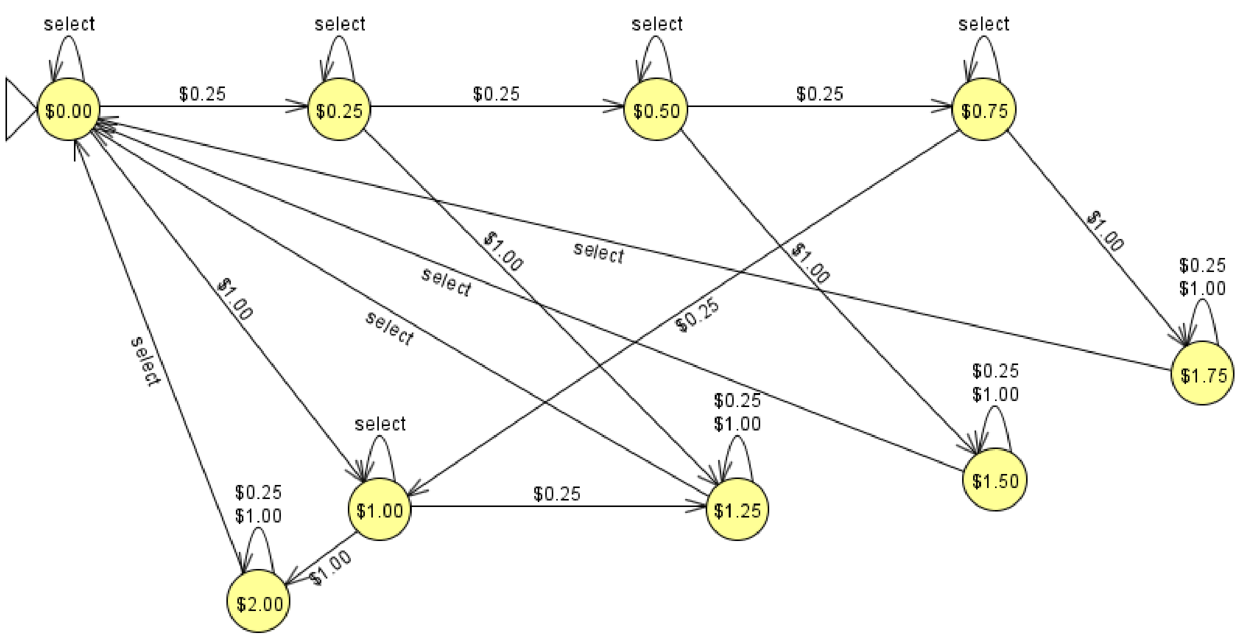
Midterm Questions

Question 1 (15 points): Complete the NFA (adding transition functions) based on the following requirements.

$\boldsymbol{Q} =$	$\{\$0.00, \$0.25, \$0.50, \$0.75, \$1.00, \$1.25, \$1.50, \$1.75, \$2.00\}$	*States*
$\boldsymbol{\Sigma} =$	$\{\$0.25, \$1.00, Select\}$	*Alphabet*
$\boldsymbol{\Delta} =$	$\delta(Q, a) \in Q$	*Transition Function*
$\boldsymbol{q_0} =$	$\{\$0.00\}$	*Start State*
$\boldsymbol{F} =$	$\emptyset$	*Set of Accept States*



Question 2 (15 points): Use regular language pumping lemma to prove that the language $\{wtw \mid w, t \in \{0,1\}^+\}$ is not regular.

$$A = \{wtw \mid w, t \in \{0,1\}^+\}$$

If we assume that A is regular, the Pumping Lemma definition tells us that any string s in A can be 'pumped' at least a 'pumping length' of p , then divided into three pieces $s = xyz$. For it to be regular, it must satisfy the following conditions:

(1) $\forall i \geq 0, xy^iz \in A$
(2) $|y| \geq 0$
(3) $|xy| \leq p$

We can test these conditions by doing the following:

