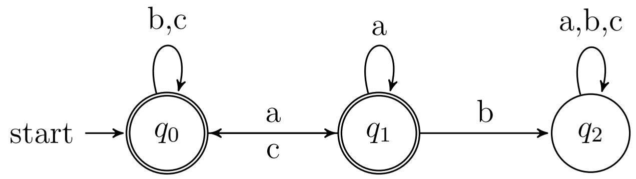
CS 360 Fall 2013
Assignment 2 - Solutions

1. (a) Consider $M = (\Sigma, Q, q_0, \delta, F)$ where $\Sigma = \{a, b, c\}$, $Q = \{q_0, q_1, q_2\}$,$F = \{q_0, q_1\}$ and δ is defined based on the table below:

Source	Input Character	Destination
q_0	a	q_1
q_0	b	q_0
q_0	c	q_0
q_1	a	q_1
q_1	b	q_0
q_1	c	q_0
q_2	a	q_2
q_2	b	q_2
q_2	c	q_2

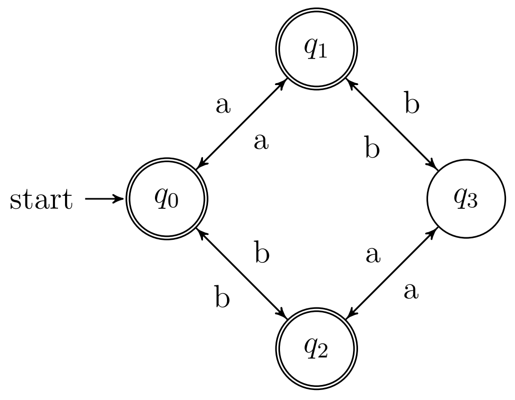
The idea is to match ab anywhere in the string. q_0 is a state that we have not matched anything. In q_1 we have matched an a and in q_2 we have already matched ab. The DFA looks like this.



(b) Consider $M = (\Sigma, Q, q_0, \delta, F)$ where $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3\}$,$F = \{q_0, q_1, q_2\}$ and δ is defined based on the table below:

Source	Input Character	Destination
q_0	a	q_1
q_0	b	q_2
q_1	a	q_0
q_1	b	q_3
q_2	a	q_3
q_2	b	q_0
q_3	a	q_2
q_3	b	q_1

The idea is to track whether n_a is even or odd (and so for n_b). So there are 4 states for combinations of these situations: q_0 for even-even, q_3 for odd-odd, and q_1 and q_2 for even-odd or odd-even cases. See below



2. (a) Let $x, y \in L(M)$ arbitrary, denote by q_f the only accepting state and by q_0 the initial state. Since x, y are accepted by M, $\delta^*(q_0, x) = \delta^*(q_0, y) = q_f$. So for any $z \in \Sigma^*$ we have

$$\delta^*(q_0, xz) = \delta^*(\delta^*(q_0, x), z) = \delta^*(q_f, z) = \delta^*(\delta^*(q_0, y), z) = \delta^*(q_0, yz),$$

so in particular xz is accepted by $L(M)$ iff yz is, that is,

$$\{z \in \Sigma^* : \ xz \in L(M)\} = \{z \in \Sigma^* : \ yz \in L(M)\}.$$

(b) Let L be defined by the regular expression $r = a{+}aa$; then L is a regular language. Assume there is a DFA, M, with only 1 accepting state, that calculates L. By part a) for any $x, y \in L$ we have that $xz \in L$ iff $yz \in L$. To get a contradiction we let $x := a \in L$, $y := aa \in L$, and $z := a$ and note that $xz = aa \in L$ but $yz = aaa \notin L$.

(c) Fix an arbitrary DFA, $M = (\Sigma, Q, q_0, \delta, F = q_1, q_2, , q_n)$. Define n DFAs with a single accepting state each:

$$A_i = (\Sigma, Q, q_0, \delta, q_i).$$

Then

$$\bigcup_{i=1}^{n} L(A_i) = \bigcup_{i=1}^{n} \{w \in \Sigma^* \mid \delta^*(q_0, w) = q_i\} = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\} = L(M),$$

where the first and last equalities are by definition, and for the middle one we have the two sided inclusion:

