Q1. Review the formal definition of a DFA (see p 35 in Sipser 2nd/3rd editions). Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. True or False: If $F$ contains at least one state, the language of $M$ must be non-empty.

- True
- False

**Feedback**

False. There is nothing in the definition that would require there are any arrows that point into the accept states. In more formal terms, it may be that the transition function $\delta$ does not lead to any computations that go through an accept state.
Q2. Consider the following DFA machine. How many different strings does it accept?

- None
- 1
- 2
- 3
- Infinitely many (but not necessarily every string)

Feedback:

This DFA will accept precisely two different strings: $\varepsilon$ (the empty string) and $ab$. The machine accepts $\varepsilon$ because the start state is an accept state (nothing in the definition prevents this). The machine accepts $ab$ because when computing on input $ab$ the machine will enter this sequence of states: $q_0$, $q_1$, $q_2$ where $q_2$ is an an accept state.
Q3. Consider the following DFA over the alphabet \(\{a, b\}\), in which we have not yet marked any accept states. In its current state the language of this DFA would be the empty set. If we want to modify the machine such that it will accept precisely those strings that contain either one or two a's somewhere, which states should be included in \(F\), the set of accept states?

\[ F = \{q_1, q_2, q_3\}. \]

The machine will be in state if \(q_0\) if it has seen only b's (and thus no a's) so we do not make \(q_0\) an accept state. The machine will be in state \(q_1\) if it has seen 0 or more b's followed by one a; it will be in \(q_3\) if it has seen one a so far but has seen b's since then. So we will include both \(q_1\) and \(q_3\) in \(F\). The machine will be in state \(q_2\) if it has seen two a's so far, so we include \(q_2\) in \(F\). Finally, the machine will be in \(q_4\) if it has seen three or more a's so we do not include \(q_3\).

Feedback

We will let \(F = \{q_1, q_2, q_3\}\). The machine will be in state if \(q_0\) if it has seen only b's (and thus no a's) so we do not make \(q_0\) an accept state. The machine will be in state \(q_1\) if it has seen 0 or more b's followed by one a; it will be in \(q_3\) if it has seen one a so far but has seen b's since then. So we will include both \(q_1\) and \(q_3\) in \(F\). The machine will be in state \(q_2\) if it has seen two a's so far, so we include \(q_2\) in \(F\). Finally, the machine will be in \(q_4\) if it has seen three or more a's so we do not include \(q_3\).
Q4. Let A and B be regular languages. Let A -- B consist of all the strings in A which are not in B (this is sometimes written A\B and is also called the set difference). Is A -- B regular? *

- Yes, always
- No, not necessarily

Feedback

Yes, if A and B are regular languages then A -- B is always regular. Note that the definition of A -- B (the set of strings which are both in A and also not in B) is the same as the intersection of A and the complement of B. We saw in lecture that regular languages are closed under both the complement operation and under intersection. Therefore A -- B is also regular.
Q5. Let $A = \{a, ab\}$ and $B = \{a, ba\}$ be two languages. How many strings are in $A \circ B$?

1. 1
2. 2
3. 3
4. 4
5. 8

Feedback

$A \circ B = \{ xy \mid x \in A \text{ and } y \in B \} = \{ aa, aba, abba \}$ so there are three strings in the concatenation. Observe that there are four ways to concatenate the strings in $A$ with the strings in $B$:

- if $x = a$ and $y = a$ then $xy = aa$.
- if $x = a$ and $y = ba$ then $xy = aba$.
- if $x = ab$ and $y = a$ then $xy = aba$.
- if $x = ab$ and $y = ba$ then $xy = abba$.

However there are only three unique strings above, so the concatenation $A \circ B$ is a set containing just three strings.