Homework 3 - 5

Menu

HW#3 Problem 1

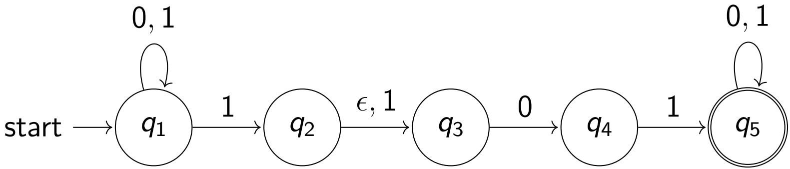
(Exercise 1.7 adapted; 10 points) For each of the following languages, give the state diagram of an NFA, with as few states as possible, that recognizes the language. In all parts, the alphabet is $\{\mathtt{0}, \mathtt{1}\}$.

(a) The language $\{w \mid w$ contains `011` or `0101` as a substring, i.e., $w = x(\mathtt{011}|\mathtt{0101})y$ for some x and $y\}$

(b) The language $1^*0^*1^+$ (Note: 1^+ is a shorthand for 11^*.)

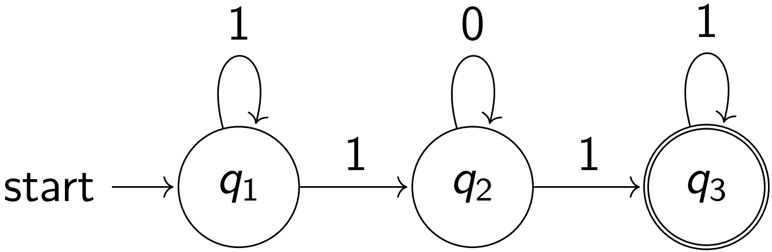
HW#3 Problem 1

(a) The language $\{\omega \mid \omega$ contains 101 or 1101 as a substring, i.e., $\omega = x(101|1101)y$ for some x and $y\}$ with five states.



HW#3 Problem 1

(b) The language $1^*0^*1^+$ with three states.

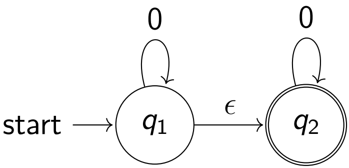


HW#3 Poblem 2

(Exercise 1.14; 10 points) Show by giving an example that, if M is an NFA that recognizes language C, swapping the accept and nonaccept states in M doesn't necessarily yield a new NFA that recognizes the complement of C. Is the class of languages recognized by NFAs closed under complement? Explain your answer.

HW#3 Problem 2

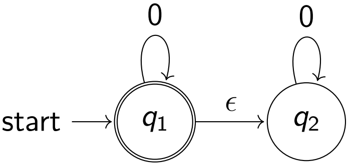
Give a example that swapping the accept and nonaccept states in an NFA does not necessarily yield a new NFA that recognizes the complement of the original language:



The above NFA recognizes the string 0^*.

HW#3 Problem 2

Give a example that swapping the accept and nonaccept states in an NFA does not necessarily yield a new NFA that recognizes the complement of the original language:



The above NFA still recognizes the string 0^*.

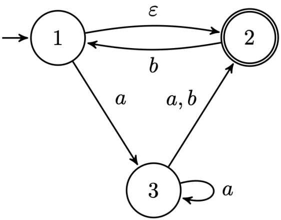
HW#3 Problem 2

Show that the class of languages recognized by NFAs is closed under complement:

Let the language L be the language recognized by an NFA M.
According to Theorem 1.39 on the slides, every NFA has an equivalent DFA. Let N be the equivalent DFA of M, the complement of N (written $\overline{N}$) recognizes the complement of L.
Similarly, every DFA has an equivalent NFA, so $\overline{N}$ must have an equivalent NFA, called D. In conclusion, the complement of L is still recognized by an NFA D, so the class of languages recognized by NFAs is closed under complement.

HW#3 Poblem 3

(Exercise 1.16 adapted; 20 points) Use the construction given in Theorem 1.39 (every NFA has an equivalent DFA) to convert the following NFA into an equivalent DFA.



HW#3 Problem 3

Use Th 1.39 (subset construction) to construct equivalent DFA.

