Qualifying-Exam Syllabus: Programming Languages and Compilers

Susan Horwitz, Somesh Jha, Ben Liblit, and Thomas Reps
University of Wisconsin–Madison

Fall 2013

The course requirements are CS 536, CS 701, CS 704, and CS 706. You should consult the syllabi for these courses when you study for the exam. If you did not take these courses, please ask the faculty for copies from recent offerings of the courses. (Required material is also sometimes covered, or covered in more detail, in CS 703; however, the topic of CS 703 varies from offering to offering.)

This document lists topics and concepts that the exam may cover, and is provided as a study aid. The general references that are provided in the different sections are places where you should be able to find a discussion of the majority of the topics listed in that section. In many cases, several general references have been given so that you have a number of different options (e.g., to find a treatment of a particular topic at a level that you find accessible). Many of the references are books. You do not have to read the entire book; consult the table of contents and the index to find where particular topics are discussed.

Most of the books are available at Wendt Library; some may be available only at the reserve desk.

1 Programming Paradigms

General References: [Sethi 1989]

- Procedural languages
- Object-oriented languages (C++ and Java)
  - Classes, objects, methods
  - Inheritance & subclass extension
- Functional languages (Lisp or Scheme, ML) [Field and Harrison 1988]
  - Programs as expressions
  - Programming without side-effects
  - Lists and list operators (cons, head, tail)
  - Tail recursion
  - Functions as first-class objects
  - Polymorphism
- λ-calculus [Glaser et al. 1984; Stoy 1977]
  - Substitution, reduction rules, normal form
  - Expressing programming constructs in λ-calculus: booleans, numerals, pairing
  - Fixed-point combinators [Stoy 1977, Chapter 5]
  - Confluence (Church-Rosser property) [Rosser 1982, Section 4]

2 Features and Properties of Languages

General References: [Pratt 1983; Sethi 1989]

- Scoping

  - Dynamic (Lisp)
  - Static
  - Extensions for importing and exporting names
- Evaluation
  - Lazy [Field and Harrison 1988, Chapter 4]
- Exceptions (Java, ML)
- Parameter-passing modes
  - Value
  - Result
  - Value-result
  - Reference
  - Name
- Data encapsulation (classes, modules)
- Overloading and coercion (C++ and Java)
- Infinite data objects [Friedman and Wise 1976]
- Multiple inheritance [Cardelli 1984]

3 Translation and Implementation

General References: [Aho et al. 1986; Fischer and LeBlanc 1988; Muchnick 1997; Waite and Goos 1985; Wilhelm and Maurer 1995]

- Lexical analysis
- Parsing
- Symbol tables and type checking
- Code generation
  - Algorithms on trees and directed acyclic graphs
  - Syntax-directed translation
- Register allocation
  - Sethi–Ullman register allocation [Aho et al. 1986, Section 9.10; Sethi and Ullman 1970]
  - Graph-coloring methods [Chaitin 1982]
- Partial Evaluation [Jones et al. 1993, Chapters 1, 3, 4, and 5]
  - Futamura projections
  - Binding-time analysis/specialization
  - Runtime code generation [Auslander et al. 1996]
- Runtime execution models
  - Activation records
  - Dynamic storage management (heap-allocated storage)
  - Garbage collection [Jones and Lins 1996]
- Interpretation

4 Formal Methods for Describing Languages and Reasoning About Programs

General References: [Aho et al. 1986; Field and Harrison 1988; Nielson and Nielson 1992; Schmidt 1986]

- Grammars
  - Regular expressions
  - Context-free grammars
- Structural induction [Burstall 1969]

- Operational semantics [Nielson and Nielson 1992]
  - Small-step
  - Large-step
- Axiomatic semantics [Hoare 1969; Nielson and Nielson 1992]
  - Partial correctness
  - Total correctness
- Denotational semantics [Gordon 1979; Schmidt 1986; Stoy 1977]
  - Continuation semantics [Gordon 1979]
- Domain theory [Schmidt 1986, Chapters 3, 6, and 11]

