Alex Hubers
CSC:341 Spring 2024

Exam #1 (Automata)

Due: February 27, 2024

Academic Honesty & Collaboration. I expect you to work alone, and for your work to represent wholly your own efforts. You are, of course, allowed to ask your classmates (and me) general questions about the material. Please include the following information at the top of your submission, along with your name.

- Written sources used: (Include textbook(s), complete citations for web or other written sources. Write none if no sources used)
- Help obtained: (Include names of anyone other than the instructor.)

Extensions. I am happy to offer extensions, but I require (for this exam) that you ask *in advance* and at least prior to *Friday, Feb 23*. Further, I expect you to be able to show either (i) proof of effort and some progress on the exam, or (ii) some minor proof of conflict in schedule. (Exceptions will be made, of course, in cases of sickness, injury, or emergency.) Finally, please talk to me *as soon as possible* if you know you have a conflict in schedule (e.g., another take-home exam in a separate course) and you need some help meeting this deadline.

The latest extension I can/will give is to Thursday, Feb 29.

Extra Credit. Extra credit questions will be graded (more harshly) on an E/O/N scale:

- **E**xcellent. No errors or only a few tiny errors. Argument is easy to follow. Would be accepted by the broader mathematical community as a valid proof of the claim. Full credit.
- **Ok**ay. Has most of the right answer, but some minor errors. Argument is easy to follow. Would be accepted by the broader mathematical community after some revision. Maximum half credit.
- **N**o credit. Major errors and/or too many minor errors. Would not be accepted by the broader mathematical community.

This is to say, I am looking for correct or nearly correct answers, and will not be offering extra credit to anything less.

Learning outcomes. As per the syllabus, this exam is designed to test your mastery of the following course learning outcomes.

1. Represent a problem using a regular model of computation.
2. Prove closure and algorithmic properties of the regular languages.
3. Prove the irregularity of a problem.
4. Represent a problem using a context-free model of computation.

N.b. For each question that calls for a proof of regularity, I will allow you to exhibit not only the regular models we discussed in class (NFAs, DFAs, and regular expressions) but also any model we showed to be equivalently regular, e.g., Pointed DFAs and All-NFAs, etc. ***Many of the problems below are made dramatically easier by choosing the right model to exhibit.***

#1 Sandwich (50 pts)

Suppose A is a language with alphabet Σ and s is some character not in Σ. Define the sandwich of s by language A, written $\|A\|_s$, as follows:

$$\|A\|_s = \{xsy \mid x, y \in A\}$$

Prove the following claim.

Claim 1. *If A is a regular language with alphabet Σ and $s \notin \Sigma$, then $\|A\|_s$ is regular.*

Observe that the sandwich operator is really just concatenation of the sets A and $\{s\}$:

$$\|A\|_s = A \circ \{s\} \circ A$$

The regular languages are closed under concatenation, which means $\|A\|_s$ is regular.

N.b. a finite automata solution is equally valid. You could also argue that $||A||_s$ can be generated by the regular expression RsR where $L(R) = A$.

#2 Intersection (50 pts)

Define the intersection of sets A and B as follows:

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}$$

Prove or disprove the following claim.

Claim 2. *If languages A and B are regular then $A \cap B$ is regular.*

The claim is true. There are a few ways to prove this. Here are two (there are more).

De Morgan's law. The easiest way to prove the claim is that, by De Morgan's law,

$$A \cap B = \overline{\overline{A} \cup \overline{B}}$$

We have already shown (in class and in homework) that regularity is preserved under complement and union, so it follows easily that $A \cap B$ is regular.

All-NFAs. Let DFAs D_1 and D_2 recognize A and B. We will build an All-NFA N that recognizes $A \cap B$. To do so, repeat the same construction we used to show the regularity of set union. That is, create a new start state q_0 with ϵ-arrows to the start states of D_1 and D_2. Keep the accept states the same. The trick is to observe that the NFA that recognizes $A \cup B$, when declared an All-NFA, will recognize $A \cap B$.

I omit a formal proof of correctness (I do not require one of you). The sketch goes like this: suppose $x \in A \cap B$. Then there are traces through D_1 and D_2 along which the string x is accepted. Both of these traces will result in accept states in the all-NFA that we have built. Because these traces are derived from DFAs, and the only point of nondeterminism is along the epsilon arrows we introduced, we can be certain that these are the only two paths taken by string x. So it is safe to conclude that the All-NFA accepts x.

Next, suppose $x \in A$ but not in B. Then x will have a valid path through N that corresponds to its path through D_1. Its path through D_2 will not end in an accept state, though. So the string is rejected (by the rules of All-NFAs). Finally, if $x \notin A \cap B$ then x is neither in A nor B and can therefore not have a valid path through N.

