A Self-Learning, Modern Computer Science Curriculum

QEK

March 26, 2017

Contents

1 Introduction

> "The great idea of constructive type theory is that there is no distinction between programs and proofs... Theorems are types (specifications), and proofs are programs" (Robert Harper)

This guide is based on Carnagie Mellon's new curriculum, for reasons why they rewrote it from scratch see Robert Harper's blog and his follow up post on the success of the new material. The goal of this guide is to build up to a

senior undergrad level with the necessary rigorous background to understand academic papers in journals to continue self-learning.

There is a (terrible) pdf of this guide as well.

1.1 How Long Will This Take?

Depends on daily effort. Each entry is about one semester worth of material and a typical undergrad student at any university will take 5 courses per semester with lectures 3 days a week for each course, and be assigned large projects plus multiple extra readings. Often these readings are entire texts, with the warning "exam questions from the assigned reading are fair game". Because this guide almost entirely uses functional programming you will write far less code and be able to complete a book like *Parallel and Sequential Algorithms* in one semester which would be impossible in an imperative language.

1.2 Build a Library

If you can, buy these books and start building yourself a library. Abe Books often has used copies, or global editions you can buy at significant discount. You will want them around for reference and it's good for motivation, you get books off your desk and into the display shelf and have a physical reference for your accomplishments. It also saves your eyes from the strain of screens. If you can't afford these then try libgen.io (Library Genesis) to get a pdf/epub.

1.3 How to Succeed

This anecdote by Cal Newton on how he was able to get the best grade in his Discrete Mathematics class involved him practicing proving every theorem given in the lectures and assigned reading over and over until he could comfortably pick randomly through his notes and prove something without assistance. The more deliberate practice you can do by completing the exercises and problem sets the faster you will complete this curriculum, since you will be able to skip review chapters and will spend less time working on problems as you will have gained mathematical maturity.

On Isaac Newton self-learning geometry:

> "He bought Descartes' Geometry and read it by himself .. when he was got over 2 or 3 pages he could understand no farther, than he began again and got 3 or 4 pages father till he came to

another difficult place, than he began again and advanced farther and continued so doing till he made himself master of the whole without having the least light or instruction from anybody" (King's Cam.,Keynes MS 130.10,fol. 2/v/)

2 Preliminaries

2.1 Intro to Programming with Types

If you've struggled with programming books in the past, worry not as this book/course has proven successs teaching a profoundly typed discipline to programming. The rest of the guide assumes you are at this level of HtDP experience in programming.

- (Full Course) Systematic Program Design (edx.org)
  - Based on the book How to Design Programs (HtDP)
  - Do some of the exercises in the book as well as the edx.org course exercises, the more you do the better you will understand the material
    - There is a Masters level bootcamp for HtDP as well if you can already program in some imperative language
  - Optional notes on version control (git)

2.2 Elementary Mathematics Survey

Build a better foundation of elementary mathematics.

- (Book) Elements of Mathematics: From Euclid to Gödel - John Stillwell
  - A Mathematicians survey of Algebra, Arithmetic, Combinatorics/Discrete Math, Calculus, Probability, Logic ect.
  - Casual read for anybody interested in gaining intuition into the big picture of elementary math topics
  - Explains how the introduction of the concept of infinity changes elementary math to advanced.
    - See Gilbert Strang's Highlights of Calculus for a survey of calculus

* Calculus will occasionally come up in some of the texts and lectures such as when studying probability, or often it will be used as an example for insight into a topic
* If you are totally lost start with the book Precalculus - Sheldon Axler

2.3 Intro to Mathematical Thinking

- (Book) How to Think Like A Mathematician - Kevin Houston
  - How to read theorems and definitions, how to write proofs
  - How to read a research paper
  - Optional Book of Proof - Hammack online free version

