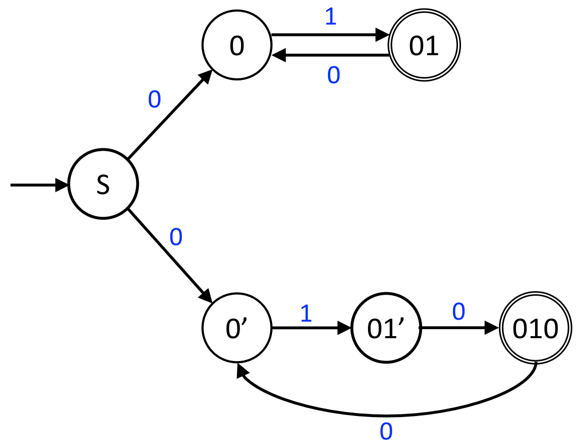
1 Construct an NFA for the language $(01)^+ + (010)^+$.

Solution:

The NFA is shown in the figure below.



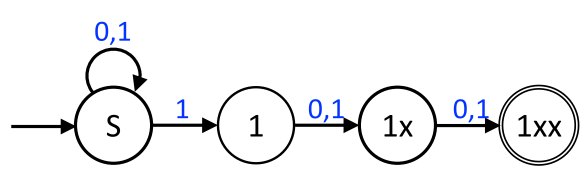
Note that we've separated the two cases of either repeated 01, or repeated 010. Why would the NFA with states labeled 0 and $0'$ merged be incorrect?

---

2 Construct an NFA that accepts all binary strings that have 1 as the third to last character; i.e., $x1ab$ for $a, b \in \{0, 1\}$ and $x \in \{0, 1\}^*$.

Solution:

The NFA is shown in the figure below.



---

3 Given a DFA $M = (\Sigma, Q, \delta, s, A)$, construct an NFA N that accepts all prefixes of $L(M)$, i.e., $w \in L(N) \Leftrightarrow wx \in L(M)$ for some $x \in \Sigma^*$.

Solution:

We construct an NFA $N = (\Sigma, Q', \delta', s', A')$ that accepts *prefix*$(L(M))$ as follows.

$$
\begin{aligned}
Q' &:= Q \\
s' &:= s \\
A' &:= \{q \in Q \mid \delta^*(s, w) = q \text{ and } \delta^*(q, x) \in A \text{ for some } w, x \in \Sigma^*\} \\
\delta'(q, a) &= \{\delta(q, a)\} \quad \forall q \in Q, a \in \Sigma
\end{aligned}
$$

We let A' be the set of states in Q that are both reachable from s via some prefix w and can reach an accepting state via some suffix x. We make every state in A' accepting .

---

4 Given an DFA $M = (\Sigma, Q, \delta, s, A)$, construct an NFA N that accepts all suffixes of $L(M)$, i.e., $w \in L(N) \Leftrightarrow xw \in L(M)$ for some $x \in \Sigma^*$.

Solution:

We construct an NFA $N = (\Sigma, Q', \delta', s', A')$ that accepts *prefix*$(L(M))$ as follows.

$$
\begin{aligned}
Q' &:= Q \cup \{t\} \quad \text{(here } t \text{ is a new state not in } Q\text{)} \\
s' &:= t \\
A' &:= A \\
\delta'(t, \epsilon) &= \{q \in Q \mid \delta^*(s, w) = q \text{ and } \delta^*(q, x) \in A \text{ for some } w, x \in \Sigma^*\} \\
\delta'(t, a) &= \emptyset \quad \forall a \in \Sigma \\
\delta'(q, a) &= \{\delta(q, a)\} \quad \forall q \in Q, a \in \Sigma
\end{aligned}
$$

We create a new start state t and create an ϵ-transition from t to every state in Q that is both reachable from s via some prefix x and can reach an accepting state via some suffix w. Note that the addition of the extra state t is necessary to avoid the ϵ-transition being taken after the NFA takes a series of steps and returns to s.

---

5 Given a DFA $M = (\Sigma, Q, \delta, s, A)$, construct an NFA N that accepts resverse of $L(M)$, i.e., $w \in L(N) \Leftrightarrow w^R \in L(M)$.

Solution:

We construct an NFA $N = (\Sigma, Q', \delta', s', A')$ that accepts *reverse*$(L(M))$ as follows.

$$
\begin{aligned}
Q' &:= Q \cup \{t\} \quad \text{(here } t \text{ is a new state not in } Q\text{)} \\
s' &:= t \\
A' &:= \{s\} \\
\delta'(t, \epsilon) &= A \\
\forall q \in Q, a \in \Sigma \quad \delta'(q, a) &= \left\{q' \in Q \;\middle|\; \delta(q', a) = q\right\}
\end{aligned}
$$

N is obtained from M by reversing all the directions of the edges, adding a new state t that becomes the new start state that is connected via ϵ edges to all the original accepting states. There is a single accepting state in N which is the start state of M. To see that N accepts *reverse*$(L(M))$ you need to see that any accepting walk of N corresponds to an accepting walk of M.

