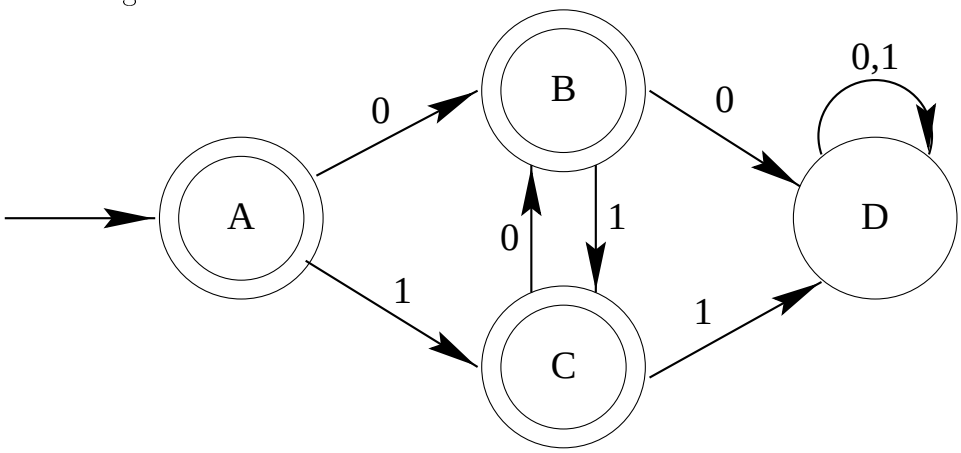
CMSC 330 Exam 1: Oct. 6, 1999

1. Give an unambiguous grammar for the language:
   L = { $w \mid w \in \{0,1\}^*$ and neither 00 nor 11 is a substring of w }

   Generate the grammar from this DFA:



   And the grammar is:

$$A' \rightarrow A \mid \epsilon$$

$$A \rightarrow 0B \mid 1C \mid 0 \mid 1$$
$$B \rightarrow 0D \mid 1C \mid 1$$
$$C \rightarrow 0B \mid 1D \mid 0$$
$$D \rightarrow 0D \mid 1D$$

   But all the productions that have a D in them can be eliminated, since they can never generate a string of all terminals (that's what the dead state in the DFA means). So a simpler grammar is:

$$A' \rightarrow A \mid \epsilon$$

$$A \rightarrow 0B \mid 1C \mid 0 \mid 1$$
$$B \rightarrow 1C \mid 1$$
$$C \rightarrow 0B \mid 0$$

2. Show a regular expression that generates all strings in the language:
   L = { $w \mid w \in \{a,b\}^*$ and w has at most one set of three consecutive b's }

$$(a \mid ba \mid bba)^* \ (b \mid bb \mid bbb \mid \epsilon) \ (a \mid ab \mid abb)^*$$

3. Prove that the following grammar is ambiguous.

$$S \rightarrow W \mid X$$

$$W \rightarrow 0W3 \mid 0U3$$

$$U \rightarrow 1U2 \mid 12$$

$$X \rightarrow YZ$$

$$Y \rightarrow 0Y1 \mid 01$$

$$Z \rightarrow 2Z3 \mid 23$$

The way to show a grammar is ambiguous is to show two leftmost (or rightmost) derivations for the same string. For this grammar, the easiest string to show is 0123:

$$S \Rightarrow X \Rightarrow YZ \Rightarrow 01Z \Rightarrow 0123$$

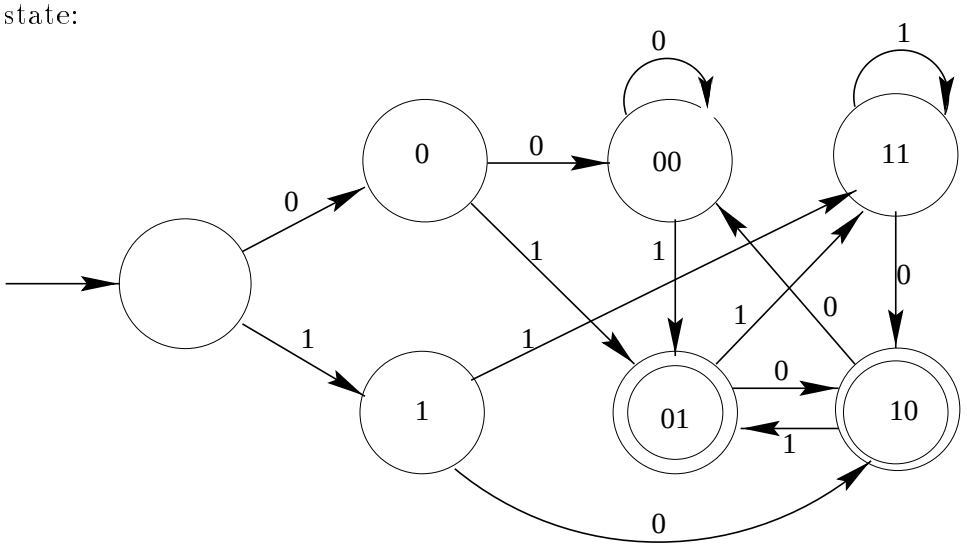
or

$$S \Rightarrow W \Rightarrow 0U3 \Rightarrow 0123$$

4. Produce a deterministic finite automaton (DFA) that recognizes the language:
L = $\{ w \mid w \in \{0,1\}^*$ and the last two symbols of w include one 0 and one 1 $\}$

Note that L contains no strings of length less than 2.

The label in the state shows the last or last 2 symbols seen in the string upon reaching that state:



5. Write an unambiguous context free grammar for Boolean expressions with operands represented by <id> and operators $\oplus$ (XOR), ! (NOT) and $\Rightarrow$ (IMPLIES). $\oplus$ and $\Rightarrow$ are infix binary operators and ! is a prefix unary operator. ! has highest precedence, $\oplus$ has next highest precedence, and $\Rightarrow$ has the lowest precedence. For the binary operators, $\oplus$ has left associativity and $\Rightarrow$ has right associativity. Parentheses are used to override precedence and associativity (i.e. if B is a boolean expression, then so is (B))

   Let B be the start symbol.

   First, rules for the lowest precedence operator, with right associativity:

$$B \rightarrow C \Rightarrow B \mid C$$

   Second, rules for the middle precedence operator, with left associativity:

$$C \rightarrow C \oplus D \mid D$$

   Last, rules for the highest precedence operator, for generating operands, and for parenthesized expressions:

$$D \rightarrow !D \mid (B) \mid < id >$$