

Midterm Review

CSC324 Fall 2005

Introduction

- What is a PL?
- fetch-execute cycle
- Von Neumann bottleneck
- Compilation vs. Interpretation
- Language Paradigms
- What makes a good PL?

Formal Language Specification

- Specification vs. Implementation
- Specification
 - Syntax (formal)
 - Semantics (informal)
- Properties of Good Syntax
- Lexical Rules
- Syntactic Structure
- Grammars
- Chomsky Hierarchy
- Regular Languages - Regular Expressions
- Context-Free Grammars (CFGs)
- Limitations of each
- BNF
- EBNF
- Parse Trees and Derivations
- Syntactic Ambiguity (grammar, sentence wrt grammar)
 - change language (e.g., delimiters)
 - change grammar (e.g., associativity, precedence)
- Dealing w/ Ambiguity
 - change language (e.g., delimiters)
 - change grammar (e.g., associativity, precedence)
- Implementation
- Parsing Techniques
- Other Applications

Functional Programming

- Pure functional languages:
 - Referential transparency
 - No assignment
 - No iteration, only recursion
 - Implicit storage management (garbage collection)
 - Functions are values
- λ -calculus
- LISP, Common LISP, Scheme
- Useful Built-In Procedures
- Lists (cons cells, proper/improper)
- Read-eval-print loop
- Inhibiting + Activating evaluation (quote, eval)
- Writing recursive procedures (cdr, car-cdr recursion)
- Procedure definition and lambda expressions
- Conditionals (if, cond)
- Equality Checking (eq?, =, equal?, eqv?)
- Recursion (practice, practice)
- Efficiency Concerns
 - helper procedures
 - let, let*, ...
 - we didn't see accumulators
- Structured Data (e.g., trees)
- Higher-order procedures (map, apply, my-reduce)
- Passing Procedures, Returning Procedures
- Anonymous Procedures
- set!, syntactic forms