HW#3 Problem 3

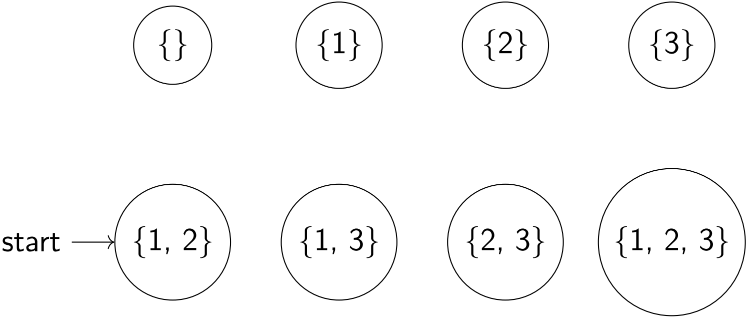
List all states

$\{\}$ $\{1\}$ $\{2\}$ $\{3\}$

$\{1, 2\}$ $\{1, 3\}$ $\{2, 3\}$ $\{1, 2, 3\}$

HW#3 Problem 3

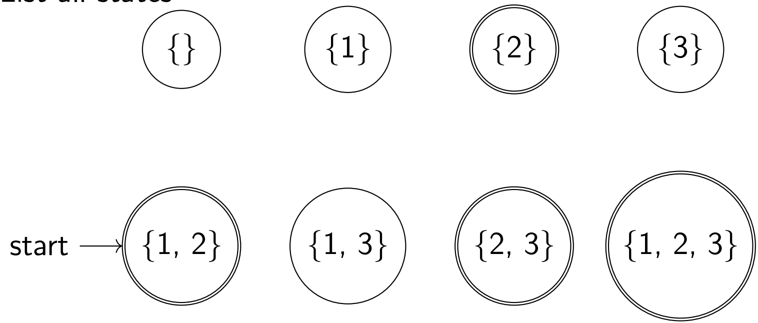
List all states



$q'_0 = E(\{1\}) = \{1, 2\}$

HW#3 Problem 3

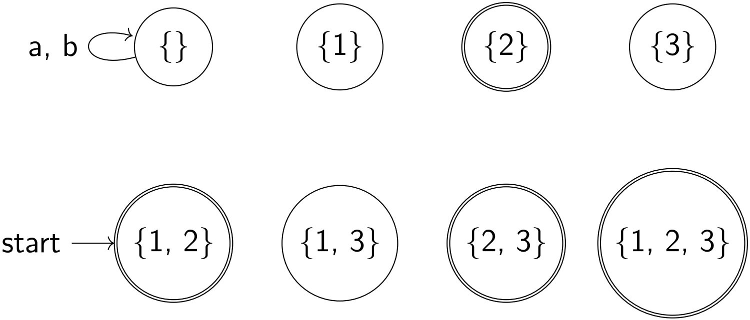
List all states



$$F' = \{\{2\}, \{1, 2\}, \{2, 3\}, \{1, 2, 3\}\}$$

HW#3 Problem 3

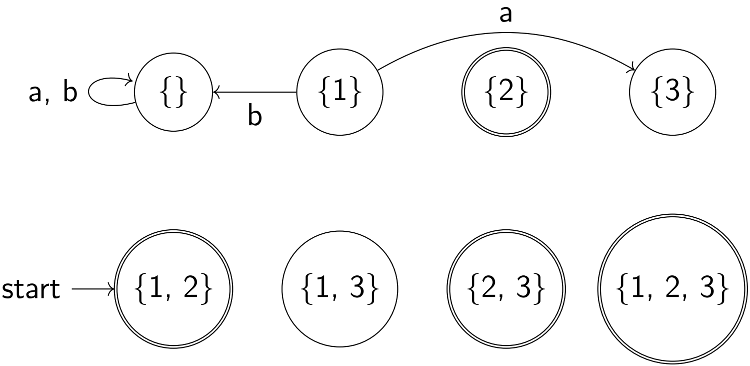
List all states



$\delta'(\{\}, a) = \{\}$
$\delta'(\{\}, b) = \{\}$

HW#3 Problem 3

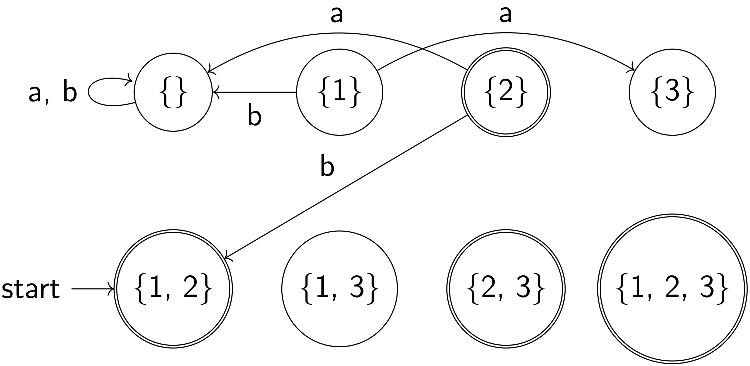
List all states



$\delta'(\{1\}, a) = \{3\}$
$\delta'(\{1\}, b) = \{\}$

HW#3 Problem 3

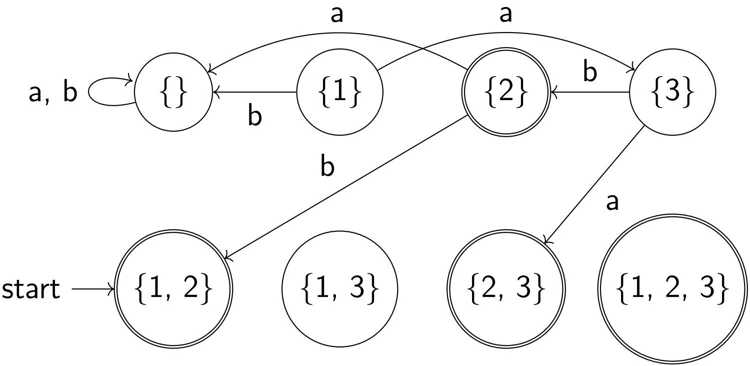
List all states



$\delta'(\{2\}, a) = \{\}$
$\delta'(\{2\}, b) = \{1, 2\}$

HW#3 Problem 3

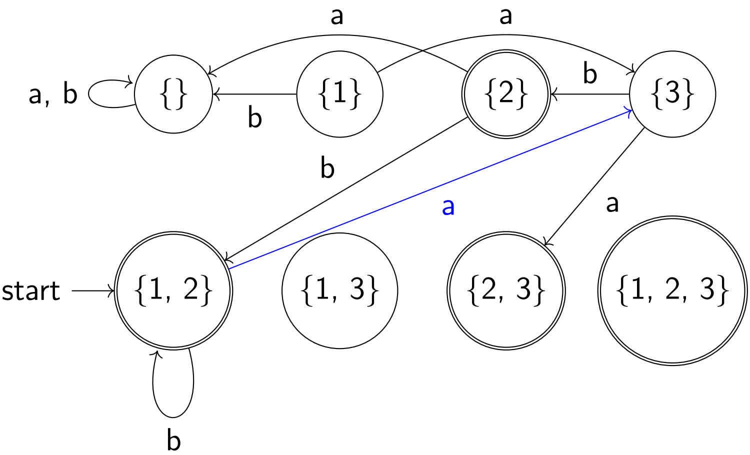
List all states



$\delta'(\{3\}, a) = \{2, 3\}$
$\delta'(\{3\}, b) = \{2\}$

HW#3 Problem 3

List all states

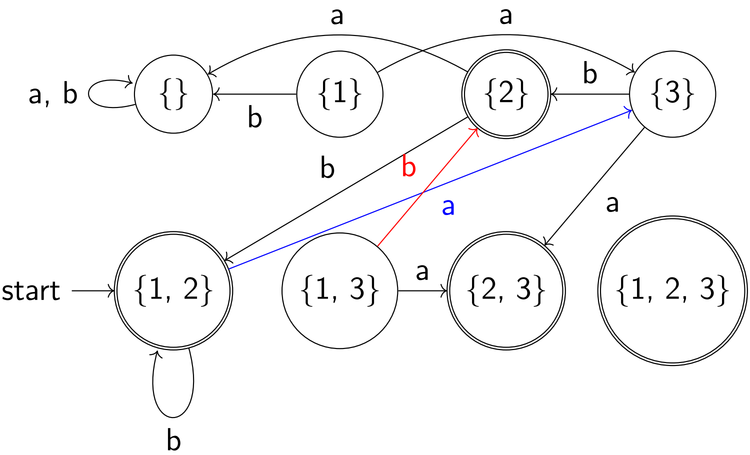


$\delta'(\{1,2\}, a) = \{3\}$

$\delta'(\{1,2\}, b) = \{1,2\}$

HW#3 Problem 3

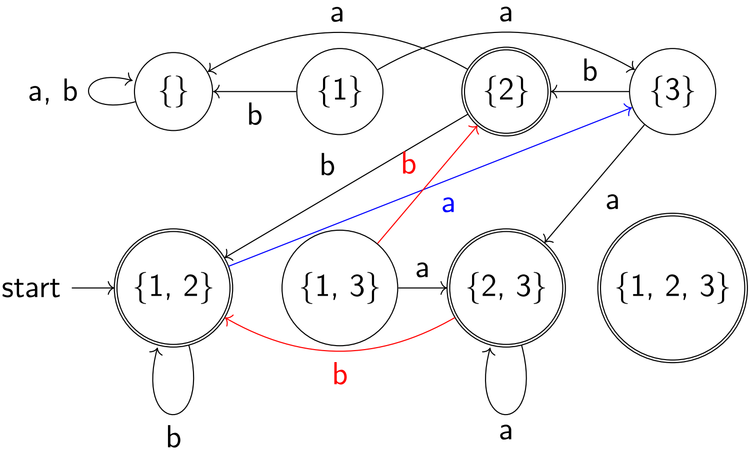
List all states



$\delta'(\{1,3\}, a) = \{2,3\}$
$\delta'(\{1,3\}, b) = \{2\}$

