Homework module #2: Compiler Technology

Aleksander Vognild Burkow

February 5, 2014

Part I
Theory

1 Regular languages

a) The following is a NFA implementation of the regular expression. **??**

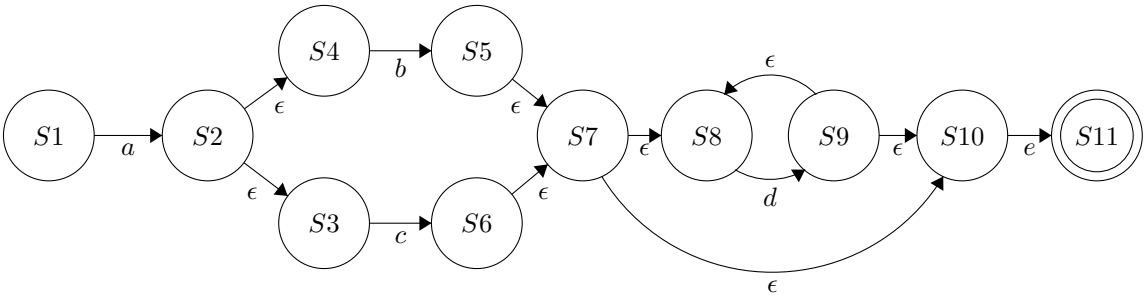

Figure 1: NFA

b) The converted NFA is presented below. **??**

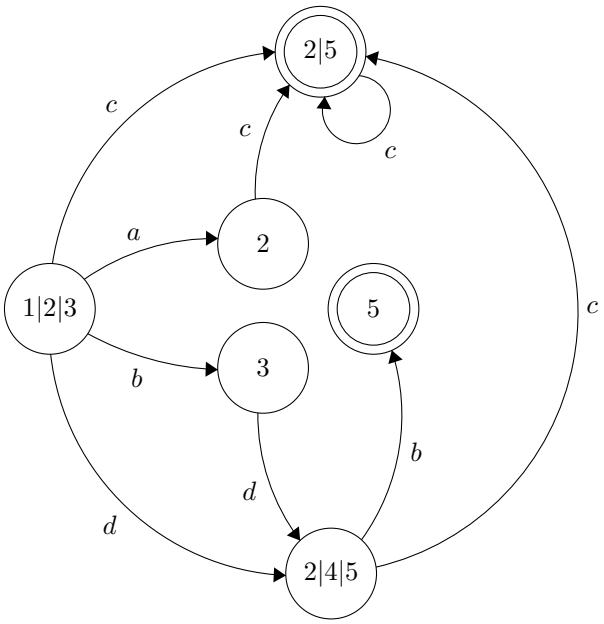

Figure 2: DFA

c) The program will be able to remove simple while loops without any nesting inside.

Issues will arise in the following cases:

i Nested statements:

```
while (COND) {
    if (1) {
    } // End will be matched
}
```

Flex will match the end of the while with the end of the if statement.

ii Do while constructs:

```
do {
    ...
} while (COND);
if (3.14) {
    // Will be matched here
}
```

The end of the do while statement be matched with the end if the if. The do part remains intact.

iii Strings:

```
while (true) {
    char omfg[] = "do not touch my while {";
}
```

Flex will mistakenly demolish the string.

The correct way to remove loops would be to ignore the node at the parser level. The lexer has no idea about the semantic meanings of the tokens may encounter, and cannot decide if the matched brackets belong to a while statement or some poor soon-to-be dismembered if statement.

Implementing removal of while-loops in bison (by ignoring the node) will ensure only well formed while-loops will be removed.

2 Grammars

a) An ambigous grammar is a formal grammar which allows for more than one leftmost derivation for any expression.

b) Ambiguity is an undecidable problem, but if we study the string *mvmp* we see that two productions arise:

i S = NvN (Use S = NvN)
NvN = mvN (Use N = m)
mvN = mvmp (Use N = mp)
mvmp

ii S = Sp (Use S = NvN)
Sp = NvNp (Use S = NvN)
NvNp = mvmp (Use N = m)
mvmp

Here we see two different parse trees for the same production. We conclude the grammar is ambigous.

c) An left-recursive grammar is a grammar in which there exists a non-terminal where one of its productions (transitive or immediate) has itself as the leftmost symbol.

d) The grammar is left recursive. This is seen by the fact that we can produce Sp from S.