2.4 Problem Solving

"So I went to Case, and the Dean of Case says to us, it's a all men's school, "Men, look at, look to the person on your left, and the person on your right. One of you isn't going to be here next year; one of you is going to fail." So I get to Case, and again I'm studying all the time, working really hard on my classes, and so for that I had to be kind of a machine. In high school, our math program wasn't much, and I had never heard of calculus until I got to college. But the calculus book that we had was (in college) was great, and in the back of the book there were supplementary problems that weren't assigned by the teacher. So this was a famous calculus text by a man named George Thomas, and I mention it especially because it was one of the first books published by Addison-Wesley, and I loved this calculus book so much that later I chose Addison-Wesley to be the publisher of my own book. Our teacher would assign, say, the even numbered problems, or something like that (from the book). I would also do the odd numbered problems. In the back of Thomas's book he had supplementary problems, the teacher didn't assign the supplementary problems; I worked the supplementary problems. I was scared I wouldn't learn calculus, so I worked hard on it, and it turned out that of course it took me longer to solve all these problems than the kids who were only working on what was assigned, at first. But after a year, I could do all of those

problems in the same time as my classmates were doing the assigned problems, and after that *I could just coast in mathematics, because I'd learned how to solve problems*" (Don Knuth)

- (Book) The Art and Craft of Problem Solving - Paul Zeitz
  - Any edition will do
  - Written by former coach of the US Math Olympiad team
  - First 3 chapters have excellent strategies, such as finding invariants
    - Other chapters are optional but are very relevant to all the material here
- (Wikipedia) How to Solve it - George Polya
  - Summary on stragies in the book, optionally read whole book
    - There is also Descartes Discourse On the Method

3 Fundamentals I: Mathematics Core

Start here for an introduction to Functional Programming and Discrete Mathematics

3.1 Recorded Lectures

Pair these lecture subjects with *Discrete Mathematics and Functional Programming* by VanDrunen

- CMU 15-251 *Great Theoretical Idea in Computer Science* lectures and course notes
- (Optional) MIT 6.042j *Mathematics for Computer Science* lectures and course notes

3.2 Discrete Math with Standard ML

- (Book/Lectures) Discrete Mathematics and Functional Programming - Thomas VanDrunen
  - Chapter 1 (Set)
    - Lectures exist above on author's book page

- Chapter 2 (List)
  - Watch author's lectures
  - Watch 15-251 Lecture 5 (Turing's Legacy)
- Chapter 3 (Proposition)
  - Watch author's lectures
  - Watch 15-251 Lecture 1 (Intro) & Lecture 02 (Deductive Systems)
  - Watch 6.042j Lecture 1 (Introduction & Proofs)
- Chapter 4 (Proof)
  - Watch 15-251 Lecture 3 (Formalization of Proof)
  - See Robert Harper's blog post on proof by contradiction
- Chapter 5 (Relation)
  - Watch 6.042j Lecture 11: Relations, Partial Orders, and Scheduling
- Chapter 6 (Self Reference)
  - Watch 6.042j Lectures 2 & 3 (Induction and Strong Induction)
- Chapter 7 (Function)
  - Watch 15-251 Lecture 6 - Uncountability & Uncomputability
- Chapter 8 (Graph)
  - Watch 15-251 Lectures 10, 11 & 12 (Graphs)
  - Watch 6.042j Lectures 6, 8, & 10 (Graph Theory)
- Chapter 9 (Complexity Class)
  - Watch 15-251 Lectures 7,8 & 9 Introduction to Computational Complexity
  - Watch 15-251 Lecture 14 - Proof of Cook-Levin Theorem
- Chapter 12 (Automaton)
  - Watch 15-251 Lecture 4 (Finite Automata)
  - Read Lambda Calculus: The Other Turing Machine
  - Watch the rest of the 15-251 lectures

3.3 Additional Exercises to Build Mathematical Maturity

The more practice you have the better you will understand this material

- Go through the lecture notes for both 15-251/6.042j and try to prove the example propositions and theorems yourself
- Try the exercises in Concrete Mathematics for deliberate practice
- Buy a used copy (any edition) of *The Art of Computer Programming Vol 1: Fundamental Algorithms* by Donald E. Knuth and try the exercises in 1.1 and 1.2.x chapters, and 2.3.4.x *Basic Mathematical Properties of Trees*

4 Fundamentals II: Abstraction

Functional programming languages support algebraic datatypes, and besides direct applications to CS this material will train your mind in building up and reasoning about abstractions.

