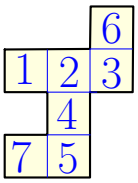
1 Consider the following "maze":



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.)

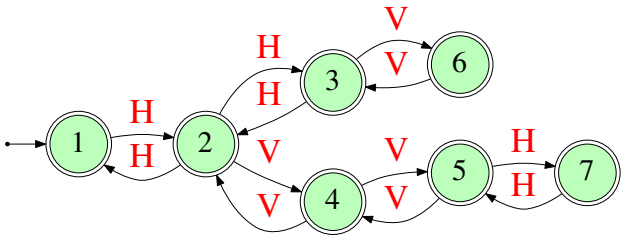
1 A. Starting at position 1, consider the (command) input HVV. Which location might the robot be in? (Same for $HVVV$ and $HVVVH$.)

Solution:

HVV: 2 or 5.
$HVVV$: 4.
$HVVVH$: This is an invalid input. The robot can not be in any valid location.

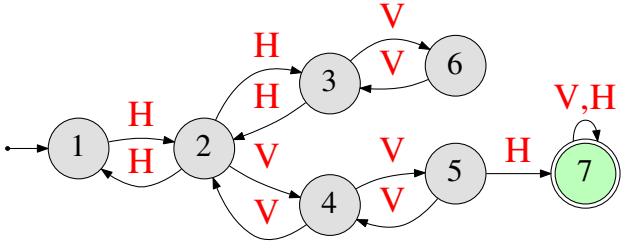
1 B. Draw an NFA that accepts all valid inputs.

Solution:



1 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:



1 D. (Extra - not for discussion section.) Write a regular expression which is all inputs that are valid solutions to the maze.
(See here for notes of how to solve such a question.)

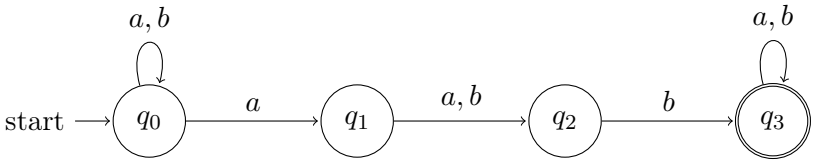
---

2 Let $L = \{w \in \{a, b\}^* \mid a \text{ appears in some position } i \text{ of } w, \text{ and a } b \text{ appears in position } i + 2\}$.

2 A. Create an NFA N for L with at most four states.

Solution:

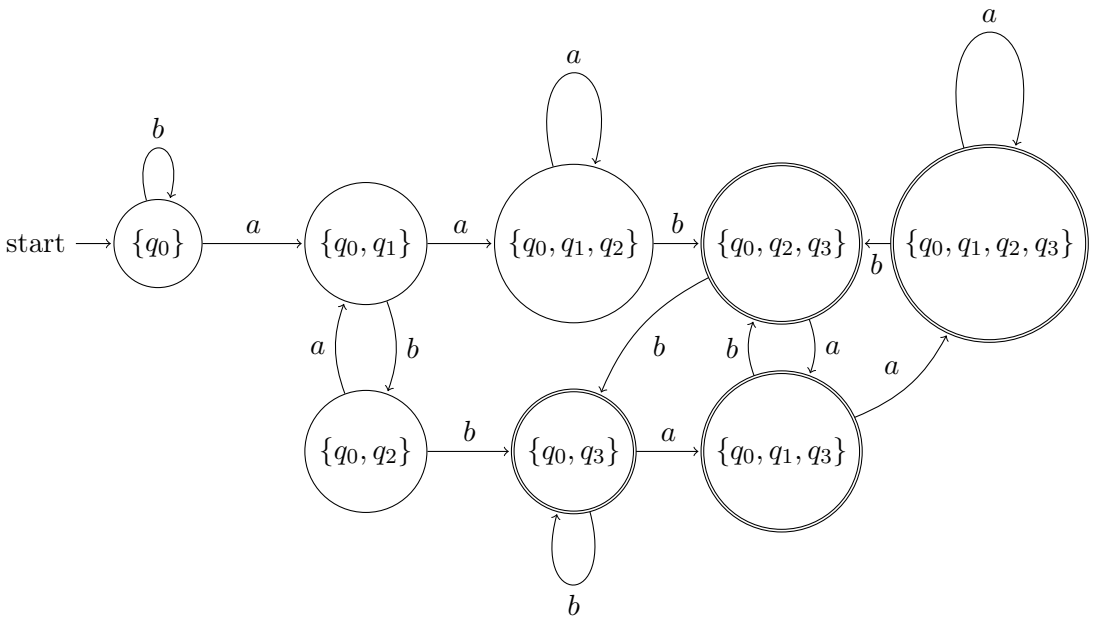
The following NFA N accepts the language. The machine starts at state q_0. On seeing the symbol a, the NFA has the choice of either staying at q_0 or to check if it is followed, 2 positions later, with a b.