5 Intermediate Representations (IRs)

General References: [Aho et al. 1986; Muchnick 1997]

- Parse trees
- Abstract syntax trees
- DAGs
- Control-flow graphs
- Call graphs
- Dependences and dependence-based IRs
  - Control dependence
  - Data dependence (flow, anti, output)
  - Program dependence graphs
  - SSA-form

6 Static and Dynamic Program Analysis

General References: [Aho et al. 1986; Muchnick 1997; Nielson et al. 1999; Wilhelm and Maurer 1995]

- Polymorphic type checking [Field and Harrison 1988; Peyton Jones 1987, Chapters 8 and 9]
  - Type variables
  - Type inference
  - Typed λ-calculus
  - Unification, substitutions, type environments, Algorithm W
- Intraprocedural dataflow analysis
  - Meet-over-all-paths (MOP) solution vs. solution to a set of equations
  - Flow-sensitive vs. flow-insensitive problems
- Interprocedural dataflow analysis
  - Meet-over-all-valid paths (MOVP) solution vs. solution to a set of equations
  - Context-sensitive vs. context-insensitive problems
- Abstract interpretation [Abramsky and Hankin 1987; Nielson et al. 1999]
  - Collecting semantics
  - Galois connection, Galois insertion (abstraction/concretization)
  - Soundness requirements
- Declarative notations for specifying dataflow-analysis problems
  - Program analysis using Datalog [Whaley and Lam 2004]
  - Set constraints [Aiken 1999]
  - Graph-reachability criteria [Reps 1997]

- Fixed-point-finding techniques
- Points-to analysis/alias analysis
- Optimization
  - Basic block
  - Intraprocedural
  - Interprocedural
- Software verification techniques [Ball and Rajamani 2001; Corbett et al. 2000; Engler et al. 2000]
- Profiling and instrumentation
  - Path profiling [Ball and Larus 1996]

7 Language-Based Security

- Software fault isolation (sandboxing) Wahbe et al. [1993]
- Stack inspection [Erlingsson and Schneider 2000]
- Safe kernel extensions without run-time checking [Necula and Lee 1996]
- Static analysis for computer security [Ashcraft and Engler 2002; Chen and Wagner 2002; Jensen et al. 1999; Wagner and Dean 2001; Wagner et al. 2000]

8 Software Engineering

- Software life cycles
  - Waterfall model [Royce 1987]
  - Spiral model [Boehm 1995]
  - Extreme programming [Beck 1999]
- Analysis of formal specifications
  - Temporal logics
  - LTL and CTL model checking [Dwyer et al. 1999; Holzmann 1997]
  - Relational modeling [Jackson 1998; Seater et al. 2012]
- Architecture and design
  - Aspect-oriented programming [Kiczales et al. 1997]
  - Design patterns [Gamma et al. 1993]
- Implementation and program analysis
  - Nonstandard type systems [Foster et al. 1999; Necula et al. 2005]
  - Abstraction-based model checking
  - Meta-level compilation [Hallem et al. 2002]
- Testing and debugging
  - Concolic testing [Godefroid et al. 2005]
  - Concurrency bugs [Park et al. 2009]
  - Delta debugging [Zeller and Hildebrandt 2002]
  - Statistical debugging [Liblit et al. 2003, 2005]
  - Empirical evaluation and human factors [Parnin and Orso 2011]
- Maintenance and evolution
  - Software metrics and code decay [Eick et al. 2001]
  - Refactoring [Kataoka et al. 2001]
  - Code clones and clone detection [Kim et al. 2005]
  - Repository mining [Zimmermann et al. 2004]
  - Semantic differencing [Apiwattanapong et al. 2004]
  - Long-term tracking and evolution [Thain et al. 2006]
- Reverse engineering and program understanding