4.1 Abstract Linear Algebra

- (Book) Linear Algebra: An Introduction to Abstract Mathematics - Robert J. Valenza
  - Surprisingly easy proofs, focuses on the geometric intuition of linear algebra
  - These YouTube presentations here can also help with understanding the topics or Gilbert Strang's lectures
  - Do the chapter on Groups in the book *Discrete Mathematics and Functional Programming* by VanDrunen
    - There is also this Linear Algebra introduction which uses Sage Math inlined with the text to test out exercises immediately if you want more practice

4.2 Abstract Algebra

- (Full Course) Math-371 Abstract Algebra
  - Has recorded lectures which are essential as you can quickly get lost in the texts at this level of abstraction

  - Uses readings from three texts
    * There is also this Harvard course with excellent recorded lectures and uses readings from *Algebra - M. Artin* (blue book) as a great compliment to Math-371
  - Read the history of Abstract Algebra by Prof Lee Lady

4.3 Hardware Abstraction

Learn the ability to move between levels of abstractions in programming and understand the x86-64 architecture.

- (Full Course) 15-213 Intro to Computer Systems (CMU)
  - Recorded lectures here
  - Buy the CS:APP book, 3rd version from Abe Books either used or global version.
    * The labs (Data Lab, Bomb Lab ect) are on the book website. The labs come with scripts for self-checking answers
  - You could read K&R's *The C Programming Language* for a brief intro, though this course will explain C as you go anyway
  - You can also do Unix programming in Standard ML(pdf)
  - Try the embedded security CTF here

4.4 Meta-Linguistic Abstraction (Optional Elective)

- (Full Course) 6.037 Structural Interpretation of Computer Programs
  - Recorded lectures from the first edition of SICP here
  - This course condenses the book into the core ideas
    * There is a unit test library for SICP as well as an online tutorial
    * Read the paper The Art of the Propagator for adventures in symbolic programming

5 Fundamentals III: Functional Programming

5.1 Principles of Functional Programming

The 2012 version by Dan Licata has the best lecture notes.

- (Lecture Notes) 15-150 Principles of Functional Programming
  - "PSML Chapters" are assigned reading from this book Programming in Standard ML
    * If you want pair these lecture notes with the videos from Unit 1 through 4 (specific to SML) at CSE341 (UW)
    * Do these homework assignments and labs plus any exercises you find in the PSML book. Try the CSE341 homework
    * Note emacs setup also here
  - Read this post about what recursion is

5.2 Parallel and Sequential Algorithms

Design, analysis and programming of sequential and parallel algorithms and data structures in a functional pseudocode similar to Standard ML.

- (Book) Parallel and Sequential Algorithms
  - Complete, self-contained book with exercises, check 15-210 course schedule for recitation pdfs and extra material
  - Do the chapter on Lattices in the VanDrunen book, it has applications to parallel datastructures and distributed computing (You've done the entire VanDrunen book, congrats)

5.3 Programming Language Theory

Learn the fundamental principles to the design, implementation, and application of programming languages. Terence Tao article on formalism and rigor you want to be *Stage 3 - Post Rigorous Stage* where you can confidently reason with intuition because you have the necessary formal background.

- (Book) Practical Foundations for Programming Languages - Robert Harper
  - Read these reasons for studying programming languages
    * Lecture videos to accompany the book Programming Languages Background — Robert Harper and Dan Licata provide great insight
    * More info on Judgements (logic) here and a series of video lectures here by Frank Pfenning on Basic Proof Theory but all this is self contained in the PFPL book.

