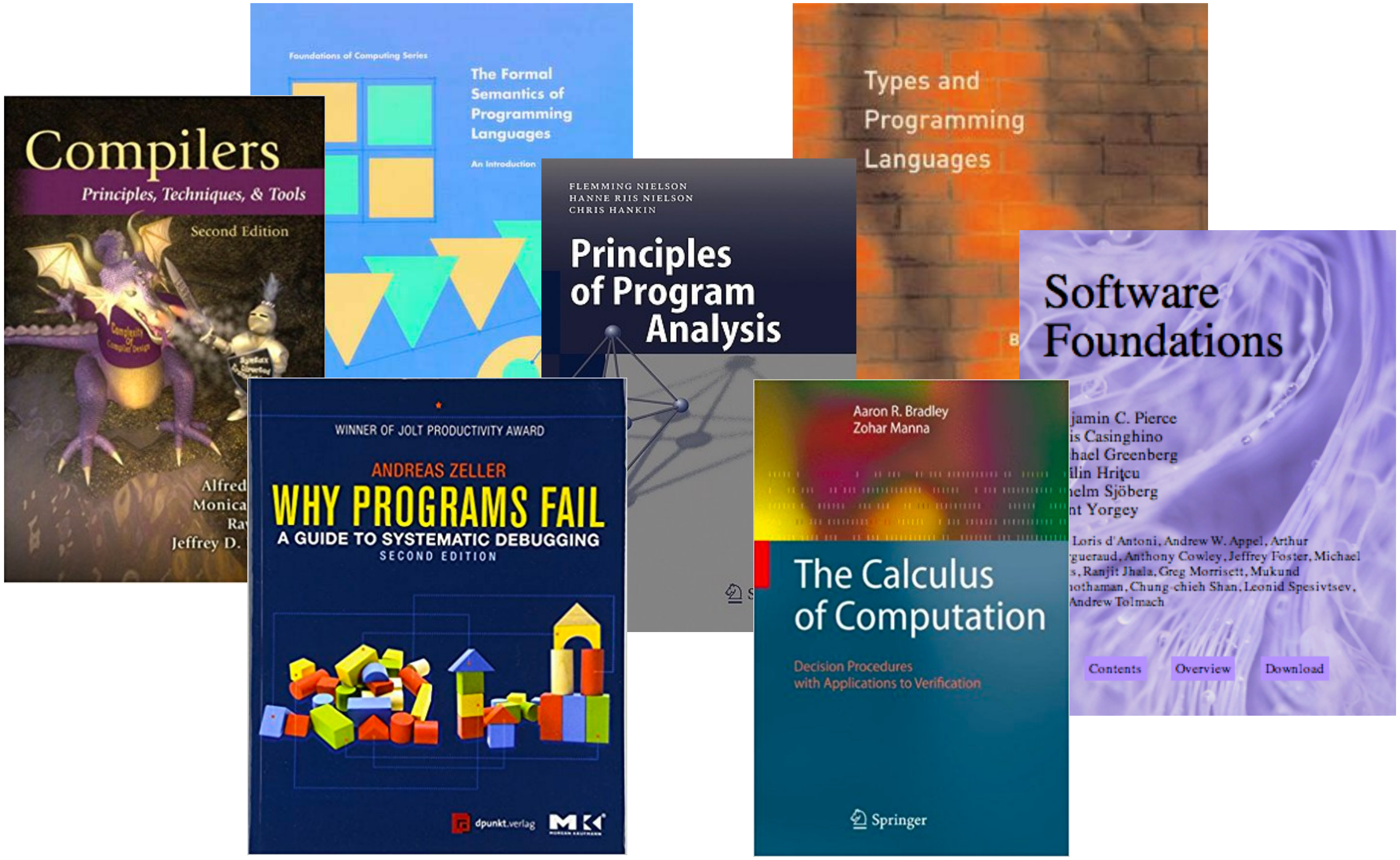
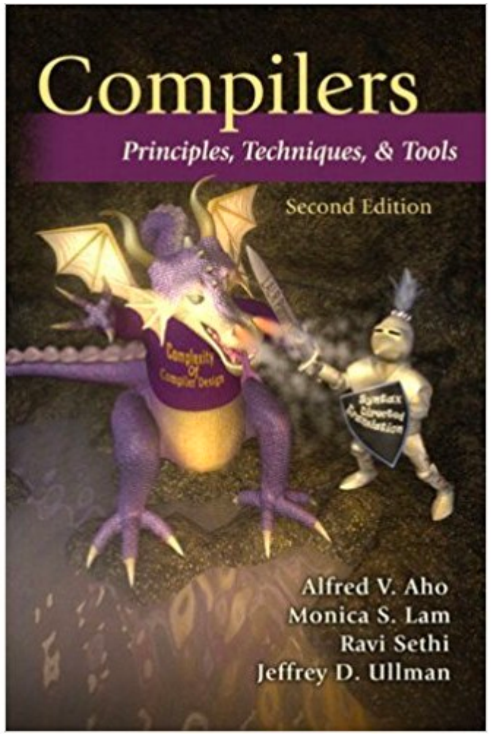
Software Analysis and Testing