6 Given a DFA $M = (\Sigma, Q, \delta, s, A)$, construct an NFA N that accepts $\mathit{insert1}(L(M)) := \{x1y \mid xy \in L(M)\}$, i.e., strings in $L(M)$ with 1 inserted somewhere. For example, if $L(M) = \{\varepsilon, OOK!\}$, then $\mathit{insert1}(L(M)) = \{1, 1OOK!, O1OK!, OO1K!, OOK1!, OOK!1\}$.

Solution:

We construct an NFA $N = (\Sigma, Q', \delta', s', A')$ that accepts $\mathit{insert1}(L(M))$ as follows:

$$\begin{aligned} Q' &:= Q \times \{\mathit{before}, \mathit{after}\} \\ s' &:= (s, \mathit{before}) \\ A' &:= \big\{(q, \mathit{after}) \bigm| q \in A\big\} \end{aligned}$$

$$\delta'((q, \mathit{before}), a) = \begin{cases} \big\{(\delta(q,a), \mathit{before}),\ (q, \mathit{after})\big\} & \text{if } a = 1 \\ \big\{(\delta(q,a), \mathit{before})\big\} & \text{otherwise} \end{cases}$$

$$\delta'((q, \mathit{after}), a) = \big\{(\delta(q,a), \mathit{after})\big\}$$

N simulates M, but inserts a single 1 into M's input string at a nondeterministically chosen location.

- The state (q, before) means (the simulation of) M is in state q and N has not yet inserted a 1.
- The state (q, after) means (the simulation of) M is in state q and N has already inserted a 1.

7 Given a DFA $M = (\Sigma, Q, \delta, s, A)$, construct an NFA N that accepts $\mathit{delete1}(L(M)) := \{xy \mid x1y \in L(M)\}$. Intuitively, $\mathit{delete1}(L(M))$ is the set of all strings that can be obtained from strings in $L(M)$ by deleting exactly one 1. For example, if $L(M) = \{101101, 00, \varepsilon\}$, then $\mathit{delete1}(L(M)) = \{01101, 10101, 10110\}$.

Solution:

We construct an NFA $N = (\Sigma, Q', \delta', s', A')$ with ε-transitions that accepts $\mathit{delete1}(L(M))$ as follows:

$$\begin{aligned} Q' &:= Q \times \{\mathit{before}, \mathit{after}\} \\ s' &:= (s, \mathit{before}) \\ A' &:= \{(q, \mathit{after})\}\, q \in A \end{aligned}$$

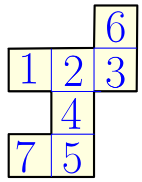
$$\begin{aligned} \delta'((q, \mathit{before}), \varepsilon) &= \big\{(\delta(q, 1), \mathit{after})\big\} \\ \delta'((q, \mathit{after}), \varepsilon) &= \varnothing \end{aligned}$$

$$\begin{aligned} \delta'((q, \mathit{before}), a) &= \big\{(\delta(q, a), \mathit{before})\big\} \\ \delta'((q, \mathit{after}), a) &= \big\{(\delta(q, a), \mathit{after})\big\} \end{aligned}$$

N nondeterministically chooses a 1 in the input string to ignore, and simulates M running on the rest of the input string.

- The state (q, before) means (the simulation of) M is in state q and N has not yet skipped over a 1.
- The state (q, after) means (the simulation of) M is in state q and N has already skipped over a 1.

8 Consider the following "maze":



A robot starts at position 1 – where at every point in time it is allowed to move only to adjacent cells. The input is a sequence of commands V (move vertically) or H (move horizontally), where the robot is required to move if it gets such a command. If it is in location 2, and it gets a V command then it must move down to location 4. However, if it gets command H while being in location 2 then it can move either to location 1 or 3, as it chooses.

An input is ***invalid***, if the robot get stuck during the execution of this sequence of commands, for any sequence of choices it makes. For example, starting at position 1, the input HVH is not valid. (The robot was so badly designed, that if it gets stuck, it explodes and no longer exists.)

8 A. Starting at position 1, consider the (command) input HVV. Which location might the robot be in? (Same for $HVVV$ and $HVVVH$.)

Solution:

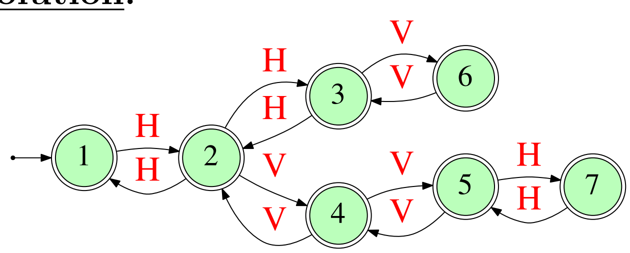
HVV: 2 or 5.

$HVVV$: 4.

$HVVVH$: This is an invalid input. The robot can not be in any valid location.

8 B. Draw an NFA that accepts all valid inputs.

Solution:



8 C. The robot *solves* the maze if it arrives (at any point in time) to position 7. Draw an NFA that accepts all inputs that are solutions to the maze.

Solution: