1 Language Principles, Methodology, and Features

This material is covered in undergraduate programming language courses such as CS312, CS400, and CS411. Students should be familiar with formal models and semantics of programming languages, programming methodology, and principles of program correctness. Students should also have good working knowledge of standard programming language paradigms, concepts, and constructs. Specific topics in language semantics, models, and methodology include:

- Inductive definitions, proof trees, proofs by induction on syntax and on derivations;
- Large-step and small-step operational semantics, configurations;
- Pure and typed lambda calculus;
- Axiomatic semantics, partial and total program correctness, weakest preconditions, Hoare-style proofs, loop invariants.

Specific language features and paradigms include:

- Imperative languages: assignment, loop constructs, arrays, procedures, functions, parameter-passing mechanisms, lexical scope, etc. (e.g., FORTRAN, Pascal, C)
- Functional languages: higher-order functions, polymorphism, lexical vs. dynamic scope (e.g., Lisp, Scheme, ML, Haskell)
- Type systems, strong vs. weak typing, static vs. dynamic typing;
- Object-oriented languages: objects, classes, inheritance, virtual vs. static methods, subtyping (e.g., Java, C++, Smalltalk)
- Logic and relational languages: propositional and first-order predicate calculus, Horn clauses, unification, backtracking (e.g., Prolog)
- Concurrent and parallel languages: threads, synchronization constructs (mutex locks, condition variables, monitors, fork/join, channels, rendezvous). (e.g. Modula-3, Java, Ada, Concurrent ML)
- Data abstraction: modules, interfaces, encapsulation, polymorphism (e.g., Modula-3, Ada, Eiffel, ML)
- Continuations, non-local control-flow, exceptions.

General references:

Specific language references:

6. A. Goldberg and D. Robson. *Smalltalk 80: The Language*. Addison-Wesley, 1989.

## 2 Language Implementation

This material is covered in an undergraduate compiler class, such as CS412. Students should be familiar with all of the phases in the compilation process, and should have good knowledge about the compilation of standard language constructs. Specific topics include:

- Lexical analysis, context-free grammars; top-down and bottom-up parsing; LL, LR, and LALR grammars; automatic parser and scanner generators;
- Type checking, typing rules; other semantic checks;
- Syntax-directed translation;
- Intermediate representations: three-address code, stack-based representations; control-flow graphs;
- Control-flow analysis and dominators;
- Dataflow analysis: lattices, transfer functions, gen/kill sets, meet-over-paths solution, termination conditions; specific analyses (e.g., live variable analysis, available expressions, etc.);
- Compiler optimizations: copy propagation, constant folding, dead code elimination, strength reduction, induction variable elimination, loop invariant code motion, etc;
- Code generation and run-time mechanisms: stacks and activation records; static and dynamic links; parameter-passing mechanisms (e.g., call-by-value, call-by-need, etc.); dynamic allocation and garbage collection; virtual method calls, dynamic dispatch; run-time checks;
- Back-end optimizations: instruction selection; register allocation.
- Compilation vs. interpretation, recursive interpreters, just-in-time compilers;

Example references: