CSE 110A - Fundamentals Of Compiler Design Day2

Elijah Hantman

Class Format

- 5:20 - 6:55 PM T/TH
- 10 mins of Q&A after class if needed

Class Schedule

1. Announcements, homeworks, tests. Announcements sent out by email from Canvas
2. Review last quiz.
3. Review of latest material
4. New Material

Office Hours

- Appointment Basis
- Google sheet (10 min slots)
- Link posted on Canvas soon
- Don't waste slots for no reason

Class Content

1. Regular languages and Expressions
2. Context free grammers and Parsing

   > Bachus Naur Form, way to formally describe context free grammer.

3. Intermediate Representations

   > Parse trees, converting complex expressions to simple IR.

4. Optimizations

   > Simple optimizations, data flow optimizations, etc.

> Schedule on Website

Assignments and Tests

- Assignments in Python
- Docker image used, don't rely too heavily on external libraries
- Must run on docker to be graded
- github classroom for automatic feedback
- May use around 3 pages of notes for exams

High Level Compiler Discussion Review

What is a compiler

> Input in a language
> Output in a different language

Lexical Analysis

- Introduction
  - Working with words we have various types of words

    * Nouns
    * Verbs
    * Articles
    * Preposition

    Before being able to understand language we first need to break it apart into individual pieces which have specific structure and roles

- Lexing

> We take the string and break it into tokens. We replace each part of the string with a structure which contains more information about its meaning.
>
> One way this manifests is in a symbol table, we can keep track of the name, type, etc. of what an identifier means as we lex in order to generate tokens which refer to repeated objects.

- How can we define a language?
  - ARTICLE: The, A, My, Your
  - NOUN: Dog, Car, Computer
  - VERB: Ran, Crashed, Accelerated
  - ADJECTIVE: Purple, Spotted, Old

  A sentence may be
  ARTICLE ADJECTIVE? NOUN VERB

> We can generate valid elements of our language by substituting the various non terminal symbols for other symbols until we are left with only terminal symbols.

- How do we expand this language?

> ARTICLE ADJECTIVE* NOUN VERB

  The star is known as the Kleene star, and means repeat zero or more times. It corrosponds to this rule:

> ADJECTIVE = NONE — ADJECTIVE ADJECTIVE

  As we can see we can either replace ADJECTIVE with NONE or we can replace it with ADJECTIVE ADJECTIVE, in addition to its other substitution rules.

  Stephen Cole Kleene formalized what a regular language was.

- Programs for Lexical Analysis

> Start with definitions of tokens. What is valid for that token and what is invalid. In C a valid identifier can begin with a letter or an underscore, and it contains any number of letters, underscores, and numbers.

> Soem programs can automatically generate Lexers. They are kinda cool but I personally am not particularly interested.