Extra Practice Problems 2

Problem One: Binary Relations

i. Let Σ be some alphabet and L be a language over Σ. Two strings $x, y \in \Sigma^*$ are called *indistinguishable* relative to L, denoted $x \equiv_L y$, iff for every $w \in \Sigma^*$ we have $xw \in L$ iff $yw \in L$. This relation arises in a more complete version of the Myhill-Nerode theorem. Prove that $\equiv_L$ is an equivalence relation over Σ^*.

Proof: We will show that $\equiv_L$ is reflexive, symmetric, and transitive.

To prove reflexivity, consider any $x \in \Sigma^*$. We will prove that $x \equiv_L x$. Note that for every string $w \in \Sigma^*$, we have $xw \in L$ iff $xw \in L$ because any statement is equivalent to itself. Therefore, $x \equiv_L x$ holds, as required.

To prove symmetry, consider any $x, y \in \Sigma^*$ where $x \equiv_L y$. We will prove that $y \equiv_L x$. Since $x \equiv_L y$, we know that for any $w \in \Sigma^*$ that $xw \in L$ iff $yw \in L$. Equivalently, for any $w \in \Sigma^*$, $yw \in L$ iff $xw \in L$. Therefore, by definition of $\equiv_L$, we see that $y \equiv_L x$, as required.

Finally, to prove transitivity, consider any $x, y, z \in \Sigma^*$ where $x \equiv_L y$ and $y \equiv_L z$. We will prove that $x \equiv_L z$. To see this, note that since $x \equiv_L y$, for any $w \in \Sigma^*$ we have $xw \in L$ iff $yw \in L$. Since $y \equiv_L z$, we see for all $w \in \Sigma^*$ that $yw \in L$ iff $zw \in L$. Combining these two statements together, we have that for any $w \in \Sigma^*$ that $xw \in L$ iff $zw \in L$. Therefore, $x \equiv_L z$, as required.

Since $\equiv_L$ is reflexive, symmetric, and transitive, it is an equivalence relation. ■

ii. Prove that a binary relation is a total order iff it is total, antisymmetric, and transitive.

Proof: Let R be a binary relation over A. We will prove that it is a total order iff it is total, antisymmetric, and transitive.

($\Rightarrow$) Let R be a total order. We will prove it is total, antisymmetric, and transitive. Since R is a total order, it is total, reflexive, antisymmetric, and transitive, Therefore, it is total, antisymmetric, and transitive, as required. ■

($\Leftarrow$) Let R be a relation that is total, antisymmetric, and transitive. We will prove that it is a total order. By definition, a total order is a relation that is total, reflexive, antisymmetric, and transitive. Therefore, we only need to show that R is reflexive. To do this, consider any $x \in A$; we will prove that xRx. Since R is total, at least one of xRx and xRx must hold, and therefore xRx is true. Thus R is reflexive, as required. ■

iii. How many equivalence relations are there over the set $\{a, b, c\}$?

Every equivalence relation over a set A defines a partition of A and vice-versa. Consequently, the number of equivalence relations over a set A is equal to the number of ways to partition the elements of A into distinct groups.

For the set $\{a, b, c\}$, there are five ways to do this:

$$\{a\}, \{b\}, \{c\}$$

$$\{a, b\}, \{c\}$$

$$\{a, c\}, \{b\}$$

$$\{b, c\}, \{a\}$$

$$\{a, b, c\}$$

So there are five equivalence relations over $\{a, b, c\}$.

(An interesting detail: the number of equivalence relations over a set of size n is given by the nth ***Bell number***, denoted B_n. The first few Bell numbers are 1, 1, 2, 5, 15, 52, 203, …)

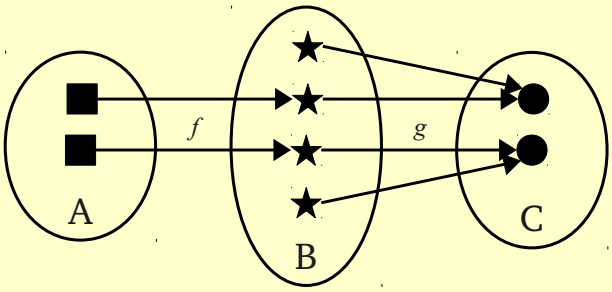
Why we asked this question: One of the topics requested on the Moderator site was more coverage of biconditional statements and how to reason about them. Parts (i) and (ii) of this problem gave examples of proofs involving biconditionals (part (i) due to the definition of the indistinguishability relation, and part (ii) due to the structure of the statement being proven). We hoped that these questions would give different perspectives on how to write proofs involving biconditionals. Part (i) proved results about biconditionals by using properties of "iff" itself, namely transitivity (P iff Q and Q iff R implies P iff R) and symmetry (P iff P always holds). Part (ii) proved the biconditional by proving two separate directions.

