Discussion Section Notes

Definitions and notation

1. Alphabet (Σ): A set of *symbols* or *letters*.
2. String (or word): A finite sequence of symbols from Σ. Length of w is denoted by $|w|$. Empty string is written ϵ. Concatenation is written $w_1 \cdot w_2$ or $w_1 w_2$.
3. Language: A set of strings (possibly infinite).
4. Operations on languages: Union is normal set union. We can also define concatenation, power, and star.

$$\begin{aligned} L_1 \cdot L_2 &= \{w_1 \cdot w_2 : w_1 \in L_1, w_2 \in L_2\} \\ L^0 &= \{\epsilon\} \\ L^{k+1} &= L \cdot L^k \\ L^* &= \bigcup_{k \geq 0} L^k \end{aligned}$$

   We can prove identities like $L \cdot \{\epsilon\} = L$ and $L \cdot \emptyset = \emptyset$ and $L^{**} = L^*$.

5. For a DFA A, $L(A)$ is the set of strings on which A reaches a final state. For an NFA N, $L(N)$ is the set of strings w where *there exists* a path in N labeled with w.
6. Regular expressions: A regular expression is a syntactic formalism. We can define the semantics (meaning) of a regular expression in terms of operations on sets. The meaning of a regular expression R is $L(R)$, which is the set of strings it accepts. If R_1 and R_2 are regular expressions, then so are the following: (1) a (for $a \in \Sigma$), (2) ϵ, (3) $\emptyset$, (4) $R_1 \cup R_2$, (5) $R_1 \cdot R_2$, (6) R_1^*. Their meaning is given below.

$$\begin{aligned} L(a) &= \{a\} & L(\epsilon) &= \{\epsilon\} \\ L(\emptyset) &= \emptyset & L(R_1 \cup R_2) &= L(R_1) \cup L(R_2) \\ L(R_1 \cdot R_2) &= L(R_1) \cdot L(R_2) & L(R_1^*) &= L(R_1)^* \end{aligned}$$

How to do the homework

When you need to construct an NFA or DFA, *first* figure out all the states you will need (this is Q). Then afterwards decide how δ should work and what F should be. Remember that Q must be finite.

To prove that an automaton A that you've constructed accepts a language L, you need to prove two things. First, that every string in L is accepted by A, and second, that every string accepted by A is in L.

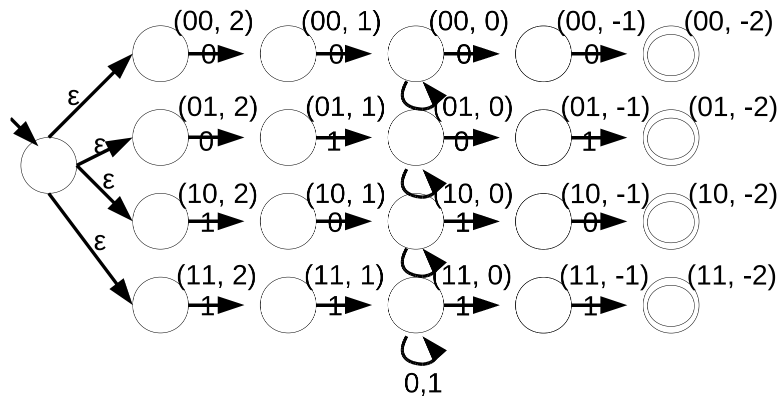
Example problem

For any fixed positive k and some alphabet Σ, prove that the following language is regular.

$$L_k = \{w \cdot w' \cdot w : w \in \Sigma^k \wedge w' \in \Sigma^*\}$$

This is the set of strings of length at least $2k$ whose first k symbols are equal to the last k symbols.

We will build an NFA that accepts this language. The NFA for L_2 over the alphabet $\{0, 1\}$ is shown below.



We immediately "guess" the string w from the start state via epsilon transitions to the states (w, k). The succeeding states, (w, i) for $i \in \{k - 1, 0\}$, check that the guess for w matches the input. The states $(w, 0)$ allow extra symbols in the middle by spinning on 0 and 1. Finally, they lead to states (w, i) for $i \in \{-1, -k\}$. These states ensure that the input ends with w. The states $(w, -k)$ are final states.

Thus, we construct an NFA where

$$Q = \{q_0\} \cup (\Sigma^k \times \{-k, -k + 1, \ldots, k\}).$$

The start state is q_0. The final states are $(w, -k)$ for $k \in \Sigma^k$. We define δ as follows. Assume $w = w_0 \ldots w_{k-1}$, where each $w_i \in \Sigma$.

$$\begin{aligned}
\delta(q_0, \epsilon) &= \{(w, k) : w \in \Sigma^k\} \\
\delta((w, i), \epsilon) &= \emptyset && \text{(for } -k \leq i \leq k\text{)} \\
\delta((w, i), w_{k-i}) &= \{(w, i - 1)\} && \text{(for } k \geq i > 0\text{)} \\
\delta((w, i), a) &= \emptyset && \text{(for } k \geq i > 0\text{, if } w_{k-i} \neq a\text{)} \\
\delta((w, 0), w_0) &= \{(w, 0), (w, -1)\} \\
\delta((w, 0), a) &= \{(w, 0)\} && \text{(if } w_0 \neq a\text{)} \\
\delta((w, -i), w_i) &= \{(w, -i - 1)\} && \text{(for } 1 \leq i < k\text{)} \\
\delta((w, -i), a) &= \emptyset && \text{(for } 1 \leq i < k \text{ if } a \neq w_i\text{)}
\end{aligned}$$

We need to prove that this NFA (call it N_k) accepts L_k.

Claim 1 *If $x \in L_k$, then $x \in L(N_k)$.*

PROOF: Let $x = w \cdot w' \cdot w$ for some $w \in \Sigma^k$. We construct a path through N_k from q_0 to $(w, -k)$. It begins with an ϵ transition to (w, k). It proceeds through $(w, k - 1), (w, k - 2), \ldots, (w, 0)$ since the input starts with w. Then it takes the self edge in $(w, 0)$ $|w'|$ times. Finally it proceeds through $(w, -1), (w, -2), \ldots, (w, -k)$, since the input ends with w. Now it is in an accept state. □

Claim 2 *If $x \in L(N_k)$, then $x \in L_k$.*

PROOF: Let $q_0, q_1, \ldots, q_n$ be an accepting path through N_k. The only edges out of q_0 are ϵ edges to states of the form (w, k), so let $q_1 = (w, k)$ for some w. From now on, the machine must follow a path to $(w, 0)$ labeled with the string w, since N_k is entirely deterministic in this region.

Now the path may loop at $(w, 0)$ some number of times. Let the string w' be the string that labels the edges on the path at this point. Eventually, the path must leave $(w, 0)$ and go to $(w, -1)$ since $(w, 0)$ is not an accept state and there are no other outgoing edges. This transition is labeled with w_0. The path must continue deterministically to $(w, -k)$ along a path labeled with the rest of w. $(w, -k)$ is an accept state and a sink, so the path must end. We have now proved that the edges on the path are labeled with $w \cdot w' \cdot w$, which implies that $x \in L_k$. $\square$