CSE401: Midterm review

David Notkin
Autumn 2000

Scope

- Everything we covered on overview topics and (especially) on the front-end issues in a compiler

Overview

- Why we study compilers
- What a compiler is, what an interpreter is
- The structure of compilation (front-end, back-end, lexer, parser, etc.)
- Engineering issues in compilation

Lexing

- Overall approach
  - Define regular expressions for tokens
  - Convert regular expressions to NFAs
  - Convert NFAs to DFAs
    - Subset construction
  - Convert DFAs to efficient implementation
    - Two approaches
- You should be able to actually do each and every one of these steps
- Language design issues (whitespace, indenting, etc.)

Formal languages

- Alphabets, grammars, languages, productions, etc.
- Relationship of languages to automaton
  - You should understand this clearly for lexing and parsing, but for the higher levels in the hierarchy, you don't need to know the nitty-gritty details

Parsing

- The AST: what and why
  - Primary and central hierarchical representation of the program
- CFGs
  - Why they are different from regular expressions
  - Why this is necessary for parsing
  - Notation and terminology
    - Derivations, parsing, etc.
  - Ambiguity and ways to overcome it

Parsing

- Algorithms
  - Top-down vs. bottom-up
  - You need to know all details of top-down parsing
    - FIRST/FOLLOW and predictive parsing, etc.
    - Eliminating common prefixes, ambiguity, etc.
    - Recursive descent parsers
  - You need to know the basics of bottom-up
  - Know the notation: LL(k), LR(k), etc.

Semantic analysis

- Perform final legality checking of program
- Perform enough analysis to enable back-end

Symbol tables

- What they are
- What goes in them
- Why they are needed
- How to implement them
- How to structure them for block-structured languages

Static vs. dynamic scoping

- Why this matters to symbol tables

Types

- What are they
- A taxonomy of types
- How we represent them
  - Including records, arrays, procedures, etc.
- Type checking terminology
  - strong vs. weak
  - static vs. dynamic
  - structural vs. name equivalence
  - overloading vs. polymorphism
- Type checking strategy
- Type conversion and coercion, casting