Theory of Computation

Spring 2024, Homework # 3 Reference Solutions

---

1. (a) L_1 is Decidable.
   Let $n = |w| + |Q| + 1$, where Q is the set of states in M. Construct a decider
   $D_1 =$ "On input $\langle M, w\rangle$ where M is a TM and w is a string:

   ```
   i) Simulate $M$ on $w$ for $n$ steps.
   ii) if $M$ ever makes a left move, accept. Otherwise reject. "
   ```

   To prove that $L_1 = L(D_1)$:
   If $\langle M, w\rangle \notin L_1$, then M never moves left while computing w. So D_1 rejects $\langle M, w\rangle$ thus $\langle M, w\rangle \notin L(D_1)$.
   If $\langle M, w\rangle \in L_1$, then M makes at least one left move while computing w. We show that M must make a left move during the first n steps. Let $p = q_1 q_2 \dots q_k$ be the shortest computation path of M ending in a left move (note that p need not be the entire computation path, just the first part until M makes a left move) . Since the tape contains only blank symbols after the first $|w|$ squares, M reads only blank symbols from the state $q_{|w|}$. Therefore we can remove any cycles (there can be different ones) that occurs afer the first $|w|$ steps from the computation path and still have a legal computation path ending in a left move. For p to contains no cycles after the first $|w|$ steps, $(k - |w|) \le (|Q| + 1)$ by the pigeonhole principle. This means M makes a left move in at most $|w| + |Q| + 1 = n$ steps thus $\langle M, w\rangle \in L(D_1)$.

   (b) L_2 is undecidable. Following is a reduction from A_{TM} to L_2.
   Suppose TM D decides L_2. Construct a TM
   $S =$ "On input $\langle M, w\rangle$ where M is a TM and w is a string:

   ```
   i) Use $\langle M, w\rangle$ to construct
      $M_w =$ "On input $x$:
        i) Simulate $M$ on $w$, but add two more moves (one right and one left) after
           every left move. (That is, immediately after simulate a left move, $S$ makes
           a right move and then another left move, without changing the state or the
           content of the tape.)
        ii) If $M$ rejects, reject.
        iii) If $M$ accepts, make three consecutive right moves following by three consec-
             utive left moves, then accept. "
   ii) Run $D$ on $M_w$. If $D$ accepts, accept. If $D$ rejects, reject. "
   ```

   S decides A_{TM} but A_{TM} is undecidable. A contradiction. So L_2 is undecidable.
   To prove $A_{TM} = L(S)$: If M accepts w, M_w accepts all input (including w) and moves left three times in a row. So D accepts M_w, which means S accepts $\langle M, w\rangle$. Otherwise (i.e., if M does not accept w) M_w makes at most two consecutive left moves so D rejects M_w, which means S rejects $\langle M, w\rangle$.

2. Since A_{CFG} is decidable (as per page 14 of lecture notes Chapter 4), let D be a TM that decides it. Construct a TM
   $N =$ "On input w:

i) Check whether $w = \langle G_1, G_2 \rangle$ for some context-free grammars G_1 and G_2. If not, accept.

ii) Nondeterministically generate a string s. Run D on $\langle G_1, s\rangle$ and $\langle G_2, s\rangle$.

- if D accepts $\langle G_1, s\rangle$ but rejects $\langle G_2, s\rangle$, accept.
- if D rejects $\langle G_1, s\rangle$ but accepts $\langle G_2, s\rangle$, accept.

iii) Reject. "

The proof of correctness is left for exercise.
Note: It is decidable whether $w = \langle G_1, G_2 \rangle$ for some context-free grammars G_1 and G_2. After nondeterministically picking a position in w to separate G_1 and G_2, the rest is simply checking the two encodings to see if they match the definitions of CFG.

3. B_1 is decidable. Asume M is deterministic. Let Q be the set of states and Γ the tape alphabet of M. Recall that a configuration of any TM includes the current state, the position of the tape head, and the content of the tape. So $k = |Q| \cdot 2^{|w|} \cdot |\Gamma|^{2^{|w|}}$ is the maximum number of configurations M can have if M accesses at most $2^{|w|}$ tape squares. We then construct a decider D_1 that recognizes B_1. For convenience, let D_1 have three tapes, one for input, one for simulation, and one for counting the number of moves of M.
D_1 = "On input $\langle M\#w\rangle$:

i) Write a special symbol (that is not in Γ) at the $(2^{|w|}+1)$-th square of the simulation tape.

ii) Copy the string w to the simulation tape and initialize the count to 0.

iii) Simulate M on w using the simulation tape as following:

(1) If the special symbol is being read, reject. Otherwise, simulate the next move (of M) and increase the count by 1.

(2) If M halts, accept.

(3) If the count is $k+1$ (i.e., already simulated $k+1$ moves of M), accept.

