Qualifying-Exam Syllabus: Programming Languages and Compilers

Spring 2008

Charles Fischer, Susan Horwitz, Somesh Jha, Ben Liblit, and Thomas Reps

University of Wisconsin

The course requirements are CS 536, CS 538, CS 701, and CS 704. 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. (A small amount of required material is covered in CS 706. 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
  - Multiple inheritance
- 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
  - Type variables
  - Type inference
- λ-calculus [Stoy 1977], [Glaser, Hankin, and Till 1984]
  - 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]
- Logic languages (Prolog)
  - Logic vs. control
— Facts and rules
— Backtracking and unification
— Depth-first and breadth-first search
— Issues concerning the ordering of rules and literals

2. Features and Properties of Languages
General References: [Sethi 1989], [Pratt 1984]
• 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 types and user-defined data types
  — Abstract data types
  — Name/structural equivalence
• 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
• 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, Sethi, Ullman 1985, Section 9.10], [Sethi and Ullman 1970]
  — Graph-coloring methods [Chaitin 1982]
• Partial Evaluation [Jones et al. 1993]
  — 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, Sethi, Ullman 1985], [Schmidt 1986], [Field and Harrison 1988], [Nielson and Nielson 1992]
• 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 [Schmidt 1986], [Stoy 1977], [Gordon 1979]
  — Continuation semantics [Gordon 1979]
• Domain theory [Schmidt 1986, Chapters 3, 6, and 11]

5. Intermediate Representations (IRs)
General References: [Aho, Sethi, and Ullman 1985], [Muchnick 2000]
• 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


• Polymorphic type checking [Field and Harrison 1988], [Hancock 1987a], [Hancock 1987b]
  — 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
  — Horn clauses
  — Set constraints [Aiken 1999]
  — Graph-reachability criteria [Reps 1998]
  — Nonstandard type systems [Necula et al. 2002], [Foster et al. 1999]

• Fixed-point-finding techniques

• Points-to analysis/alias analysis

• Optimization
  — Basic block
  — Intraprocedural
  — Interprocedural

• Software verification techniques [Ball and Rajamani 2001], [Engler et al. 2000], [Corbett 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
  [Wagner et al. 2000], [Wagner and Dean 2001], [Chen and Wagner 2002], [Jensen et al. 1999],
  [Ashcraft and Engler 2002]

8. Software Engineering

• Architecture and design [Gamma et al. 1993], [Kiczales et al. 1997]

• Testing and debugging [Zeller and Hildebrandt 2002]

9. References

[Aiken 1999]

[Ashcraft and Engler 2002]

[Auslander et al. 1996]

[Abramsky and Hankin 1987]

[Aho, Sethi, Ullman 1985]

[Ball and Larus 1996]

[Ball and Rajamani 2001]

[Burstall 1969]

[Cardelli 1984]

[Chaitin 1982]

[Chen and Wagner 2002]

[Corbett et al. 2000]

[Engler et al. 2000]

[Erlingsson and Schneider 2000]

[Field and Harrison 1988]

[Fischer and LeBlanc 1988]

[Foster et al. 1999]
Foster, Fähndrich, and Aiken, A theory of type qualifiers, PLDI 1999.

[Friedman and Wise 1976]

[Gamma et al. 1993]

[Glaser, Hankin, and Till 1984]

[Gordon 1979]

[Hancock 1987a]
[Hancock 1987a]  

[Hoare 1969]  

[Jensen et al. 1999]  

[Jones et al. 1993]  

[Jones and Lins 1996]  

[Kiczales et al. 1997]  

[Muchnick 2000]  

[Necula and Lee 1996]  

[Necula et al. 2002]  
Necula, McPeak, and Weimer, CCured: Type-safe retrofitting of legacy code, POPL 2002.  

[Nielson and Nielson 1992]  

[Nielson et al. 1999]  

[Pratt 1984]  

[Reps 1998]  
[Rosser 1982]

[Schmidt 1986]

[Sethi 1989]

[Sethi and Ullman 1970]

[Stoy 1977]

[Wagner and Dean 2001]

[Wagner et al. 2000]

[Wahbe et al. 1993]

[Waite and Goss 1984]

[Wilhelm and Maurer 1995]

[Zeller and Hildebrandt 2002]