CS 3304
Comparative Languages

Lecture 29:
Final Exam Review

1 May 2012

Final Exam

- The final exam is a closed book exam. Please review The Virginia Tech Undergraduate Honor System:

  `http://www.honorsystem.vt.edu`.
- Location: Hutcheson Hall 310 (HUTCH 310) – section 12059
- Date: 4 May 2012
- Time: 7:00pm
- Duration: 120 minutes
- It covers Chapters 1-13 (except Chapter 5).
- Consists of two parts:
  - Twenty quiz questions (multiple choice, true/false), two points each, 40 points total.
  - Six essay/problem questions, ten points each, a total of 60 points.
- It is a comprehensive exam, all material covered in lectures and assignments is included.

Chapter 1: Introduction

- Content:
  - The Art of Language Design
  - The Programming Language Spectrum
  - Why Study Programming Languages?
  - Compilation and Interpretation
  - Programming Environments
  - An Overview of Compilation
- Issues:
  - Language groups:
    - Imperative (von Neumann, scripting, object-oriented)
    - Declarative (functional, logic, dataflow, template-based)
  - The von Neumann architecture
  - Implementation: compilation, interpretation, hybrid.
  - Compilation phases: scanning, parsing, semantic analysis, …:
    - Symbol table

Chapter 2: Programming Language Syntax

- Content:
  - Specifying Syntax: Regular Expressions and Context-Free Grammars
  - Scanning
  - Parsing
  - Theoretical Foundations
- Issues:
  - Regular expressions: concatenation, alternation, Kleene closure
    - Scanning – deterministic finite automaton (DFA)
  - Context-free grammars – parse trees, derivations:
    - Set of terminals, set of non-terminals, start symbol, set of productions

    Generator for a context-free language
  - Parsing – language recognizer:
    - The two most important classes are called LL and LR
    - Table-driven parsing
    - Push-down automaton (PDA)

Chapter 3: Names, Scopes, and Bindings

- Content:
  - The Notion of Binding Time
  - Scope Rules
  - Implementing Scope
  - The Meaning of Names within a Scope
  - The Binding of Referencing Environments
  - Macro Expansion
  - Separate Compilation
- Issues:
  - Static (before run time) and dynamic (at run time):
    - Efficiency versus flexibility
  - Storage allocation: static, stack, dynamic:
    - Stack and heap-based allocation
  - Static and dynamic scoping
  - Deep and shallow binding

Chapter 4: Semantic Analysis

- Content:
  - The Role of Semantic Analyzer
  - Attribute Grammars
  - Evaluating Attributes
  - Action Routines
  - Space Management for Attributes
  - Decorating a Syntax Tree
- Issues:
  - Semantics rules:
    - Static: enforced by the compiler at compile time (semantic analyzer)
    - Dynamic: enforced by the compiler generated code at runtime
  - Static analysis:
    - Compile-time algorithms that predict run-time behavior
  - Annotation:
    - The process of evaluating attributes

Chapter 4: Continued

- Issues (continued):
  - Attribute grammars:
    - Synthesized and inherited attributes
    - Bottom-up and top-down attribute grammars
  - Semantic analyzer: automatic tools or ad-hoc/handwritten
  - Action routine - a semantic function that we tell the compiler to execute at a particular point in the parse:
    - Most compilers rely on action routines that evaluate attribute rules at specific points in a parse
  - Space management for attributes: symbol table
  - Syntax tree:
    - Tree grammar represents the possible structure

Chapter 6: Control Flow

- Content:
  - Expression Evaluation
  - Structured and Unstructured Flow
  - Sequencing
  - Selection
  - Iteration
  - Recursion
  - Non-determinacy
- Issues:
  - Eight categories of language mechanisms used to specify ordering
  - Operator, operands, and function calls
  - Precedence and associativity
  - Assignment and its semantics:
    - Value model of variables
    - Reference model of variables