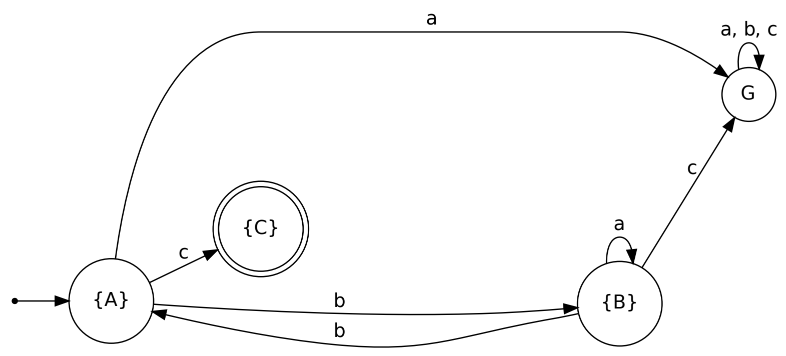
"$\subseteq$" $w \in \bigcup_{i=1}^{n}\{w \in \Sigma^* \mid \delta^*(q_0, w) = q_i\} \Rightarrow w \in \Sigma^*$ and $\exists \hat{i}$ s.t. $\delta^*(q_0, w) = q_{\hat{i}}$
$\Rightarrow w \in \Sigma^*$ and $\delta^*(q_0, w) \in F$,

"$\supseteq$" $w \in \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\} \Rightarrow \exists \hat{i}$ s.t. $\delta^*(q_0, w) = q_{\hat{i}}$
$\Rightarrow w \in \bigcup_{i=1}^{n}\{w \in \Sigma^* \mid \delta^*(q_0, w) = q_i\}$.

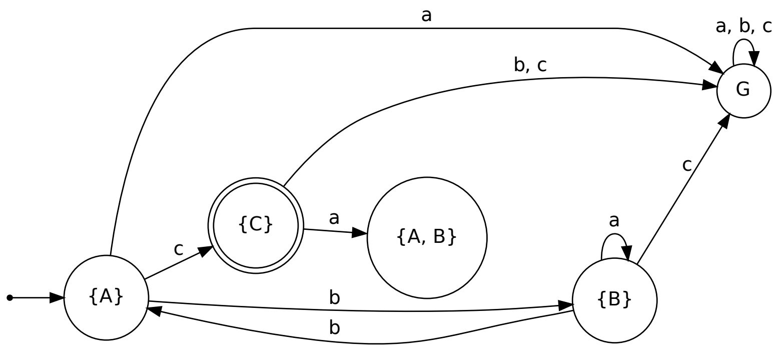
3. We follow the subset construction, where the idea is that the resulting DFA simulates the original NFA; it will keep track of what states the NFA may be in as it reads the input. Denote the original NFA by $(Q = \{A, B, C\}, \Sigma, A, C, \delta)$, the resulting DFA will be $(Q' \cup G, \Sigma, q_0, F, \delta')$, where

   - $Q' \subseteq \mathcal{P}(Q)$ for keeping track of all possible instances of the NFA;
   - G will act as a garbage/dead-end state: recall that in DFAs the transition function must always map to a state;
   - $q_0 = \{A\}$ denotes that the NFA is exactly in the A state to begin with;
   - $F = \{S \in Q' \mid C \in S\}$, that is, if any instance of the NFA is in an accepting state, the word will be accepted in the DFA;
   - for $S \in Q'$, $x \in \Sigma$ set $\delta'(S, x) = \bigcup_{q \in S} \{\delta(q, x)\}$ unless this is empty, in which case δ' takes the value of G. This definition corresponds to observing, for any state the NFA may be in, what the resulting state will be after reading input x and keeping track of this information. Since G acts as a dead end state, all input simply loops back to itself.

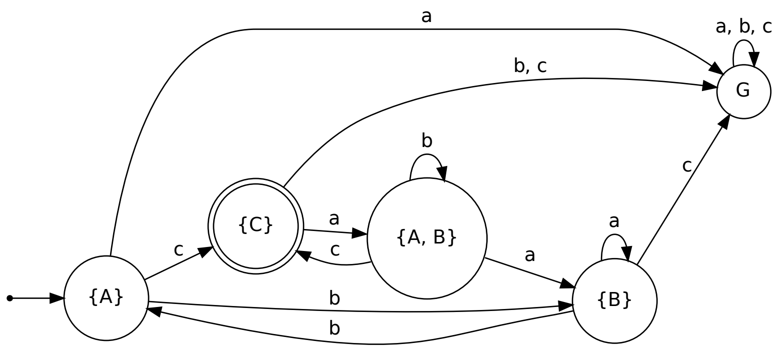
   We now build the DFA using these rules, starting from the initial state. The first few steps are very similar to the original NFA:



   Since in the NFA from the state C on input a we may transition to either state A or B we introduce a new state to represent this:



Finally, we add transitions from this new state $\{A, B\}$. To find the transition for input a we consider: $\delta(A, a) = \emptyset$ and $\delta(B, a) = B$. Hence $\delta'(\{A, B\}, a) = \{B\}$. The other transitions are calculated similarly. The resulting DFA is:



4. (a) Basically, we have a chain of states for each word. We can put these chains in parallel (and connect them in the intial state) using NFA. We should have loops for initial and accepting states to be able to ignore prefix/suffixes. We assume that λ is not a one of the words, otherwise $L = \Sigma^*$ and the solution is trivial.