  - See `http://mlton.org/` if you are going to build your own DSL, it's one of the few whole-program optimizing compilers

5.4 Isolating Software Failure and Proving Safety

How to verify software, and strategies of programming that minimize catastrophe during failure.

- (Book) Verified Functional Algorithms
  - Part 3 of the Deep Specifications interactive book series
    * A good introduction to Dependent Types by Dan Licata is here
- (Lecture Notes) 15-316 Software Foundations of Security and Privacy
  - Read about tests, strategies to safe program design (in OCaml)
  - Additional readings

5.5 Designing Compilers (Optional Elective)

The labs have starter code files you don't have access to, but you can fill this in with Standard ML/OCaml yourself. The starter kit mainly exists for students who wish to do this course in an imperative language and require large amounts of starter code or else they'd never finish the course in one semester. Alternative follow the Appel book and implement the Tiger language using the lecture notes as supplemental material.

- (Lecture Notes) 15-411 Compiler Design
  - Implement C0 (for background info, see 15-122 lecture notes or the specification of C0 which is a garbage collected C-like syntax language with an interpreter shell designed for teaching imperative language contracts
    * Uses the Andrew Appel book Modern Compiler Implementation in ML

5.6 Physical Systems Software Security (Optional Elective)

- (Full Course) 15-424 Foundations of Cyber-Physical Systems

  - Course (with lecture videos) if you're interested in programming drones/space shuttles/robots/cars and other things that cannot fail from avoidable errors.
  - Req some knowledge of differential equations

6 Fundamentals IV: Algorithms & Functional Persistent Data Structures

6.1 Advanced Algorithms

Graduate level algorithms design course from Harvard with recorded lectures. Since you have (hopefully) already done the *Parallel and Sequential Algorithms* book, you satisfy the prerequisites.

- (Full Course) CS224 Advanced Algorithms
- Pair this with Purely Functional Data Structures book by Chris Okasaki (the full book, not the thesis .pdf it's missing a lot of material)
  - Splay trees/heaps/binomial heaps/hashing ect are all in the Okasaki book as well as CS224
  - Compare both to see how to make persistent functional data structures
  - Also see the external links in this Wikipedia entry on persistent data structures for other material
- Try the psets

6.2 Computational Complexity Theory (Optional Elective)

- (Book) Computational Complexity - A Modern Approach
  - Free book draft
  - Lecture notes here
    * The book *The Nature of Computation* which is the standard reference for Complexity Theory

7 You're Done the Fundamentals

If you feel at this level there are gaps in your mathematical understanding, there is the book series *Analysis I & II* by Terence Tao to fill those. You start at the very beginning, building up the naturals and reals. This book *Real Analysis for Graduate Students* by Richard Bass is a crash course in graduate mathematics that can help fill those gaps as well with Tao or Baby Rudin as a prereq. Real (and Complex) Analysis has many applications to CS Theory. There is also these scribed lecture notes that give you a diverse background in math useful for theoretical CS.

8 Graduate Research Elective: Type Theory

Read these slides from *A Theorist's ToolKit* on how to find research, how to write math in LaTeX, how to give a talk, where to ask on stackexchange ect.

8.1 Basic Proof Theory

- Start with the lectures on *Basic Proof Theory* by Frank Pfenning
  - These 15-317 Constructive Logic notes go with the seminar, there is another set here
  - Read Martin-Löf's lecture notes

8.2 Intro to Category Theory

- (Book) Category Theory - Steve Awodey
  - The most accessible introduction see notes for each chapter and YouTube lectures
  - Watch this seminar *Basic Category Theory: Semantics of Proof Theory* by Ed Morehouse
    - Also see this post here explaining Category Theory and as types in ML
  - Notes on Categorical Logic

8.3 Intermediate Category Theory and Homology

- (Book) Algebra: Chapter 0 - Paolo Aluffi

  - Designed for self-learning, this is a fully self-contained graduate introduction of Abstract Algebra that uses Category Theory in the first chapter to define the rest of the book
    * Tons of exercises, some very difficult

