CS 581: Theory of Computation
Fall 2010
Mid-term exam
James Hook

This is a closed-notes, closed-book exam.

1. PDA construction [25 points]

   (a) Construction

   Given the alphabet $\{(,), [,]\}$ build a PDA that recognizes strings with properly nested balanced parentheses and braces. That is, it should accept (), [], [()], and ([]), but it should reject ([)].

   (b) Justify your construction.

   (c) Illustrate a computation of your machine on the string "([()][])".

2. Not Context Free [25 points]

   Show that the language $\{w\#t \mid w \text{ is a substring of } t, \text{ where } w, t \in \{a, b\}^\star\}$ is not context free.

3. Minimization [25 + 25 points]

   The Myhill Nerode theorem guarantees the existence of a minimal size DFA for any Regular Language. In this problem we explore algorithms to calculate a minimal DFA from an arbitrary DFA.

   There are two basic approaches to the minimization problem. One is to start with (possibly) too many states, and combine them if they are redundant. The other is to start with (possibly) too few states, and divide them as necessary. The two questions below explore aspects of these two strategies.

   The following two questions (labeled parts a and b), while topically related, are independent. They may be attempted independently.

   (a) Size of test [25 points]

   The algorithm that tests for redundant states is possible because there is an upper bound on the length of stings that must be checked to verify indistinguishability.

   Given that A is recognized by some DFA M, can you bound the size of the strings you must test to show that two strings are indistinguishable? That is, can you determine a value n such that: $\forall z.|z| \leq n \rightarrow xz \in A \leftrightarrow yz \in A$ implies $\forall z.xz \in A \leftrightarrow yz \in A$?

   Prove your result.

(b) Partition refinement [25 points]

Notational aside. A *partition* of a set is a set of disjoint sets, called *parts*, the union of which is equal to the set.

The alternative algorithm begins by partitioning the states into two sets (parts), the final states and the non-final states. It then iteratively refines the partition, dividing parts of the partition into new parts as necessary. The necessity of a division is witnessed by testing the states in a part of a partition to see if they are all mapped to the same parts. In particular, if for some input symbol a, some states in part S transition into part R and some states transition into part T, then the partition must be refined by dividing part S into those elements that map to R and those elements that map to T. (The algorithm requires several easily verified assumptions, including that there be at least one string in the language and one string not in the language and that all states in the initial machine are reachable.)

Consider the non-optimal machine to recognize binary numbers congruent to 0 mod 3 constructed as follows:

$$\begin{array}{lcl} M & = & \langle \{0,1,\ldots,5\}, \{0,1\}, \delta, 0, \{0,3\} \rangle \\ \delta(q,a) & = & 2 * q + a \bmod 6 \end{array}$$

i. Test parts [10 points]
Explore the algorithmic idea by testing the initial partition $A = \{0,3\}, B = \{1,2,4,5\}$. For each symbol in the alphabet, test each part to see if it is "good", that is, it contains no "bad pairs" of states p, q where $p, q \in S$ (for some part S), $\delta(p,a) = p'$, $\delta(q,a) = q'$, and $p' \in T$ and $q' \notin T$ for a partition part T.

ii. Refine [10 points]
How can the partition be refined to eliminate the bad pairs without introducing any unnecessary parts in the partition?

iii. Finish [5 points]
Does this refinement contain any bad pairs? Can you recover the optimal 3 state machine from this?