Gianna Julio

CMPT432 - Design of Compilers
Lab 1

---

1. [***Crafting a Compiler* - Exercise 1.11**] The Measure Of Software Similarity (MOSS) [SWA03] tool can detect similarity of programs written in a variety of modern programming languages. Its main application has been in detecting similarity of programs submitted in computer science classes, where such similarity may indicate plagiarism (students, beware!). In theory, detecting equivalence of two programs is undecidable, but MOSS does a very good job of finding similarity in spite of that limitation. Investigate the techniques MOSS uses to find similarity. How does MOSS differ from other approaches for detecting possible plagiarism?

   MOSS differs from other code plagiarism detectors in that it supports a large number of languages, including C, C++, Java, C#, Python, Visual Basic, Javascript, FORTRAN, Lisp, Pascal, Ada, Perl, MIPS assembly, a8086 assembly, and more. Most other code comparison tools are built for one specific language. MOSS also uses an abstract syntax tree for comparisons so that the logical structure of the program can be compared and not just the text. MOSS also includes a reference database for known algorithms to ensure that using a common algorithm is not marked as plagiarism - most other plagiarism checkers for code do not have reference databases. In addition, MOSS is built the handle large-scale programs and allows the user to customize the similarity thresholds for if a program is marked as similar or not. This is useful when comparing assignments that may have the same general structure due to the nature of the assignment. MOSS also generates an email report after completing a scan of the programs, a technique I have not heard of in other program plagiarism checkers.

   references: https://theory.stanford.edu/~aiken/moss/

2. [***Crafting a Compiler* - Exercise 3.1**] Assume the following text is presented to a C scanner. What token sequence is produced? For which tokens must extra information be returned in addition to the token code?

```
main(){
    const float payment = 384.00;
    float bal;
    int month = 0;
    bal=15000;
    while (bal>0){
        printf("Month: %2d  Balance: %10.2f\n", month, bal);
        bal=bal-payment+0.015*bal;
        month=month+1;
    }
}
```

The tokens produced would be as follows:

```
IDENTIFIER: main
LEFT_PAREN: (
RIGHT_PAREN: )
LEFT_BRACE: {
CONST: const
FLOAT: float
IDENTIFIER: payment
ASSIGN: =
FLOAT_CONSTANT: 384.00
SEMICOLON: ;
FLOAT: float
IDENTIFIER: bal
SEMICOLON: ;
INT: int
IDENTIFIER: month
ASSIGN: =
INT_CONSTANT: 0
SEMICOLON: ;
IDENTIFIER: bal
ASSIGN: =
INT_CONSTANT: 15000
SEMICOLON: ;
WHILE: while
LEFT_PAREN: (
IDENTIFIER: bal
GREATER_THAN:
INT_CONSTANT: 0
RIGHT_PAREN: )
LEFT_BRACE: {
IDENTIFIER: printf
LEFT_PAREN: (
STRING_LITERAL: "Month:  %2d Balance:  %10.2f\n"
COMMA: ,
IDENTIFIER: month
COMMA: ,
IDENTIFIER: bal
RIGHT_PAREN: )
SEMICOLON: ;
IDENTIFIER: bal
ASSIGN: =
IDENTIFIER: bal
MINUS: -
IDENTIFIER: payment
PLUS: +
FLOAT_CONSTANT: 0.015
ASTERISK: *
```

```
IDENTIFIER: bal
SEMICOLON: ;
IDENTIFIER: month
ASSIGN: =
IDENTIFIER: month
PLUS: +
INT_CONSTANT: 1
SEMICOLON: ;
RIGHT_BRACE: }
RIGHT_BRACE: }
```

For identifier, constant, and string literal tokens, extra information (lexemes) need to be returned in addition to the token itself.

3. [***"Dragon" Textbook* - Exercise 1.1.4**] A compiler that translates a high-level language into another high-level language is called a source-to-source translator. What advantages are there to using C as a target language for a compiler?

A source-to-source translator with C as its target language will benefit from all of the advantages of C as a language. C is a highly portable language, available for a broad range of different platforms, making the translated code portable across different systems without major modifications. C is also already known for being a highly efficient language because of its low-level feature, so the generated code that it likely also optimized in the translation process is likely highly efficient in terms of both time and space. Additionally, C also has a wide variety of tools, libraries, and frameworks that can be utilized by the translated code. Many systems also already run on existing C code or legacy applications written in C, so a source-to-source translator with a target language of C can be utilized to integrate programs written in other languages into existing programs.

4. [***"Dragon" Textbook* - Exercise 1.6.1**] For the block-structured C code if Fig 1.13(a), indicate the values assigned to w, x, y, and z.

```
w = (6 + 7) = 13
x = (6 + 5) = 11
y = (8 + 5) = 13
z = (6 + 5) = 11
```

```
int w, x, y, z;
int i = 4; int j = 5;
{   int j = 7;
    i = 6;
    w = i + j;
}
x = i + j;
{   int i = 8;
    y = i + j;
}
z = i + j;
```