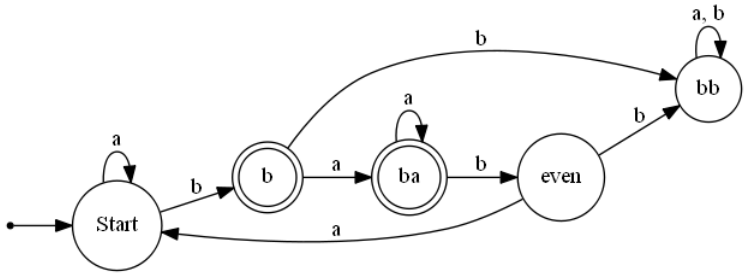
CS2200 Homework 5

Evan Wilcox

Due April 2, 2019

1. Find a deterministic finite-state automaton that recognizes the language, L, consisting of all strings in {a, b}* that contain an odd number of b's such that there is at least one a between every two b's in the string.



File: prob1.ndfsa

```
A ab
S start, 0
S b, 1
S ba, 1
S bb, 0
S even, 0
B start
D start, a, start
D start, b, b
D b, a, ba
D b, b, bb
D bb, a, bb
D bb, b, bb
D ba, a, ba
D ba, b, even
D even, a, start
D even, b, bb
T a
O Rejected
T b
O Accepted
T ab
O Accepted
T ba
O Accepted
T aba
O Accepted
T bab
O Rejected
T abaaaaba
O Rejected
T baaaaababa
O Accepted
T bb
O Rejected
T abbabbaba
O Rejected
```

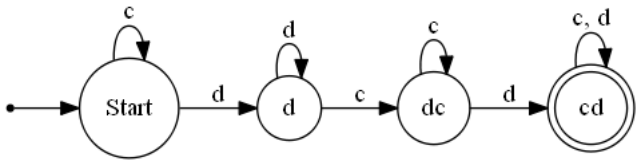
File: prob1.gv

```
digraph finite_state_machine {
  rankdir=LR;
  _ize="8,5"

  node [shape = point] x
  node [shape = circle] Start
  node [shape = doublecircle] b
  node [shape = doublecircle] ba
  node [shape = circle] bb
  node [shape = circle] even

  x -> Start
  Start -> Start [label = "a"]
  Start -> b [label = "b"]
  b -> ba [label = "a"]
  b -> bb [label = "b"]
  ba -> ba [label = "a"]
  ba -> even  [label = "b"]
  even -> Start [label = "a"]
  even -> bb  [label = "b"]
  bb -> bb [label = "a, b"]
}
```

2. Construct a regular expression that generates the language, L, defined in Problem 1.

$$L = a^*b((aa^*b)(aa^*b))^*a^*$$

3. Let $A = \{c, d\}$. Let $L_2 \subseteq A^*$ be the language consisting of all strings not in the language, L_3, which is generated by the regular expression $c^*d^*c^*$. Find a deterministic finite-state automaton that recognizes L_2.



File: prob3.ndfsa

```
A cd
S Start, 0
S d, 0
S dc, 0
S cd, 1
B Start
D Start, c, Start
D Start, d, d
D d, c, dc
D d, d, d
D dc, c, dc
D dc, d, cd
D cd, c, cd
D cd, d, cd
T
O Rejected
T c
O Rejected
T dc
O Rejected
T cdc
O Rejected
T ccccddddccccc
O Rejected
T ccccddddccccddddd
O Accepted
T cdcdcdc
O Accepted
T dddccdcdccddcdcdcdcdc
O Accepted
T ccccccddddddddccccccddddddddccccccccddddd
O Accepted
T dddddcdcdcdcdcddddddddddccccccccc
O Accepted
```

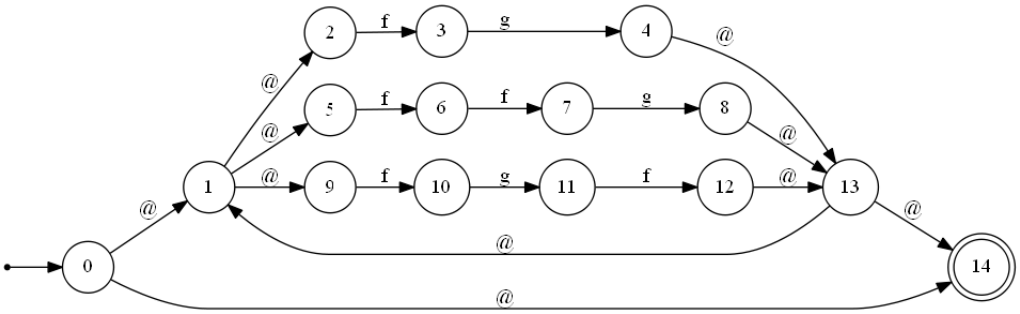
File: prob3.gv

```
digraph finite_state_machine {
 rankdir=LR;
 _ize="8,5"

 node [shape = point] x
 node [shape = circle] Start
 node [shape = circle] d
 node [shape = circle] dc
 node [shape = doublecircle] cd

 x -> Start
 Start -> Start [label = "c"]
 Start -> d [label = "d"]
 d -> dc [label = "c"]
 d -> d [label = "d"]
 dc -> dc [label = "c"]
 dc -> cd [label = "d"]
 cd -> cd [label = "c, d"]
}
```