#3 (Ir)regularity (50 pts)

Let $\Sigma = \{0, 1\}$ and let

$$A = \{w \mid w \text{ has an equal number of zeros and ones}\}$$
$$B = \{w \mid w \text{ contains an equal number of occurrences of the substrings 01 and 10}\}$$

For example, $101 \in B$ because 101 contains a single 01 and a single 10, but $1010 \notin B$ because 1010 contains two 10s and one 01.

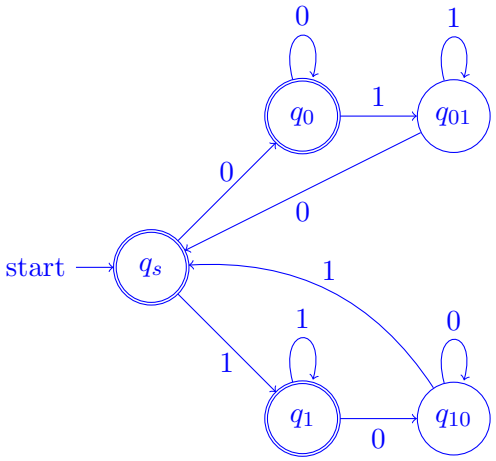
Exactly one of A and B is regular.

1. (25 pts). Which language is regular? Give a proof that it is regular.
2. (25 pts). Which language is irregular? Use the pumping lemma to prove it is irregular.

A is irregular and B is regular.

1. Towards a contradiction, suppose A is regular, and so it has some pumping length p. Consider $s = 0^p1^p$, which has an equal number of zeros and ones and is therefore an element of A. By the pumping lemma, we should have $s = xyz$ such that $xyyz \in A$. However, we know by the pumping lemma that $|xy| \leq p$. Since the first p digits of s are zero, we may conclude that y consists only of 0s. But this yields a contradiction, as the string $xyyz$ will have more zeros than ones. Thus A is irregular.

2. The NFA below recognizes B.



#4 Regular Grammars (50 pts)

Consider the following definition, which connects your study of context-free grammars (CFGs) with regular languages.

Definition 1 (Right-Regular Grammars)**.** *A right-regular grammar (RRG) is a tuple* (N, Σ, P, S) *where*

1. N *is a finite set of nonterminal symbols,*
2. Σ *is an alphabet of terminal symbols disjoint from* N*,*
3. $P \subseteq N \times \Sigma_\epsilon N_\epsilon$ *is a finite set of production rules, and*
4. $S \in N$ *is the start symbol.*

Observe that the type of production rules P is different than that of CFGs. To be clear, the production rules of a right-regular grammar each have one of the following forms:

- $A \to a$,
- $A \to aB$, or
- $A \to B$.

That is, the right hand side of a right-regular grammar production rule is *only* ever: a terminal (alone), a terminal before a nonterminal, or just a nonterminal.

Prove the following claim by structural induction.

Claim 3. *Every regular expression* R *can be translated into a right-regular grammar* G *such that* $L(R) = L(G)$.

1. Case $R = \emptyset$. There are a number of grammars that generate no strings. This one should work fine: $G = (\{A\}, \emptyset, \{A \to A\}, A)$.

2. Case $R = \epsilon$. Consider $G = (\{A\}, \{\epsilon\}, \{A \to \epsilon\}, A)$.

3. Case $R = a$. Consider $G = (\{A\}, \{a\}, \{A \to a\}, A)$.

4. Case $R = R_1 \cup R_2$. For this and the next few cases, suppose we have (By the IH) grammars G_1 and G_2, defined by:

$$G_1 = (N_1, \Sigma_1, P_1, S_1)$$
$$G_2 = (N_2, \Sigma_2, P_2, S_2)$$

   such that $L(G_1) = L(R_1)$ and $L(G_2) = L(R_2)$. (To make things easy for us, assume N_1 and N_2 are disjoint.) We will now build a grammar that recognizes $L(G_1) \cup L(G_2)$. Call it G, defined by:

   (a) $N = N_1 \cup N_2 \cup \{S\}$,
   (b) $\Sigma = \Sigma_1 \cup \Sigma_2$,
   (c) $P = P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}$,
   (d) $S = S$.

   The idea is very simple and mirrors our strategy in building an NFA that recognizes set union. We simply create a new start state S that "branches" off to S_1 and S_2.

5. Case $R = R_1 R_2$. We build a grammar G that recognizes $L(G_1)L(G_2)$, defined by:

   (a) $N = N_1 \cup N_2$,
   (b) $\Sigma = \Sigma_1 \cup \Sigma_2$,
   (c) P,
   (d) S_1

   To form P, place all of P_2 into P. Then for each $A \to aB \in P_1$, if $B \neq \epsilon$, place $A \to aB$ into P. Otherwise, place the production rule $A \to aS_2$ into P instead. In short, we force each string derived from S_1 to continue its derivation through S_2 to reach a terminal state.

