Logic and Computability
Homework 6: Distinguishability and Simulations

Please read Handout 2 before embarking on this assignment. It shows the level of rigor expected for proofs on this assignment.

Challenges

Challenge 1: Proving that Languages are not Regular! [20 Points]

Use the *Pairwise Distinguishability Theorem* to prove that each of the following languages is not regular. Please read Handout 2 for an example of how to write such a proof.

1. The language of all binary strings that contain exactly twice as many 0's as 1's.
2. The language of binary strings $L = \{x | \exists y \in \{0, 1\}^* \text{ s.t. } x = yy\}$. Note that a string of the form yy is a string y appearing twice "back-to-back" as in 011011 or 11 or 10001000.

Answer 1:

Note, I will cite the Handout given for some of the specific verbage in this solution.

1. Consider the infinite set of strings, $S = \{0^{2i} | i \geq 0\}$. We show that S is pairwise distinguishable by showing that every pair of strings in S are distinguishable. Thus, consider an arbitrary pair of strings, $x, y \in S$, we can see clearly then that we can write $x = 0^{2k}$ and $y = 0^{2l}$ such that $k \neq l$. Thus, consider then that we can append the string $z = 1^k$ to x, which gives us $xz = 0^{2k}1^k$ which is an accepting state for our language, but, since $k \neq l$, it follows that $0^{2l}1^k$, we cannot say the same for yz. Therefore, it follows that x and y are pairwise distinct.Thus, S is pairwise distinguishable and has infinite cardinality, and thus no DFA can exist for our language.
2. Consider the infinite set of strings, $S = \{0^i1^i | i \geq 0\}$. We show that S is pairwise distinguishable by showing that every pair of strings in S are distinguishable. Thus, consider an arbitrary pair of strings, $x, y \in S$, we can see clearly then

that we can write $x = 0^k 1^k$ and $y = 0^l 1^l$ such that $k \neq l$. Thus, consider then that we can append the string $z = 0^k 1^k$ to x, which gives us $xz = 0^k 1^k 0^k 1^k = xx$ which is an accepting state for our language, but, since $k \neq l$, it follows that $01^l 01^k$, we cannot say the same for yz. Therefore, it follows that x and y are pairwise distinct.Thus, S is pairwise distinguishable and has infinite cardinality, and thus no DFA can exist for our language.

Challenge 2: Multiples of Three in Binary [25 Points]

Use the Pairwise Distinguishability Theorem to prove that any DFA that accepts the language of multiples of three in binary requires at least three states.

Answer 2:

Consider the set $S = \{\epsilon, 1, 10\}$. We claim that this set is pairwise distinguishable. First, consider the pair $(\epsilon, 1)$. These two strings are distinguished by $z = \epsilon$ as $\epsilon z = \epsilon \in L$, but that $1z = z \notin L$. Next, consider the paring $(\epsilon, 10)$. Once again let $z = \epsilon$, from which we can see that, $\epsilon z = \epsilon \in L$, but that $10z = 10 \notin L$. Thus, the pair are pairwise distinguishable with respect to the string ϵ. Now, consider that the pair $(10, 1)$ are pairwise with respect to the string $z = 1$, as we can see that $10z = 101 \notin L$, but that $1z = 11 \in L$. Therefore, since each element of S is pairwise distinguishable and $|S| = 3$, the Pairwise Distinguishability Theorem asserts that any DFA for L must have at least 3 states.

Challenge 3: 01 and 10! [25 Points]

Consider the language of binary strings in which the number of occurrences of `01` is equal to the number of occurrences of `10`, For example, `00011110` is in the language because it has one `01` pattern and one `10` pattern. The string `01010` is also in the language. But, the string `011001` is not in the language. Is this language regular? If so, submit a Prolog file that represents the DFA (using the DFA representation that we used in HW #5) called `problem3.pl`. If it's not a regular language, submit a proof using the Pairwise Distinguishability Theorem in a file called `problem3.pdf` using the same problem upload (we will grade your proof manually).

Answer 3:

See attatched Prolog File.

Challenge 4: Variants of DFAs [30 Points]

Professor I. Lai wakes up one morning and finds himself in a completely unfamiliar place. He quickly realizes that he's been warped into another universe. The one-eyed aliens there are very friendly, providing him with kiwi-lime-watermelon lollipops and showing him several of their intriguing models of computation. For each of the following models of computation, indicate whether it's **more powerful** or has the **same power** as DFAs and prove your answer.

