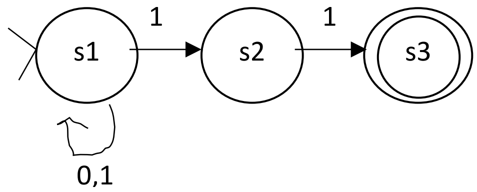
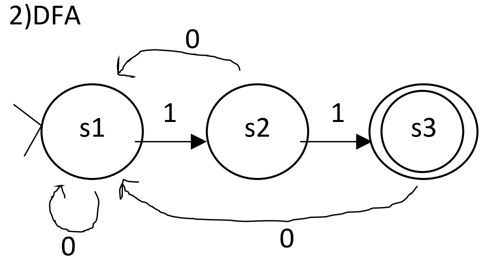
CS 314 HW 2

1.

1)NFA



2)DFA



1)NFA

	1	0
s1	s1, s2	s1
s2	s3	
s3		

S={s1,s2,s3}

s=s1

F=s3

T= (s1,1)↦s1 , (s1,1)↦s2, (s1,0)↦s1 ,(s2,1)↦s3

2)DFA

	1	0
s1	s2	s1
s2	s3	s1
s3	s3	s1

S={s1,s2,s3}

s=s1

F=s3

T= (s1,1)↦s2 , (s1,0)↦s1, (s2,1)↦s3 ,(s2,0)↦s1, (s3,1)↦s3 ,(s3,0)↦s1

2.

1)

<S>::=<A><B><C>

<A>::=<A>a | b

<B>::=<B>b | b

<C>::= <C>c | c

it has a regular expression -> a*$b^{+}c^{+}$

2)

<S>::=<A><B><C>

<A>::=a<A>b | ε

<B>::=<B>b | b

<C>::= <C>c | c

3)

Not a CFG because we cannot use pushdown automata for this language. we cannot put same amount of a, b and c (we cannot guarantee that we have same amount of b or c follow by a)

4)

<S>::=<A>

<A>::=00<A>111 | ε

5)

<S>::=<A>

<A>::= a<A>a | b<A>b | ε

6)

<S>::=<AD>

<AD>::=<BC> | a<AD>d | ε

<BC> ::=b<BC>c | ε

7)

Not a CFG because in pushdown automata we cannot guarantee that the grammar can generate same number of alternative characters.

8)

<S>::= <AB><CD>

<AB>::=a<AB>b | ε

<CD>::=c<CD>d| ε

9)

<S>::=<A><B>

<A>::=a<A>a | ε

<B>::=b<B>b | ε

10)

<S>::=<A><A><A><A><B>

<A>::=a | b

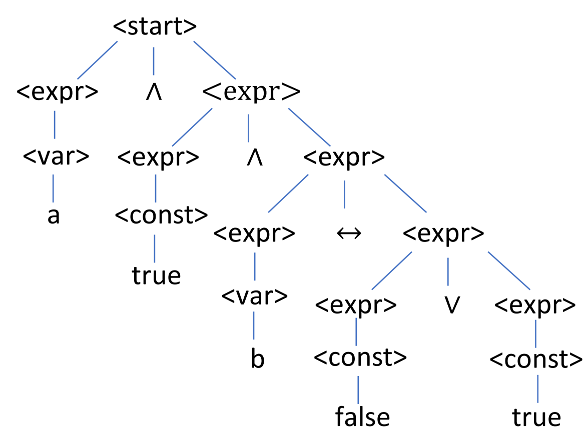
<B>::= ε | a<B> | b<B>

it has regular expression -> (a|b) (a|b) (a|b) (a|b) (a|b)*

3.

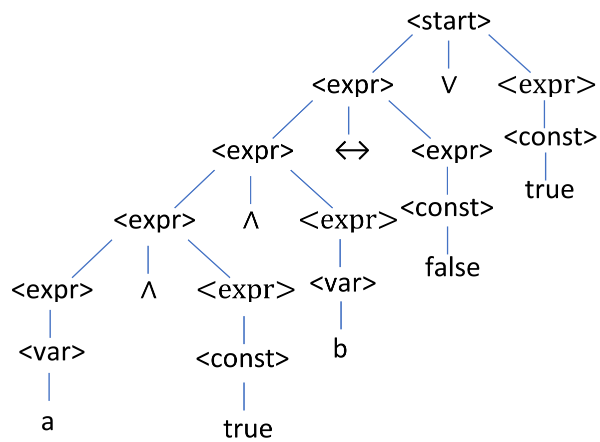
1. Leftmost derivation
   <start>
   =>$_L$ <expr>
   =>$_L$ <expr>∧<expr>
   =>$_L$ <var>∧<expr>
   =>$_L$ a ∧<expr>
   =>$_L$ a ∧<expr>∧<expr>
   =>$_L$ a ∧<const>∧<expr>
   =>$_L$ a ∧ true ∧<expr>
   =>$_L$ a ∧ true ∧<expr>↔<expr>
   =>$_L$ a ∧ true ∧<var>↔<expr>
   =>$_L$ a ∧ true ∧ b ↔<expr>
   =>$_L$ a ∧ true ∧ b ↔<expr>∨<expr>
   =>$_L$ a ∧ true ∧ b ↔<const>∨<expr>
   =>$_L$ a ∧ true ∧ b ↔ false ∨<expr>
   =>$_L$ a ∧ true ∧ b ↔ false ∨<const>
   =>$_L$ a ∧ true ∧ b ↔ false ∨ true

2. leftmost derivation parse tree

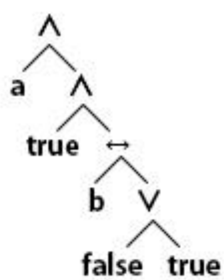


Rightmost derivation
<start>
=>R <expr>
=>R <expr>∨<expr>
=>R <expr>∨<const>
=>R <expr>∨ true
=>R <expr>↔ <expr>∨ true
=>R <expr>↔ <const>∨ true
=>R <expr>↔ false ∨ true
=>R <expr>∧<expr>↔ false ∨ true
=>R <expr>∧<var>↔ false ∨ true
=>R <expr>∧ b ↔ false ∨ true
=>R <expr>∧<expr> ∧ b ↔ false ∨ true
=>R <expr>∧<const> ∧ b ↔ false ∨ true
=>R <expr>∧ true ∧ b ↔ false ∨ true
=>R <var>∧ true ∧ b ↔ false ∨ true
=>R a ∧ true ∧ b ↔ false ∨ true

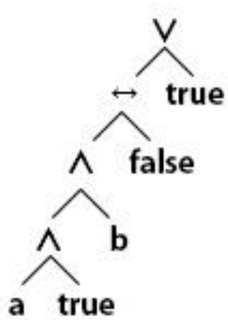
Rightmost derivation parse tree



3. AST
Leftmost



Rightmost



4. Since our leftmost parse tree and rightmost parse tree is distinct but produce same language, the grammar is ambiguous.

5. <start>::=<ex>
<ex>::=<ex>↔<orex> | <orex>
<orex>::=<orex>∨<andex>|<andex>
<andex>::=<term>∧<andex>|<term>
<term>::=<const>|<var>
<const>::=true|false
<var>::=a|b|c|⋯|z

6.