6. Case $R = R_1^*$. We will build a grammar G that recognizes $L(G_1)^*$, defined as

$$G' = (N_1 \cup \{L\}, \Sigma_1, P', L)$$

   Our goal is to make L a "loop" symbol by replacing terminal production rules with rules that instead produce L. To construct P', add the production rules $L \to S_1$ and $L \to \epsilon$ to P'. Next consider each production rule $A \to aB$ in P_1. If $B \neq \epsilon$, add $A \to aB$ to P'. Otherwise, add $A \to aL$ to P' instead.

#5 Distinguishability, again (40 pts extra credit)

Recall the definition of distinguishability that was first described in Lab 1A.

Definition 2 (Distinguishability)**.** *Let L be a set of strings and x and y be strings (not necessarily in L).*

- *We say that x and y are distinguishable by L if there exists a string z (also not necessarily from L itself) such that exactly one of xz and yz are in L.*

- *We say that x and y are indistinguishable by L if they are not distinguishable.*

Furthermore, if x and y are strings indistinguishable by L, we write $x \equiv_L y$.

Distinguishability can generalize to n many strings that are each mutually distinguishable. We call this pairwise distinguishability.

Definition 3 (Pairwise distinguishability & Index)**.** *Let L be a language and let X be a set of strings. We say that X is pairwise distinguishable by L if every two distinct strings in X are distinghuishable by L. Define the index of L to be the cardinality of the greatest such X that is pairwise distinguishable by L. The index of L may be finite or infinite.*

For full credit, give a correct proof of the following claim.

Claim 4. *The language A is regular if and only if it has finite index.*

Recall that $\equiv_A$ is an equivalence relation on A. In particular, the index of A is equal to the number of partitions induced on A by the relation $\equiv_A$. As a corollary, the following claim is equivalent.

Claim 5. *The language A is regular iff the number of equivalence classes induced by $\equiv_A$ is finite.*

(You may thus prove either.)

In one direction, we show that if the language L is recognized by a DFA with k states, then L has index at most k. So let M be a k-state DFA that recognizes L and, towards a contradiction, suppose L has index greater than k. By definition, this means that there exists a set X (with $|X| \geq k$) of pairwise distinguishable strings. Our proof relies on the insight that each string in X must transition M to distinct states. To see why, suppose, for example, that $x, y \in X$ each transition M to the state q. Recall that, for x, y to be distinguishable, there *must* exist some z such that exactly one of xz and yz are in L. So we may suppose that this state q has a path along string z to an accept state $f \in F$. But this string takes both x and y to f, and so it is the case that $xz \in L$ and $yz \in L$. So x and y are indistinguishable, a contradiction!

We arrive at a contradiction by the pigeonhole principle: if we have more than k distinct strings in X but only k states, naturally there must be at least two strings $x, y \in X$ that transition M to the same state. And so we may conclude that $|X| \leq k$—in other words, L has finite index.

In the other direction, we show that a language L with finite index k is recognized by a DFA with k states. This direction follows by construction: let $X = \{s_1, ..., s_k\}$ be the largest set of strings pairwise distinguishable by L. Define M to recognize L as $M = (Q, \Sigma, \delta, q_0, F)$ with such that $|Q| = k$ according to:

1. $Q = \{q_1, ..., q_k\}$ (that is, one state per element in X)
2. $\delta(q_i, a) = q_j$ where $s_j \equiv_L s_i a$—that is, state q_i reaches q_j along a iff s_j is indistinguishable from $s_i a$. Note that such an s_j necessarily exists: otherwise, $X \cup \{s_i a\}$ would be an even larger set of pairwise distinguishable elements, contradicting the maximality of X.
3. $F = \{q_i \mid s_i \in L\}$.
4. $q_0 =$ the q_i such that $s_i \equiv_L \epsilon$.

We now argue our construction is correct. Let $q_i \in Q$ be arbitrary. Observe that M is constructed so that

$$\{s \mid \delta(q_0, s) = q_i\} = \{s \mid s \equiv_L s_i\}$$

In other words, the set of strings that take q_0 to q_i is exactly the set of strings indistinguishable from s_i. Now, let $x \in L$. The string x is necessarily indistinguishable with some $s_f \in X$ where s_f corresponds to some $q_f \in F$. (This is a consequence of our decision to let F be the set of states $\{q_i \mid s_i \in L\}$). Hence $\delta(q_0, x) = q_f$, and so δ transitions M to an accept state on input x.