Recommended Books


Compilers
Principles, Techniques, & Tools
Second Edition
Alfred
Monica
Ra
Jeffrey D.
Foundations of Computing Series
The Formal Semantics of Programming Languages
An Introduction
FLEMMING NIELSON
HANNE RIIS NIELSON
CHRIS HANKIN
Principles of Program Analysis
Types and Programming Languages
Software Foundations
jamin C. Pierce
is Casinghino
hael Greenberg
ălin Hriţcu
helm Sjöberg
nt Yorgey
Loris d'Antoni, Andrew W. Appel, Arthur
rgueraud, Anthony Cowley, Jeffrey Foster, Michael
s, Ranjit Jhala, Greg Morrisett, Mukund
nothaman, Chung-chieh Shan, Leonid Spesivtsev,
Andrew Tolmach
Contents
Overview
Download
WINNER OF JOLT PRODUCTIVITY AWARD
ANDREAS ZELLER
WHY PROGRAMS FAIL
A GUIDE TO SYSTEMATIC DEBUGGING
SECOND EDITION
dpunkt.verlag
MK
Aaron R. Bradley
Zohar Manna
The Calculus of Computation
Decision Procedures with Applications to Verification
Springer

Compilers: Principles, Techniques & Tools
by Aho, Lam, Sethi, Ullman

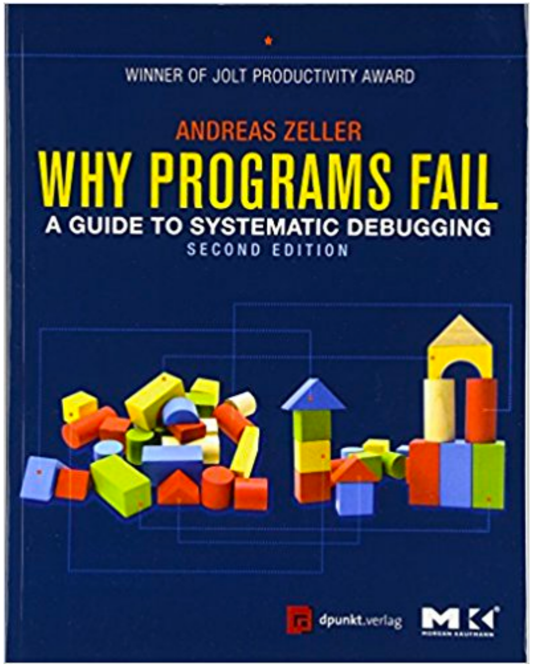


A classic book on Compilers (also called "Dragon book").

Relevant to course lessons on:

- Dataflow Analysis
- Pointer Analysis
- Type Systems, and
- Constraint-Based Analysis (including Datalog).