2 B. Using the "power-set" construction, create a DFA M from N. Rather than writing down the sixteen states and trying to fill in the transitions, build the states as needed, because you won't end up with unreachable or otherwise superfluous states.

Solution:

Using the "power-set" construction, we obtain the following DFA M.

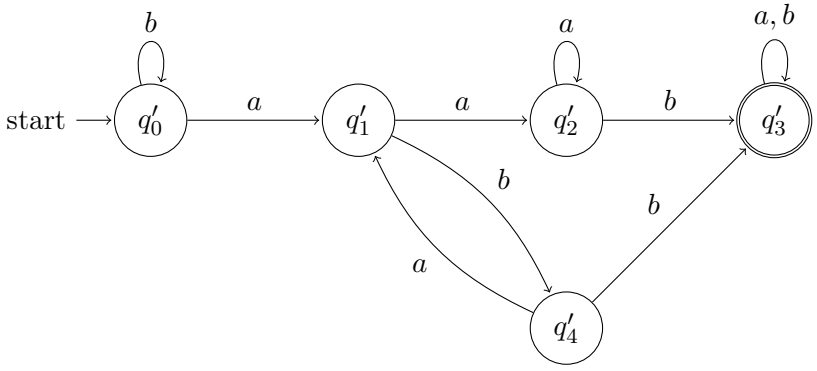


2 C. Now directly design a DFA M' for L with only five states, and explain the relationship between M and M'.

Solution:

The DFA M' is as follows. M' remembers the last two symbols seen so far.

- q'_0 is the start state. M'
- q'_1 corresponds to having seen ba as the last two symbols (or just a if this is the first symbol).
- q'_2 corresponds to having seen aa as the last two symbols.
- q'_3 is the accepting state.
- q'_4 corresponds to having seen ab as the last two symbols.



Note that if we contract all the accepting to states in M (from part (b)) to one state, then we obtain M'.

For the rest of the problems assume that L is an arbitrary regular language.

---

3 Prove that the language $\mathit{reverse}(L) := \{w^R \mid w \in L\}$ is regular. *Hint:* Consider a DFA M that accepts L and construct a NFA that accepts $\mathit{reverse}(L)$.

Solution:

Let $M = (\Sigma, Q, s, A, \delta)$ be a DFA that accepts L. We construct an NFA $M' = (\Sigma, Q', s', A', \delta')$ that accepts $\mathit{reverse}(L)$ as follows.

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

M' 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 M' which is the start state of M. To see that M' accepts $\mathit{reverse}(L)$ you need to see that any accepting walk of M' corresponds to an accepting walk of M.

Another way to show that $\mathit{reverse}(L)$ is regular is via regular expressions. For any regular expression r you can construct a regular expression r' such that $L(r') = \mathit{reverse}(L)$ using the inductive definition of regular languages. We ignore the base cases as exercise and consider the inductive cases.

- If r_1 and r_2 are regular expressions and r'_1 and r'_2 are regular expressions for the reverse languages then the reverse for $r_1 + r_2$ is $r'_1 + r'_2$.

- For $r_1 r_2$ we have $r'_2 r'_1$.
- For $(r_1)^*$ we have $(r'_1)^*$.

---

4 Prove that the language $\mathit{insert1}(L) := \{x1y \mid xy \in L\}$ is regular.

Intuitively, $\mathit{insert1}(L)$ is the set of all strings that can be obtained from strings in L by inserting exactly one 1. For example, if $L = \{\varepsilon, OOK!\}$, then $\mathit{insert1}(L) = \{1, 1OOK!, O1OK!, OO1K!, OOK1!, OOK!1\}$.