4. Find a regular expression that generates the language, L_2, defined in Problem 3.

$$L_2 = c^*dd^*cc^*d(c+d)^*$$

5. Let $L_3 \subseteq \{f, g\}^*$ be the language of all strings generated by the regular expression $(fg + ffg + fgf)^*$. Construct a non-deterministic finite-state machine that recognizes L_3.



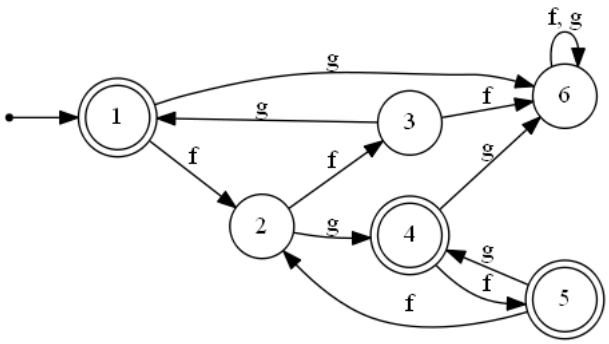
File: prob5.ndfsa

```
A fg
S 0, 0
S 1, 0
S 2, 0
S 3, 0
S 4, 0
S 5, 0
S 6, 0
S 7, 0
S 8, 0
S 9, 0
S 10, 0
S 11, 0
S 12, 0
S 13, 0
S 14, 1
B 0
D 0, @, 1
D 0, @, 14
D 1, @, 2
D 1, @, 5
D 1, @, 9
D 2, f, 3
D 3, g, 4
D 4, @, 13
D 5, f, 6
D 6, f, 7
D 7, g, 8
D 8, @, 13
D 9, f, 10
D 10, g, 11
D 11, f, 12
D 12, @, 13
D 13, @, 1
D 13, @, 14
T
O Accepted
T f
O Rejected
T g
O Rejected
T fg
O Accepted
T ffg
O Accepted
T fgf
O Accepted
T fgffgfgffg
O Accepted
T ffggffg
O Rejected
T ffgfggfgf
O Rejected
T fgfgfffg
O Accepted
```

File: prob5.gv

```
digraph finite_state_machine {
  rankdir=LR;
  _ize="8,5"

  node [shape = point] x
  node [shape = circle] 0
  node [shape = circle] 1
  node [shape = circle] 2
  node [shape = circle] 3
  node [shape = circle] 4
  node [shape = circle] 5
  node [shape = circle] 6
  node [shape = circle] 7
  node [shape = circle] 8
  node [shape = circle] 9
  node [shape = circle] 10
  node [shape = circle] 11
  node [shape = circle] 12
  node [shape = circle] 13
  node [shape = doublecircle] 14

  x -> 0
  0 -> 1 [label = "@"]
  0 -> 14 [label = "@"]
  1 -> 2 [label = "@"]
  1 -> 5 [label = "@"]
  1 -> 9 [label = "@"]
  2 -> 3 [label = "f"]
  3 -> 4 [label = "g"]
  4 -> 13 [label = "@"]
  5 -> 6 [label = "f"]
  6 -> 7 [label = "f"]
  7 -> 8 [label = "g"]
  8 -> 13 [label = "@"]
  9 -> 10 [label = "f"]
  10 -> 11 [label = "g"]
  11 -> 12 [label = "f"]
  12 -> 13 [label = "@"]
  13 -> 1 [label = "@"]
  13 -> 14 [label = "@"]
}
```

6. Using the subset construction described in class, construct a deterministic finite-state machine based on the non-deterministic finite-state machine that you constructed in Problem 5.



File: prob6.ndfsa

```
A fg
S 1, 1
S 2, 0
S 3, 0
S 4, 1
S 5, 1
S 6, 0
B 1
D 1, f, 2
D 1, g, 6
D 2, f, 3
D 2, g, 4
D 3, f, 6
D 3, g, 1
D 4, f, 5
D 4, g, 6
D 5, f, 2
D 5, g, 4
D 6, f, 6
D 6, g, 6
T
```

```
O Accepted
T f
O Rejected
T g
O Rejected
T fg
O Accepted
T ffg
O Accepted
T fgf
O Accepted
T ffggfgf
O Rejected
T fgfgf
O Accepted
T ffgfggfgf
O Rejected
T fgfgfffg
O Accepted
```