Why Programs Fail: A Guide to Systematic Debugging
by Andreas Zeller


WINNER OF JOLT PRODUCTIVITY AWARD
ANDREAS ZELLER
WHY PROGRAMS FAIL
A GUIDE TO SYSTEMATIC DEBUGGING
SECOND EDITION
dpunkt.verlag
MK

Book Website:

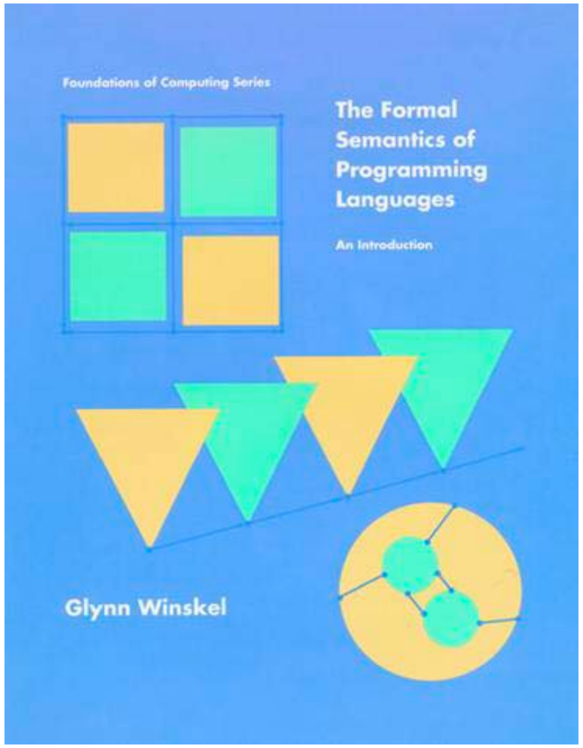
http://www.whyprogramsfail.com

Relevant to course lessons on:

- Delta Debugging
- Automated Test Generation
- Statistical Debugging

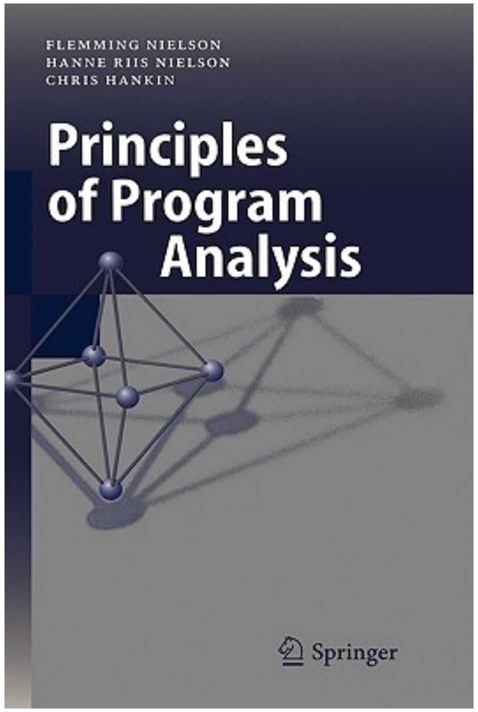
By author of Delta Debugging

The Formal Semantics of Programming Languages
by Glynn Winskel


Foundations of Computing Series
The Formal Semantics of Programming Languages
An Introduction
Glynn Winskel

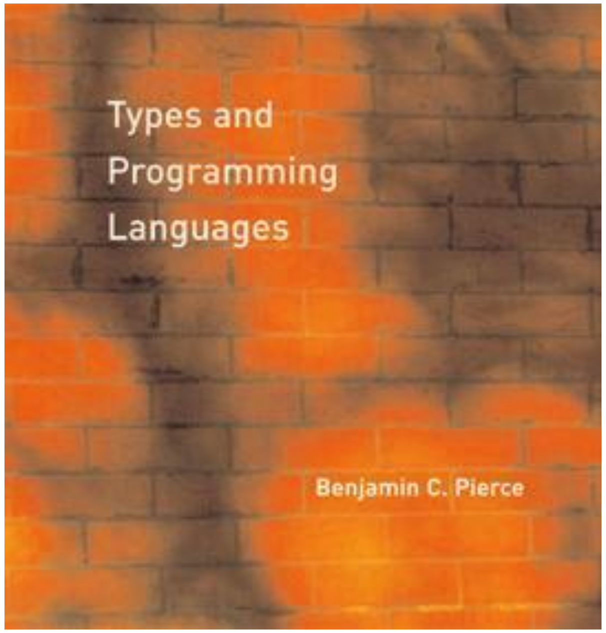
Describes formal ways to rigorously specify the meaning of programs in a given language (e.g. the WHILE language from the course lesson on Dataflow Analysis).

