CS 172 — Spring 2000
Computability and Complexity
Prelim #2 Solutions

Solution 1. The language A is context free. We call a word w such that $w = w^R$ a palindrome. We build a CFG for palindromes based on a recursive definition of a palindrome. Note that 0, 1, and ϵ are palindromes. Moreover, if w is palindromatic, so are $0w0$ and $1w1$, and these are the only ways that we can generate palindromes. Hence, a CFG for A is

$$\begin{aligned} S &\rightarrow 0 \mid 1 \mid \epsilon \\ S &\rightarrow 0S0 \mid 1S1 \end{aligned}$$

The language B is also context-free. A CFG for B is

$$\begin{aligned} S &\rightarrow 0S0 \mid 1S1 \mid D \\ D &\rightarrow 1T0 \mid 0T1 \\ T &\rightarrow 0T \mid 1T \mid \epsilon \end{aligned}$$

Intuitively, S generates strings with matched symbols, or D, and D generates strings whose first and last characters do not match (the nonterminal T generates the language $(0 \cup 1)^*$).

Solution 2a. If q is dead, then q is redundant. Suppose q is dead, and let w be a word in $L(M)$. Then, the machine M has an accepting run on w that does not go through q (since q is dead), and we can duplicate this run on the machine $M \setminus q$. On the other hand, if $M \setminus q$ has an accepting run on a word w, this can again be duplicated in M, and moreover, this run does not go through q.

However, even if q is redundant, q may not be dead. This may occur, for example, if M is nondeterministic, and has two runs on the same word. As an example, consider an NFA M. We construct the NFA M' which consists of two identical copies of M, with an additional initial state that nondeterministically chooses to go to either the start state of the first copy or the first state of the second copy. Then, each state in M' (except the initial state) is redundant, but there are states that are not dead.

Solution 2b.

The dead-state problem is the emptiness problem in disguise.

1. D_{NFA} is recursive. This problem can be reduced to graph reachability: $(M, q) \in D_{\mathsf{NFA}}$ iff in the transition graph, there is no path from the start state to the state q.

2. D_{PDA} is recursive. Given a PDA (that accepts on final state) and a state q, make q the only accepting state. Then run the algorithm for emptiness. If the language is empty, then q is dead, and if q is dead, the language is empty.

3. D_{TM} is co-r.e. : guess a string w and simulate the machine M until it hits the state q. It is not recursive, though. We reduce from TMEmptiness. Given a Turing machine M, one can construct an equivalent Turing machine N with only one accepting state q_A, moreover, the Turing machine halts whenever it accepts. Now, suppose we wish to check that N is empty. We ask if $(N, q_A) \in D_{\mathsf{TM}}$. It is clear that M is empty iff N is empty iff (N, q_A) is in D_{TM}.

1. R_{NFA} is recursive. This is because language equivalence is recursive.

2. R_{PDA} is co-r.e. : guess a string w that is in $L(M)$, but not in $L(M \setminus \{q\})$ (or a string w that is in $L(M \setminus \{q\})$ but not in $L(M)$), and run the recursive algorithm for membership on the two machines interleaved. It is not recursive, however. We reduce from CFG universality (which is not recursive). Given a PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, construct the PDA M' by adding a new state q_A, and an edge from q_0 to q_A labeled with ϵ. From q_A, the machine trivially accepts everything. Thus, the PDA nondeterministically decides to go to q_A and accept trivially, or to simulate the original machine. Then the language of M' is clearly Σ^*. However, the language of $M' \setminus q_A$ is the same as the language of M. Then, $L(M') = L(M' \setminus q_A)$ iff $L(M) = \Sigma^*$. But the latter is not recursive, hence the former cannot be recursive either.

3. R_{TM} is neither r.e. nor co-r.e. We reduce from TMUniversality, the idea is similar to the reduction for R_{PDA}. Given a TM M, we construct the TM M' as follows. M' has an initial state q_I from which it nondeterministically decides to go to either the start state of M, or to a new state q_A from which it accepts all inputs trivially. Then, $L(M') = \Sigma^*$, and $L(M' \setminus \{q_A\}) = L(M)$. Hence, $L(M) = \Sigma^*$ iff $L(M') = L(M' \setminus \{q_A\})$.