CISS362: Introduction to Automata Theory, Languages, and Computation
Test 1 Part B

We fix an alphabet Σ throughout. The following are facts you have already seen.

Concatenation. For the following x, y, z are words in Σ^*.

C1 xy is a word in Σ^*

C2 $(xy)z$ can be rewritten as $x(yz)$

C3 $x(yz)$ can be rewritten as $(xy)z$

C4 ϵx can be rewritten as x

C5 x can be rewritten as ϵx

C6 $x\epsilon$ can be rewritten as x

C7 x can be rewritten as $x\epsilon$

C8 If $x \in \Sigma^*$ is not ϵ, then $x = x'x''$ for some $x' \in \Sigma$ and $x'' \in \Sigma^*$.

Length function. Let x be a word over Σ^*.

L1 If $x = \epsilon$, then $|x| = 0$.

L2 If $x \neq \epsilon$, then $x = x' \cdot x''$ where $x \in \Sigma$ (i.e. x is a symbol in our alphabet) and $x'' \in \Sigma^*$ (i.e. x'' is a word over Σ). Then $|x| = 1 + |x''|$.

L3 $|x| \geq 0$

L4 $|x|$ is an integer

L5 If $|x| = 0$, then $x = \epsilon$.

L6 $|x| \neq 0 \iff x \neq \epsilon$

L5-L6 were proven in an earlier assignment. L6 is essentially L1 and L5.

Reverse function. Let $x \in \Sigma^*$.

R1 If $x = \epsilon$, then $\epsilon^R = \epsilon$.

R2 If $x \neq \epsilon$, then $x = x' \cdot x''$ where $x' \in \Sigma$ (x' is a symbol in our alphabet Σ) and $x'' \in \Sigma^*$ (i.e., x'' is a word over Σ) and $x^R = (x'')^R \cdot x'$.

R3 If x, y are words in Σ^*, then $(xy)^R = y^R x^R$

R3 was proven in an earlier assignment.

Read this document very carefully. I'll be using very formal notation for DFA. Remember that I prefer to write the formal definition of a DFA as $(\Sigma, Q, q_0, \delta, F)$. This is slightly different from your textbook which begins with Q instead.

The goal of this assignment is to formalize acceptance. Our definition of DFA acceptance, i.e., "start at q_0, read one character at a time left-to-right, following the transition edge; if you stop at an accept state then the word is accepted, otherwise it's not" is correct but not precise enough for a clear proof.

We will not formalize the above definition.

First of all the action of "follow the transitions" is done using the edges (pictorially) or using the transition function. The fact

"at state q if we read symbol a we go to q'"

is the same as

$$\delta(q, a) = q'$$

The two are the same. However the transition function is better suited for a clear and formally correct proof.

The transition function only describes one step in the computation. Here "computation" means "go from one state to another". Acceptance is carrying out a sequence of computations. So the first thing is to extend the concept of one computation to a sequence to computation until all symbols in a string is completely processed. We will therefore define a function

$$\delta^*$$

(In some books this is denoted $\widehat{\delta}$.) Basically this is what we want δ^* to do: Suppose a δ function does this for a word aba:

$$\begin{aligned}\delta(q_0, a) &= q_5\\ \delta(q_5, b) &= q_3\\ \delta(q_3, a) &= q_2\end{aligned}$$

and q_2 is an accept state, then we accept aba. Pictorially this means

$$\begin{tikzcd} {} \arrow[r] & q_0 \arrow[r, "a"] & q_5 \arrow[r, "b"] & q_3 \arrow[r, "a"] & q_2 \end{tikzcd}$$

(other parts of the DFA diagram is of course not shown.) If that's the case, then for this δ we want the δ^* to do this

$$\delta^*(q_0, aba) = q_2$$

In other words, δ^* basically computes the resulting state after consuming all the symbols in the word.

Obviously given a δ, the definition of δ^* depends on δ. Specifically, we will define it recursively. First if there's nothing to read of course we have

$$\delta^*(q, \epsilon) = q$$

for every state q in the DFA. We have already handled the case where $|x| = 0$. Therefore we only need to look at the case where $|x| > 0$. In that case $x = x'x''$ where x' is a symbol and x'' is a word. To model the behavior of "processing one symbol at a time going left-to-right and following δ", the only reasonble definition of δ^* is

$$\delta^*(q, x'x'') = \delta^*\Bigg(\delta(q, x'),\ \ x''\Bigg)$$

[If there's exactly one symbol to process, of course δ^* is exactly the same as δ:

$$\delta^*(q, c) = \delta(q, c)$$

for any state q and any symbol c in the alphabet of the DFA. You will be proving this later.] Finally we want to define

$$\delta^*(q, x)$$

where x is a a word.

Make sure you really understand the above definition!

In summary, given a function

$$\delta : Q \times \Sigma \to Q$$

we define the corresponding δ^* to be the function

$$\delta^* : Q \times \Sigma^* \to Q$$

by

$$\delta^*(q, x) = \begin{cases} q & \text{if } x = \epsilon \\ \delta^*(\delta(q, x'), x'') & \text{if } x = x'x'', x' \in \Sigma, x'' \in \Sigma^* \end{cases}$$

The function δ^* is defined recursively in terms of δ.