- Dynamic invariant detection [Ernst et al. 2001]
- Development environments [Bragdon et al. 2010]

References

Note: you should not need to pay to access any documents on this list. "Online copy available at" notes link to free copies of some readings. Many resources are free when accessed on campus, or from within the Department of Computer Sciences. See https://csl.cs.wisc.edu/services/remote-access/department-vpn for instructions on how to take advantage of this from non-departmental machines. The campus library offers free access to many resources at http://ezproxy.library.wisc.edu/menu#ezproxy-testing-menu. If you find that any links given below no longer work, or you cannot find any online copies of some paper, please contact the faculty members listed at the top of this document for help.

S. Abramsky and C. Hankin. An introduction to abstract interpretation. In *Abstract Interpretation of declarative languages*, volume 1, pages 63–102. Ellis Horwood, 1987.

A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers: principles, techniques, and tools*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986. ISBN 0-201-10088-6.

A. Aiken. Introduction to set constraint-based program analysis. *Sci. Comput. Program.*, 35(2-3):79–111, Nov. 1999. ISSN 0167-6423. doi: 10.1016/S0167-6423(99)00007-6. URL http://dx.doi.org/10.1016/S0167-6423(99)00007-6. Online copy available at http://www.cs.berkeley.edu/~aiken/publications/papers/scp99.ps.

T. Apiwattanapong, A. Orso, and M. J. Harrold. A differencing algorithm for object-oriented programs. In *Proceedings of the 19th IEEE international conference on Automated software engineering*, ASE '04, pages 2–13, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2131-2. doi: 10.1109/ASE.2004.5. URL http://dx.doi.org/10.1109/ASE.2004.5. Online copy available at http://www.cc.gatech.edu/~orso/papers/term.orso.harrold.ASE04.pdf.

K. Ashcraft and D. Engler. Using programmer-written compiler extensions to catch security holes. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP '02, pages 143–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1543-6. URL http://dl.acm.org/citation.cfm?id=829514.830533. Online copy available at http://www.stanford.edu/~engler/sp-ieee-02.pdf.

J. Auslander, M. Philipose, C. Chambers, S. J. Eggers, and B. N. Bershad. Fast, effective dynamic compilation. In *Proceedings of the ACM SIGPLAN 1996 conference on Programming language design and implementation*, PLDI '96, pages 149–159, New York, NY, USA, 1996. ACM. ISBN 0-89791-795-2. doi: 10.1145/231379.231409. URL http://doi.acm.org/10.1145/231379.231409. Online copy available at http://www.cs.washington.edu/research/dyncomp/Papers/pldi96-abstract.html.

T. Ball and J. R. Larus. Efficient path profiling. In *Proceedings of the 29th annual ACM/IEEE international symposium on Microarchitecture*, MICRO 29, pages 46–57, Washington, DC, USA, 1996. IEEE Computer Society. ISBN 0-8186-7641-8. URL http://dl.acm.org/citation.cfm?id=243846.243857.

T. Ball and S. K. Rajamani. Automatically validating temporal safety properties of interfaces. In *Proceedings of the 8th international SPIN workshop on Model checking of software*, SPIN '01, pages 103–122, New York, NY, USA, 2001. Springer-Verlag New York, Inc. ISBN 3-540-42124-6. URL http://dl.acm.org/citation.cfm?id=380921.380932.

K. Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, Oct. 1999. ISSN 0018-9162. doi: 10.1109/2.796139. URL http://dx.doi.org/10.1109/2.796139. Online copy available at http://faculty.salisbury.edu/~xswang/research/papers/serelated/xp/rx070.pdf.

B. W. Boehm. Human-computer interaction. In R. M. Baecker, J. Grudin, W. A. S. Buxton, and S. Greenberg, editors, *Human-computer interaction*, chapter A spiral model of software development and enhancement, pages 281–292. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995. ISBN 1-55860-246-1. URL http://dl.acm.org/citation.cfm?id=212925.212952.

