical and quantum information theory. Finally, Chapter 12 discusses the information carrying properties of quantum states and quantum communication channels, detailing many of the strange and interesting properties such systems can have for the transmission of information both classical and quantum, and for the transmission of secret information.