   Denote by $w_i[j]$ the j'th letter of w_i. Consider $M = (\Sigma, Q, q_0, \delta, F)$ where $\Sigma = \{a, b\}$, $Q = \{q_0\} \cup \{q_{i,j}|$ for all (i, j) that $i \in \{1, 2, .., k\}, 0 < j < |w_i|\}$,$F = \{q_{1,|w_1|}, q_{2,|w_2|}, .., q_{k,|w_k|}\}$ and δ as:

   $\delta(q_0, a) = \{q_0\} \cup \{q_{i,1}|w_i[1] = a\}$
   $\delta(q_0, b) = \{q_0\} \cup \{q_{i,1}|w_i[1] = b\}$
   $\delta(q_{i,|w_i|}, a) = q_{i,|w_i|}$
   $\delta(q_{i,|w_i|}, b) = q_{i,|w_i|}$

and for $j < |w_i|$, we have $\delta(q_{i,j}, w_i[j+1]) = q_{i+1,j}$

The number of states is $|w_i|$ for each chain in addtion to the intial state, which adds up to $1 + \sum |w_i|$.

(b) The idea is to construct a forward chain to accept the word, where we have loops at the terminal (i.e., accept) state to allow suffixes. Furthermore, we need to define transitions other than the main chain for the cases that we encounter other characters in the text. In this case, we cannot simply restart at the intial state, because in this case we may miss some characters that have been encountered (and have been a prefix of w_1). Therefore, we should jump back to the longest suffix of the current input that is also a prefex of w_1.

We assume that $w_1 \neq \lambda$, otherwise $L = \Sigma^*$ and the solution is trivial. Consider $M = (\Sigma, Q, q_0, \delta, F)$ where $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, ..., q_{|w_1|}|\}$,$F = \{q_{|w_1|}\}$ and δ as:

$\delta(q_{|w_1|}, a) = q_{|w_1|}$
$\delta(q_{|w_1|}, b) = q_{|w_1|}$
for $0 \leq i < |w_1|$, we have $\delta(q_i, w_1[i+1]) = q_{i+1}$
for $0 \leq i < |w_1|$, we have $\delta(q_i, \Sigma \setminus w_1[i+1]) = q_k$ where k is the length of the longest prefix of w_1 that is also a suffix of $w[1].w[2]....w[i].(\Sigma \setminus w[i+1])$

The number of states is clearly $1 + |w_1|$.

(c) Let P be the set of all prefixes of all of the words, that is $P = \{u | \exists v, uv \in \bigcup w_i\}$. First of all, we can bound the size of this set.

$|P| = |\{u | \exists v, uv \in \bigcup_i w_i\}|$
$= |\bigcup_i \{u | \exists v, uv \in w_i\}|$
$= 1 + |\bigcup_i \{u | \exists v, uv \in w_i, |u| > 0\}|$
$\leq 1 + \sum_i |\{u | \exists v, uv \in w_i, |u| > 0\}|$
$= 1 + \sum_i |w_i|$

where we separated the empty prefix from other prefixes, and the inequality comes from the union bound.

We define the DFA based on P. In particular, we assume a one-to-one correspondance between P and Q (the state set). With a little bit **abuse of notation**, we use P as the state set and its members as the states. The terminal (i.e., accept) states are $F = \{p \in P | \exists w_i, p = w_i\}$. Now, we need to define the transition function.

for $p \neq w_i$: $\delta(p, \sigma) = argmax(|u|)$ where $u \in \Sigma^*$ and $\exists v \in \Sigma^*, vu = p.\sigma$.

for $p = w_i$ for any i: $\delta(p, \sigma) = p$

We just need a preprocess: if a word $w_i = uw_jv$, we should remove w_j. This preprocess can only reduce the number of states.

Note that if this DFA goes to an accept state, it shows that we have encountered the word in the suffix of the output (and from now on it is ok to ignore other input characters and accept the word). Furthermore, the DFA accepts the language that we want: consider the word uw_iv where w_i is one of the words in our set. After encountering uw_i, DFA should be in an accept states. Otherwise, it has already matched the longest prefix, which should be w_i (because there is no word w_j in the set that $w_j = uw_iv$).