Homework 3 Solutions
15-212 Java, Spring 1998
Out: 2/4/98. In: 2/11/98
Adam Berger

1. *Say $L_1$ and $L_2$ are regular languages. Is $L' = \{w : w \text{ is in } L_1 \text{ but not in } L_2\}$ necessarily regular? Why or why not?*

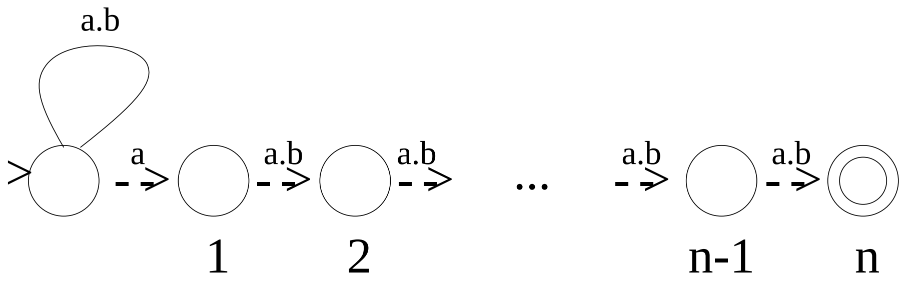
   $L'$, which is just $L_1 \cap \overline{L_2}$, is indeed regular, since regular languages are closed under intersection and complement.

2. *In proving that a DFA is as powerful as an NFA, we started with an n-state NFA and constructed an equivalent DFA with $2^n$ states. In this problem you will prove that in general, there is no more concise way to deterministically simulate a NFA.*

   *Consider the language $L = \{w = (a, b)^\star :$ the n'th symbol from the right end of $w$ is an $a\}$. Prove:*

   (a) *There is an n+1-state NFA that accepts L.*

   Here it is:




   (b) *There is no DFA which accepts L and has fewer than $2^n$ states. (Hint: Use the pigeonhole principle.)*

   If we apply the familiar NFA-to-DFA conversion to the above NFA, we get a DFA with $2^{n+1}$ states. But this doesn't mean there isn't some more clever construction with fewer than $2^n$ states. After all, often when applying the NDA-to-DFA construction we find that many of the resulting states are inaccessible from the start state of the machine and therefore can be eliminated. Not in this case, though. Here's a proof—about as rigorous as we expect from you—that no cleverly-constructed DFA with less than $2^n$ states can exist for $L$.

   To start, assume by way of contradiction that there indeed exists a DFA $M$, with $m < 2^n$ states, which accepts $L$.

   There are $2^n$ strings over $(a, b)$ of length $n$. By the pigeonhole principle, at least two of these strings—let's call them $w_a$ and $w_b$—must drive $M$ to the same state. Without loss of generality, let's say $w_a$ and $w_b$ differ in the $k$'th position, at which point $w_a$ has an $a$ and $w_b$ has the symbol $b$:

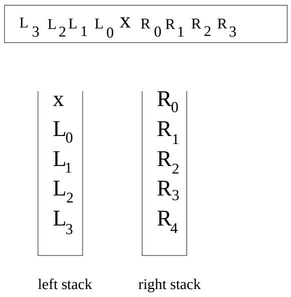
$$\begin{aligned} w_a &: \ aabbababa\ldots ba \\ w_b &: \ \underbrace{aabbb}_{k}\underbrace{baba\ldots ba}_{n-k} \end{aligned}$$

   Now introduce $s$, an arbitrary string of length $k - 1$. $M$ must accept the string $w_a s$, since the $n$'th symbol from the right end of $w_a s$ is an $a$. However, $M$ also accepts $w_b s$, since we've already shown that $M$ is in the same state after reading $w_a$ and $w_b$. Since the $n$'th symbol from the right end of $w_b s$ is an $b$, we've shown that $M$ accepts a string *not* in $L$. We've arrived at a contradiction, and thus our original assumption—that there exists a DFA $M$, with $m < 2^n$ states, which accepts $L$—must be false.

3. *The stack of a pushdown automaton can be considered as a tape that can be written and erased only at one end. In this sense a Turing machine is a generalization of a pushdown automaton. But we can generalize in the other direction as well, by considering a pushdown automaton with* two *stacks. Using a simulation argument, show that a two-stack dpda is just as powerful as a Turing machine. (We are looking for a clear but informal explanation, rather than an explicit, state by state machine specification.)*

   To show the equivalence between these two machines, we need to show that we can simulate a Turing machine by a two-stack dpda. The simulation works as follows.

   The contents of the TM will be stored on the two dpda tapes, which we'll call the left and right stacks. The contents to the left of the tape head (including the square pointed to by the tape head) will be stored on the left stack, and the contents to the right will be on the right tape head.