- Assume A is regular
- Let p be the pumping length

- Choose a string $s = 0^p110^p1$ to test
  - As we assume A is regular, s is a member of A
  - Because $|s| = 2p + 3$, and $2p + 3 \geq p$, then $|s| \geq p$ and therefore can be split into three pieces $s = xyz$
- For conditions (2) $|y| \geq 0$ and (3) $|xy| \leq p$ to be met, piece y must contain only 0's
  - For example, consider $p = 5$

$$0^p110^p1 = 0000011000001$$

  - For $|xy| \leq 5$, y can only contain 0's

$$\underbrace{00}_{x}\underbrace{000}_{y}\underbrace{11000001}_{z}$$

- According to condition (1) $\forall i \geq 0, xy^iz \in A$, for any instance i, $xy^iz \in A$
  - assume $i = 2$

$$xy^iz \in A$$
$$xy^2z \in A$$

  - Visual Representation

$$\underbrace{00}_{x}\underbrace{000}_{y}\underbrace{000}_{y}\underbrace{11000001}_{z}$$

  - $xy^2z \notin A$
    - However, there is no possible division of this string that will result in the required wtw format. This also remains true as i increases as additional zeros are placed at the beginning of the string, creating a further imbalance between the beginning and end of the string (which should be identical w's). Therefore, language A does not meet condition (1) for Pumping Lemma.

Since not all pumping lemma conditions are met, the assumption that A is regular is contradicted and proves that A is not regular.

Question 3 (20 points): Categorize the following languages (note: provide answers only, no need to proof):

a) $B = \{w \mid w \text{ has at least three } a's \text{ and at least two } b's\}$, assume $\Sigma = \{a, b\}$
b) $B = \{ww \mid w \in \{0,1\}^*\}$
c) $B = \{ww^R \mid w \in \{0,1\}^*\}$
d) $B = \{a^nb^n \mid n \geq 0\}$
e) $B = \{a^nb^nc^n \mid n \geq 0\}$

Regular languages: a

Context-free languages but non-regular languages: c, d

Non-context-free languages: b, e

Question 4 (20 points): Answer each part for the following context-free grammar G.

$R \to XRX \mid S$
$S \to aTb \mid bTa$
$T \to XTX \mid X \mid \varepsilon$
$X \to a \mid b$

a) What are the variables of G?
- R, S, T, X

b) What are the terminals of G?
- a, b

c) Which is the start variable of G?
- R

d) Give three strings in L(G).
- $abbaa$

$\boldsymbol{R}$	starting variable
$\boldsymbol{XRX}$	$R \to XRX$
$\boldsymbol{aRX}$	$X \to a$
$\boldsymbol{aSX}$	$R \to S$
$\boldsymbol{abTaX}$	$S \to bTa$
$\boldsymbol{abXaX}$	$T \to X$
$\boldsymbol{abbaX}$	$X \to b$
$\boldsymbol{abbaa}$	$X \to a$

- bba

$\boldsymbol{R}$	starting variable
$\boldsymbol{S}$	$R \to S$
$\boldsymbol{bTa}$	$S \to bTa$
$\boldsymbol{bXa}$	$T \to X$
$\boldsymbol{bba}$	$X \to b$

- $baabbab$

$\boldsymbol{R}$	starting variable
$\boldsymbol{XRX}$	$R \to XRX$
$\boldsymbol{bRX}$	$X \to b$
$\boldsymbol{bXRXX}$	$R \to XRX$
$\boldsymbol{baRXX}$	$X \to a$
$\boldsymbol{baSXX}$	$R \to S$
$\boldsymbol{baaTbXX}$	$S \to aTb$
$\boldsymbol{baaXbXX}$	$T \to X$
$\boldsymbol{baabbXX}$	$X \to b$
$\boldsymbol{baabbaX}$	$X \to a$
$\boldsymbol{baabbab}$	$X \to b$

e) Give three strings not in L(G).
- a
- b
- $abba$

f) True or False: $T \Rightarrow aba$
- False, T can only derive XTX, X, or ε in a single step

g) True or False: $T \overset{*}{\Rightarrow} aba$.
- True, as the definition for $u \Rightarrow v$ states the following:

> u derives v, if $u = v$ or if:
> a sequence $u_1, u_2, u_3, \cdots, u_k$ exists for $\mathrm{k} \geq 0$, and
> $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$

(Sipser, pg 103)

- The sequence that satisfies this condition is listed below

$\boldsymbol{T}$	
$\boldsymbol{XTX}$	$T \to XTX$
$\boldsymbol{aTX}$	$X \to a$
$\boldsymbol{aXX}$	$T \to X$
$\boldsymbol{abX}$	$X \to b$
$\boldsymbol{aba}$	$X \to a$

h) True or False: $T \Rightarrow$T.
- False, T can only derive XTX, X, or ε in a single step

i) True or False: $T \overset{*}{\Rightarrow} T$.
- True, as the definition for $u \Rightarrow v$ states the following:

