Xiangrui Kong

HW1

1.1

The input /\*aa/\*\*\*/ is accepted by the DFA.

States visited: 1, 2, 3, 3, 3, 3, 4, 4, 4, 5

1.2

The input /\*\*\*a\*/\*/ is not accepted by DFA.

The error will be detected at state 5. The states visited is 1, 2, 3, 4, 4, 3, 4, 5. (chars consumed /\*\*\*a\*/) And then the input will stuck at state 5, since there no way to consume the next input character \*.

2.

**0**

**1**

**1**

**0**

**0**

**2**

**2**

**1**

**2**

3.

“/”“\*” ([^\*] | “\*”+[^\*/])\* “\*”“/”

The logic behind this regular expression is the middle part besides of the beginning and end of the comment (“/\*”, “/\*”), is any string without the substring “/\*”, thus there are 2 cases, one without “\*” at all, one with “\*” but not directly follows by “/”.

4.

<prog> -> (rule 10) L

-> (rule 8’) S L’

-> (rule 1) ID = E; L’

-> (rule 4’) ID = F E’; L’

-> (rule 6) ID = ID E’; L’

-> (rule 5’’) ID = ID + F; L’

-> (rule 7) ID = ID + num; L’

-> (rule 9’’) ID = ID + num; L

-> (rule 8’) ID = ID + ID; S L’

-> (rule 2) ID = ID + ID; write (ID); L’

-> (rule 9’) ID = ID + ID; write (ID);

Since x, y, a are IDs, the derivation equals to the input “x = a+2; write ( y );”.

5.

The first leftmost derivation:

A -> (rule 1) - A

-> (rule 2) - A – id

-> (rule 3) - id – id

The second leftmost derivation:

A -> (rule 2) A – id

-> (rule 1) – A – id

-> (rule 3) - id – id

we have more than one leftmost derivation for the same input, thus the grammar is ambiguous.

6.

The grammar is ambiguous.

An example: If(Stmt) If(Stmt) else Stmt

The first leftmost derivation:

S -> Stmt

-> IfStmt

-> (rule 4) if (Expr) Stmt

-> (rule 2) if (Expr) IfStmt

-> (rule 5) if (Expr) if (Expr) else Stmt

The second leftmost derivation:

S -> Stmt

-> IfStmt

-> (rule 5) if (Expr) Stmt else Stmt

-> (rule 2) if (Expr) IfStmt else Stmt

-> (rule 4) if (Expr) if (Expr) else Stmt