

Solution Set 7

Posted: March 12

Of course, if you have not yet turned in PS7, you should not consult these solutions.

1. IN-PLACE ACCEPTANCE is in PSPACE because we can simulate M on input x , rejecting immediately if M attempts to leave the first $|x|$ cells of the tape, and otherwise accepting iff M accepts. Since our simulation only needs to keep track of $|x| \leq |(M, x)|$ cells of the tape, it runs in PSPACE.

Now we argue that this language is PSPACE-hard. Given an arbitrary language L in PSPACE, let M be the machine deciding L using polynomial space, and let x be an input to M . We know that M uses at most $|x|^k$ cells of its tape in the course of its computation, for some constant k . Our reduction produces the instance (M, x') where x' is x followed by $|x|^k - |x|$ blank symbols. Clearly, M accepts x' without leaving the first $|x'| = |x|^k$ cells of the tape iff M accepts x iff $x \in L$.

2. (a) DEADLOCK is in NP because given a state $v = (v_1, \dots, v_n)$ that is a purported deadlock state, we can check in polynomial time that for all i, j pairs, there is no pair $((v_i, v'_i), (v_j, v'_j))$ in the list P (there are n^2 pairs of i, j to check, and for each we need to traverse the list P).

To prove that DEADLOCK is NP-hard, we reduce from 3-SAT. Given an 3-CNF formula ϕ with n variables and m clauses, we produce m directed graphs G_1, \dots, G_m . Each $G_i = (V_i, E_i)$ is a directed cycle on 3 vertices; the vertices are labelled with the literals appearing in clause i . Our list P consists of all pairs of edges $(e_i = (v_i, v'_i), e_j = (v_j, v'_j))$ with $e_i \in E_i$ and $e_j \in E_j$ for which the literals labelling v_i and v_j are negations of each other. Now, suppose we started with a satisfiable 3-CNF formula, and fix a satisfying assignment. Then the state $v = (v_1, v_2, \dots, v_m)$ in which each v_i is labelled with a true literal in that satisfying assignment is a deadlock state. In the other direction, if $v = (v_1, v_2, \dots, v_m)$ is a deadlock state, then the literals labelling the vertices v_1, \dots, v_m must be consistent (meaning that no two of these literals are negations of each other), because otherwise, there is an available transition, and so v could not have been a deadlock state. But then the assignment that sets the literals labelling v_1, v_2, \dots, v_m to TRUE is a satisfying assignment to the original 3-CNF.

- (b) We reduce from IN-PLACE ACCEPTANCE. Let (M, x) be an instance of IN-PLACE ACCEPTANCE. We first make some minor modifications to M to prevent it from moving off the beginning or the end of the input. First we place a special marker at the beginning of the input (shifting the input to the right by 1) and then another special marker at the end of the input. Any move by M onto the left-marker is now be followed by a move back to the right (onto the first “real” tape square); this prevents moves off the left of the tape. And, any move by M onto the right marker now results in the machine entering

an infinite loop. Similarly if M is about to reject, it should also enter an infinite loop. So, the modified machine simulates M , never moves outside the $|x| + 2$ tape squares (the input x plus the two end markers), and it halts if (M, x) was a positive instance, and enters an infinite loop if (M, x) was a negative instance.

Define $\Gamma = \Sigma \cup (\Sigma \times Q)$. These are the symbols that appear in a tableau of M 's computation on x (as we used in the reduction showing that CIRCUIT-SAT was NP-complete). Let $n = |x| + 2$, and define the graphs G_1, G_2, \dots, G_n to each be the complete directed graph in $|\Gamma|$ vertices. We identify the vertices of each G_i with Γ . Now we produce the list P consisting of all pairs $((v_i, v'_i), (v_{i+1}, v'_{i+1}))$ for $v_i, v'_i \in V_i$ and $v_{i+1}, v'_{i+1} \in V_{i+1}$ for which

- $v_i, v'_{i+1} \in (\Sigma \times Q)$ and $v_{i+1}, v'_i \in \Sigma$, or
- $v'_i, v_{i+1} \in (\Sigma \times Q)$ and $v'_{i+1}, v_i \in \Sigma$,

and adjacent cells v_i, v_{i+1} in one row of the tableau become v'_i, v'_{i+1} in the next row of the tableau, according to M 's transition function.

If we take $v = (v_1, \dots, v_m)$ to be the sequence in Γ^m corresponding to the first row of the tableau (of M with input x), then there is exactly one reachable state (since M is deterministic), which corresponds to the second row of the tableau. From that state, there is exactly one reachable state corresponding to the third row of the tableau, and so on. Thus if M eventually accepts, there will be a deadlock state (since M halts upon accepting, and there are no further rows of the tableau from that point). The other alternative (recall our initial modification to M) is for M to loop forever; consequently we will never reach a deadlock state.

This completes our reduction from IN-PLACE ACCEPTANCE to REACHABLE DEADLOCK.

3. We want to compute $f(g(x))$ in logspace. We cannot compute $g(x)$ and then evaluate f on it, because we don't have enough storage space to write-down the intermediate result $g(x)$, which may be polynomially long in the input length $|x|$. Let M_f and M_g be Turing Machines that compute f and g in logspace, respectively. We build a new Turing Machine that simulates M_f , and whenever M_f would have read the i -th bit of its input, we pause (remembering M_f 's state in $O(\log |x|)$ bits), and simulate M_g on input x (ignoring what it would have written on its output tape) *until it outputs the i -th bit of $g(x)$* . At this point have the necessary bit to continue simulating M_f . At every point in this simulation we need only remember the state of one of the two machines (requiring space $O(\log |x|)$), while running the other, so the total space is $O(\log |x|)$ as desired.