HW#3 Problem 3

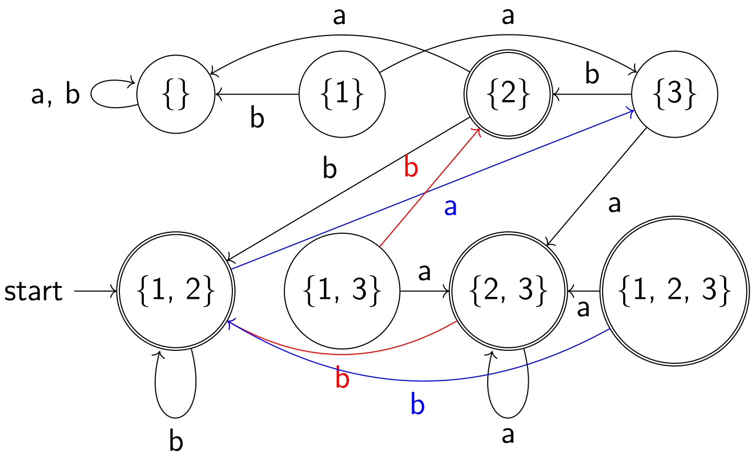
List all states



$$\delta'(\{2,3\}, a) = \{2,3\}$$
$$\delta'(\{2,3\}, b) = \{1,2\}$$

HW#3 Problem 3

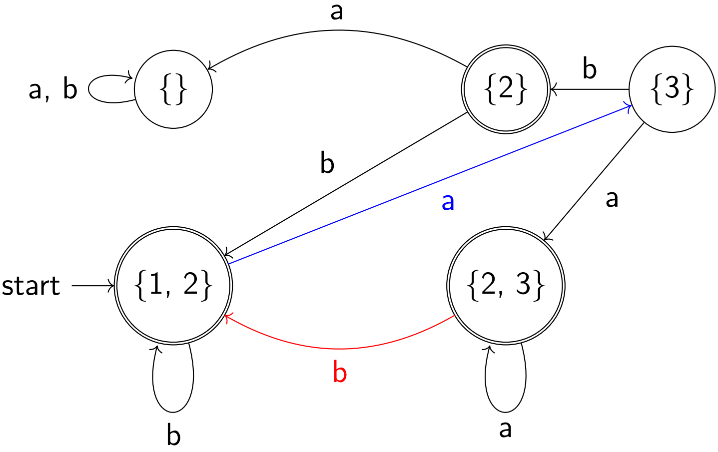
List all states



$\delta'(\{1,2,3\}, a) = \{2,3\}$
$\delta'(\{1,2,3\}, b) = \{1,2\}$

HW#3 Problem 3

List all states



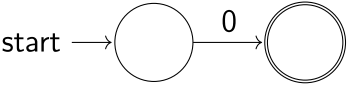
delete unreachable states {1}, {1, 3} and {1, 2, 3}.

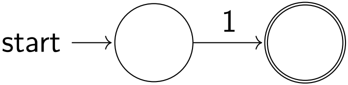
HW#3 Problem 4

(Exercise 1.18 adapted; 10 points) Use the procedure described in Lemma 1.55 to convert the regular expression $(0 \cup 1)^{+}011(0 \cup 1)^{*}$ into an NFA. Be sure to show the intermediate automata.

HW#3 Problem 4

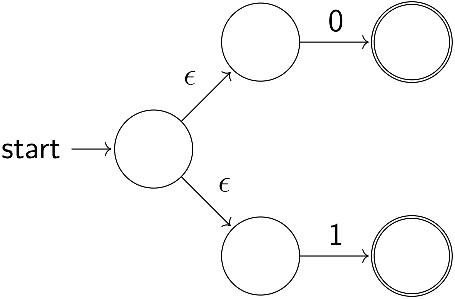
0 1





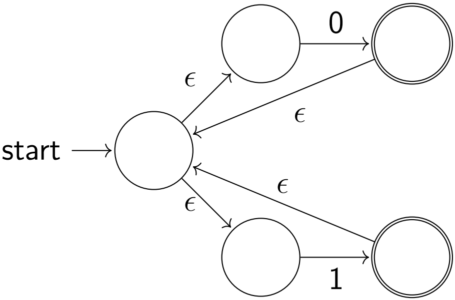
HW#3 Problem 4

$0 \cup 1$



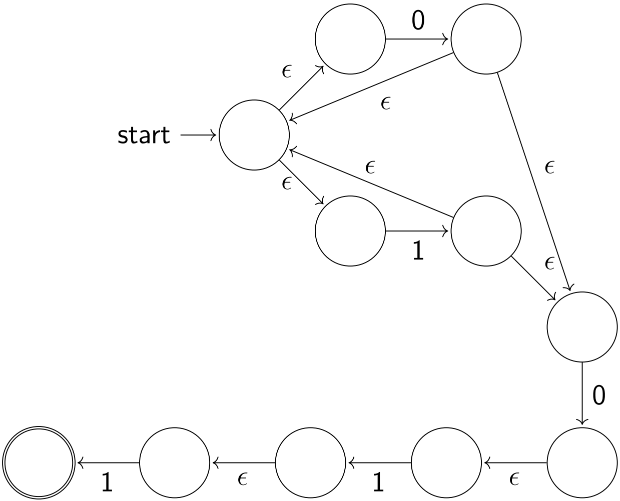
HW#3 Problem 4

$(0 \cup 1)^+$



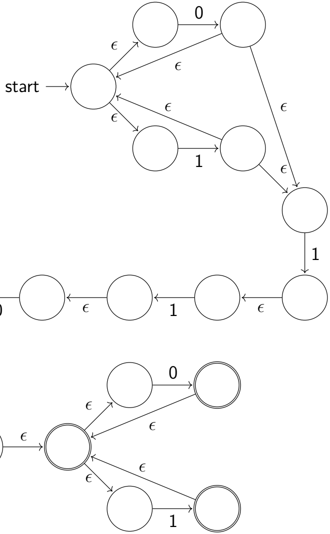
HW#3 Problem 4

$(0 \cup 1)^{+}011$



HW#3 Problem 4

$(0 \cup 1)^{+}011(0 \cup 1)^{*}$



HW#3 Poblem 5

(Exercise 1.20; 10 points) Give regular expressions generating the following languages, where the alphabet is $\{\texttt{0}, \texttt{1}\}$:

