

Cooperation: Students should talk to each other about the subject matter of this class and help each other. It is fine to discuss the problems and ask questions about them. I encourage such questions in class and office hours as well as elsewhere. However, there is a line past which you must not go, e.g., sharing or copying a solution is not okay and could result in failure. If a significant part of one of your solutions is due to someone else, or something you've read then **you must acknowledge your source!** Failure to do so is a serious academic violation, likely to result in failure of the course or worse. Furthermore, **all solutions must be written by yourself, in your own words.** You may get an idea from somewhere or someone and acknowledge that, but you must still understand it and explain it yourself. A copied solution, even with the source acknowledged will be considered plagiarism. The exception is if it is in quotation marks and cited specifically. But in this case, don't bother because you won't get credit for quoting someone else's solution.

Note: This whole problem set is meant to help with your review of regular languages. If you are already quite up on this topic then this problem set could be quite easy. That's okay, the problem sets should get more interesting soon. I want everyone to write solutions that are **clear** and **correct!** Now and always feel free to ask questions about the homework. The sooner you ask, the more your question and the answer will help you and your classmates. In particular, **please arrive at the discussion section on Friday having thought about the problems due the following Monday**, and prepared to ask any questions you might have about them. If you find typos on the homework, please email me right away – don't wait until Friday. Thanks.

General Strategy for doing the homeworks: My desire for the homeworks is that they help you **understand** the concepts from lecture and readings, i.e., that these concepts not only seem believable, but you can employ them.

For almost all the problems I give, you should be able to look at a few tiny examples, and try to solve the problem on those examples. If you can do it for the tiny examples, then you are part way to giving a rule that will solve the problem in general. If you cannot solve or are confused by what the problem means on one of your small examples, then that would be a great time to ask me or the TA a question about it.

Review: This week's problems encourage you to review and consolidate your knowledge of regular languages, in particular, you should be familiar with the meaning and proof of the following results:

Theorem 1 (Kleene's Theorem) *Let $A \subseteq \Sigma^*$ be any language. Then the following are equivalent:*

1. $A = \mathcal{L}(D)$, for some DFA D .
2. $A = \mathcal{L}(N)$, for some NFA N .
3. $A = \mathcal{L}(R)$, for some regular expression R .
4. A is regular.

Recall that a language *homomorphism* is a function $h : \Sigma^* \rightarrow \Gamma^*$ such that

$$(\forall x, y \in \Sigma^*)(h(xy) = h(x)h(y)) \quad (1.0)$$

Thus, using Equation 1.0, any map $h_0 : \Sigma \rightarrow \Gamma^*$ uniquely determines a homomorphism h .

Theorem 2 (Closure Theorem for Regular Languages) Let $A, B \subseteq \Sigma^*$ be regular languages and let $h : \Sigma^* \rightarrow \Gamma^*$ and $g : \Gamma^* \rightarrow \Sigma^*$ be homomorphisms. Then the following languages are regular:

- | | | |
|-------------------------------|---------------|----------------|
| 1. $A \cup B$ | 2. $A \cap B$ | 3. AB |
| 4. $\bar{A} = (\Sigma^* - A)$ | 5. $h(A)$ | 6. $g^{-1}(A)$ |

[In the above $h(A)$ is the image of the set A under the map h , i.e. $h(A) = \{h(a) \mid a \in A\}$, and $g^{-1}(A)$ is the pre-image of A , i.e. $g^{-1}(A) = \{w \in \Gamma^* \mid g(w) \in A\}$.]

