Lab1

Cassidy Heulings

February 13, 2026

1 *Crafting A Compiler*

1.1 Exercise 1.1

MOSS uses structural similarity in programs to detect plagiarism. Using this method, MOSS can then detect if two pieces of code are similar but are just using different variable names, reordered code, or spacing that has been reformatted.

MOSS uses a few methods to achieve this. First, it breaks code into tokens and normalizes them. This will replace identifiers with a general name, so any name differences will be eliminated. Additionally, comments and spacing are ignored in token streams, so any style choices that differ between the code will also be eliminated. Next, MOSS generates a sequence of hashes of k-grams. Finally, MOSS chooses some of these hashes as the code's fingerprint and compares it to other fingerprints and flags pairs that have a high number of matches.

MOSS is different from other approaches as it doesn't just search for matching text, but rather matching token patterns. Matching text will not catch plagiarism if variables are renamed and code is switched around. There are other effective methods like Abstract Syntax Tree comparisons, which, like MOSS, will look for structure of the code, but these are more expensive and require a full parser for every language. There is also semantic equivalent methods, which pay attention to similarity in the code's output after working through the code with sybol inputs. This will track the flow of data as the code runs. MOSS doesn't look for equivalence, it just looks for similar structural patterns.

1.2 Exercise 3.1

1. ID(main)
2. LPAREN
3. RPAREN
4. LBRACE
5. CONST
6. FLOAT
7. ID(payment)
8. ASSIGN
9. FLOAT_LITERAL(384.00)
10. SEMICOLON
11. FLOAT
12. ID(bal)
13. SEMICOLON
14. INT
15. ID(month)
16. ASSIGN
17. INT_LITERAL(0)
18. SEMICOLON
19. ID(bal)
20. ASSIGN
21. INT_LITERAL(15000)
22. SEMICOLON
23. WHILE
24. LPAREN
25. ID(bal)
26. GREATER_THAN
27. INT_LITERAL(0)

28. RPAREN
29. LBRACE
30. ID(printf)
31. LPAREN
32. STRING_LITERAL("Month: %2d Balance: %10.2f\n")
33. COMMA
34. ID(month)
35. COMMA
36. ID(bal)
37. RPAREN
38. SEMICOLON
39. ID(bal)
40. ASSIGN
41. ID(bal)
42. MINUS
43. ID(payment)
44. PLUS
45. FLOAT_LITERAL(0.015)
46. TIMES
47. ID(bal)
48. SEMICOLON
49. ID(month)
50. PLUS
51. INT_LITERAL(1)
52. SEMICOLON
53. RBRACE
54. RBRACE

Some tokens require a pointer to the symbol table. These are required by IDs so that the compiler does not make multiple copies of the same ID and so the compiler knows what ID it is pointing to.

Some tokens require a number value. These are integer literals and float literals. Without the numeric value, the scanner would only return it's type without the value.

Some tokens require a pointer to it's string storage. String literals require this pointer, so the compiler can hold the contents of the string and later on, be able to call back to it.

2 *Dragon*

2.1 Exercise 1.1.4

C is a great target language for a compiler due to several advantages. C has many compilers and tools for many platforms, making C reliable for producing machine code and able to be compiled on multiple different platforms. These existing compilers also are optimized to generate code efficiently and effectively. C is also very portable, meaning just about every architecture has a C compiler. A source-to-source compiler can use C as the intermediate language through standard C compilers to generate reliable translation.

2.2 Exercise 1.6.1

w = i + j

In the block w is in, i gets assigned 6 and j is a new integer declaration assigned to 7.

w = 6 + 7 = 13

Since j is a new integer declaration assigned to 7 in that block, the following code will use the initial declaration of j = 5 as j does not get reassigned going forward.

x = i + j

i has been reassigned in the previous block of code to 6, and j remains as 5.

x = 6 + 5 = 11

y = i + j

A new integer declaration for i is in the block y is assigned in, so for this block i = 8.

y = 8 + 5 = 13

z = i + j

i has a new integer declaration in the last block, and since this operation is outside that block, i is remains assigned to 6.

z = 6 + 5 = 11