(a) $\{w \mid \text{every odd position of } w \text{ is a } \texttt{1}\}$ (Note: see w as $w_1 w_2 \cdots w_n$, where $w_i \in \{\texttt{0}, \texttt{1}\}$)

(b) $\{w \mid w \text{ doesn't contain the substring } \texttt{011}\}$

HW#3 Problem 5

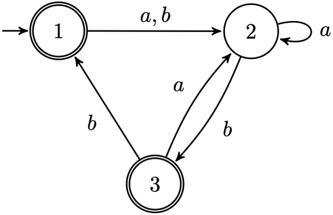
(a) odd position is 1
$(1(0 \cup 1))^*(1 \cup \epsilon)$

HW#3 Problem 5

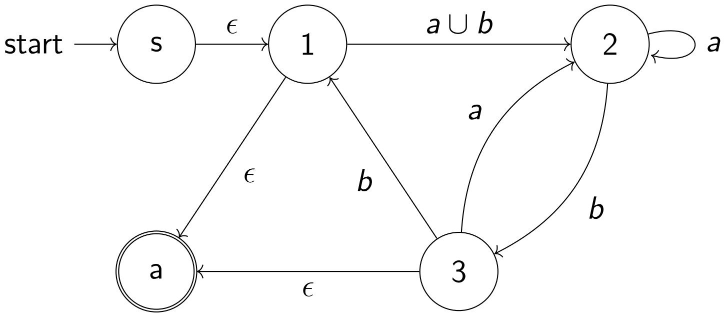
(b) doesn't contain the substring 011
$1^*(0 \cup 01)^*$

HW#3 Poblem 6

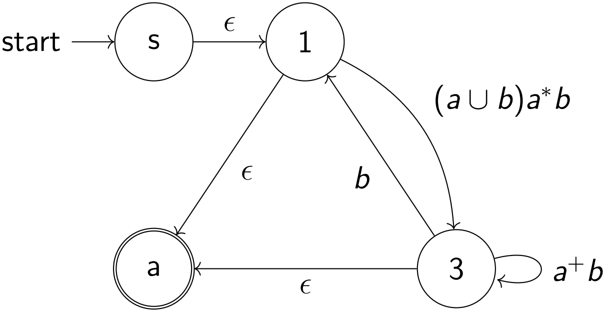
(Exercise 1.21 adapted; 20 points) Use the procedure described in Lemma 1.60 to convert the following finite automaton into a regular expression.



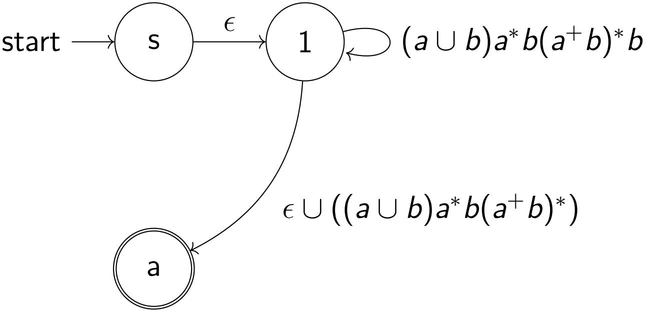
HW#3 Problem 6



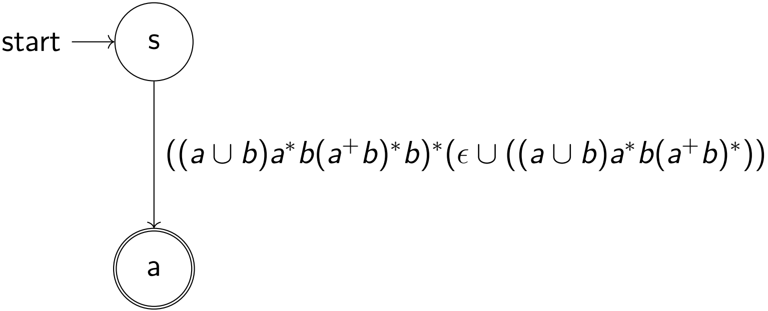
HW#3 Problem 6



HW#3 Problem 6

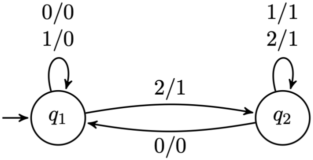


HW#3 Problem 6

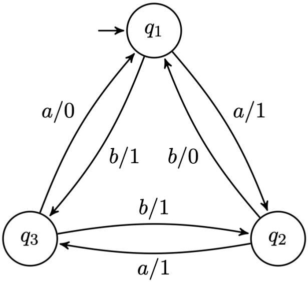


HW#3 Poblem 7

(Exercise 1.24 adapted; 10 points) A *finite-state transducer* (FST) is a type of deterministic finite automaton whose output is a string rather than *accept* or *reject*. The following are state diagrams of finite state transducers T_1 and T_2.



T_1



T_2

HW#3 Poblem 7

Each transition of an FST is labeled with two symbols, one designating the input symbol for that transition and the other designating the output symbol. The two symbols are written with a slash, /, separating them. In T_1, the transition from q_1 to q_2 has input symbol 2 and output symbol 1. Some conditions may have multiple input-output pairs, such as the transition in T_1 from q_1 to itself. When an FST computes on an input string w, it takes the input symbols $w_1 \cdots w_n$ one by one and, starting from the start state, follows the transitions by matching the input labels with the sequence of symbols $w_1 \cdots w_n = w$. Every

time it goes along a transition, it outputs the corresponding output symbol. For example, on input 2212011, machine T_1 enters the sequence of states $q_1, q_2, q_2, q_2, q_2, q_1, q_1, q_1$ and produces output 1111000. On input **abbb**, T_2 outputs 1011. Give the sequence of states entered and the output produced in each of the following parts.

(a) T_1 on input 120221

(b) T_2 on input **abaabb**

HW#3 Problem 7

(a) q_1
q_1: input 1, output 0, transfer to q_1
q_1: input 2, output 1, transfer to q_2
q_2: input 0, output 0, transfer to q_1
q_1: input 2, output 1, transfer to q_2
q_1: input 2, output 1, transfer to q_2
q_2: input 1, output 1, transfer to q_2
Output: 010111

HW#3 Problem 7

(b) abaabb q_1
q_1: input a, output 1, transfer to q_2
q_2: input b, output 0, transfer to q_1
q_1: input a, output 1, transfer to q_2
q_2: input a, output 1, transfer to q_3
q_3: input b, output 1, transfer to q_2
q_2: input b, output 0, transfer to q_1
Output: 101110

HW#3 Poblem 8

(Exercise 1.25 adapted; 10 points) Read the informal definition of the finite state transducer given in Exercise 1.24. Give a formal definition of this model, following the patterns in Definition 1.5 (Page 35 in Sipser's book or Page 7 of the slides). Assume that an FST has an input alphabet Σ and an output alphabet Γ but not a set of accept states. Include a formal definition of the computation of an FST. (Hint: an FST is a 5-tuple. Its transition function is of the form $\delta : Q \times \Sigma \longrightarrow Q \times \Gamma$.)