A. Bragdon, S. P. Reiss, R. Zeleznik, S. Karumuri, W. Cheung, J. Kaplan, C. Coleman, F. Adeputra, and J. J. LaViola, Jr. Code bubbles: rethinking the user interface paradigm of integrated development environments. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 455–464, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-719-6. doi: 10.1145/1806799.1806866. URL http://doi.acm.org/10.1145/1806799.1806866.

R. Burstall. Proving properties of programs by structural induction. *The Computer Journal*, 12(1):41–48, 1969. URL http://www.cs.wisc.edu/~cs704-1/Papers/induction.69.pdf.

L. Cardelli. A semantics of multiple inheritance. In *Proc. of the international symposium on Semantics of data types*, pages 51–67, New York, NY, USA, 1984. Springer-Verlag New York, Inc. ISBN 3-540-13346-1. URL http://dl.acm.org/citation.cfm?id=1096.1098.

G. J. Chaitin. Register allocation & spilling via graph coloring. In *Proceedings of the 1982 SIGPLAN symposium on Compiler construction*, SIGPLAN '82, pages 98–105, New York, NY, USA, 1982. ACM. ISBN 0-89791-074-5. doi: 10.1145/800230.806984. URL http://doi.acm.org/10.1145/800230.806984.

H. Chen and D. Wagner. MOPS: an infrastructure for examining security properties of software. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS '02, pages 235–244, New York, NY, USA, 2002. ACM. ISBN 1-58113-612-9. doi: 10.1145/586110.586142. URL http://doi.acm.org/10.1145/586110.586142.

J. C. Corbett, M. B. Dwyer, J. Hatcliff, S. Laubach, C. S. Păsăreanu, Robby, and H. Zheng. Bandera: extracting finite-state models from Java source code. In *Proceedings of the 22nd international conference on Software engineering*, ICSE '00, pages 439–448, New York, NY, USA, 2000. ACM. ISBN 1-58113-206-9. doi: 10.1145/337180.337234. URL http://doi.acm.org/10.1145/337180.337234.

M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Patterns in property specifications for finite-state verification. In *Proceedings of the 21st international conference on Software engineering*, ICSE '99, pages 411–420, New York, NY, USA, 1999. ACM. ISBN 1-58113-074-0. doi: 10.1145/302405.302672. URL http://doi.acm.org/10.1145/302405.302672.

S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus. Does code decay? Assessing the evidence from change management data. *IEEE Trans. Softw. Eng.*, 27(1):1–12, Jan. 2001. ISSN 0098-5589. doi: 10.1109/32.895984. URL http://dx.doi.org/10.1109/32.895984.

D. Engler, B. Chelf, A. Chou, and S. Hallem. Checking system rules using system-specific, programmer-written compiler extensions. In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation - Volume 4*, OSDI'00, pages 1–1, Berkeley, CA, USA, 2000. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1251229.1251230.

U. Erlingsson and F. B. Schneider. IRM enforcement of Java stack inspection. In *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP '00, pages 246–, Washington, DC, USA, 2000. IEEE Computer Society. ISBN 0-7695-0665-8. URL http://dl.acm.org/citation.cfm?id=882494.884407. Online copy available at http://www.cs.cornell.edu/fbs/publications/sasiOakland.ps.

M. D. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin. Dynamically discovering likely program invariants to support program evolution. *IEEE Trans. Softw. Eng.*, 27(2):99–123, Feb. 2001. ISSN 0098-5589. doi: 10.1109/32.908957. URL http://dx.doi.org/10.1109/32.908957.

A. J. Field and P. G. Harrison. *Functional Programming*. Addison-Wesley, 1988. ISBN 0-201-19249-7.

C. N. Fischer and R. J. LeBlanc, Jr. *Crafting a compiler*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1988. ISBN 0-8053-3201-4.