> u derives v, if $u = v$ or if:
> a sequence $u_1, u_2, u_3, \cdots, u_k$ exists for $\mathrm{k} \geq 0$, and
> $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_k \Rightarrow v$

(Sipser, pg 103)

- Therefore, if $k = 0$, the zero-step derivation $T \overset{k=0}{\Longrightarrow} T$ meets this condition.

j) True or False: $S \overset{*}{\Rightarrow} \varepsilon$
- False, there are no series of steps where S can derive ε

Question 5 (15 points): Give context-free grammars that generate the languages $L = \{0^i 1^n 2^n | n \geq 1, i \geq 1\}$

1. Define

CFG 4-tuple (V, Σ, R, S)

V	Variables *finite set*	$\{S, S_1, S_2\}$
Σ	Terminals *finite set, disjoint from V*	$\{0,1,2\}$
R	Rules *finite set, variable and string of variables/terminals*	
$S \mid S \in V$	Start Variable	S

2. Construct First Grammar for $\{0^i \big| i \geq 1\}$
   - $S_1 \to 0S_1 | \ 0$

3. Construct Second Grammar for $\{1^n 2^n | n \geq 0\}$
    - $S_2 \rightarrow 1S_2 2 \mid 12$
4. Merge rules $S \rightarrow S_1 | S_2$ to get the CFG

S	$\rightarrow$	$S_1 \| S_2$
S_1	$\rightarrow$	$0S_1 \| 0$
S_2	$\rightarrow$	$1S_2 2 \| 12$

Question 6 (15 points): Show that G is ambiguous. Let $G = (V, \Sigma, R,$<STMT>$)$ be the following grammar.

CFG 4-tuple (V, Σ, R, S)

V	Variables	{ <STMT>, <IF-THEN>, <IF-THEN-ELSE>, <ASSIGN> }
Σ	Terminals	{if, condition, then, else, a:=1}
R	Rules	*listed below*
$S \mid S \in V$	Start Variable	<STMT>

<STMT>	→	<ASSIGN> | <IF-THEN> | <IF-THEN-ELSE>
<IF-THEN>	→	if condition then <STMT>
<IF-THEN-ELSE>	→	if condition then <STMT> else <STMT>
<ASSIGN>	→	a:=1