Part (iii) of this problem was designed to refresh your understanding of what equivalence relations naturally mean by calling back to the intuition that equivalence relations define partitions of the underlying set into different equivalence classes.

Problem Two: Injections, Surjections, and Bijections

i. Find functions $f : A \to B$ and $g : B \to C$ where $g \circ f$ is a bijection but neither f nor g are bijections.

One possible way to do this is to make the function f not surjective and the function g not injective. Here is a drawing of one possible way to do this:



ii. An *involution* is a function $f : A \to A$ where $f(f(x)) = x$. Prove that all involutions are bijections.

Proof: Let $f : A \to A$ be an involution. We will prove f is a bijection by showing it is injective and surjective.

First, we will prove that f is injective. Consider any $x, y \in A$ where $f(x) = f(y)$. We will prove that $x = y$. To see this, note that since $f(x) = f(y)$, we have $f(f(x)) = f(f(y))$. Since f is an involution, we know that $f(f(x)) = x$ and $f(f(y)) = y$. Therefore, since $f(f(x)) = f(f(y))$, we see $x = y$, as required.

Next, we will prove that f is surjective. Note that for any $x \in A$, there is some y such that $f(y) = x$; namely, choose $y = f(x)$, since $f(y) = f(f(x)) = x$. Therefore, f is surjective, as required. ■

Why we asked this question: Part (i) of this question was designed to get you thinking about properties of bijections and how to construct a bijection between two sets out of functions that weren't actually bijections themselves. We hoped this would help remind you of the intuition behind injections, surjections, and bijectinos. Part (ii) of this question was designed to get you thinking about how to prove that various functions are bijections by using only limited knowledge about their behavior.

Problem Three: Regular and Nonregular Languages

i. Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and let $L = \{ \mathtt{a}^n\mathtt{b}^m \mid n, m \in \mathbb{N} \text{ and } n \neq m \}$. Prove that L is not regular.

Proof: Let $S = \{ \mathtt{a}^n \mid n \in \mathbb{N} \}$. This set is infinite, since it contains one string per natural number. Moreover, we claim that all strings in S are distinguishable relative to L. To see this, consider any strings $\mathtt{a}^n$, $\mathtt{a}^m \in S$ where $n \neq m$. Then $\mathtt{a}^n\mathtt{b}^n \notin L$ but $\mathtt{a}^m\mathtt{a}^n \in L$. Thus $\mathtt{a}^n$ and $\mathtt{a}^m$ are distinguishable relative to L, and since they were chosen arbitrarily we can conclude that any pair of strings in S are distinguishable relative to L. Therefore, by the Myhill-Nerode theorem, L is not regular. ■

ii. Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and let $L = \{ w \in \Sigma^* \mid w \text{ is a palindrome} \}$. Prove that L is not regular.

Proof: Let $S = \{ \mathtt{a}^n \mid n \in \mathbb{N} \}$. This set is infinite, since it contains one string per natural number. Moreover, we claim that all strings in S are distinguishable relative to L. To see this, consider any strings $\mathtt{a}^n$, $\mathtt{a}^m \in S$ where $n \neq m$. Then $\mathtt{a}^n\mathtt{b}\mathtt{a}^n \in L$ but $\mathtt{a}^m\mathtt{b}\mathtt{a}^n \notin L$. Thus $\mathtt{a}^n$ and $\mathtt{a}^m$ are distinguishable relative to L, and since they were chosen arbitrarily we can conclude that any pair of strings in S are distinguishable relative to L. Therefore, by the Myhill-Nerode theorem, L is not regular. ■

iii. Let $\Sigma = \{\mathtt{a}\}$ and let $L = \{ w \in \Sigma^* \mid w \text{ is a palindrome} \}$. Prove that L *is* regular.

