

The following problems are not for submission or grading. No solutions for them will be provided but you can discuss them on Piazza (however, some of them already contain a solution).

1 (100 PTS.) Draw me a sheep.

For each of the following languages, draw (or describe formally) an NFA that accepts them. Your automata should have a small number of states. Provide a short explanation of your solution, if needed.

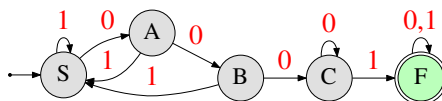
- 1.A. All strings over $\{a, b, c\}^*$ in which every nonempty maximal substring of consecutive a s is of even length.
- 1.B. $\Sigma^* a \Sigma^* b \Sigma^* c \Sigma^*$.
- 1.C. $(a(a + b)^* a + b(b + c)^* b + c(c + a)^* c)^*$.
- 1.D. $\left(((aa + aab)^* (bab + bb)^* + c)b \right)^* + bb$.
- 1.E. All strings in 1^* of length that is divisible by at least one of the following numbers 2, 3, 5, 7. For full credit your automata should have less than (say) 20 states.
- 1.F. All strings in a^* of length that is NOT divisible by any of the following numbers 2, 3, 5, 7.

2 (100 PTS.) Blip blip.

For two binary strings $x, y \in \{0, 1\}^*$, of the same length, their **Hamming distance** $d_H(x, y)$ is the number of bits in which they differ. For example $d_H(1111, 1111) = 0$, $d_H(0001, 1111) = 3$, and $d_H(1111001, 1111011) = 1$. As a negative example, observe that $d_H(11, 1011)$ is not defined.

Let $L \subseteq \{0, 1\}^*$ be a regular language.

- 2.A. Consider the language $L_{\leq 1} = \{x \in \{0, 1\}^* \mid \exists y \in L \text{ s.t. } d_H(x, y) \leq 1\}$. Describe in words what the language $L_{\leq 1}$ is.
- 2.B. Consider the following DFA M .



What is its language $L = L(M)$?

- 2.C. By modifying the given DFA give above, describe an NFA that that accepts the language $L_{\leq 1}$. Explain your construction.

- 2.D.** More generally, demonstrate that if a language $L \subseteq \{0, 1\}^*$ is regular, then $L_{\leq 1}$ is a regular language (for simplicity, you can assume $\varepsilon \notin L$). Specifically, consider a DFA for L , and describe in detail how to modify it to an NFA for $L_{\leq 1}$. (The description of the NFA does not have to be formal here.) Explain why the constructed NFA accept the desired language.
- 2.E.** Prove, that for any constant k , the language $L_{\leq k}$ is regular. Your proof has to be formal and provide all necessary details. (I.e., you need to provide an explicit formal description of the resulting NFA for the new language, and prove that the NFA accepts the language $L_{\leq k}$).

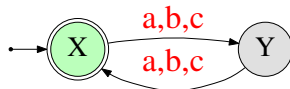
3 (100 PTS.) Codes.

Let Σ be finite alphabet. A **code** is a mapping $f : \Sigma \rightarrow \{0, 1\}^+$. For example, if $\Sigma = \{a, b, c\}$, a code f might be $f(a) = 00010$, $f(b) = 000$, and $f(c) = 1$. (To simplify things, we assume that $f(a) \neq \varepsilon$, for any character $a \in \Sigma$.)

For a string $w_1 w_2 \cdots w_m \in \Sigma^*$, we define $f(w) = f(w_1) f(w_2) \cdots f(w_m)$. In the above code, we have

$$f(abcba) = 00010 \bullet 000 \bullet 1 \bullet 000 \bullet 00010. = 00010000100000010.$$

- 3.A.** (10 PTS.) Let L be the language of the following DFA M . What is L ?



- 3.B.** (20 PTS.) Working directly on the DFA M from (A) construct an NFA for the language $f(L)$. Here $f(L) = \{f(w) \mid w \in L\}$ is the *code language*. Where f is code from the above example.
- 3.C.** (30 PTS.) Let $L \subseteq \Sigma^*$ be an arbitrary regular language. Prove that the encoded language $f(L) = \{f(w) \mid w \in L\}$ is regular.

Specifically, given a DFA $M = (Q, \Sigma, \delta, s, A)$ for L , describe how to build an NFA M' for $f(L)$. Give an upper bound on the number of states of M' .

