In class quiz:

Give a DFA to the following language:

\[ L(D) = \{ w \mid w \text{ has } 3 \text{ zeros or } 2 \text{ ones and } \Sigma = \{0,1\} \} \]

Answer:

[Wylie’s obviously biased DFA: Use of imagination required]

Bias against DFA’s aside, here is a simple and beautiful DFA solution:
To illustrate the difference between a DFA and an NFA, here is the simple NFA solution:

![NFA Diagram]

**Lecture:**

Question: How do you know which state to transition to, given the same option in two different outward directions? In this case: the outward direction from $q_1$ via $\varepsilon$.

*(Sipser p48)*

How does an NFA compute? Suppose that we are running an NFA on an input string and come to a state with multiple ways to proceed. For example, say that we are in state $q_1$ in NFA $N_1$ and that the next input symbol is a 1. After reading that symbol, the machine splits into multiple copies of itself and follows all the possibilities in parallel. Each copy of the machine takes one of the possible ways to proceed and continues as before. If there are subsequent choices, the machine splits again. If the next input symbol doesn’t appear on any of the arrows exiting the state occupied by a copy of the machine, that copy of the machine dies, along with the branch of the computation associated with it. Finally, if any one of these copies of the machine is in an accept state at the end of the input, the NFA accepts the input string.

Question: What does the $\varepsilon$ symbol mean?

*(Sipser p48)*

If a state with an $\varepsilon$ symbol on an exiting arrow is encountered, something similar happens (as explained above). Without reading any input, the machine splits into multiple copies, one following each of the exiting $\varepsilon$ labeled arrows and one staying at the current state. Then the machine proceeds non-deterministically as before.
Let’s consider some sample runs of the NFA $N_1$ shown in Figure 1.27. The computation of $N_1$ on input 010110 is depicted in the following figure.
\[ L(N_t) = \{ w \mid w \text{ contains either } 101 \text{ or } 11 \text{ as a substring} \} \]

**Figure 1.27 (Sipser p48)**
The nondeterministic finite automaton \( N_1 \)

Transition examples for \( N_1 \):

1. \( q_1 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \)
2. \( q_1 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \)
3. \( q_1 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow \text{die} \)

**Definition 1.37 (Sipser p53) (Maheshwari p39)**
A nondeterministic finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \( Q \) is a finite set of states,
2. \( \Sigma \) is a finite alphabet,
3. \( \delta: Q \times \Sigma \rightarrow P(Q) \) is the transition function,
4. \( q_0 \in Q \) is the start state, and
5. \( F \subseteq Q \) is the set of accept states.

For any set \( Q \) we write \( P(Q) \) to be the collection of all subsets of \( Q \). Here \( P(Q) \) is called the power set of \( Q \).

For any alphabet \( \Sigma \) we write \( \Sigma_\varepsilon \) to be \( \Sigma \cup \{ \varepsilon \} \).
Example 1.38 (Sipser p54)
Recall the NFA $N_1$:

The formal description of $N_1$ is $(Q, \Sigma, \delta, q_0, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$
2. $\Sigma = \{0, 1\}$
3. $\delta$ is given as

<table>
<thead>
<tr>
<th>State</th>
<th>0</th>
<th>1</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>${q_1}$</td>
<td>${q_1, q_2}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>${q_3}$</td>
<td>$\emptyset$</td>
<td>${q_3}$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$\emptyset$</td>
<td>${q_4}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>${q_4}$</td>
<td>${q_4}$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

4. $q_1$ is the start state, and
5. $F = \{q_4\}$
Figure 1.34 (Sipser p52)
The nondeterministic finite automaton $N_3$

$\Sigma = \{0\}$

$A = \{0^k : 0 = k \mod 2 \text{ or } 0 = k \mod 3\}$

$A_1 = \{0^k : 0 = k \mod 2\}$

$A_2 = \{0^k : 0 = k \mod 3\}$

$A = A_1 \cup A_2$
Formal definition of computation (DFA) (Sipser p40)

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1w_2 ... w_n$ be a string where each $w_i$ is a member of the alphabet $\Sigma$. Then $M$ accepts $w$ if a sequence of states $r_0, r_1, ..., r_n$ in $Q$ exists with three conditions:

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, ..., n - 1$
3. $r_n \in F$

Condition 1 says that the machine starts in the start state.

Condition 2 says that the machine goes from state to state according to the transition function.

Condition 3 says that the machine accepts its input if it ends up in an accept state.

We say that $M$ recognizes language $A$ if $A = \{ w \mid M$ accepts $w\}$

Formal definition of computation (NFA) (Sipser p54)

The formal definition of computation for an NFA is similar to that for a DFA.

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and $w$ a string over the alphabet $\Sigma$. Then we say that $N$ accepts $w$ if we can write $w$ as $w = y_1y_2 ... y_m$, where each $y_i$ is a member of $\Sigma_{\epsilon}$ and a sequence of states $r_0, r_1, ..., r_m$ exists in $Q$ with three conditions:

1. $r_0 = q_0$
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, ..., m - 1$
3. $r_m \in F$

Condition 1 says that the machine starts out in the start state.

Condition 2 says that state $r_{i+1}$ is one of the allowable next states when $N$ is in state $r_i$ and reading $y_{i+1}$. Observe that $\delta(r_i, y_{i+1})$ is the set of allowable next states and so we say that $r_{i+1}$ is a member of that set.

Condition 3 says that the machine accepts its input if the last stats is an accept state.