   We describe the simulation on a case-by-case basis, considering all possible actions by the TM:

   - *Write symbol $s$ on tape head*: pop (anything) from left stack, push $s$ onto left stack.
   - *Read symbol from tape head*: pop (anything) from left stack, push popped symbol back onto left stack
   - *Move right*: pop (anything) from right stack; push popped symbol onto left stack.
   - *Move left*: pop (anything) from left stack; push popped symbol onto right stack.

   In the above description, (anything) means "any symbol in $\Sigma$."

   (Surely some of you turned in more profuse simulation descriptions. But the above constitutes, to the 212 staff, a minimal "clear but informative explanation.")

4 *A* linear-bounded automaton *(LBA) is a type of Turing machine where the tape head isn't allowed to move off the part of the tape containing the input. All the "regular" rules still apply; in particular, the alphabet of the LBA is finite.*

   (a) *Let $M$ be an LBA with $q$ states and $g$ symbols in the tape alphabet. How many configurations of $M$ are there, for a tape of length $n$?*

   A configuration of the machine consists of the contents of the tape, the position of the tape pointer, and the state of the machine. There are $g^n$ different possible tape contents, $n$ possible head positions, and $q$ states, so that's $nqg^n$ different configurations.

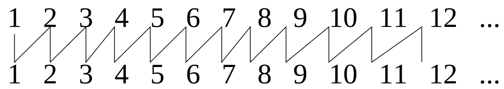
   (b) *Is $L = \{(M, w) : M$ is an LBA that accepts string $w\}$ decidable? Why or why not?*

$L$ is decidable. Given an $(M, w)$, just simulate the behavior of $M$ on $w$, and keep track of how many steps of $M$ you've simulated. If the machine is still running after $nqg^n$ steps, then we know (by the pigeonhole principle) the LBA is in a configuration it's already been in, so it's in an infinite loop. The point is that you only need to simulate $M$ on $w$ for at most $nqg^n$ steps until you can answer the question with a yes or no.

5. *(Flavors of infinity) As discussed in class, a* countably infinite *set is one which can be put in a one-to-one correspondence with the natural numbers. Prove that*

   (a) *The union of two countably infinite sets is countably infinite.*

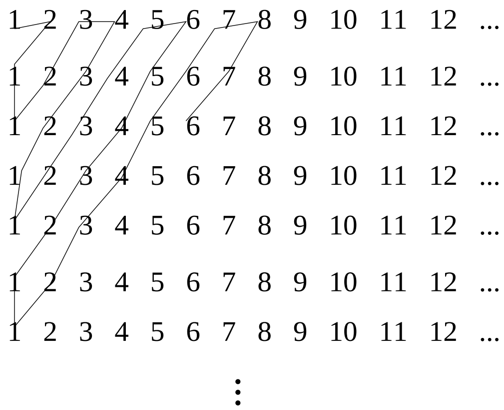
   We illustrate this by example, showing how to put $\mathcal{N} \times \mathcal{N}$, the union of the natural numbers with itself, in a one-to-one correspondence with the natural numbers. The idea is simply to *dovetail* between the first and second set, like this:




   You might wonder if dovetailing is really necessary. That is, couldn't we just enumerate all the elements of the first set, and then enumerate the second? But this doesn't work. Take the first element of the second set. When would we ever count it, if we had to count the entire first set before beginning with the second? In order for a set to be countable, there has to be a way to count it such that every element of the set eventually gets counted.

   (b) *The union of countably many countably infinite sets is countably infinite.*

This is just a generalization of part (a). The dovetailing is a little more complicated, but still works:




6. *One cannot write a program to decide the halting problem on all Java programs less than n characters long, for a fixed value of n. True or false? Substantiate your answer.*

(We decided not to count this problem, but we've provided a solution for your edification.)

Rephrasing the question in terms of languages, we're asking whether

$$L = \{(M, w) : \text{length}(M) < n \text{ and } M \text{ halts on } w\}$$

is a decidable language. Clearly, without the length$(M) < n$ restriction, $L$ is undecidable, since it's just $H$, the halting problem. But does the length restriction change things?

The answer depends on the value of $n$. If $n$ is sufficiently small, then $L$ is decidable. For example, for $n = 1$, there are *no* syntactically valid Java programs and so $L$ is empty. But with enough characters (i.e. $n$ large enough), one could encode a Java simulator—that is, a Java program that takes another Java program and its input and runs the program on itself. This simulator is just like the "universal Turing machine" presented in class. With this Java simulator $S$ we could do the opposite of $M(M)$:

$$S(M) = \begin{cases} \text{halt} & \text{if } M(M) \text{ goes into an infinite loop} \\ \text{go into an infinite loop} & \text{if } M(M) \text{ halts} \end{cases}$$

Now the question, as you should be able to guess by now, is what does $S(S)$ do. The program $S$ is the Java version of the "paradox problem" presented in class. It cannot logically exist, and yet its existence depends only on the decidability of $L$. So therefore $L$ can't be decidable.

**END OF ASSIGNMENT 3**