Proof: All strings made only of **a**'s are palindromes, since they're the same forwards and backwards. Thus $L = \Sigma^*$, which is a regular language. ■

Why we asked this question: Regular and nonregular languages were one of the popular topics on the Moderator site and so far we've only had a few examples about how to reason about them. We hoped that parts (i) and (ii) of this problem would help you get a better feel for how to use the Myhill-Nerode theorem.

Part (iii) of this problem was mostly for fun. I chose to put it here because it helps show that the way a language is described often influences your perception of how "hard" it is. Asking you to prove the set of all palindromes over $\Sigma = \{a\}$ is regular seems more challenging than asking you to show that Σ^* is regular.

Problem Four: Closure Properties and Nonregular Languages

The regular languages are closed under intersection; if L_1 and L_2 are regular, then $L_1 \cap L_2$ is regular as well.

i. Is the converse of this statement true? That is, if $L_1 \cap L_2$ is regular, then are L_1 and L_2 regular? Prove or disprove this statement.

This statement is false. Take, for example, L_D and $\overline{L}_D$. Neither of these languages are regular, since they're undecidable. However, $L_D \cap \overline{L}_D = \varnothing$, which is indeed regular.

ii. Let $L' = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$. Find a regular language R such that $L \cap R = L'$.

Theorem: $L \cap \mathtt{a^*b^*} = L'$.

Proof: We will prove that $w \in L \cap \mathtt{a^*b^*}$ iff $w \in L'$ by proving both directions of implication. From this, we can conclude that $L \cap \mathtt{a^*b^*} = L'$, since the sets have the same elements.

($\Rightarrow$) Consider any $w \in L \cap \mathtt{a^*b^*}$. Then $w \in L$, so w has the same number of **a**'s and **b**'s, and $w \in \mathtt{a^*b^*}$, so all **a**'s in w appear before all **b**'s in w. Let n be the number of **a**'s in w, which is also the number of **b**'s in w. Then $w = \mathtt{a}^n\mathtt{b}^n$, and so $w \in L'$, as required.

($\Leftarrow$) Consider any $w \in L'$. Then w has the form $\mathtt{a}^n\mathtt{b}^n$ for some $n \in \mathbb{N}$. Thus w has the same number of **a**'s and **b**'s, so it belongs to L, and all **a**'s in w appear before all **b**'s in w, so $w \in \mathtt{a^*b^*}$. Thus $w \in L \cap \mathtt{a^*b^*}$, as required. ■

iii. Using your result from (ii), prove that L is not regular.

Proof: By contradiction; assume that L is regular. Then since the regular languages are closed under intersection, we see that $L \cap \mathtt{a^*b^*}$ is regular. As proven above, $L \cap \mathtt{a^*b^*} = L'$. However, as proven in lecture, L' is not regular. We have reached a contradiction, so our assumption must have been wrong. Thus L is not regular. ■

Why we asked this question: We briefly studied closure properties of regular languages when exploring the regular expressions, but beyond that have not really discussed them. This problem was designed to get you to explore them more. Along the way, we also hoped that these problems would give you more exposure to biconditionals and when they fail. For example, in part (i) you saw that closure properties are only a one-way implication; *if* the input languages are regular, *then* the output languages are regular, but not necessarily the other way around.

Parts (ii) and (iii) were designed to show you a new way to prove a language is not regular: show that after applying certain closure properties to the language, we end with a nonregular language. Notice that the proof in part (ii) is a biconditional, since we're proving two sets are equal.

Problem Five: R, RE, co-RE Languages

i. Prove that the **R** languages are closed under intersection.

Proof: Let L_1 and L_2 be decidable languages and let M_1 and M_2 be deciders for them. Consider the following TM M defined in terms of M_1 and M_2:

> M = "On input w:
>
> Run M_1 on w.
>
> If M_1 rejects w, then M rejects w.
>
> If M_1 accepts w:
>
> Run M_2 on w.
>
> If M_2 accepts w, then M accepts w.
>
> If M_2 rejects w, then M rejects w."

We claim that M is a decider for $L_1 \cap L_2$, meaning that $L_1 \cap L_2$ is decidable, proving the theorem. To show this, we will prove that M is a decider and that $\mathscr{L}(M) = L_1 \cap L_2$.

