CS 4700 Programming Languages  Review Sheet

For specific examples of how the material may be tested, see the homework and first two exams.

1 Introduction

1.1 What is a Programming Language?
1.2 Abstractions in Programming Languages
1.3 Computational Paradigms
1.4 Interpretation/compilation

2 History

2.1 Early History: The First Programmer
2.2 The 1950s: The First Programming Languages
2.3 The 1960s: An Explosion in Programming Languages
2.4 The 1970s: Simplicity, Abstraction, Study

3 Language Design Principles

3.1 History and Design Criteria: portability, orthogonality, readability, writability, simplicity, efficiency, reliability, regularity
   operator overloading
3.2 Efficiency
3.3 Regularity
3.4 Further Language Design Principles

4 Syntax

4.1 Lexical Structure of Programming Languages
   Regular Expressions, Thompson construction, NFA to DFA construction, lexical analysis
4.2 Regular, Context-Free, Context Sensitive Grammars and BNFs
4.3 Parse Trees and Abstract Syntax Trees
4.4 Ambiguity, Associativity, and Precedence - from the grammar
4.5 Elimination of left recursion, left factoring
4.5 syntax diagram

5 Basic Semantics

5.1 Attributes, Binding, and Semantic Functions
5.2 Declarations, Blocks, and Scope (static/dynamic)
5.3 The Symbol Table
5.4 Name Resolution and Overloading
5.5 Allocation, Lifetimes, and the Environment
5.6 Variables and Constants
5.7 Aliases, Dangling References, and Garbage

6 Data Types
6.1 Static/Dynamic typing
6.2 Type Equivalence: name equivalence, declaration equivalence, structural equivalence
6.3 Type Checking
6.4 Type Conversion

7 Control I—Expressions and Statements
7.1 Expressions - evaluation order, lazy/eager evaluation, aliasing, side effect
7.2 Conditional Statements and Guards
7.3 Loops and Variation on WHILE
7.4 The GOTO Controversy
7.5 Exception Handling
7.6 Binding times: language definition, language implementation, translation time, link time, load time, execution time.
7.7 Short circuit evaluation
7.8 Array access formula, row major, column major, virtual origin, run time descriptor
7.9 Switch Statement - jump table

8 Control II—Procedures and Environments
8.1 Default arguments
8.2 Call stack, activation record, environment pointer
8.3 Parameter Passing Mechanisms: value, value-result, name, need, reference
8.4 Procedure Environments, Activations, and Allocation
8.5 Efficiency considerations: central referencing table, display, deep access, shallow access.

11 Functional Programming
11.1 Haskell, curried function

12 Logic Programming
12.1 Prolog, unification, backtracking

13 Scripting Languages
13.1 Ruby, list comprehension