Let $\mathcal{L} \subseteq \Sigma^*$ be any language. Define the *right-equivalence relation* $\sim_{\mathcal{L}}$ on Σ^* as follows:

$$x \sim_{\mathcal{L}} y \iff (\forall w \in \Sigma^*)(xw \in \mathcal{L} \leftrightarrow yw \in \mathcal{L})$$

Lemma 3 (Pumping Lemma for Regular Sets) Let $D = (Q, \Sigma, \delta, s, F)$ be a DFA. Let $n = |Q|$. Let $w \in \mathcal{L}(D)$ s.t. $|w| \geq n$. Then $\exists x, y, z \in \Sigma^*$ s.t. the following all hold:

- $xyz = w$
- $|xy| \leq n$
- $|y| > 0$, and
- $(\forall i \geq 0)xy^iz \in \mathcal{L}(D)$

Corollary 4 The language $E = \{a^r b^r \mid r \in \mathbf{N}\}$ is not regular.

Proof: (of Lemma 3 and Corollary 4) See slides 5 and 6 of Lecture 3. □

Theorem 5 (Myhill-Nerode) The language \mathcal{L} is regular iff $\sim_{\mathcal{L}}$ has a finite number of equivalence classes. Furthermore, this number of equivalence classes is equal to the number of states in the minimum-state DFA that accepts \mathcal{L} .

Problems:

1. [25 pts.] Consider the language: $A_1 = \{a^n b c^{2n} \mid n \in \mathbf{N}\}$.
 - (a) Using the Pumping Lemma, prove that A_1 is not regular.
 - (b) Identify all the equivalence classes of \sim_{A_1} . You should show that every string in $\{a, b, c\}^*$ is a member of exactly one of the classes you have described. Since there are infinitely many equivalence classes, this provides another proof that A_1 is not regular, using the Myhill-Nerode Theorem.
2. [30 pts.] Give linear-time algorithms for the following two problems:
 - (a) Given a DFA, $D = (Q, \Sigma, \delta, s, F)$, determine whether or not $\mathcal{L}(D) = \Sigma^*$.

- (b) Given a DFA, D , determine whether or not $\mathcal{L}(D)$ is infinite. [Hint: linear time graph-theoretic algorithms that you would see in any algorithms course should help here. First clearly state the conditions on D that characterize these properties and then show how to test them. Please do this carefully for part (b) because it is common to not get exactly the right condition.]

3. [20 pts.] Let $d \geq 2$ be a natural number. Show that the following language is regular:

$$A_d = \{w \in \{0, 1\}^* \mid w \text{ as a binary number is divisible by } d; \text{ ignore leading } 0\text{'s}\}$$

[Hint: show how to construct a DFA D_d accepting A_d . Think of the empty string as representing 0, and thus D_d should accept the empty string. You might want to try this first for $d = 3$. D_d can be constructed with exactly d states, but no fewer.]

4. [25 pts.] In this problem let $\Sigma = \{0, 1\}$ and for $j \in \Sigma$, let $\bar{j} = (1 - j)$. You will prove in two different ways that if R is a regular language over Σ^* then the following language is regular:

$$R_{1\text{-error}} = \{w \in \Sigma^* \mid (\exists u, j, v)(w = u j v \ \& \ u \bar{j} v \in R \ \& \ j \in \Sigma \ \& \ u, v \in \Sigma^*)\}$$

That is $R_{1\text{-error}}$ is the set of all strings from R in which exactly one 0 is changed to a 1 or one 1 is changed to a 0. As an example, if R is the language $\mathcal{L}(0^*)$, then $R_{1\text{-error}}$ would be the language $\mathcal{L}(0^*10^*)$.

- (a) In this part, give a machine proof: let N be an NFA accepting R . Construct an NFA, N' accepting $R_{1\text{-error}}$. [Hint: if N has n states, then you can do this with an NFA, N' , having $2n$ states. The idea is to non-deterministically guess which symbol to change.]
- (b) In this part, give a proof using the Closure Theorem for Regular Sets. Starting from R give a series of transformations that preserve the property of being regular that take us from R to $R_{1\text{-error}}$. The hint is that you can use an inverse homomorphism to guess which symbol to change, you can intersect with a regular language to insure that exactly one symbol will be changed, and then use a homomorphism to actually do the change.