1. **Middle-Window DFAs:** A Middle-Window DFA is like a DFA but the machine can always see the symbol at the middle of the tape. If the length of the input string is even, there are two middle symbols, in which case the middle-window DFA sees the "left middle". For example, if the input was "milk", the middle-window DFA would see the "i". *Note that each time the tape head moves to the right, the middle window floats to always be at the middle of the remaining string. So, after reading the "m" in milk, the remaining string is "ilk" and the middle window now sees the "l".* More formally, a Middle-Window DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where the components are analogous to a DFA but $\delta : Q \times \Sigma \times \Sigma \to Q$ is a transition function from the current state, current symbol, and middle window symbol to the next state.

2. **Rear-Window DFAs:** A Rear-Window DFA is like a DFA but the machine can always see the last symbol on the tape. A Rear-Window DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where all the components are analogous to a DFA except that $\delta : Q \times \Sigma \times \Sigma \to Q$ is a transition function from the current state, current symbol, and last symbol to the next state.

Answer 4:

1. Consider the proof that Middle-Window DFAs are more powerful than regular DFAs. First, we will show that any language accepted by regular DFAs would also be accepted by our Middle-Window DFAs. Given a DFA, $D = (Q, \Sigma, \delta, q_0, F)$, we can construct a Middle-Window DFA M $= (Q, \Sigma, \delta', q_0, F)$ that is identical to the DFA, with regards to the fact that $\delta(q_i, \sigma) = \delta'(q_i, \sigma, \sigma_2)$. Thus, this constructed Middle-Window DFA behaves just like the DFA but does not pay attention to what the middle value is.

   However, consider the language L such that we accept any string with middle element 1, but that we reject any other string. Thus, consider the following proof that L is non-regular for a regular DFA. Consider the infinite set $S = \{0^i 1 | i \geq 0\}$. We show that S is pairwise distinguishable by showing that every pair of strings in S are distinguishable. To that end, consider an arbitrary pair of two distinct strings $x, y \in S$. Then there exist k, l such that $k \neq l$ and $l > k + 1$ and $x = 0^k 1$ and $y = 0^l 1$. Let $z = 0^k$. We can see then that $xz = 0^k 1 0^k \in S$, but that $yz = 0^l 1 0^k \notin S$. Therefore, S is pairwise distinguishable and has infinite cardinality, and thus no DFA can exist for L.

Now, consider that we can define a Middle-Widow DFA that does accept this simply by defining the transition $\delta'(q_0, 0, 1) = q_1$, $\delta'(q_0, 1, 1) = q_1$, $\delta'(q_0, 0, 0) = q_2$, $\delta'(q_0, 1, 0) = q_2$, where q_1 is the sink state for acceptance (cannot leave), and q_2 is the sink state for rejection (also cannot leave). We note that this machine accepts exactly those strings in L and thus accepts a non-regular language. Thus, we have shown that Middle-Window DFAs are more powerful than DFAs.

2. Consider the proof that Rear-Window DFAs are as powerful as our regular DFAs.

   We show that DFAs and NFAs are equally powerful. To show that, we first show that any language accepted by a DFA is also accepted by a Rear-Window DFA and then we show that every language accepted by a Rear-Window DFA is accepted by a DFA. First, consider a language L that is accepted by a DFA, $D = (Q, \Sigma, \delta, q_0, F)$. We can see that L can also be accepted by a Rear-Window DFA $R = (Q, \Sigma, \delta', q_0, F)$, where we define $\delta(q_i, \sigma_1) = \delta'(q_i, \sigma_1, \sigma_2)$, or in other words, that we just use all of the transitions from our DFA, and only look at the next symbol in our string (ignoring the last symbol). Under this construction, N accepts exactly the inputs accepted by D.

   Conversely, let L be a language accepted by a Rear-Window DFA, $R = (Q, \Sigma, \delta, q_0, F)$, then we can construct a NFA, $N = (Q', \Sigma, \delta', q_0', F')$ where we can define the following properties of N as follows,

   - $Q' = Q \times \Sigma \times \Sigma \cup \{q_0'\} \cup (q_0 \times \{\epsilon\} \times \Sigma)$d $q' = \delta(q, \sigma, \sigma_2)$.
   - For each state, $(q, \sigma_1, \sigma_2) \in Q'$, we can define $\delta'((q, \sigma_1, \sigma_f), \sigma) = (q', \sigma, \sigma_f)$ where we define $q' = \delta(q, \sigma, \sigma_f)$ where σ_f is the last symbol on our string.
   - q_0' is defined as the start state that we transition from using ϵ transitions, defined as follows, $\delta(q_0', \epsilon) = (q_0, \epsilon, \sigma_f)$.

- $F' = (q, \sigma_f, \sigma_f)$

Thus, as we have proven that NFAs are computationally equivalent to DFAs, by the transitive property, it follows that Rear-Window DFAs are computationally equivalent to DFAs as desired.