CS 461

Programming Language Concepts

Gang Tan
Computer Science and Engineering
Penn State University

1ST EXAM REVIEW

Format

- Exam Time and Place
  - Feb 21st, 12:20-1:10am (50 minutes)
  - Wartick Lab 110 (this room)
- In-class 50-min exam
  - Closed book and notes
  - All calculators, cell phones, and portable audio players must be put away for the duration of the test
- **Bring your clicker to sign in**
- Scope: everything before parameter passing
- Office hours moved this week
  - Tu 2:30-4:30pm Westgate W358

Types of Questions

- True/false questions
  - Decide whether a claim is true or false and explain why (even when it's true)
- Multiple-choice questions
- Short answer questions
- Versions of homework problems
- You will be expected to be able to read and write small programs

Ch1 Introduction

- Language = syntax + semantics + philosophy
- Programming paradigms
  - Imperative programming, OO, functional programming, logic programming
- Interpreter vs. compiler
- Possible kinds of questions
  - True/false questions; multiple-choice questions
- You do not need to know
  - ~~Date when a language was designed~~

Ch2 Grammar

- BNF grammar (CFG)
  - Terminals, nonterminals, production rules, start symbol
  - Derivation, left-most vs. right-most derivations, parse tree
- Ambiguity
  - Definition
  - Removing operator ambiguity
    - Adding explicit parenthesis
    - Design a new grammar to enforce associativity and precedence rules
    - Ambiguous grammar + informal spec. of associativity and precedence rules

Ch2 Grammar

- E-BNF
  - Writing more concise grammars
  - Alternatives, repetitions, optional parts
  - E-BNF is no more expressive than BNF
    - Can always convert E-BNF to BNF
- Possible questions
  - Given a grammar, write derivations and parse trees
  - Conversion from E-BNF to BNF
  - Design new BNF grammars
    - E.g., given associativity and precedence
  - Decide whether a grammar is ambiguous and explain why
  - Remove ambiguity

Ch2 Lexical and Syntactic Analysis

- Goal: algorithms for constructing a parse tree from input based on grammars
- Lexical vs. syntactic analysis
  - Lexical analysis: seq of chars to seq of tokens; regular expressions
  - syntactic analysis: seq of tokens to parse trees; BNF

Ch2 Lexical and Syntactic Analysis

- Regular expressions
  - epsilon, a, rs, r|s, r*
  - extended regular expressions
    - r+, r?, [a-z]
- FSA
  - states, input alphabet, transition function
  - accepting an input
  - deterministic FSA vs. nondeterministic FSA
  - Th: each RE corresponds to a deterministic FSA
- The construction of lexers and parsers
  - Lexer: a single FSA for all tokens; nextToken()
  - Parser: recursive-descent parsing

Ch2 Lexical and Syntactic Analysis

- Possible kinds of questions
  - Write regular expressions for syntax of tokens
  - Given a regular expression, develop an FSA
  - Conversion form extended regular expressions to regular expressions

Ch3 Names, Scopes and Bindings

- Syntactic issues for Naming
  - lexical rules;
- Binding: compile time vs. runtime
- Variable
  - common bindings; naming convention
  - l-values vs. r-values
- Scope: decides when a name is visible
  - nested scopes
  - "holes" in scopes
  - scope not the same as lifetime
- Constructs that can introduce a scope
  - Blocks, functions, for-loops, classes, packages (module), namespace

Ch3 Names, Scopes and Bindings

- Static scoping vs. dynamic scoping
  - Algorithm based on stack of dictionaries
- Possible kinds of questions
  - Scoping

Comparing the syntax of E-BNF and Regexps

- They have similar concepts but with different syntax
- E-BNF
  - Alternative parts in parentheses and separated with vertical bars
    - <exp> -> <exp> (+ | -) <exp>
  - 0-or-more repetitions in curly braces { }
    - <num> -> <digit> {<digit>}
  - Optional parts in square brackets [ ]
    - <if-stmt> -> if <test> then <stmt> [else <stmt>]

Comparing the syntax of E-BNF and Regexps

- Regexps
  - Alternation: r1 | r2
  - 0-or-more repetition: r*
  - Optional: r?
  - …