J. S. Foster, M. Fähndrich, and A. Aiken. A theory of type qualifiers. In *Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation*, PLDI '99, pages 192–203, New York, NY, USA, 1999. ACM. ISBN 1-58113-094-5. doi: 10.1145/301618.301665. URL http://doi.acm.org/10.1145/301618.301665. Online copy available at http://www.cs.umd.edu/~jfoster/papers/pldi99.pdf.

D. P. Friedman and D. S. Wise. CONS should not evaluate its arguments. In *ICALP*, pages 257–284, 1976.

E. Gamma, R. Helm, R. E. Johnson, and J. M. Vlissides. Design patterns: Abstraction and reuse of object-oriented design. In *Proceedings of the 7th European Conference on Object-Oriented Programming*, ECOOP '93, pages 406–431, London, UK, UK, 1993. Springer-Verlag. ISBN 3-540-57120-5. URL http://dl.acm.org/citation.cfm?id=646151.679366. Online copy available at http://citeseer.ist.psu.edu/gamma93design.html.

H. Glaser, C. Hankin, and D. Till. *Principles of functional programming*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1984. ISBN 0-13-709148-6.

P. Godefroid, N. Klarlund, and K. Sen. DART: directed automated random testing. In *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '05, pages 213–223, New York, NY, USA, 2005. ACM. ISBN 1-59593-056-6. doi: 10.1145/1065010.1065036. URL http://doi.acm.org/10.1145/1065010.1065036.

M. J. C. Gordon. *The Denotational Description of Programming Languages: An Introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1979. ISBN 0387904336.

S. Hallem, B. Chelf, Y. Xie, and D. Engler. A system and language for building system-specific, static analyses. In *Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation*, PLDI '02, pages 69–82, New York, NY, USA, 2002. ACM. ISBN 1-58113-463-0. doi: 10.1145/512529.512539. URL http://doi.acm.org/10.1145/512529.512539.

C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, Oct. 1969. ISSN 0001-0782. doi: 10.1145/363235.363259. URL http://doi.acm.org/10.1145/363235.363259.

G. J. Holzmann. The model checker SPIN. *IEEE Trans. Softw. Eng.*, 23(5):279–295, May 1997. ISSN 0098-5589. doi: 10.1109/32.588521. URL http://dx.doi.org/10.1109/32.588521.

D. Jackson. Boolean compilation of relational specifications. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1998.

T. P. Jensen, D. Le Métayer, and T. Thorn. Verification of control flow based security properties. In *IEEE Symposium on Security and Privacy*, pages 89–103. IEEE Computer Society, 1999. ISBN 0-7695-0176-1.

N. D. Jones, C. K. Gomard, and P. Sestoft. *Partial evaluation and automatic program generation*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-020249-5. Online copy available at http://www.dina.kvl.dk/~sestoft/pebook/jonesgomardsestoft.ps, http://www.dina.kvl.dk/~sestoft/pebook/jonesgomardsestoft.pdf.

R. Jones and R. Lins. *Garbage collection: algorithms for automatic dynamic memory management*. John Wiley & Sons, Inc., New York, NY, USA, 1996. ISBN 0-471-94148-4.

Y. Kataoka, D. Notkin, M. D. Ernst, and W. G. Griswold. Automated support for program refactoring using invariants. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM'01)*, ICSM '01, pages 736–, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1189-9. doi: 10.1109/ICSM.2001.972794. URL http://dx.doi.org/10.1109/ICSM.2001.972794. Online copy available at http://homes.cs.washington.edu/~mernst/pubs/refactoring-icsm2001.pdf.

G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. V. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. In *ECOOP*, pages 220–242, 1997. Online copy available at http://www.cs.ubc.ca/~gregor/papers/kiczales-ECOOP1997-AOP.pdf.

