Reminder

- Finite automata (FA) is formally represented as a 5-tuple of $(Q, \Sigma, \delta, q_0, F)$.
- Language $A$ is regular if there exists some finite automata that recognizes it.
- Regular languages are closed under regular operations: If $A, B$ are regular, then so is their union $(A \cup B)$, concatenation $(A \circ B = AB)$, and star $(A^*)$.
- Non-Regular languages:
  - $\{1^n0^n \mid n \geq 0\}$, and
  - $\{w \mid w$ has $=$ numbers of 0s and 1s$\}$.

Example 1 — Even number of $a$’s

Let $A = \{w \mid w$ has even number of $a$’s$\}$, where $\Sigma = \{a, b\}$. Prove that $A$ is regular.

Solution

The idea is to keep track whether there are **even** or **odd** number of $a$’s read so far in the string $w$. This implies that we will have 2 states $q_0$ and $q_1$, where $q_0$ contains even number of $a$’s and $q_1$ contains odd number of $a$’s. Reading an alphabet $a$ will alternate between $q_0$ and $q_1$, whereas reading $b$ will not change the state.

Here is an automaton that recognizes $A$:

```
start  ↓
         ↓
q0      ↓
       ↓      ↓
a      a
         ↓      ↓
         ↓      ↓
         ↓      ↓
q1      ↓
       ↓      ↓
         ↓      ↓
         ↓      ↓
         ↓      ↓
b
```

What about the empty string? An empty string has 0 number of $a$’s, which is even, and so our automaton accepts an empty string.

Example 2 — Number of $a$’s is a multiple of 3

Let $B = \{w \mid$ number of $a$’s in $w$ is a multiple of 3$\}$, where $\Sigma = \{a, b\}$. Prove that $B$ is regular.
Solution

We use the same idea as Example 1. Instead of even/odd, we now keep track of number of a’s in remainder modulo 3. To be precise, we will have 3 states, each representing remainder 0,1, and 2 when divided by 3. Reading an a will shift from remainder 0 to 1, 1 to 2, and 2 back to 0. Reading b won’t change state. The automaton can be drawn as follows:

Example 3 — Number of a’s is a multiple of n

Let \( C = \{w \mid \text{number of a’s in } w \text{ is a multiple of } n\} \), where \( \Sigma = \{a,b\} \). Prove that \( C \) is regular.

Solution

Again, we use the same trick of keeping track of remainder modulo \( n \). Here we show a formal description of an automaton that recognizes \( C \):

- \( n \) states: \( Q = \{q_0, q_1, \ldots, q_{n-1}\} \), where \( q_i \) represents remainder \( i \) modulo \( n \).
- starting state: \( q_0 \)
- accepting states: \( F = \{q_0\} \). Note that \( F \) is a set.
- transition: For each \( i = 0,1,\ldots,n-1 \), we have
  - \( \delta(q_i, b) = q_i \)
  - \( \delta(q_i, a) = q_{i+1}, \text{ where } q_n = q_0 \)
- alphabet: \( \Sigma = \{a,b\} \) is given.

Can we construct a different automaton that recognizes the same language \( C \)? The answer is yes. We can simply keep track of remainder modulo \( 2n \) instead and accept any strings that has either remainder 0 or \( n \) modulo \( 2n \). That is, our automaton will have \( 2n \) states \( Q = \{q_0, q_1, \ldots, q_{2n-1}\} \). However, the accepting states will now be \( F = \{q_0, q_n\} \) instead. The rest of the components in the 5-tuple will be straightforwardly similar to the \( n \)-state FA described above.

Example 4 — Single string

Let \( w \) be an arbitrary string, using alphabets from \( \Sigma \). Let \( D = \{w\} \). Show that \( D \) is regular.
Solution

Let $w = w_1 w_2 w_3 \ldots w_k$. Since we only have 1 string to check with, we can simply check if the first letter is $w_1$, second letter is $w_2$, and so on. That is, we should have sequences of states $q_0, q_1, \ldots, q_k$, where $q_i$ represents the first $i$ letters read is $w_1 w_2 \ldots w_{i-1}$. By reading $w_i$ at state $q_i$, we get to the next state $q_{i+1}$. However, if we read other alphabet $x \neq w_i$ at state $q_i$, we will never be correct. Thus, we must have an extra stuck state $q_{\text{reject}}$, where strings entering this state will always get rejected from that point on. So, our formal construction will look like:

- $k + 2$ states: $Q = \{q_0, q_1, \ldots, q_k, q_{\text{reject}}\}$.
- starting state: $q_0$
- accepting states: $F = \{q_k\}$.
- transition: For each $i = 1, 2, \ldots, k$,
  - $\delta(q_{i-1}, w_i) = q_i$
  - $\delta(q_{i-1}, x) = q_{\text{reject}}$, for all $x \neq w_i$
  - $\delta(q_{\text{reject}}, x) = q_{\text{reject}}$, for all $x$
- alphabet: $\Sigma$ is given.

Example 5 — Finite set

Let $E$ be a finite set of strings. Show that $E$ is regular.

Solution 1

The first solution simply uses closure property of regular languages. We have shown in Example 4 that every set of single element is regular, and we have already shown in class that regular languages are closed under union. We can then easily show by induction that any set of size 1,2,3,... are also regular, which completes the proof.

Does this induction imply that infinite set is also regular (since infinite set can also be written as unions of singletons)? Unfortunately no. Closure property does not allow infinite unions.

Solution 2

The second solution will directly construct an automaton. The key idea is to create states that will cover every possible strings in the set $E$. That is, if we let $k$ be the length of the longest string in $E$, we create a graph of depth $k + 1$ where each node has expands every possible alphabet. Then, given each string in $E$, we can walk down the tree and mark the final state as an accepting state. Note that we also need $q_{\text{reject}}$ state for the rest of the transitions to go there and get stuck in reject state forever.

Here is an example automaton diagram when $E = \{aba, ba, abb\}$, and $\Sigma = \{a, b\}$.