Solution:

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

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

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

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

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

Solutions for extra problems

---

5 Prove that the language $\mathit{delete1}(L) := \{xy \mid x1y \in L\}$ is regular.

Intuitively, $\mathit{delete1}(L)$ is the set of all strings that can be obtained from strings in L by deleting exactly one 1. For example, if $L = \{101101, 00, \varepsilon\}$, then $\mathit{delete1}(L) = \{01101, 10101, 10110\}$.

Solution:

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

$$\begin{aligned} Q' &:= Q \times \{\mathit{before}, \mathit{after}\} \\ s' &:= (s, \mathit{before}) \\ A' &:= \{(q, \mathit{after})\}\, q \in A \\ \delta'((q, \mathit{before}), \varepsilon) &= \{(\delta(q, 1), \mathit{after})\} \\ \delta'((q, \mathit{after}), \varepsilon) &= \varnothing \\ \delta'((q, \mathit{before}), a) &= \{(\delta(q, a), \mathit{before})\} \\ \delta'((q, \mathit{after}), a) &= \{(\delta(q, a), \mathit{after})\} \end{aligned}$$

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

---

6 Consider the following recursively defined function on strings:

$$\mathit{stutter}(w) := \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ aa \bullet \mathit{stutter}(x) & \text{if } w = ax \text{ for some symbol } a \text{ and some string } x \end{cases}$$

Intuitively, $\mathit{stutter}(w)$ doubles every symbol in w. For example:

- $\mathit{stutter}(\mathit{PRESTO}) = \mathit{PPRREESSTTOO}$
- $\mathit{stutter}(\mathit{HOCUS\square POCUS}) = \mathit{HHOOCCUUSS\square\square PPOOCCUUSS}$

Let L be an arbitrary regular language.

6.A. Prove that the language $\mathit{stutter}^{-1}(L) := \{w \mid \mathit{stutter}(w) \in L\}$ is regular.

Solution:

Let $M = (\Sigma, Q, s, A, \delta)$ be a DFA that accepts L.
We construct an DFA $M' = (\Sigma, Q', s', A', \delta')$ that accepts $\mathit{stutter}^{-1}(L)$ as follows:

$$\begin{aligned} Q' &= Q \\ s' &= s \\ A' &= A \\ \delta'(q, a) &= \delta(\delta(q, a), a) \end{aligned}$$

M' reads its input string w and simulates M running on $\mathit{stutter}(w)$. Each time M' reads a symbol, the simulation of M reads two copies of that symbol.

6.B. Prove that the language $\mathit{stutter}(L) := \{\mathit{stutter}(w) \mid w \in L\}$ is regular.

Solution:

Let $M = (\Sigma, Q, s, A, \delta)$ be a DFA that accepts L.

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

$$\begin{aligned} Q' &= Q \times (\{\bullet\} \cup \Sigma) \cup \{\mathit{fail}\} \quad \text{for some } \bullet \notin \Sigma \\ s' &= (s, \bullet) \\ A' &= \{(q, \bullet)\}\, q \in A \\ \delta'((q, \bullet), a) &= (q, a) \\ \delta'((q, a), b) &= \begin{cases} (\delta(q, a), \bullet) & \text{if } a = b \\ \mathit{fail} & \text{if } a \neq b \end{cases} \\ \delta'(\mathit{fail}, a) &= \mathit{fail} \end{aligned}$$

M' reads the input string $\mathit{stutter}(w)$ and simulates M running on input w.

- State $(q, \bullet)$ means M' has just read an even symbol in $\mathit{stutter}(w)$, so M should ignore the next symbol (if any).
- For any symbol $a \in \Sigma$, state (q, a) means M' has just read an odd symbol in $\mathit{stutter}(w)$, and that symbol was a. If the next symbol is an a, then M should transition normally; otherwise, the simulation should fail.
- The state *fail* means M' has read two successive symbols that should have been equal but were not; the input string is not $\mathit{stutter}(w)$ for any string w.