HW#3 Problem 8

An FST T is a 5-tuple $(Q, \Sigma, \Gamma, \delta, q_0)$
Q is a finite set of states
Σ is a finite set of input symbols
Γ is a finite set of output symbols
$\delta : Q \times \Sigma \to Q \times \Gamma$ is the transition function
$q_0 \in Q$ is the start state

Let $w = w_1 w_2 ... w_n$ be a string over Σ and $x = x_1 x_2 ... x_n$ a string over Γ
We say T produces output x on input w with the sequence of states $r_0, r_1, ..., r_n$ when

- $r_0 = q_0$
- $\delta(r_i, w_{i+1}) = (r_{i+1}, x_{i+1})$ for $i = 0, 1, ..., i - 1$

HW#4 Problem 1

(Problem 1.43; 10 points) An *all*-NFA M is a 5-tuple $(Q, \Sigma, \delta, q, F)$ that accepts $x \in \Sigma^*$ if every possible state that M could be after reading input x is a state from F. Note, in contrast, that an ordinary NFA accepts a string if *some* state among these possible states is an accept state. Prove that all-NFAs recognize the class of regular languages.

HW#4 Problem 1

We need to prove the following two claims:

- All regular languages can be recognized by an *all*-NFA.
- All languages *all*-NFAs recognize are regular.

Claim: All regular languages can be recognized by an *all*-NFA.

Proof: All regular languages are recognized by a DFA, and DFA is also an *all*-NFA because DFA has only one run for each input string, namely, all the accepting runs (only one) terminate at the accepting states.

HW#4 Problem 1

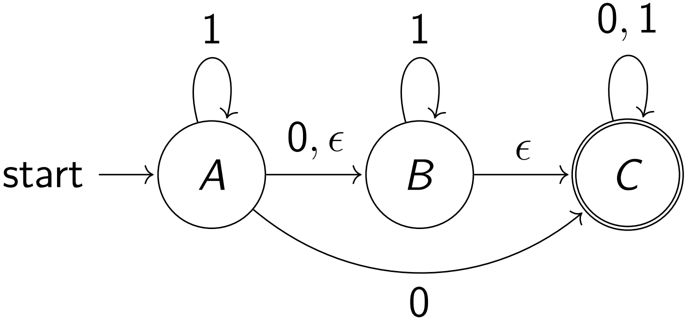
Claim: All languages *all*-NFAs recognize are regular.

Proof: Suppose that A is the language that an *all*-NFA $N = (Q, \Sigma, \delta, q, F)$ recognizes. Now we can construct a DFA $M = (Q', \Sigma, \delta', q', F')$ that recognizes A as follows:

- $Q' = P(Q)$ (the power set of Q).
- δ' is the ϵ-closure of transitions from the elements of the state-set.
- $q' = \{q\}$.
- $F' = P(F) - \{\{\}\}$.

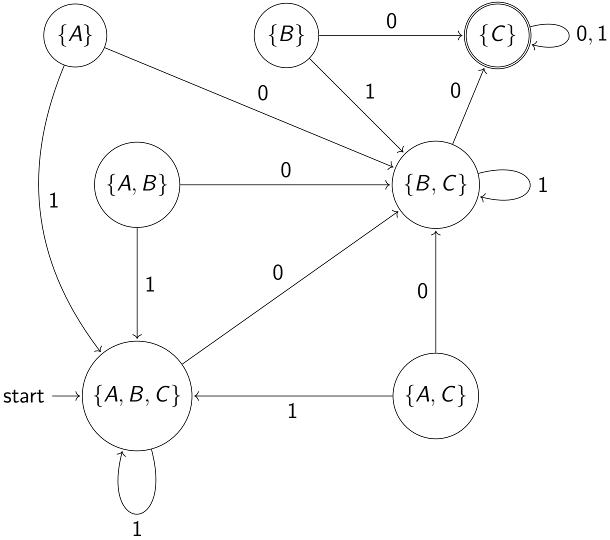
HW#4 Problem 1

For example: *all*-NFA N:



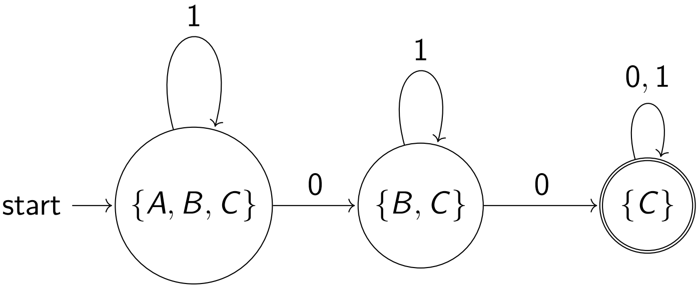
HW#4 Problem 1

For example: DFA M:



HW#4 Problem 1

Simplify M:



HW#4 Problem 2

(Problem 1.66; 20 points) Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let h be a state of M called its "home". A *synchronizing sequence* for M and h is a string $s \in \Sigma^*$ where $\delta(q, s) = h$ for every $q \in Q$. Say that M is *synchronizable* if it has a synchronizing sequence for some state h. Prove that, if M is a k-state synchronizable DFA, then it has a synchronizing sequence of length at most k^3. (Note: $\delta(q, s)$ equals the state where M ends up, when M starts from state q and reads input s.)

HW#4 Problem 2

We first start from two states q_A and q_B of Q.

q_A and q_B can reach the same state since M is synchronizable. Let s_{AB} be a string with the minimum length that leads q_A and q_B into the same state g.

The length of s_{AB} is at most $k * (k-1)$. Because the pairs of different two states in Q are at most $k * (k-1)$, if the length of s_{AB} is $k * (k-1) + 1$, there must be two repeated pairs, which means that the substring between them could be removed.

For example: if s_{AB} can be divided as $s_1 s_2 s_3$ such that

$$(q_A, q_B) \xrightarrow{s_1} (q'_A, q'_B) \xrightarrow{s_2} (q'_A, q'_B) \xrightarrow{s_3} (g, g)$$

Then s_2 can be removed.

HW#4 Problem 2

Now we have k states in Q. We can first run s_{AB} with the length at most $k * (k-1)$ so that q_A and q_B will transfer to the same state. Then, we can similarly run s_{BC} to make q_B and q_C transfer to the same state, which means that q_A, q_B and q_C are in the same state.