M. Kim, V. Sazawal, D. Notkin, and G. Murphy. An empirical study of code clone genealogies. *SIGSOFT Softw. Eng. Notes*, 30(5):187–196, Sept. 2005. ISSN 0163-5948. doi: 10.1145/1095430.1081737. URL http://doi.acm.org/10.1145/1095430.1081737.

B. Liblit, A. Aiken, A. X. Zheng, and M. I. Jordan. Bug isolation via remote program sampling. *SIGPLAN Not.*, 38(5): 141–154, May 2003. ISSN 0362-1340. doi: 10.1145/780822.781148. URL http://doi.acm.org/10.1145/780822.781148.

B. Liblit, M. Naik, A. X. Zheng, A. Aiken, and M. I. Jordan. Scalable statistical bug isolation. *SIGPLAN Not.*, 40(6): 15–26, June 2005. ISSN 0362-1340. doi: 10.1145/1064978.1065014. URL http://doi.acm.org/10.1145/1064978.1065014.

S. S. Muchnick. *Advanced compiler design and implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. ISBN 1-55860-320-4.

G. C. Necula and P. Lee. Safe kernel extensions without run-time checking. In *Proceedings of the second USENIX symposium on Operating systems design and implementation*, OSDI '96, pages 229–243, New York, NY, USA, 1996. ACM. ISBN 1-880446-82-0. doi: 10.1145/238721.238781. URL http://doi.acm.org/10.1145/238721.238781. Online copy available at http://www.cs.berkeley.edu/~necula/papers.html.

G. C. Necula, J. Condit, M. Harren, S. McPeak, and W. Weimer. CCured: type-safe retrofitting of legacy software. *ACM Trans. Program. Lang. Syst.*, 27(3):477–526, May 2005. ISSN 0164-0925. doi: 10.1145/1065887.1065892. URL http://doi.acm.org/10.1145/1065887.1065892. Online copy available at http://www.cs.berkeley.edu/~necula/Papers/ccured_popl02.pdf.

F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999. ISBN 3540654100.

H. R. Nielson and F. Nielson. *Semantics with applications: a formal introduction*. John Wiley & Sons, Inc., New York, NY, USA, 1992. ISBN 0-471-92980-8. Online copy available at http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.ps, http://www.daimi.au.dk/~bra8130/Wiley_book/wiley.pdf.

S. Park, S. Lu, and Y. Zhou. CTrigger: exposing atomicity violation bugs from their hiding places. *SIGPLAN Not.*, 44(3):25–36, Mar. 2009. ISSN 0362-1340. doi: 10.1145/1508284.1508249. URL http://doi.acm.org/10.1145/1508284.1508249.

C. Parnin and A. Orso. Are automated debugging techniques actually helping programmers? In *Proceedings of the 2011 International Symposium on Software Testing and Analysis*, ISSTA '11, pages 199–209, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0562-4. doi: 10.1145/2001420.2001445. URL http://doi.acm.org/10.1145/2001420.2001445.

S. L. Peyton Jones. *The Implementation of Functional Programming Languages (Prentice-Hall International Series in Computer Science)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1987. ISBN 013453333X.

T. W. Pratt. *Programming languages: design and implementation (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1983. ISBN 0-13-730580-X.

T. Reps. Program analysis via graph reachability. In *Proceedings of the 1997 international symposium on Logic programming*, ILPS '97, pages 5–19, Cambridge, MA, USA, 1997. MIT Press. ISBN 0-262-63180-6. URL http://dl.acm.org/citation.cfm?id=271338.271343. Online copy available at http://www.cs.wisc.edu/wpis/papers/tr1386.ps, http://www.cs.wisc.edu/wpis/papers/tr1386.pdf.

J. B. Rosser. Highlights of the history of the lambda-calculus. In *Proceedings of the 1982 ACM symposium on LISP and functional programming*, LFP '82, pages 216–225, New York, NY, USA, 1982. ACM. ISBN 0-89791-082-6. doi: 10.1145/800068.802153. URL http://doi.acm.org/10.1145/800068.802153.

