Final Review

CSC324 Winter 2007

## 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)
- Dealing w/ Ambiguity
  - change language (e.g., delimiters)
  - change grammar (e.g., associativity, precedence)
- Implementation
- Parsing Techniques (briefly)
- Other Applications

## Functional Programming

- Pure functional languages:
  - Referential transparency
  - No assignment
  - No iteration, only recursion
  - Implicit storage management (garbage collection)
  - Functions are values
- $\lambda$-calculus
- LISP, Common LISP, Scheme
- Built-In Procedures
- Lists (cons cells, proper/improper)
- Read-eval-print loop
- Inhibiting + Activating evaluation (quote, eval)
- Procedure definition and lambda expression

- Conditionals (if, cond)
- Equality Checking (eq?, =, equal?, eqv?)
- Recursion (practice, practice)
- Efficiency Concerns
  - helper procedures
  - let, let*, ...
  - accumulators
- Higher-order functions (map, apply, reduce)
- Passing Procedures, Returning Procedures
- Anonymous Procedures

## ML

- Definitions: typing, type safety, type checking, type inference, strongly typed language
- ML Features
- Declarations
- Pattern Matching
- Functions, including anonymous and recursive
- Exception Handling: user-defined, built-in
- Datatypes!
- Type Synonyms
- User-Defined Datatypes, Enumerated types, Variant types
- Recursive types, Mutual Recursive types
- Polymorphism

## Prolog

- Logic Programming
- Prolog vs. Scheme (relational vs. functional)
- Logic Programming vs. Prolog (nondeterministic vs. deterministic, e tc.)
- Prolog Syntax (Horn Clauses (w/ variables), Facts, translating from english to Prolog)
- Writing Recursive Predicates (e.g., family relations)
- Lists (internal representation (dot predicate), head/tail)
- Recursive Predicates for List Manipulation (including accumulators)
- Other Structures (functions, e.g., resistor, parse tree, double exa mples)
- How Prolog Works
  - Unification
  - Goal-Directed Reasoning

  - Rule Ordering
  - Backtracking DFS
- Improving Efficiency
  - Anonymous Variables
  - Accumulators
  - CUT
- Negation as Failure (NAF) (safety conditions, etc.)
- Arithmetics
- Cut (!)

## Procedural Language Design Issues

- Components of a procedure
  - name
  - parameters
  - body
  - optional result
- Parameter passing
  - pass by value
  - pass by result
  - pass by value-result
  - pass by reference
  - pass by name
- Aliasing through parameter passing
- Procedure Activations
- Stack frames
- Lexical scope

- Dynamic scope
- **NOT** Implementing scope with stack frames
- **NOT** Displays