By repeating the steps above $k-1$ times, all k states will be transferred to the same state, which is h. And we can obtain our synchronizing sequence s with the length at most $k * (k-1)^2 \leq k^3$.

HW#4 Problem 3

(Problem 1.61; 20 points) Let the *rotational closure* of language A be $RC(A) = \{yx \mid xy \in A\}$.

(a) Show that, for any language A, we have $RC(A) = RC(RC(A))$ (i.e., rotational closure, as an operation/function, is idempotent).

(b) Show that the class of regular languages is closed under rotational closure.

HW#4 Problem 3 (a)

First of all, it is obvious that, for any language A, we have $A \subseteq RC(A)$ (by taking x or y in the definition of RC to be the empty string). Therefore, for any language A, we have $RC(A) \subseteq RC(RC(A))$ readily.
It remains to be proven that $RC(RC(A)) \subseteq RC(A)$.
For this, we let Σ be the alphabet and show that, for every $w \in \Sigma^*$, if $w \in RC(RC(A))$, then $w \in RC(A)$.

HW#4 Problem 3 (a)

Suppose $w \in RC(RC(A))$. Let $w = yx$ for some $x, y \in \Sigma^*$ such that $xy \in RC(A)$. For $xy \in RC(A)$ to hold, either $xy = x_1x_2y$ and $x_2yx_1 \in A$ for some $x_1, x_2 \in \Sigma^*$ or $xy = xy_1y_2$ and $y_2xy_1 \in A$ for some $y_1, y_2 \in \Sigma^*$. In the first case where $x_2yx_1 \in A$, we have $yx_1x_2 \in RC(A)$ and hence $w = yx = yx_1x_2 \in RC(A)$; analogously, for the second case.

HW#4 Problem 3 (b)

Let A be an arbitrary regular language and $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ be a DFA that recognizes A. To prove that $RC(A)$ is also regular, we construct from M_A (as a building block) an NFA N that recognizes $RC(A)$. We first elaborate on the basic ideas and then give a formal definition for N.

HW#4 Problem 3 (b)

Suppose N is given an input $w = yx$ for some $x, y \in \Sigma^*$ such that $xy \in A$. Let q_x be the state in which M_A ends up after reading x. Starting from q_x, M_A should end at some final state after reading y. For N to accept w, we let N simulate M_A from q_x and, after reading y and reaching a final state, make an epsilon transition (which needs to be added to M_A) to the initial state q_x of M_A and continue simulating M_A with the rest of the input.
If N eventually ends up at q_x, then the input w is of the correct form of yx such that $xy \in A$. Any state of M_A may act as q_x.

HW#4 Problem 3 (b)

For N to start and finish the simulation at the same state, we need $|Q_A|$ copies of M_A, one for each state in Q_A, with an epsilon transition added from every final state to the initial state. To start the simulation of M_A from any state, N has an epsilon transition from its initial state to every state of M_A.

So, $N = (Q_A \times Q_A \cup \{q_0\}, \Sigma_\varepsilon, \delta, q_0, \bigcup_{q \in Q_A} \{(q, q)\})$, where

$$\begin{cases} \delta(q_0, \varepsilon) = \bigcup_{q \in Q_A} \{(q, q)\} & \\ \delta((q_1, q_2), a) = \{(q, q_2) \mid \delta_A(q_1, a) = q\} & q_1, q_2 \in Q_A \text{ and } a \in \Sigma \\ \delta((q_1, q_2), \varepsilon) = \{(q_A, q_2)\} & q_1 \in F_A \text{ and } q_2 \in Q_A \\ \delta(q, a) = \emptyset & \text{otherwise} \end{cases}$$

HW#4 Problem 4

(Problem 1.64; 20 points) If A is any language, let $A_{\frac{1}{2}-}$ be the set of all first halves of strings in A so that

$$A_{\frac{1}{2}-} = \{x \mid \text{ for some } y, |x| = |y| \text{ and } xy \in A\}.$$

Show that if A is regular, then so is $A_{\frac{1}{2}-}$.

HW#4 Problem 4

The idea is that two DFA works simultaneously, one starts from the start state q and recognizes A, and the other starts from one of the accepting states $r \in F$ and recognizes A^R. Whenever the former DFA reads in an input a, we feed a letter c to the latter DFA to let both DFAs move forward for one step.
So, if both DFAs stop at the same state, we know that the two strings are of same length and the concatenation of them are in A.

HW#4 Problem 4

Suppose that A is the language that an DFA $D = (Q, \Sigma, \delta, q, F)$ recognizes. Now we can construct a NFA $N = (Q', \Sigma, \delta', q', F')$ that recognizes $A_{\frac{1}{2}-}$ as follows:

- $Q' = \{Q \times Q\} \cup \{q_0\}$.
- $\delta'(q_0, \epsilon) = (q, r)$ for all $r \in F$
  $\delta'((r_1, r_2), a) = (\delta(r_1, a), z)$ for any z such that there exists some $c \in \Sigma$ with $\delta(z, c) = r_2$.
- $q' = q_0$.
- $F' = \{(r, r) | r \in Q\}$.

HW#4 Problem 5

(Problem 1.40; 10 points) Let

$$\Sigma_2 = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}.$$

Here, Σ_2 contains all columns of 0s and 1s of length two. A string of symbols in Σ_2 gives two rows of 0s and 1s.

Consider the top and bottom rows to be strings of 0s and 1s and let

$$E = \{w \in \Sigma_2^* \mid \text{the bottom row of } w \text{ is the reverse of the top row of } w\}.$$

Show that E is not regular.

HW#4 Problem 5

Use the pumping lemma: Let s be $\begin{bmatrix} 0 \\ 1 \end{bmatrix}^p \begin{bmatrix} 1 \\ 0 \end{bmatrix}^p$, where p is the pumping length for E.

When dividing s as xyz, because $|xy| \leq p$, y must consist of $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ s.

And obviously, $xy^2z \notin E$ (the number of 0 is different between the top and the bottom rows).

HW#4 Problem 6

(Problem 1.71; 20 points) Let $\Sigma = \{0, 1\}$.

(a) Let $A = \{1^k x \mid x \in \Sigma^*$ and x contains at least k 1s, for $k \geq 1\}$. Show that A is regular.

(b) Let $B = \{1^k x \mid x \in \Sigma^*$ and x contains at most k 1s, for $k \geq 1\}$. Show that B is not regular.

HW#4 Problem 6

(a) Regular expression: $10^*1(0 \cup 1)^*$
(b) Use the pumping lemma:
Let s be $1^p0^p1^p$, where p be the pumping length given by the pumping lemma.
When dividing s as xyz, because $|xy| \leq p$, $y = 1^i$ for some $i \geq 1$.
$xy^0z = 1^{p-i}0^p1^p \notin B$ ($p - i < p$ so x contains more than k 1s).