Solution:

Let R be an arbitrary regular *expression*. We recursively construct a regular expression $\mathit{stutter}(R)$ as follows:

$$\mathit{stutter}(R) := \begin{cases} \varnothing & \text{if } R = \varnothing \\ \mathit{stutter}(w) & \text{if } R = w \text{ for some string } w \in \Sigma^* \\ \mathit{stutter}(A) + \mathit{stutter}(B) & \text{if } R = A + B \text{ for some regular expressions } A \text{ and } B \\ \mathit{stutter}(A)\,\mathit{stutter}(B) & \text{if } R = A\,B \text{ for some regular expressions } A \text{ and } B \\ (\mathit{stutter}(A))^* & \text{if } R = A^* \text{ for some regular expression } A \end{cases}$$

To prove that $L(\mathit{stutter}(R)) = \mathit{stutter}(L(R))$, we need the following identities for arbitrary *languages* A and B:

- $\mathit{stutter}(A \cup B) = \mathit{stutter}(A) \cup \mathit{stutter}(B)$
- $\mathit{stutter}(A \bullet B) = \mathit{stutter}(A) \bullet \mathit{stutter}(B)$
- $\mathit{stutter}(A^*) = \mathit{stutter}(A)^*$

These identities can all be proved by inductive definition-chasing, after which the claim $L(\mathit{stutter}(R)) = \mathit{stutter}(L(R))$ follows by induction. We leave the details of the induction proofs as an exercise for ~~a future semester~~ ~~an exam~~ the reader.

Equivalently, we can directly transform R into $\mathit{stutter}(R)$ by replacing every explicit string $w \in \Sigma^*$ inside R with $\mathit{stutter}(w)$ (with additional parentheses if necessary). For example:

$$\mathit{stutter}\big((1 + \varepsilon)(01)^*(0 + \varepsilon) + 0^*\big) \;=\; (11 + \varepsilon)(0011)^*(00 + \varepsilon) + (00)^*$$

Although this may look simpler, actually *proving* that it works requires the same induction arguments.

7 Consider the following recursively defined function on strings:

$$evens(w) := \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ \varepsilon & \text{if } w = a \text{ for some symbol } a \\ b \cdot evens(x) & \text{if } w = abx \text{ for some symbols } a \text{ and } b \text{ and some string } x \end{cases}$$

Intuitively, $evens(w)$ skips over every other symbol in w. For example:

- $evens(EXPELLIARMUS) = XELAMS$
- $evens(AVADA\square KEDAVRA) = VD\square EAR$.

Once again, let L be an arbitrary regular language.

7.A. Prove that the language $evens^{-1}(L) := \{w \mid evens(w) \in L\}$ is regular.

Solution:

Let $M = (\Sigma, Q, s, A, \delta)$ be a DFA that accepts L. We construct an DFA $M' = (\Sigma, Q', s', A', \delta')$ that accepts $evens^{-1}(L)$ as follows:

$$\begin{aligned} Q' &= Q \times \{0,1\} \\ s' &= (s, 0) \\ A' &= A \times \{0,1\} \\ \delta'((q,0),a) &= (q,1) \\ \delta'((q,1),a) &= (\delta(q,a),0) \end{aligned}$$

M' reads its input string w and simulates M running on $evens(w)$.

- State $(q, 0)$ means M' has just read an even symbol in w, so M should ignore the next symbol (if any).
- State $(q, 1)$ means M' has just read an odd symbol in w, so M should read the next symbol (if any).

7.B. Prove that the language $evens(L) := \{evens(w) \mid w \in L\}$ is regular.

Solution:

Let $M = (\Sigma, Q, s, A, \delta)$ be a DFA that accepts L. We construct an NFA $M' = (\Sigma, Q', s', A', \delta')$ that accepts $evens(L)$ as follows:

$$\begin{aligned} Q' &= Q \\ s' &= s \\ A' &= A \cup \{q \in Q \mid \delta(q,a) \cap A \neq \varnothing \text{ for some } a \in \Sigma\} \\ \delta'(q,a) &= \bigcup_{b \in \Sigma} \{\delta(\delta(q,b),a)\} \end{aligned}$$

M' reads the input string $evens(w)$ and simulates M running on string w, while nondeterministically guessing the missing symbols in w.

- When M' reads the symbol a from $evens(w)$, it guesses a symbol $b \in \Sigma$ and simulates M reading ba from w.

- When M' finishes $evens(w)$, it guesses whether w has even or odd length, and in the odd case, it guesses the last character of w.