File: prob6.gv

```
digraph finite_state_machine {
  rankdir=LR;
  _ize="8,5"

  node [shape = point] x
  node [shape = doublecircle] 1
  node [shape = circle] 2
  node [shape = circle] 3
  node [shape = doublecircle] 4
  node [shape = doublecircle] 5
  node [shape = circle] 6

  x -> 1
  1 -> 2 [label = "f"]
  1 -> 6 [label = "g"]
  2 -> 3 [label = "f"]
  2 -> 4 [label = "g"]
  3 -> 6 [label = "f"]
  3 -> 1 [label = "g"]
  4 -> 5 [label = "f"]
  4 -> 6 [label = "g"]
  5 -> 2 [label = "f"]
  5 -> 4 [label = "g"]
  6 -> 6 [label = "f, g"]
}
```

7. Let G be the context-free grammar $\{\{Q, V\}, \{c, d\}, \{Q \to QQ|V, V \to cVd|cd\}, Q\}$.

   (a) Describe the language $L_6 = L(G)$. Prove that G is ambiguous.

   (b) Give an unambiguous grammar H, such that $L(H) = L(G)$. Give some justification for the claim that H is unambiguous.

8. Consider the CFG $G = \{\{S\}, \{a, b\}, \{S \rightarrow aSbS|bSaS|\epsilon\}, S\}$. Prove by induction that $L(G) = \{w \in \{a, b\}^* |$ the number of a's in $w =$ the number of b's in $w\}$.

   Proof 1

   (a) **Define the problem.**
   Prove that a string generated by the CFG $G = \{\{S\}, \{a, b\}, \{S \rightarrow aSbS|bSaS|\epsilon\}, S\}$ has the same number of a's as it does b's.

   (b) **Base case and two more.**
   The base case for this CFG would be substituting the empty string ϵ for S. In the empty string there is an equal amount of a's and b's, both zero. When you substitute S for $aSbS$ then substitute the remaining S's for the empty string you end up with the string 'ab' which contains an equal amount of a's and b's. If you substitute S for $bSaS$ then substitute the remaining S's with the empty string you get the string 'ba' which has an equal amount of a's and b's.

   (c) **Inductive case.**
   When you substitute S you have three options: $aSbS$ which adds 1 a and 1 b to the string, $bSaS$ which adds 1 a and 1 b to the string, and the empty string which adds no a's or b's. Because its not possible to add an a without a b or a b without an a when substituting S, the amount of a's and b's in the string will always be equal.

   (d) **Conclusion.**
   After showing the previous two steps, we can conclude that strings generated byt the CFG $G = \{\{S\}, \{a, b\}, \{S \rightarrow aSbS|bSaS|\epsilon\}, S\}$ will have equal amounts of a's and b's.

   Proof 2

   (a) **Define the problem.**
   Prove that a string made of a's and b's that has an equal number of a's and b's can be generated by the CFG $G = \{\{S\}, \{a, b\}, \{S \rightarrow aSbS|bSaS|\epsilon\}, S\}$.

   (b) **Base case and two more.**
   The base case would be to generate the empty string. The empty string contains an equal amount of a's and b's and can be generated directly be S. To generate the string 'ab' you can substitute S for $aSbS$ then substitute S for the empty string. To generate the string 'abab' you must substitute S for $aSbS$ then substitute the first S for $bSaS$ and substitute the remaining S's for empty strings.

   (c) **Inductive case.**
   The CFG's rules allows you to create every permutation of a string made of a's and b's that contain an equal mount of a's and b's. The rules also allow you to chain these permutations.

   (d) **Conclusion.**
   After showing the previous two steps we can conclude that all strings made of a's and b's containing the same amount of a's and b's can be generated by the CFG $G = \{\{S\}, \{a, b\}, \{S \rightarrow aSbS|bSaS|\epsilon\}, S\}$.

9. Let G be a CFG as described. Using the algorithms discussed in class, convert G into an equivalent grammar that is in Chomsky Normal Form.

Vars = $\{A, B, C, D, E, F, G, H, I, J, P, Q, R, S, T, U.V.W.X, Y, Z\}$.
Alph = $\{a, b, c\}$.
Start = S.
Rules =
$S \to AD|YE|UF|UG|PH|AX|YB|PQ|BZ|WC|AT|a|BT|b|UC|c|UB$
$P \to AT|a$
$Q \to QT|AQ$
$T \to CT|c$
$U \to AU|a$
$X \to AX|a|AI|AB$
$R \to AI|AB$
$Y \to YB|b|AI|AB$
$V \to BJ|BC$
$W \to WC|c|BJ|BC$
$Z \to BZ|b|BJ|BC$
$A \to a$
$B \to b$
$C \to c$
$D \to XT$
$E \to BT$
$F \to BZ$
$G \to WC$
$H \to QT$
$I \to RB$
$J \to VC$