To see that M is a decider, take any string w. We will prove that M halts on w. When M is run on w, it first runs M_1 on w. Since M_1 is a decider, M_1 halts on w. If M_1 rejects w, then M rejects w and so M halts on w. Otherwise, M_1 accepts w, and so M runs M_2 on w. Since M_2 is a decider, M_2 halts on w. If M_2 rejects w, then M rejects w and so M halts on w. Otherwise, M_2 accepts w, so M accepts w and thus halts on w. Therefore, in all cases, M halts on w. Since our choice of w was arbitrary, this means that M is a decider.

Next, we prove that $\mathscr{L}(M) = L_1 \cap L_2$. To see this, note that if M_1 and M_2 accept w, then M accepts w, and if either M_1 or M_2 do not accept w, then M does not accept w. Therefore, M accepts w iff M_1 and M_2 accept w. In turn, M_1 accepts w iff $w \in L_1$ and M_2 accepts w iff $w \in L_2$. Combined with our previous statement, this means that M accepts w iff $w \in L_1$ and $w \in L_2$. Finally, note that $w \in L_1$ and $w \in L_2$ iff $w \in L_1 \cap L_2$. Therefore, we have that M accepts w iff $w \in L_1 \cap L_2$, so $\mathscr{L}(M) = L_1 \cap L_2$, as required. ■

ii. Prove that the **RE** languages are closed under intersection.

Proof: Let L_1 and L_2 be **RE** languages and let M_1 and M_2 be recognizers for them. Consider the following TM M defined in terms of M_1 and M_2:

M = "On input w:

Run M_1 on w.

If M_1 rejects w, then M rejects w.

If M_1 accepts w:

Run M_2 on w.

If M_2 accepts w, then M accepts w.

If M_2 rejects w, then M rejects w."

We claim that M is a recognizer for $L_1 \cap L_2$, meaning that $L_1 \cap L_2$ is **RE**, proving the theorem. To see this, note that if M_1 and M_2 accept w, then M accepts w, and if either M_1 or M_2 do not accept w, then M does not accept w. Therefore, M accepts w iff M_1 and M_2 accept w. In turn, M_1 accepts w iff $w \in L_1$ and M_2 accepts w iff $w \in L_2$. Combined with our previous statement, this means that M accepts w iff $w \in L_1$ and $w \in L_2$. Finally, note that $w \in L_1$ and $w \in L_2$ iff $w \in L_1 \cap L_2$. Therefore, we have that M accepts w iff $w \in L_1 \cap L_2$, so $\mathscr{L}(M) = L_1 \cap L_2$, as required. ■

iii. Prove that the co-**RE** languages are closed under intersection.

Proof: We will prove that if $L_1 \in$ co-**RE** and $L_2 \in$ co-**RE**, then $L_1 \cap L_2 \in$ co-**RE**. By a theorem proven in lecture, $L_1 \in$ co-**RE** iff $\overline{L_1} \in$ **RE** and $L_2 \in$ co-**RE** iff $\overline{L_2} \in$ **RE**. Since the **RE** languages are closed under union, we know that $\overline{L_1} \cup \overline{L_2} \in$ **RE**. By the same theorem from lecture, we know that the language $\overline{\overline{L_1} \cup \overline{L_2}}$ must therefore be co-**RE**. As proven in lecture when discussing the regular languages, we know that $\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2$. Therefore, $L_1 \cap L_2 \in$ co-**RE**, as required. ■

Why we asked this question: R, **RE**, and co-**RE** languages were another popular topic on the Moderator page. We hoped that this question would give you practice writing high-level descriptions and proofs about Turing machines.

The last part of this problem (about co-**RE**) can also be proven along the same lines as the other results, but I figured that it would be good to show that it's also possible to prove this result by leveraging off the other results we've built up over the course of the quarter.

Problem Six: R, RE, and co-RE Languages II

In lecture, we sketched a proof that if M is a recognizer for L, then the machine M' formed by swapping the accept and reject states of M is a co-recognizer for $\overline{L}$. However, the machine M' won't in general be a recognizer for $\overline{L}$.

Prove that the machine M' formed by swapping the accept and reject states of M is a recognizer for the language $\overline{L}$ iff M is a decider.

Proof: We will prove both directions of implication. First, note that in lecture, we already proved that if M is a decider, then the TM formed by swapping the accept and reject states of M is a decider for the complement of $\mathscr{L}(M)$. Therefore, we only need to prove the other direction of implication: namely, that if M is a recognizer for L that is not a decider, then M' will not be a recognizer for $\overline{L}$.