8.4 Type Theory Foundations

- Read Chapters 1-12, 23 and 24 from TAPL
- Watch seminar on *Type Theory Foundations* by Robert Harper
  - Pair with Programming in Martin-Löf's Type Theory and the rest of the suggested reading

8.5 Higher Dimensional Type Theory

Start with this talk *A Functional Programmer's Guide to Homotopy Type Theory* with intro to Cubical Type Theory

- (Full Course) Homotopy Type Theory
  - Self-contained grad seminar with recorded lectures
    * Assigns readings from self-contained HoTT book `https://homotopytypetheory.org/book/`
    * Try the homework, see some of the open problems in HoTT research
  - This interactive tutorial uses HoTT to teach you automated theorem proving using Lean 2 software

8.6 Further Research

- The Type Theory podcast
- Use the Sci-Hub proxy to get access to journals in Applied Logic or Functional Programming or visit a local library to ask for a printing, in a few cases the authors of a paper have a draft copy on their personal pages as well.
- Look up your closest university's event listings for the Mathematics or Computer Science departments to find seminars, these are open to the public usually

- Try an advanced course in Modal or Substructural logic or Linear type theory

9 Graduate Research Elective: Machine Learning

Read these slides from *A Theorist's ToolKit* on how to find research, how to write math in LaTeX, how to give a talk, where to ask on stackexchange ect.

9.1 Graduate Introduction to ML

- (Full Course) 10-601 Masters Introduction to ML
  - Recorded lectures and recitation videos
  - Self contained, assumes you are grad level standing so have familiarity with basic probability and algebra
  - Prepares you in the foundations to understand journals and research papers
    - You may want to also take an introduction to Statistics like Penn State's Stat-500 or look at the Harvard Graduate Statistics Course family tree

9.2 Advanced Introduction to ML

- (Full Course) 10-701 PhD Introduction to ML
  - Recorded lectures, provides a more in-depth treatment
  - This 10-715 Advanced Introduction to ML also has recorded lectures

9.3 Algorithms for Big Data

- (Full Course) CS229r Algorithms for Big Data
  - A grad level course in designing/analyzing algorithms for massive data.

9.4 Further Research

- The Journal of Machine Learning Research
- Open challenges in ML

10 Graduate Research Elective: Cryptography

Read these slides from *A Theorist's ToolKit* on how to find research, how to write math in LaTeX, how to give a talk, where to ask on stackexchange ect.

10.1 Graduate Cryptography Intro

- (Full Course) Masters Level Cryptology
  - Recorded lectures here
    - Covers some post-quantum crypto systems and ECC crypto, taught by premiere researcher Tanja Lange
    - Pair with notes from 18.783 Elliptic Curves
  - See DJ Bernsteins talks and papers from his personal site, and post-quantum crypto site
  - Read all his posts on the IETF Crypto Forum Research Group Discussion archive
  - There is also Matthew D. Green's Practical Crypto Course

10.2 Theoretical Cryptography

- (Lecture Notes) 15-859 Theoretical Cryptography
  - Free book *A Course in Cryptography* by Rafael Pass and Abhi Shelat
    - Optional book *Introduction to Modern Cryptography* by Jonathan Katz and Yehuda Lindell
    - Course uses Vol2 of *The Art of Computer Programming* by Don Knuth for sections on testing randomness
  - Read *The Arithmetic of Elliptic Curves* and it's sequel *Advanced Topics in the Arithmetic of Elliptic Curves* by Silverman

10.3 Coding Theory

- A draft book on Coding Theory with some lecture notes

10.4 Future Research

- Attend a djb conference talk
- Read a book on Random Graphs
- Try the Cryptopals challenges
- Try and get into TU/e as a direct Masters or PhD candidate to do research with djb and Tanja Lange. Yes, this is possible even if you don't have a bachelors degress