Let me state formally the properties of δ^* for the sake of your proof writing:

Let $\delta : Q \times \Sigma \to Q$ be a function. We define an associated function

$$\delta^* : Q \times \Sigma^* \to Q$$

recursively so that:

DS1 $\delta^*(q, \epsilon)$ can be rewritten as q.

DS2 q can be rewritten as $\delta^*(q, \epsilon)$.

DS3 If $x' \in \Sigma$ and $x'' \in \Sigma^*$, then $\delta^*(q, x'x'')$ can be rewritten as $\delta^*(\delta(q, x'), x'')$.

DS4 If $x' \in \Sigma$ and $x'' \in \Sigma^*$, then $\delta^*(\delta(q, x'), x'')$ can be rewritten as $\delta^*(q, x'x'')$.

Your are strongly advised to test your understanding of the above definition of δ^* with the following example:

You are given the following DFA M where $\Sigma = \{a, b\}$, $Q = \{q_0, q_1, q_2, q_3\}$, the initial state is q_0, $F = \{q_0, q_1, q_2\}$, and δ is given by

$$\begin{aligned}
\delta(q_0, a) &= q_0 \\
\delta(q_0, b) &= q_1 \\
\delta(q_1, a) &= q_0 \\
\delta(q_1, b) &= q_2 \\
\delta(q_2, a) &= q_0 \\
\delta(q_2, b) &= q_3 \\
\delta(q_3, a) &= q_3 \\
\delta(q_3, b) &= q_3
\end{aligned}$$

Compute $\delta^*(q_0, abab)$, $\delta^*(q_1, abbb)$ using DS1-DS4 and of course δ, listing down carefully which of the above DS1-DS4 you are using for each step of the computation. When you're done, draw the DFA and check that your computation is correct.

Again, just like the case of the reverse function and the length function, you can very quickly write a δ^* function given a δ. For instance in C++, if states are represented using integers, it might look like this where verb!delta! is a transition function already defined:

```
int delta(int q, char c)
{
    ...
}

int deltastar(int q, const char s[])
{
    if (strlen(s) == 0)
    {
        return q;
    }
    else
    {
        return deltastar(delta(q, s[0]), s + 1);
    }
}
```

Acceptance

With this precise definition of δ^* that computes the resulting state for a word (and not just a symbol), we can formalize the concept of acceptance: A word $x \in \Sigma^*$ is accepted by a DFA $M = (\Sigma, Q, q_0, \delta, F)$ if

$$\delta^*(q_0, x) \in F$$

This acceptance criteria is defined in terms of our δ^* which depends on δ. This is now formally our definiton of acceptance. From this we can quickly define the concept of the language accepted by our DFA M:

$$L(M) = \{x \in \Sigma^* \mid \delta^*(q_0, x) \in F\}$$

Let's list the following for proof writing. Given a DFA $M = (\Sigma, Q, q_0, \delta, F)$, we have:

AC1 "x is accepted by M" can be replaced by $\delta^*(q_0, \epsilon) \in F$.

AC2 $\delta^*(q_0, \epsilon) \in F$ can be replaced by "x is accepted by M"

AC3 $L(M)$ can be replaced by $\{x \mid \delta^*(q_0, x) \in F\}$

AC4 $\{x \mid \delta^*(q_0, x) \in F\}$ can be replaced by $L(M)$

Here's a warmup:

Q1. Let $(\Sigma, Q, q_0, \delta, F)$ be a DFA and $q \in Q$ be fixed. Prove that if $x, y \in \Sigma^*$, then

$$\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$$

[Make sure you state the $P(n)$, the smallest value of n you need for $P(n)$ to hold. Make sure you prove the base case. Make sure you prove the inductive case. Make sure your proof is clear, concise, correct and refers to only basic axioms/properties listed. Follow the format of previous proof problems. Proper math writing includes proper writing in general.

If you need some fact but can't prove it, you might ask if I'm willing to include it as a list of facts that you can quote. That's not a promise that I'll say yes.]

SOLUTION. We will prove the above statement by Mathematical Induction. For $n \geq ?$, we define

$$P(n): \text{ If ?, then ?}$$

<u>Base case.</u> ?

<u>Inductive case.</u> ?

Hence by Mathematical Induction for all $n \geq ?$

$$?$$

We conclude that if $x, y \in \Sigma^*$, then

$$\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$$

QED.

Let $M' = (\Sigma, Q', q'_0\delta', F')$ and $M'' = (\Sigma, Q', q'_0\delta', F')$ be two DFAs. From theδ' from M', we have δ'^*. Likewise we have a δ''^* from the δ'' of M''. In both the union and intersection construction we define a $\delta : (Q' \times Q'') \times \Sigma \to (Q' \times Q'')$ to be

$$\delta((q', q''), c) = (\delta'(q', c), \delta''(q'', c))$$

for $q' \in Q'$, $q'' \in Q''$, and $c \in \Sigma$. Note that δ depends on δ', δ''.

This involves the cross product. Here are some basic facts about the cross product:

CP1 $(x, y) \in X \times Y$ can be replaced by $x \in X, y \in Y$.

