Demo of "Dragonstar" DFA using the "Turky Kit"

Theorem: If a DFA $M = (Q, \Sigma, \delta, s, F)$ accepts a language $L$, then the DFA $M' = (Q, \Sigma, \delta, s, Q \setminus F)$ accepts $\overline{L}$.

Example: $L = L_A = \{ x \in \{0, 1, 2, 3\}^* : x$ leaves light A on $\}$.

Let $A = \{9, 9\}$.

By the Cartesian Product theorem in recitations, if $L_1$ is accepted by a DFA $M$, and $L_2$ by $M_2$, then we can build a DFA $M_3$ s.t. $L(M_3) = L_1 \cap L_2$.

$M_3 = (Q_1 \times Q_2, \Sigma, \delta_3, (s_1, s_2), F_3)$

Then $L(M_4) = L(M_1) \cup L(M_2)$.

$M_5 = (Q_1 \times Q_2, \Sigma, \delta_5, (S_1, S_2), F_5)$

Then $L(M_5) = \{ x : M_1$ accepts $x \times M_2$ accepts $x \times 2 \}$ XOR $I_2 \in F_2$.

Do $M_6 = \{ x \in \{0, 1, 2, 3\}^* : x$ leaves one or both lights on $\}$.

$\widehat{L} = \{ x \in \{0, 1, 2, 3\}^* : x$ leaves one or both lights on $\}$.

$\widehat{L} = \widehat{L}_A \cup \widehat{L}_B$.

Third idea: For U, try an NFA.
Build an NFA \( N_3 = (Q_3, \Sigma, \delta_3, s_3, F_3) \) such that \( L(N_3) = L(M'_A) \cup L(M'_B) \).

By:

\[ Q_3 = \{ s_3 \} \cup Q_1, Q_2 \]

\[ F_3 = F_1 \cup F_2 = \{ q_1, q_2 \} \]

\[ S_3 = S_1 \cup S_2 \cup \{ (s_3, \varepsilon, s_1), (s_3, \varepsilon, s_2) \} \]

**Formal Definition of NFA (with \( \varepsilon \)-arcs)**

**Def**: A nondeterministic finite automaton (NFA) is a 5-tuple \( N = (Q, \Sigma, \delta, s, F) \) where:

- \( Q \) is a finite set of states
- \( \Sigma \) is the input alphabet
- \( s \) is a member of \( Q \), is the start state
- \( F \) is a subset of \( Q \), the final states

\( \delta \) is a function with domain all of \( Q \times \Sigma \) and range \( \subseteq Q \).

Typical instruction: \( \langle p, \sigma, q \rangle \) \( \sigma \in \Sigma \) or \( \langle p, \varepsilon, q \rangle \) \( p = q \) allowed.

An NFA is a DFA if for all \( q \in Q \) and \( \sigma \in \Sigma \) there is exactly one instruction \( \langle q, \sigma, r \rangle \in \delta \) where \( r \in Q \). (And no instruction have \( \varepsilon \).)

\[ \Sigma = \{ \$0, \$1, \$D \} \]

\[ L = \{ \varepsilon \in \Sigma^* \} \]

Is this a DFA?

No: lacks an instruction for \( (q_0, 0) \).

Formally, this diagram needs to be "completed" by adding a dead state. Then we can complement the machine.
Defn: A computation path that processes a string $x$ is a sequence $(q_0, w_1, q_1, w_2, q_2, \ldots, q_{m-1}, w_m, q_m)$ such that

1. For all $j$, $0 \leq j \leq m\!-\!1$, $(q_{j-1}, w_j, q_j) \in S$
2. The string $w_1 \cdot w_2 \cdots w_m$ equals $x$.

Then we also say that $N$ can process $x$ from state $q_0$ to state $q_m$.

Formally, $L(N) = \{ x \in \Sigma' : N$ can process $x$ from $s$ to a state in $F \}.$

Example: $x = 1\, 3$ (leaves both lights on)

Path: $(s_3, \varepsilon, s_1, 1, s_1, 3, q_1)$

Another Acc Path: $(s_3, \varepsilon, s_2, 1, s_2, 3, q_2)$

With a DFA, always $m = n = l(x)\!$, and every step has just one option.

Note Added: We will use the concept in cases where $q_0$ is not the start state. Indeed for any $p, q \in Q$ we can define

$L_{pq} = \{ x \in \Sigma' : N$ can process $x$ from $p$ to $q \}.$

So $L(N) = \bigcup_{q \in F} L_{sq}$. This again is why I like to use separate notation in "$s$" from $q_0$ for the start state.