DISCLAIMER: These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

- Comments from last time:
  1. Missing transitions?

- Today’s topics:
  1. Review of NFAs and how to simulate them.
  2. Building DFAs from NFAs.
  3. Nondeterministic transitions for union, concatenation, and star.

Last time, we formally defined an non-deterministic finite automaton (NFA). It is still a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), except the transition function \(\Sigma\) now maps to a set of states. Formally:

1. \(Q\) is a finite set of states.
2. \(\Sigma\) is a finite alphabet.
3. \(\delta : Q \times (\Sigma \cup \epsilon) \mapsto \mathcal{P}(Q)\) is the transition function (~\(\mathcal{P}(Q)\) is the power-set of \(Q\)).
4. \(q_0 \in Q\) is the starting state.
5. \(F \subseteq Q\) is the set of accept states.

Note that the only difference from a DFA is in \(\sigma\). The first difference is that the transitions can accept \(\epsilon\). We sometimes refer to the \(\epsilon\)-inclusive alphabet as \(\Sigma_\epsilon\). Secondly, each transition can now lead to multiple, including possibly, none, states.

On the diagrams, we put an arrow \(q_1 \rightarrow q_2\) for some \(a \in \Sigma_\epsilon\) if

\[ q_2 \in \delta(q_1, x). \]
1 From NFAs to DFAs

The most important fact about NFAs is that the languages accepted by them is precisely the regular languages.

A DFA is a NFA in that there are no $\epsilon$ transitions, and all transitions go to exactly one state.

So we need to show that any NFA has an equivalent DFA that accepts the same states.

Recall that an NFA $N = (Q, \Sigma, \delta, q_0, F)$ accepts a string $w$ if

- we can write $w$ as $w = y_1y_2 \cdots y_m$, where each $y_i \in \Sigma$;
- there exists a sequence of states $r_0r_1 \cdots r_m$ such that
  - $r_0 = q_0$
  - $r_m \in F$
  - $r_{i+1} \in \delta(r_i, y_{i+1})$

The way that we check whether a string is accepted is similar to reachability check on graphs: we create a set of all reachable states after we have examined $y_1 \ldots y_i$. That is, we successively update

$$R_{i+1} \leftarrow \bigcup_{r \in R_i} \delta(r, y_{i+1}).$$

For example, for the NFA with two states $Q = \{q_0, q_1\}$, $F = \{q_0\}$, $\Sigma = \{0, 1\}$, and transitions

\[
\begin{array}{c|cc}
Q \backslash \Sigma & 0 & 1 \\
\hline
q_0 & \{q_0, q_1\} & \{q_1\} \\
q_1 & \{q_0\} & \emptyset \\
\end{array}
\]

on the input $y = 01101$, we get

- $R_0 = \{q_0\}$
- $R_1 = \{q_0, q_1\}$
- $R_2 = \{q_1\}$
- $R_3 = \{q_0\}$
- $R_4 = \{q_0, q_1\}$
- $R_5 = \{q_1\}$

And because $R_5 \cap F = \emptyset$, this NFA does not accept this input.

Formally, the $R_i$s sets is precisely how we convert this NFA into a DFA. This is because the $R_i$s are all members of $P(Q)$, the power set of $Q$. Furthermore, for each $R \in P(Q)$, we get:

$$\delta(R, y) = \{\delta(r, y) \mid r \in R\}.$$  \hspace{1cm} (1)

And the new starting state is $\{q_0\}$, while the accepting state is all $R \in P(Q)$ with non-empty intersection with $F$. 
Going with the example above, we get a DFA with states

\[ \emptyset, \{q_0\}, \{q_1\}, \{q_0, q_1\}. \]

Its starting state is \(\{q_0\}\), and its accepting states are \(\{\{q_1\}, \{q_0, q_1\}\}\). Finally, we can work out its transitions using Equation 1.

<table>
<thead>
<tr>
<th>( P(Q) \setminus \Sigma )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( {q_0} )</td>
<td>( {q_0, q_1} )</td>
<td>( {q_1} )</td>
</tr>
<tr>
<td>( {q_1} )</td>
<td>( {q_0} )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( {q_0, q_1} )</td>
<td>( {q_0, q_1} )</td>
<td>( {q_1} )</td>
</tr>
</tbody>
</table>

2 Regular Operations with NFAs

The three regular operations that we discussed can also be applied to languages recognized by non-deterministic automatas (NFAs). In each case, the change that we need to make is significantly more straight-forward with NFAs compared with DFAs.

1. \( A_1 \cup A_2 \): a new ‘global’ starting state \( q_0 \) with \( \epsilon \)-transitions to \( q_1 \), the starting state for \( M_1 \), as well as \( q_2 \), the starting state for \( M_2 \).

2. \( A_1 \circ A_2 \): an \( \epsilon \) transition from each accepting state of \( A_1, F_1 \), to the starting state of \( M_2, q_2 \).

3. \( A^* \): an \( \epsilon \) transition from each accepting state in \( F \) to \( q_0 \), the starting state.

As an example, we will build a NFA for the language \( \{0, 01\}^* \)

First, consider a DFA \( M \) for \( \{0, 01\} \). It has 4 states, \( q_1, q_2, q_3 \) (with starting state \( q_0 = q_1 \)), and \( bad = q_4 \). The transitions are:

<table>
<thead>
<tr>
<th>( \Sigma \setminus \Sigma )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 )</td>
<td>( q_2 )</td>
<td>( q_4 )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>( q_4 )</td>
<td>( q_3 )</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>( q_4 )</td>
<td>( q_4 )</td>
</tr>
<tr>
<td>( q_4 )</td>
<td>( q_4 )</td>
<td>( q_4 )</td>
</tr>
</tbody>
</table>

and states \( q_2 \) and \( q_3 \) are accepting.

To get a NFA for \( L(M)^* \), we simply add an \( \epsilon \) transition from \( q_2 \) and \( q_3 \) to \( q_1 \). That is, we can ‘move on’ to the next word at any point.