Lab One

Nicholas Fiore

Nicholas.Fiore1@Marist.edu

February 3, 2024

1 Exercises

1.1 Crafting a Compiler

1.1.1 1.11 (MOSS)

MOSS uses document fingerprinting, specifically using k-gram hashing. It takes a few steps to process the document this way, first by removing the whitespace, then creating the hashes based on the selection of k-grams (k being the size of the string of characters), then selects a subset of those hashes to use as the "fingerprints", and then compares documents with those fingerprints. This process levies a much smaller computational tax than a naive approach of scanning the entire document.

Fingerprint selection is done by an algorithm called winnowing. A high level understanding of the algorithm is that it creates windows of the hash values, and selects the lowest hash value in each row as long as it has not been recorded already.

These methods of copy-detection allow MOSS to detect the slight modifications (variable names, comments, etc.) while still seeing similarities between the programs structurally. Unlike systems that read entire documents or just compare them for exact copies, MOSS is able to find subtler changes and attempts to obfuscate stolen code.

All information sourced from here: https://yangdanny97.github.io/blog/2019/05/03/MOSS

1.1.2 3.1 (token sequence)

```
KEY_MAIN [main] found at (1:1)
SYM_L_PAREN [(] found at (1:5)
SYM_R_PAREN [)] found at (1:6)
SYM_L_BRACE [{] found at (1:7)
KEY_CONST [const] found at (2:5)
F_TYPE [float] found at (2:11)
ID [payment] found at (2:17)
SYM_ASSIGN [=] found at (2:25)
NUM_REAL [384.00] found at (2:27)
SYM_END_STATE [;] found at (2:33)
F_TYPE [float] foudn at (3:5)
```

```
ID [bal] found at (3:11)
SYM_END_STATE [;] found at (3:14)
I_TYPE [int] found at (4:5)
ID [month] found at (4:9)
SYM_ASSIGN [=] found at (4:15)
NUM_DIGIT [0] found at (4:17)
SYM_END_STATE [;] found at (4:18)
ID [bal] found at (5:5)
SYM_ASSIGN [=] found at (5:8)
NUM_DIGIT [15000] found at (5:9)
SYM_END_STATE [;] found at (5:14)
KEY_WHILE [while] found at (6:5)
SYM_L_PAREN [(] found at (6:11)
ID [bal] found at (6:12)
SYM_GREATER_THAN [>] found at (6:15)
NUM_DIGIT [0] found at (6:16)
SYM_R_PAREN [)] found at (6:17)
SYM_L_BRACE [{] found at (6:18)
ID [printf] found at (7:9)
SYM_L_PAREN [(] found at (7:15)
SYM_QUOTE ["] found at (7:16)
CHAR [M] found at (7:17)
...
CHAR [\n] found at (7:44)
SYM_QUOTE ["] found at (7:45)
SYM_COMMA [,] found at (7:46)
ID [month] found at (7:48)
SYM_COMMA [,] found at (7:53)
ID [bal] found at (7:55)
SYM_R_PAREN [)] found at (7:58)
SYM_END_STATE [;] found at (7:59)
ID [bal] found at (8:9)
SYM_ASSIGN [=] found at (8:12)
ID [bal] found at (8:13)
SYM_MINUS [-] found at (8:16)
ID [payment] found at (8:17)
SYM_PLUS [+] found at (8:24)
NUM_REAL [0.015] found at (8:25)
SYM_MULTI [*] found at (8:30)
ID [bal] found at (8:31)
SYM_END_STATE [;] found at (8:34)
ID [month] found at (9:9)
SYM_ASSIGN [=] found at (9:14)
ID [month] found at (9:15)
SYM_PLUS [+] found at (9:20)
NUM_DIGIT [1] found at (9:21)
SYM_END_STATE [;] found at (9:22)
SYM_R_BRACE [}] found at (10:5)
SYM_R_BRACE [}] found at (11:1)
```

ID names and values (such as strings or numbers) need to be sent along with the token code.

1.2 Dragon

1.2.1 1.1.4 (advantages of C as a target language)

C is the ancestor of a lot of modern high-level languages (Java, C++, Python (Alan's favorite)), and so they are structurally very similar to C, which can simplify building a compiler to C. Furthermore, C is a much lower-level language than modern languages, which gives it the advantage of requiring less processing to reach machine-level code (or interpretation) (https://www.bestcolleges.com/bootcamps/guides/java-vs-c/).

1.2.2 1.6.1 (VARIABLES IN BLOCK-STRUCTURED CODE)

Indicate the values assigned to w, x, y, and z.

```
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;
```

$w = 13$

$x = 11$

$y = 13$

$z = 11$