HW#5 Problem 1

(Exercise 2.1; 10 points) Consider the following CFG discussed in class, where for convenience the variables have been renamed with single letters.

$$\begin{array}{lcl} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T \times F \mid F \\ F & \rightarrow & (E) \mid a \end{array}$$

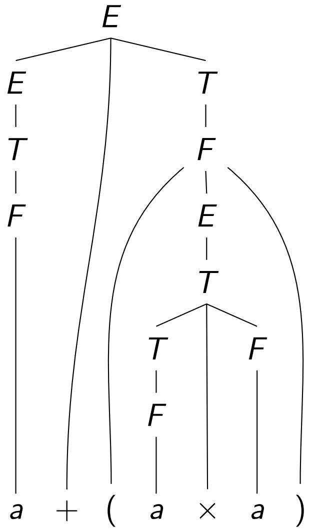
Give (leftmost) derivations and the corresponding parse trees for the following strings.

(a) $a + (a \times a)$

(b) $((a) \times a)$

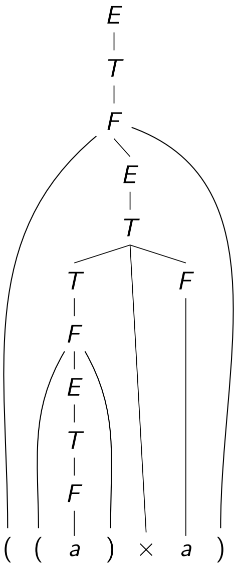
HW#5 Problem 1 (a)

$a + (a \times a)$



HW#5 Problem 1 (b)

$((a) \times a)$



HW#5 Problem 2

(Exercise 2.4; 10 points) Give CFGs that generate the following languages. In all parts the alphabet Σ is $\{0, 1\}$.

(a) $\{w \mid \text{the length of } w \text{ is a multiple of } 3\}$

(b) $\{w \mid w = w^R, \text{ that is, } w \text{ is a palindrome}\}$

HW#5 Problem 2 (a)

$\{\omega \mid \text{the length of } \omega \text{ is multiple of } 3\ \}$

$S \rightarrow AAAS \mid \epsilon$
$A \rightarrow 0 \mid 1$

HW#5 Problem 2 (b)

$\{\omega \mid \omega = \omega^R$, that is, ω is palindrome $\}$

$S \rightarrow 0S0 \mid 1S1 \mid C \mid \epsilon$
$C \rightarrow 0 \mid 1$

HW#5 Problem 3

(Exercise 2.6d; 10 points) Give a CFG that generates the language $\{x_1\#x_2\#\cdots\#x_k \mid k \geq 1,$ each $x_i \in \{a,b\}^*,$ and for some i and j, $x_i = x_j^R\}$.

HW#5 Problem 3

The pattern of generated string can be considered as the following:
Lx_iMx_jR, where $x_i = x_j^R$.
Let $X = \{a, b\}^*$.
L can generate:
1. ϵ
2. $...X\#X\#X\#$

R can generate:
1. ϵ
2. $\#X\#X\#X...$

HW#5 Problem 3

Lx_iMx_jR

M can generate:

1. $\#$
2. $\#X\#X\#X\#$ ... $\#X\#X\#X\#$
3. ϵ, a, b (when $i = j$, namely $x_i = x_j$ is a palindrome)

HW#5 Problem 3

$$
\begin{aligned}
&S \to LM'R \\
&M' \to aM'a \mid bM'b \mid M \\
&M \to \#XMX\# \mid \# \mid a \mid b \mid \epsilon \\
&L \to X\#L \mid \epsilon \\
&R \to R\#X \mid \epsilon \\
&X \to Xa \mid Xb \mid \epsilon
\end{aligned}
$$

HW#5 Problem 4

(Problem 2.33; 20 points) Let $\Sigma = \{a, b\}$. Give a CFG generating the language of strings with twice as many a's as b's (no restriction is imposed on the order in which the input symbols may appear). Prove that the CFG is correct.

HW#5 Problem 4

The CFG G generates the language $C = \{w \mid w$ contains twice as many a's as b's$\}$:
$S \rightarrow aaSb \mid aSbSa \mid bSaa \mid SS \mid \epsilon$

Let the string $s \in C$ is of length k, we can prove that G generates s by strong induction on k :

Base case($k = 0$): $s = \epsilon \in L(G)$.
Inductive step: Let $s = s_1 \cdots s_k$ and $c_i =$ the number of a's minus twice the number of b's in $s_1 \cdots s_i$, consider two cases:
(1) There exists $c_i = 0$ for some $0 < i < k$, then we can let $s = pq$ where p is the first i letters of s, by induction hypothesis we know both p and $q \in L(G)$. Therefore the rule $S \rightarrow SS$ generates s.

HW#5 Problem 4

(2) $c_i \neq 0$ for all $0 < i < k$, then there are three subcases:

(i) s starts with b, then $c_i < 0$ for all $0 < i < k$, and s must end with aa. Therefore $s = bpaa$ where $p \in L(G)$, and the rule $S \to bSaa$ generates s.

(ii) s starts with a and $c_i >= 0$ for all $0 < i < k$, then $s = aapb$ where $p \in L(G)$, and the rule $S \to aaSb$ generates s.

(iii) s starts with a and $c_i < 0$ for some $0 < i < k$, then $s = apbqa$ where $p, q \in L(G)$, and the rule $S \to aSbSa$ generates s.

HW#5 Problem 5

(Exercise 2.8 adapted; 10 points) Show that the string "**`the boy sees a girl with a telescope`**" has two different leftmost derivations in the following CFG.

$$
\begin{array}{rcl}
\langle\text{SENTENCE}\rangle & \to & \langle\text{NOUN-PHRASE}\rangle\langle\text{VERB-PHRASE}\rangle \\
\langle\text{NOUN-PHRASE}\rangle & \to & \langle\text{CMPLX-NOUN}\rangle \mid \\
 & & \langle\text{CMPLX-NOUN}\rangle\langle\text{PREP-PHRASE}\rangle \\
\langle\text{VERB-PHRASE}\rangle & \to & \langle\text{CMPLX-VERB}\rangle \mid \\
 & & \langle\text{CMPLX-VERB}\rangle\langle\text{PREP-PHRASE}\rangle \\
\langle\text{PREP-PHRASE}\rangle & \to & \langle\text{PREP}\rangle\langle\text{CMPLX-NOUN}\rangle \\
\langle\text{CMPLX-NOUN}\rangle & \to & \langle\text{ARTICLE}\rangle\langle\text{NOUN}\rangle \\
\langle\text{CMPLX-VERB}\rangle & \to & \langle\text{VERB}\rangle \mid \langle\text{VERB}\rangle\langle\text{NOUN-PHRASE}\rangle \\
\langle\text{ARTICLE}\rangle & \to & \texttt{a} \mid \texttt{the} \\
\langle\text{NOUN}\rangle & \to & \texttt{boy} \mid \texttt{girl} \mid \texttt{flower} \mid \texttt{telescope} \\
\langle\text{VERB}\rangle & \to & \texttt{touches} \mid \texttt{likes} \mid \texttt{sees} \\
\langle\text{PREP}\rangle & \to & \texttt{with}
\end{array}
$$