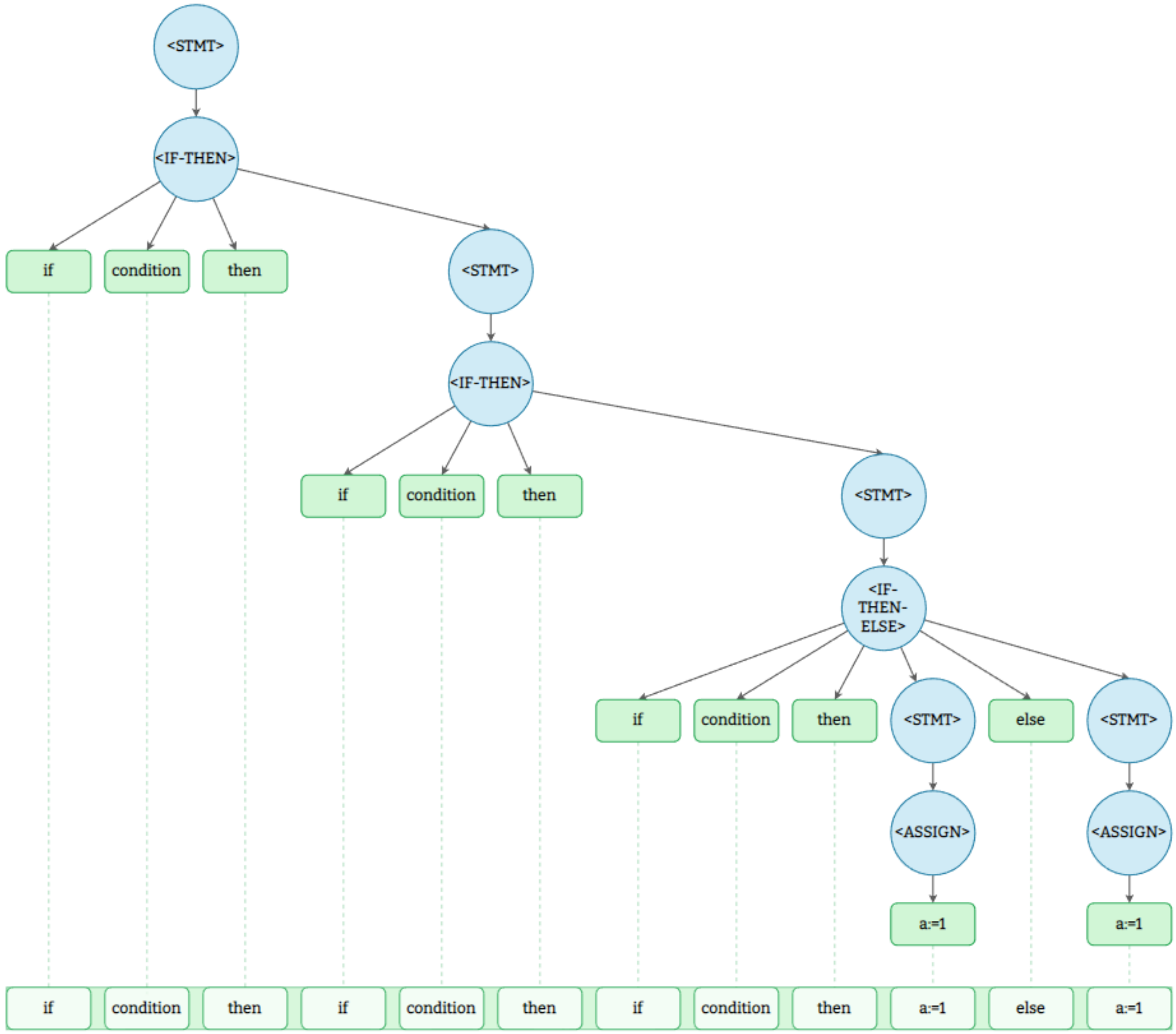
if condition then if condition then if condition then a:=1 else a:=1

The string above shows that G is ambiguous, as it can be derived multiple ways. Each instance of ambiguity, including their derivation and parsing tree, is shown below.

- First Instance
    - Derivation

<STMT>	starting variable
<IF-THEN>	<STMT> → <IF-THEN>
if condition then <STMT>	<IF-THEN>→if condition then <STMT>
if condition then <IF-THEN>	<STMT> →<IF-THEN>
if condition then if condition then <STMT>	<IF-THEN>→if condition then <STMT>
if condition then if condition then <IF-THEN-ELSE>	<STMT> →<IF-THEN-ELSE>
if condition then if condition then if condition then <STMT> else <STMT>	<IF-THEN-ELSE> → if condition then <STMT> else <STMT>
if condition then if condition then if condition then <ASSIGN> else <STMT>	<STMT> → <ASSIGN>
if condition then if condition then if condition then a:=1 else <STMT>	<ASSIGN>→ a:=1
if condition then if condition then if condition then a:=1 else <ASSIGN>	<STMT> → <ASSIGN>
if condition then if condition then if condition then a:=1 else a:=1	<ASSIGN>→ a:=1

- Parsing Tree



- Second Instance
  - Derivation

<STMT>	starting variable
<IF-THEN-ELSE>	<STMT> →<IF-THEN-ELSE>
if condition then <STMT> else <STMT>	<IF-THEN-ELSE> → if condition then <STMT> else <STMT>
if condition then <IF-THEN> else <STMT>	<STMT> →<IF-THEN>
if condition then if condition then <STMT> else <STMT>	<IF-THEN>→if condition then <STMT>
if condition then if condition then <IF-THEN> else <STMT>	<STMT> →<IF-THEN>
if condition then if condition then if condition then <STMT> else <STMT>	<IF-THEN>→if condition then <STMT>
if condition then if condition then if condition then <ASSIGN> else <STMT>	<STMT> → <ASSIGN>
if condition then if condition then if condition then a:=1 else <STMT>	<ASSIGN>→ a:=1
if condition then if condition then if condition then a:=1 else <ASSIGN>	<STMT> → <ASSIGN>
if condition then if condition then if condition then a:=1 else a:=1	<ASSIGN>→ a:=1

- Parsing Tree