CP2 $x \in X, y \in Y$ can be replaced by $(x, y) \in X \times Y$.

CP3 $(x, y) = (x', y')$ can be replaced by $x = x', y = y'$

CP4 $x = x', y = y'$ can be replaced by $(x, y) = (x', y')$.

You will be proving something about the intersection construction so here are some basic things about intersections:

IN1 $x \in X \cap Y$ can be replaced by $x \in X, x \in Y$.

IN2 $x \in X, x \in Y$ can be replaced by $x \in X \cap Y$

[Don't forget that in CS and Math, a comma "," means "and".]

Q2. Assuming the given M', M'' above. **Suppose δ' is the transition for M' and δ'' is the transition function for M''. We define δ by**

$$\delta((q', q''), c)$$

where $q' \in Q'$, $q'' \in Q''$ and $c \in \Sigma$. Associated to $\delta', \delta'', \delta''$, we have the functions $\delta'^*, \delta''^*, \delta^*$ respectively. Prove that if $q' \in Q', q'' \in Q'', x \in \Sigma^*$, then

$$\delta^*((q', q''), x) = (\delta'^*(q', x), \delta''^*(q', x))$$

[Formulate a $P(n)$ and prove the above by mathematical induction.

Although not necessary, it might be a good idea to prove the following Lemma: If δ is any transition function and δ^* the associated function, then for $c \in \Sigma$, we have $\delta^*(q, c) = \delta(q, c)$ for any $q \in Q$. A Lemma is a small fact that is used in a more important theorem. It's equivalent to the helper function for programmers. The proof of this Lemma is about 3 computations long. The Lemma is used twice in the proof of the result, so if you don't have this fact, you would have to more or less go through the proof of the Lemma twice in the proof of the actual statement that we want to prove. So the Lemma simply allows us to cleanup the main proof.]

SOLUTION. Before we prove the above statement we will prove the following Lemma that we use later:

Lemma. Let $\delta : Q \times \Sigma \to Q$ be any transition function and $\delta^* : Q \times \Sigma^* \to Q$ be the associated function. Then if $q \in Q$ and $c \in \Sigma$

$$\delta^*(q, c) = \delta(q, c)$$

Proof of Lemma. Let $c \in \Sigma$ (i.e. $c \in \Sigma^*$ of length 1). We have the following:

$$\begin{aligned} \delta^*(q, c) &= \delta^*(?, c\epsilon) && \text{by ?} \\ &= \delta^*(?, \epsilon) && \text{by ?} \\ &= \delta(q, c) && \text{by ?} \end{aligned}$$

The Lemma is proved. QED.

We will prove the above statement by Mathematical Induction. For $n \geq ?$, we define

$$P(n): \text{ If } x \in \Sigma^* \text{ with } |x| = n, \text{ then ...}$$

<u>Base case.</u> ?

Inductive case. ?

Hence by Mathematical Induction for all $n \geq ?$

$$?$$

We conclude that if $x \in \Sigma^*$, then

$$?$$

With the result in Q2 we can now prove that the intersection construction is absolutely true for any pair of DFAs in the following sense:

Q3. Let $M' = (\Sigma, Q', q_0', \delta', F')$ and $M'' = (\Sigma, Q'', q_0'', \delta'', F'')$ be two DFAs. Define a new DFA $M = (\Sigma, Q' \times Q'', (q_0', q_0''), \delta, F' \times F'')$ where

$$\delta((q', q''), c) = (\delta'(q', c), \delta''(q'', c))$$

for $q' \in Q'$, $q'' \in Q''$, and $c \in \Sigma$. Then

$$L(M) = L(M') \cap L(M'')$$

[Recall: Two sets X and and Y are the same i.e. $X = Y$, if $X \subseteq Y$ and $Y \subseteq X$. This is the formal definition of set equality. To prove that $X \subseteq Y$, you need to prove $x \in X \implies x \in Y$. To prove that $Y \subseteq X$, you need to prove $x \in Y \implies x \in X$.]

SOLUTION. Let $x \in \Sigma^*$. If $x \in L(M)$ we have the following:

$$\begin{aligned}
& x \in L(M) & \\
\therefore\ & \delta^*((q_0', q_0''), x) \in F' \times F'' & \text{by ?} \\
\therefore\ & ? \in F' \times F'' & \text{by ?} \\
\therefore\ & ? \in F' \text{ and } ? \in F'' & \text{by ?} \\
\therefore\ & x \in L(M') \text{ and } x \in L(M'') & \text{by ?} \\
\therefore\ & x \in L(M') \cap L(M'') &
\end{aligned}$$

We have shown

$$L(M) \subseteq L(M') \cap L(M'')$$

Now we will show that $L(M') \cap L(M'') \subseteq L(M)$.

If $x \in L(M') \cap L(M'')$ then we have the following:

$$\begin{gathered}
x \in L(M') \cap L(M'') \\
?
\end{gathered}$$

We have shown

$$L(M') \cap L(M'') \subseteq L(M)$$

We conclude that

$$L(M) = L(M') \cap L(M'')$$

QED.