W. W. Royce. Managing the development of large software systems: concepts and techniques. In *Proceedings of the 9th international conference on Software Engineering*, ICSE '87, pages 328–338, Los Alamitos, CA, USA, 1987. IEEE Computer Society Press. ISBN 0-89791-216-0. URL http://dl.acm.org/citation.cfm?id=41765.41801.

D. A. Schmidt. *Denotational semantics: a methodology for language development*. William C. Brown Publishers, Dubuque, IA, USA, 1986. ISBN 0-697-06849-2.

R. Seater, G. Dennis, D. Le Berre, and F. Chang. Tutorial for Alloy Analyzer 4.0, Aug. 2012. URL http://alloy.mit.edu/alloy/tutorials/online/.

R. Sethi. *Programming languages: concepts and constructs*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989. ISBN 0-201-10365-6.

R. Sethi and J. D. Ullman. The generation of optimal code for arithmetic expressions. *J. ACM*, 17(4):715–728, Oct. 1970. ISSN 0004-5411. doi: 10.1145/321607.321620. URL http://doi.acm.org/10.1145/321607.321620.

J. E. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press, Cambridge, MA, USA, 1977. ISBN 0262191474.

D. Thain, T. Tannenbaum, and M. Livny. How to measure a large open-source distributed system: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(15):1989–2019, Dec. 2006. ISSN 1532-0626. doi: 10.1002/cpe.v18:15. URL http://dx.doi.org/10.1002/cpe.v18:15.

D. Wagner and D. Dean. Intrusion detection via static analysis. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, SP '01, pages 156–, Washington, DC, USA, 2001. IEEE Computer Society. URL http://dl.acm.org/citation.cfm?id=882495.884434. Online copy available at http://www.cs.berkeley.edu/~daw/papers/ids-oakland01.pdf.

D. Wagner, J. S. Foster, E. A. Brewer, and A. Aiken. A first step towards automated detection of buffer overrun vulnerabilities. In *NDSS*. The Internet Society, 2000. ISBN 1-891562-07-X, 1-891562-08-8. Online copy available at http://www.cs.berkeley.edu/~daw/papers/overruns-ndss00.ps.

R. Wahbe, S. Lucco, T. E. Anderson, and S. L. Graham. Efficient software-based fault isolation. In *Proceedings of the fourteenth ACM symposium on Operating systems principles*, SOSP '93, pages 203–216, New York, NY, USA, 1993. ACM. ISBN 0-89791-632-8. doi: 10.1145/168619.168635. URL http://doi.acm.org/10.1145/168619.168635.

W. M. Waite and G. Goos. *Compiler Construction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1985. ISBN 0387908218.

J. Whaley and M. S. Lam. Cloning-based context-sensitive pointer alias analysis using binary decision diagrams. In *Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation*, PLDI '04, pages 131–144, New York, NY, USA, 2004. ACM. ISBN 1-58113-807-5. doi: 10.1145/996841.996859. URL http://doi.acm.org/10.1145/996841.996859.

R. Wilhelm and D. Maurer. *Compiler Design*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1995. ISBN 0201422905.

A. Zeller and R. Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Trans. Softw. Eng.*, 28(2):183–200, Feb. 2002. ISSN 0098-5589. doi: 10.1109/32.988498. URL http://dx.doi.org/10.1109/32.988498. Online copy available at http://www.st.cs.uni-sb.de/papers/tse2002/.

T. Zimmermann, P. Weisgerber, S. Diehl, and A. Zeller. Mining version histories to guide software changes. In *Proceedings of the 26th International Conference on Software Engineering*, ICSE '04, pages 563–572, Washington, DC, USA, 2004. IEEE Computer Society. ISBN 0-7695-2163-0. URL http://dl.acm.org/citation.cfm?id=998675.999460. Online copy available at http://www.st.cs.uni-saarland.de/papers/icse2004/icse.pdf.