Chapter 6: Continued

- Issues (continued):
  - Operand evaluation order
  - Short-circuit evaluation
  - Iteration - loops: enumeration-controlled, logically controlled, combination and iterators
  - Recursion requires no special syntax
    - Fundamental to functional programming
  - Evaluation order: applicative and normal
  - Lazy evaluation
  - Guarded commands

Chapter 7: Data Types

- Content:
  - Type Systems
  - Type Checking
  - Records (Structures) and Variants (Unions)
  - Arrays
  - Strings
  - Sets
  - Pointers and Recursive Types
  - Lists
  - Files and Input/Output
  - Equality Testing and Assignment
- Issues:
  - Type equivalence, type compatibility, type inference
  - Polymorphism
  - Classification of types: discrete (ordinal), scalar (complex), composite

Chapter 7: Continued

- Issues (continued):
  - Type conversion and casts
  - Overloading and coercion
  - Reference and value models
  - Dangling references and memory leaks:
    - Garbage collection
  - Input/output: one of the most difficult aspect of a language to design
  - L-values and r-values:
    - Shallow and deep comparison

Chapter 8: Subroutines and Control Abstraction

- Content:
  - Review of Stack Layout
  - Calling Sequences
  - Parameter Passing
  - Generic Subroutines and Modules
  - Exception Handling
  - Coroutines
  - Events
- Issues:
  - Abstraction: control and data
  - Subroutine nesting: static chains and dynamic link
  - Typical stack frame
  - Parameter modes: call-by-value, call-by-reference, call by value/result
    - Closures as parameters
    - Call-by-name

Chapter 8: Continued

- Issues (continued):
  - Generics: parameterizing control abstraction
  - Exception handling: moves error-checking code “out of line”
    - Multilevel returns
    - Simulated in a language that does not have them
  - Cactus stack
  - Event handling: sequential and thread-based

Chapter 9: Data Abstraction and Object Orientation

- Content:
  - Object-Oriented Programming
  - Encapsulation and Inheritance
  - Initialization and Finalization
  - Dynamic Method Binding
  - Multiple Inheritance
  - Object-Oriented Programming Revisited
- Issues:
  - Three key factors: encapsulation, inheritance, dynamic binding
  - Constructors
  - Virtual functions:
    - Non-virtual functions require no space at run time
  - Dynamic method binding extends polymorphism

Chapter 10: Functional Languages

- Content:
  - Historical Origins
  - Functional Programming Concepts
  - A Review/Overview of Scheme
  - Evaluation Order Revisited
  - Higher-Order Functions
  - Theoretical Foundations
  - Functional Programming in Perspective
- Issues:
  - Church's model of computing: the lambda calculus
    - 1st class and higher-order functions
  - Applicative and normal order
  - Monads
  - Currying

Chapter 11: Logic Languages

- Content:
  - Logic Programming Concepts
  - Prolog
  - Theoretical Foundations
  - Logic Programming in Perspective
- Issues:
  - Symbolic logic and inferencing process
    - First order predicate calculus
  - Horn clause:
    - Rules, fact (empty body), query (empty head)
  - Resolution and unification:
    - Bottom-up/top-down resolution
    - Forward/backward chaining
  - Imperative control flow: cut
  - Negation and closed-world assumption

Chapter 12: Concurrency

- Content:
  - Background and Motivation
  - Concurrent Programming Fundamentals
  - Implementing Synchronization
  - Language-Level Mechanisms
  - Message Passing
- Issues:
  - Concurrent, parallel and distributed
  - Race condition, synchronization, shared memory
  - Thread creation syntax
    - Two-level thread implementation
  - Scheduling/schedulers: uniprocessor and multiprocessor
  - Synchronization – busy-wait, spin locks, barriers
  - Semaphores, monitors, message passing
  - Transactional memory

Chapter 13: Scripting Languages

- Content:
  - What is a Scripting Language?
  - Problem Domains
  - Scripting the World Wide Web
  - Innovative Features
- Issues:
  - Shell languages - #! convention
  - Glue languages
  - Extension languages
  - Server and client-side scripting
    - JavaScript
  - String and pattern matching
  - Data types: numeric and composite