HW#5 Problem 5

S $\Rightarrow$ NP VP $\Rightarrow$ CN VP $\Rightarrow$ A N VP $\Rightarrow$ the N VP $\Rightarrow$ the boy VP $\Rightarrow$
the boy CV $\Rightarrow$ the boy V NP $\Rightarrow$ the boy sees NP $\Rightarrow$
the boy sees CN PP $\Rightarrow$ the boy sees A N PP $\Rightarrow$
the boy sees the N PP $\Rightarrow$ the boy sees the girl PP $\Rightarrow$
the boy sees the girl P CN $\Rightarrow$ the boy sees the girl with CN $\Rightarrow$
the boy sees the girl with A N $\Rightarrow$ the boy sees the girl with a N $\Rightarrow$
the boy sees the girl with a telescope

HW#5 Problem 5

S $\Rightarrow$ NP VP $\Rightarrow$ CN VP $\Rightarrow$ A N VP $\Rightarrow$ the N VP $\Rightarrow$ the boy VP $\Rightarrow$
the boy CV PP $\Rightarrow$ the boy V NP PP $\Rightarrow$ the boy sees NP PP $\Rightarrow$
the boy sees CN PP $\Rightarrow$ the boy sees A N PP $\Rightarrow$
the boy sees the N PP $\Rightarrow$ the boy sees the girl PP $\Rightarrow$
the boy sees the girl P CN $\Rightarrow$ the boy sees the girl with CN $\Rightarrow$
the boy sees the girl with A N $\Rightarrow$ the boy sees the girl with a N $\Rightarrow$
the boy sees the girl with a telescope

HW#5 Problem 6

(Exercise 2.9; 20 points) Give a CFG that generates the language

$$A = \{a^i b^j c^k \mid i = j \text{ or } j = k \text{ where } i, j, k \geq 0\}.$$

Is your grammar ambiguous? Why or why not?

HW#5 Problem 6

To design a CFG to that generates $a^i b^j c^k$ where $i = j \vee j = k$
We can consider two paths: $i = j$ or $j = k$
If we choose $i = j$, then the left part should have equal a and b.
If we choose $j = k$, then the right part should have equal b and c.

HW#5 Problem 6

$S \to UC \mid AV$
$U \to aUb \mid \epsilon$
$V \to bVc \mid \epsilon$
$A \to aA \mid \epsilon$
$C \to cC \mid \epsilon$

Is the CFG ambiguous?
Consider the $s = abc$, there are two ways to generate s
$S \Rightarrow UC \Rightarrow aUbC \Rightarrow abC \Rightarrow abcC \Rightarrow abc$
$S \Rightarrow AV \Rightarrow aAV \Rightarrow aV \Rightarrow abVc \Rightarrow abc$

HW#5 Problem 7

(Exercise 2.14; 20 points) Convert the following CFG (where A is the start variable) into an equivalent CFG in Chomsky normal form, using the procedure given in Theorem 2.9.

$$\begin{array}{rcl} A & \rightarrow & BAB \mid B \mid \varepsilon \\ B & \rightarrow & 0B1 \mid \varepsilon \end{array}$$

HW#5 Problem 7

$$A \to BAB \mid B \mid \epsilon$$
$$B \to 0B1 \mid \epsilon$$

HW#5 Problem 7

1. Add a new start symbol

Add $S_0 \to A$

$S_0 \to A$

$A \to BAB \mid B \mid \epsilon$

$B \to 0B1 \mid \epsilon$

HW#5 Problem 7

2. Remove ϵ rules

Remove $B \rightarrow \epsilon$

$S_0 \rightarrow A$

$A \rightarrow BAB \mid B \mid \epsilon \mid BA \mid AB \mid A$

$B \rightarrow 0B1$

HW#5 Problem 7

2. Remove ϵ rules

Remove $A \rightarrow \epsilon$

$S_0 \rightarrow A \mid \epsilon$

$A \rightarrow BAB \mid B \mid BA \mid AB \mid A \mid BB$

$B \rightarrow 0B1$

HW#5 Problem 7

3. Remove unit rules

Remove $A \rightarrow A$

$S_0 \rightarrow A \mid \epsilon$

$A \rightarrow BAB \mid B \mid BA \mid AB \mid BB$

$B \rightarrow 0B1$

HW#5 Problem 7

3. Remove unit rules

Remove $A \rightarrow B$

$S_0 \rightarrow A \mid \epsilon$

$A \rightarrow BAB \mid BA \mid AB \mid BB \mid 0B1$

$B \rightarrow 0B1$

HW#5 Problem 7

3. Remove unit rules

Remove $S \to A$

$S_0 \to BAB \mid BA \mid AB \mid BB \mid 0B1 \mid \epsilon$

$A \to BAB \mid BA \mid AB \mid BB \mid 0B1$

$B \to 0B1$

HW#5 Problem 7

4. Split other rules

Remove $S_0 \rightarrow BABA \rightarrow BAB$

$S_0 \rightarrow BC_1 \mid BA \mid AB \mid BB \mid 0B1 \mid \epsilon$

$A \rightarrow BC_2 \mid BA \mid AB \mid BB \mid 0B1$

$B \rightarrow 0B1 \quad C_1 \rightarrow AB$

$C_2 \rightarrow AB$

HW#5 Problem 7

4. Split other rules

Remove $S \rightarrow 0B1A \rightarrow 0B1B \rightarrow 0B1$

$S_0 \rightarrow BC_1 \mid BA \mid AB \mid BB \mid C_3 1 \mid \epsilon$

$A \rightarrow BC_2 \mid BA \mid AB \mid BB \mid C_4 1$

$B \rightarrow C_5 1 \quad C_1 \rightarrow AB$

$C_2 \rightarrow AB \quad C_3 \rightarrow 0B$

$C_4 \rightarrow 0B$

$C_5 \rightarrow 0B$

HW#5 Problem 7

$S_0 \to BC_1 \mid BA \mid AB \mid BB \mid C_3 I_1 \mid \epsilon$
$A \to BC_2 \mid BA \mid AB \mid BB \mid C_4 I_2$
$B \to C_5 I_3 \quad C_1 \to AB$
$C_2 \to AB \quad C_3 \to O_1 B$
$C_4 \to O_2 B$
$C_5 \to O_3 B$
$I_1 \to 1$
$I_2 \to 1$
$I_3 \to 1$
$O_1 \to 0$
$O_2 \to 0$
$O_3 \to 0$