(I.e., You need to prove the correctness of your construction – that the language of the constructed NFA is indeed the desired language $f(L)$.)

(Rubric: Half the credit is for a correct construction, and the other half is for a correct proof of correctness.)

- 3.D.** (40 PTS.) Let $L \subseteq \{0, 1\}^*$ be a regular language. Consider the decoded language $L_f = \{w \in \Sigma^* \mid f(w) \in L\}$.

Prove that L_f is a regular language. As above, given a DFA M for L , describe how to construct an NFA for L_f .

(Rubric: Half the credit is for a correct construction, and the other half is for a correct proof of correctness.)

4 Suppose $N_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$ are NFAs. Formally describe a DFA that accepts the language $L(N_1) \setminus L(N_2)$. This combines subset construction and product construction to give you practice with formalism.

5 Suppose $M = (Q, \Sigma, \delta, s, A)$ is a DFA. For states $p, q \in Q$ (p can be same as q) argue that $L_{p,q} = \{w \mid \delta^*(p, w) = q\}$ is regular. Recall that $\text{PREFIX}(L) = \{w \mid wx \in L, x \in \Sigma^*\}$ is the set of all prefixes of strings in L . Express $\text{PREFIX}(L(M))$ as $\cup_{q \in Z} L_{s,q}$ for a suitable set of states $Z \subseteq Q$. Why does this prove that $\text{PREFIX}(L(M))$ is regular whenever L is regular?

6 For a language L let $\text{MID}(L) = \{w \mid xwy \in L, x, y \in \Sigma^*\}$. Prove that $\text{MID}(L)$ is regular if L is regular.

- 7**
1. Draw an NFA that accepts the language $\{w \mid \text{there is exactly one block of 0s of even length}\}$. (A “block of 0s” is a maximal substring of 0s.)
 2. (a) Draw an NFA for the regular expression $(010)^* + (01)^* + 0^*$.
 (b) Now using the powerset construction (also called the subset construction), design a DFA for the same language. Label the states of your DFA with names that are sets of states of your NFA.

8 This problem is to illustrate proofs of (the many) closure properties of regular languages.

1. For a language L let $\text{FUNKY}(L) = \{w \mid w \in L \text{ but no proper prefix of } w \text{ is in } L\}$. Prove that if L is regular then $\text{FUNKY}(L)$ is also regular using the following technique. Let $M = (Q, \Sigma, \delta, s, A)$ be a DFA accepting L . Describe a NFA N in terms of M that accepts $\text{FUNKY}(L)$. Explain the construction of your NFA.
2. In Lab 3 we saw that $\text{insert1}(L)$ is regular whenever L is regular. Here we consider a different proof technique. Let r be a regular expression. We would like to show that there is another regular expression r' such that $L(r') = \text{insert1}(L(r))$.
 - (a) For each of the base cases of regular expressions \emptyset, ϵ and $\{a\}, a \in \Sigma$ describe a regular expression for $\text{insert1}(L(r))$.
 - (b) Suppose r_1 and r_2 are regular expressions, and r'_1 and r'_2 are regular expressions for the languages $\text{insert1}(L(r_1))$ and $\text{insert1}(L(r_2))$ respectively. Describe a regular expression for the language $\text{insert1}(L(r_1 + r_2))$ using r_1, r_2, r'_1, r'_2 .
 - (c) Same as the previous part but now consider $L(r_1 r_2)$.
 - (d) Same as the previous part but now consider $L((r_1)^*)$.

9 Recall that for any language L , $\bar{L} = \Sigma^* - L$ is the complement of L . In particular, for any NFA N , $\overline{L(N)}$ is the complement of $L(N)$.

Let $N = (Q, \Sigma, \delta, s, A)$ be an NFA, and define the NFA $N_{\text{comp}} = (Q, \Sigma, \delta, s, Q \setminus A)$. In other words we simply complemented the accepting states of N to obtain N_{comp} . Note that if M is DFA then M_{comp} accepts $\Sigma^* - L(M)$. However things are trickier with NFAs.

1. Describe a concrete example of a machine N to show that $L(N_{\text{comp}}) \neq \overline{L(N)}$. You need to explain for your machine N what $\overline{L(N)}$ and $L(N_{\text{comp}})$ are.
2. Define an NFA that accepts $\overline{L(N)} - L(N_{\text{comp}})$, and explain how it works.
3. Define an NFA that accepts $L(N_{\text{comp}}) - \overline{L(N)}$, and explain how it works.

