

Plan for Today

Motivation

- Why study compilers?

Programming Assignment Overview

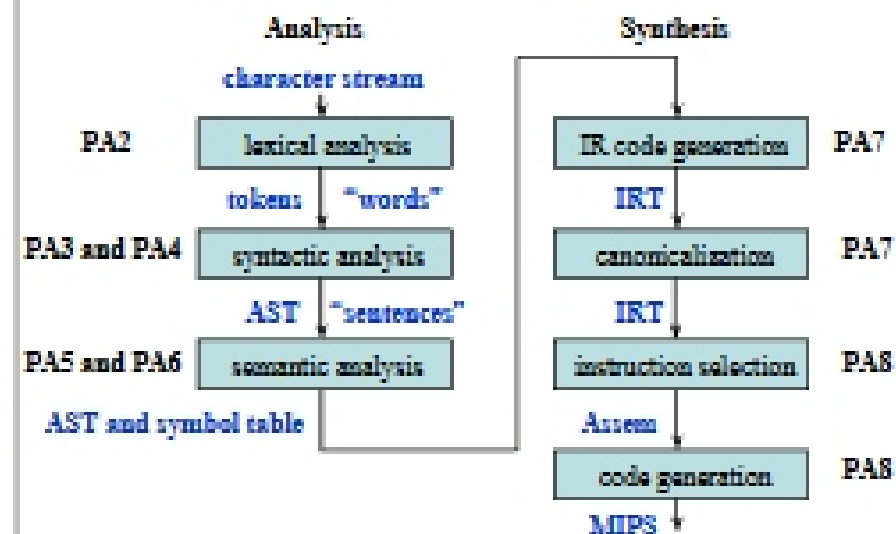
- The compiler we will be building

Lexing/Scanning

- Representing tokens with regular expressions
- Using DFAs to recognize a string of tokens

The screenshot shows a web interface for a course management system. At the top, there is a navigation bar with links like 'myWebCT', 'Resume Course', 'Course Map', 'Check Browser', 'Log Out', and 'Help'. Below this, the page title is 'CS 453 Introduction to Compiler Construction'. A 'Control Panel' is visible on the left, and a 'Designer Options' tab is active. The main content area is titled 'Calculation Editor: TotalPoints' and features a numeric keypad with buttons for digits 0-9, '+', '-', '*', '/', and '='. There are also buttons for 'Update' and 'Cancel'. The interface is designed for entering mathematical formulas.

Structure of the MiniJava Compiler



Specifying Tokens with SableCC

Theory meets practice

- Regular expressions, formal language, grammar, parsing...

SableCC example input file:

```

Package miniJava;

Helpers
  int = [0..9];
  char = ' ';

digit = [0..9];
letter = [a..z] | [A..Z];
underscore = '_';

not_asterisk = [^*];
not_asterisk_block = [^*]*;

c_comment = [^/*]*;
not_asterisk_block [^/*]*;
not_asterisk_block [^/*]*;

Tokens
  plus = '+';
  LP = '(';
  id = letter (letter | digit | underscore)*;
  block = [^* | * | *]*;
  comment = c_comment | line_comment;

Ignored Tokens
  blank,
  comment;

```

Example DFA for Recognizing a String of Tokens

```
digit = ['0'..'9'];  
letter = ['a'..'z'] | ['A'..'Z'];  
not_f = [letter - {'f'}];  
not_o = [letter - {'o'}];  
not_r = [letter - {'r'}];  
tokens  
for = 'for';  
id = letter (letter | digit | underscore)*;
```