Principles of Program Analysis
by Nielson, Nielson, Hankin



Describes how to define dataflow analyses and prove them correct (i.e. sound) with respect to the concrete semantics of the language.

Types and Programming Languages
by Benjamin Pierce

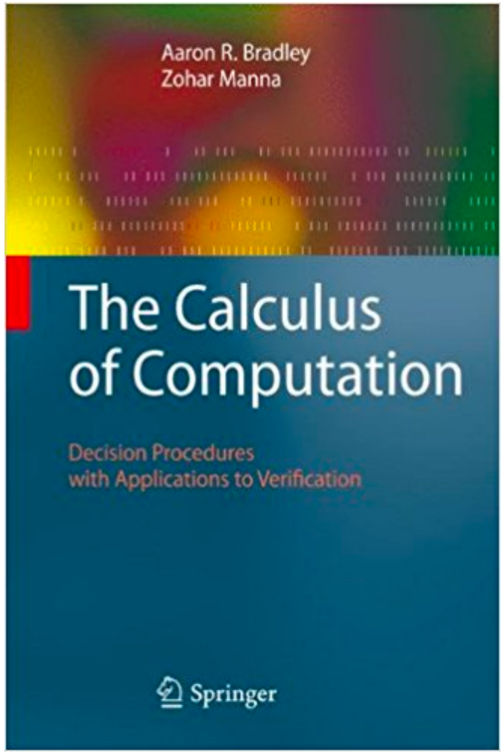


Book Website:

http://www.cis.upenn.edu/~bcpierce/tapl/

Relevant to course lesson on
Type Systems

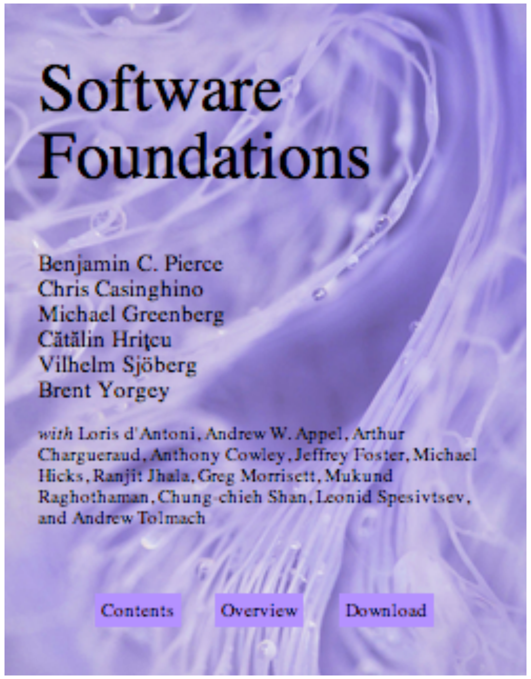
The Calculus of Computation
by Bradley and Manna


Aaron R. Bradley
Zohar Manna
The Calculus of Computation
Decision Procedures with Applications to Verification
Springer

Describes decision procedures with applications to program verification.

Useful in understanding how constraint solvers such as those used in course lessons on Constraint-Based Analysis and Dynamic Symbolic Execution work. These solvers can be viewed as decision procedures.

Software Foundations
by Benjamin Pierce et al.


Software Foundations

Benjamin C. Pierce
Chris Casinghino
Michael Greenberg
Cătălin Hriţcu
Vilhelm Sjöberg
Brent Yorgey

with Loris d'Antoni, Andrew W. Appel, Arthur Chargueraud, Anthony Cowley, Jeffrey Foster, Michael Hicks, Ranjit Jhala, Greg Morrisett, Mukund Raghothaman, Chung-chieh Shan, Leonid Spesivtsev, and Andrew Tolmach

Book website:

http://www.cis.upenn.edu/~bcpierce/sf/current/
(Available free online!)

Covers interactive theorem proving, which can prove richer program properties but is less automated.

Static analyses (e.g. dataflow analyses or type systems) can be viewed as lightweight theorem provers for *automatically* proving simpler program properties.