Suppose that M is a recognizer for L that is not a decider. Then there is some string w such that M loops on w. Since M loops on w, we know that $w \notin L$, so it must be true that $w \in \overline{L}$. Since $w \in \overline{L}$, in order for M' to be a recognizer for $\overline{L}$, M' would have to accept w. However, since M' was formed by swapping the accept and reject states of M and M loops on w, we know that M' loops on w. Therefore, M' is not a recognizer for $\overline{L}$. ■

Why we asked this question: The interplay between **RE** and co-**RE** is an important topic that we touched on in lecture but not much in the problem sets. This question asks you to see exactly what happens if you flip the accept and reject states of a TM – you know that you'll end up with a co-recognizer for $\overline{L}$, and now you know that unless the TM was a decider, you won't end up with a recognizer for $\overline{L}$.

Problem Seven: Mapping Reducibility

i. A *nontrivial language* is a language other than Ø and Σ*. Prove that all nontrivial decidable languages are mapping reducible to one another.

Proof: Let L_1 and L_2 be any two nontrivial decidable languages. We will prove that $L_1 \leq_M L_2$. Since our choice of L_1 and L_2 are arbitrary, this proves that any two nontrivial decidable languages are mapping reducible to one another.

Since L_2 is nontrivial, we know that $L_2 \neq \emptyset$ and $L_2 \neq \Sigma^*$. Therefore, there is some string $w_{yes} \in L_2$ and some string $w_{no} \in L_2$. Consider the following function f, which we claim is a mapping reduction from L_1 to L_2:

$$f(w) = \begin{cases} w_{yes} & \text{if } w \in L_1 \\ w_{no} & \text{otherwise} \end{cases}$$

First, note that this function is computable. This is the case because we can run a decider for L_1 to decide whether $w \in L_1$ and then output w_{yes} or w_{no} based on the result. Moreover, we claim that $w \in L_1$ iff $f(w) \in L_2$. To see this, note that if $w \in L_1$, then $f(w) = w_{yes} \in L_2$. Otherwise, if $w \notin L_1$, then $f(w) = w_{no} \notin L_2$. This means that f is a mapping reduction from L_1 to L_2, so $L_1 \leq_M L_2$, as required. ■

ii. Find an example of an **RE** language and a co-**RE** language that are mapping reducible to one another.

Take any two nontrivial decidable languages. By part (i) of this problem, these languages are reducible to one another. Since **R** = **RE** ∩ co-**RE**, these languages are both **RE** and co-**RE**. Therefore, any two nontrivial decidable languages give a pair of an **RE** language and a co-**RE** language that are mapping reducible to one another.

iii. Prove that $L \leq_M \Sigma^*$ iff $L = \Sigma^*$.

Proof: We prove both directions of implication.

($\Rightarrow$) First, we prove that if $L \leq_M \Sigma^*$, then $L = \Sigma^*$. Since $L \leq_M \Sigma^*$, there is a computable function f such that $w \in L$ iff $f(w) \in \Sigma^*$. Since every string belongs to Σ^*, we see that $f(w) \in \Sigma^*$ is always true. Therefore, since $f(w) \in \Sigma^*$ holds for all w and $f(w) \in \Sigma^*$ iff $w \in L$, this means that $w \in L$ is always true. Therefore, $L = \Sigma^*$.

($\Leftarrow$) Next, we prove that if $L = \Sigma^*$, then $L \leq_M \Sigma^*$. This is equivalent to showing that $\Sigma^* \leq_M \Sigma^*$. To see this, let $f(w) = w$. This function is computable by a TM because the TM can just output its input. Moreover, note that $w \in \Sigma^*$ iff $f(w) \in \Sigma^*$, since $w = f(w)$. Therefore, f is a mapping reduction from Σ^* to Σ^*, so $\Sigma^* \leq_M \Sigma^*$, as required. ■

Why we asked this question: This question explores some details about mapping reducibility that aren't covered in Problem Set 8. Part (i) stresses that a computable function can perform any calculation it needs as a subroutine as long as that calculation eventually terminates. This then makes it possible to reduce any nontrivial decidable language to any other nontrivial decidable language. Part (ii) complements Problem Three from Problem Set 8 by asking you to show that you can indeed find **RE** languages that are reducible to co-**RE** languages and vice-versa. Part (iii) rounded out the picture by exploring why, in part (i), we singled out Σ^* and Ø as "special" when talking about mapping reducibility.