Hint: For all three parts it is useful to classify strings in Σ^* based on whether N takes them to accepting and non-accepting states from s .

10 Let L be an arbitrary regular language. Prove that the language $half(L) := \{w\}ww \in L$ is also regular.

Solution:

Let $M = (\Sigma, Q, s, A, \delta)$ be an arbitrary DFA that accepts L . We define a new NFA $M' = (\Sigma, Q', s', A', \delta')$ with ε -transitions that accepts $half(L)$, as follows:

$$Q' = (Q \times Q \times Q) \cup \{s'\}$$

s' is an explicit state in Q'

$$A' = \{(h, h, q) \mid h \in Q \text{ and } q \in A\}$$

$$\delta'(s', \varepsilon) = \{(s, h, h) \mid h \in Q\}$$

$$\delta'((p, h, q), a) = \{(\delta(p, a), h, \delta(q, a))\}$$

M' reads its input string w and simulates M reading the input string ww . Specifically, M' simultaneously simulates two copies of M , one reading the left half of ww starting at the usual start state s , and the other reading the right half of ww starting at some intermediate state h .

- The new start state s' non-deterministically guesses the “halfway” state $h = \delta^*(s, w)$ without reading any input; this is the only non-determinism in M' .
- State (p, h, q) means the following:
 - The left copy of M (which started at state s) is now in state p .
 - The initial guess for the halfway state is h .
 - The right copy of M (which started at state h) is now in state q .
- M' accepts if and only if the left copy of M ends at state h (so the initial non-deterministic guess $h = \delta^*(s, w)$ was correct) and the right copy of M ends in an accepting state.

Rubric: 5 points =

- + 1 for a formal, complete, and unambiguous description of a DFA or NFA
 - No points for the rest of the problem if this is missing.
- + 3 for a correct NFA
 - –1 for a single mistake in the description (for example a typo)
- + 1 for a *brief* English justification. We explicitly do *not* want a formal proof of correctness, but we do want one or two sentences explaining how the NFA works.

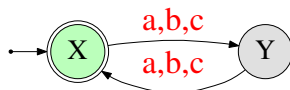
11 (100 PTS.) Codes.

Let Σ be finite alphabet. A **code** is a mapping $f : \Sigma \rightarrow \{0, 1\}^+$. For example, if $\Sigma = \{a, b, c\}$, a code f might be $f(a) = 00010$, $f(b) = 000$, and $f(c) = 1$. (To simplify things, we assume that $f(a) \neq \varepsilon$, for any character $a \in \Sigma$.)

For a string $w_1w_2\cdots w_m \in \Sigma^*$, we define $f(w) = f(w_1)f(w_2)\cdots f(w_m)$. In the above code, we have

$$f(abcb) = 00010 \bullet 000 \bullet 1 \bullet 000 \bullet 00010. = 00010000100000010.$$

- 11.A.** (10 PTS.) Let L be the language of the following DFA M . What is L ?



- 11.B.** (20 PTS.) Working directly on the DFA M from (A) construct an NFA for the language $f(L)$. Here $f(L) = \{f(w) \mid w \in L\}$ is the *code language*. Where f is code from the above example.
- 11.C.** (30 PTS.) Let $L \subseteq \Sigma^*$ be an arbitrary regular language. Prove that the encoded language $f(L) = \{f(w) \mid w \in L\}$ is regular.

Specifically, given a DFA $M = (Q, \Sigma, \delta, s, A)$ for L , describe how to build an NFA M' for $f(L)$. Give an upper bound on the number of states of M' .

(I.e., You need to prove the correctness of your construction – that the language of the constructed NFA is indeed the desired language $f(L)$.)

(Rubric: Half the credit is for a correct construction, and the other half is for a correct proof of correctness.)

- 11.D.** (40 PTS.) Let $L \subseteq \{0, 1\}^*$ be a regular language. Consider the decoded language $L_f = \{w \in \Sigma^* \mid f(w) \in L\}$.

Prove that L_f is a regular language. As above, given a DFA M for L , describe how to construct an NFA for L_f .

(Rubric: Half the credit is for a correct construction, and the other half is for a correct proof of correctness.)