(4) Repeat (1). "

D_1 is a decider since it halts after simulating at most $k+1$ moves. In step iii), the correctness of steps (1) and (2) should be obvilus. So here we only prove that (3) is correct. Assume M does not accesses more than $2^{|w|}$ tape squares during the first $k+1$ moves. By pigeonhole principle, at least one configuration would have been repeated (since k is the maximum number of configurations). This indicates a cycle in the computation history. Since M is deterministic, upon entering the cycle, it will continue to repeat the same cycle forever. So M does not access beyond the $2^{|w|}$ tape squares and $\langle M\#w\rangle$ should be accepted.
Note: For the case when M is nondeterministic, the definition of *spaceBound* can be confusing. Specifically, should you take the view that *spaceBound*(M, w, n) is true if M never access more than n tape squares in *any* paths of its computation tree, or do so only when w is not accepted. Also, even though we can build an equivalent deterministic TM from M, it only guarantees that resulting languages are the same but not necessarily the TM behaviors (i.e., how many squares of the tape were actually used). So for the sake of this homework, we only consider the deterministic TM's.

B_2 is undecidable. We prove this by the following reduction from A_{TM}.
Let R be a TM deciding B_2. Construct the TM
$S =$ "On input $\langle M, w\rangle$, where M is a TM and w is a string:

i) Use $\langle M\rangle$ to construct the LBA
$L_{B_2} =$ "On input $\langle C_1, C_2, \ldots, C_l\rangle$, where C_i's are configurations of M:
   (1) If C_1 is not the start configuration of M on w, reject.
   (2) If C_l is not an accepting configuration of M, reject.
   (3) For each $1 \leq i \leq l$, if C_i does not yield C_{i+1}, reject.
   (4) Otherwise, accept. "

ii) Use $\langle M\rangle$ and $\langle L_{B_2}\rangle$ to construct the TM
$M_{B_2} =$ "On input $\langle C_1, C_2, \ldots, C_l\rangle$, where C_i's are configurations of M:
   (1) Simulate L_{B_2} on $\langle C_1, C_2, \ldots, C_l\rangle$.
   (2) If L_{B_2} rejects, reject.
   (3) If L_{B_2} accepts, first access (either read from or write to) the n-th position on the tape where $n = 1 + 2^{|\langle C_1, C_2, \ldots, C_l\rangle|}$, then accept. "

iii) Run R on $\langle M_{B_2}\rangle$.

iv) If R rejects, accept. If R accepts, reject. "

Since L_{B_2} is an LBA, step ii)(1)and ii)(2) can be completed without exceeding the limitation placed by B_2. So the only scenario where M_{B_2} exceeds the limit is if step ii)(3) is executed. This is the key to the reduction.
Proof of the correctness by showing that $L(S) = A_{TM}$: If M accepts w, there must be an accepting computation history (let it be $\alpha_0, \alpha_1, \ldots, \alpha_k$) for M on w. From the definition of M_{B2}, it accepts $\langle \alpha_0, \alpha_1, \ldots, \alpha_k\rangle$ and also accesses the part of the tape not allowed by B_2. Therefore R rejects M_{B_2} and S accepts $\langle M, w\rangle$. Similarly, if M does not accept w, all computation history of M on w will be non-accepting. So $M_{B_2} = \emptyset$, which means step ii)(3) never gets executed. Therefore R accepts $\langle M_{B_2}\rangle$ and S rejects $\langle M, w\rangle$. So S decides A_{TM} which is a contradiction. So B_2 is undecidable.

4. (a) $PCP_\star$ is undecidable.

Proof by reducing $PCP_\star$ from PCP: Let $u = u_1 \ldots u_n$ be a string where $u_i \neq \epsilon$ for $1 \leq i \leq n$. Define $\star u = \star u_1 \star u_2 \star \cdots \star u_n$. For consistency, let $\star\epsilon = \epsilon$ (although we won't be using it). Given a PCP $P : \{[\frac{t_1}{b_1}], \ldots, [\frac{t_k}{b_k}]\}$ where $t_i \neq \epsilon$ and $b_i \neq \epsilon$ for all $1 \leq i \leq k$, construct $P_\star : \{[\frac{\star t_1}{\star b_1}], \ldots, [\frac{\star t_k}{\star b_k}]\}$. Clearly every domino in $P_\star$ is a $\star$-domino. If $P \in PCP$, let $[\frac{t_{i_1}}{b_{i_1}}][\frac{t_{i_2}}{b_{i_2}}] \cdots [\frac{t_{i_m}}{b_{i_m}}]$ be a match in P. So $t_{i_1} t_{i_2} \cdots t_{i_m} = b_{i_1} b_{i_2} \cdots b_{i_m}$. Since $t_{i_l} \neq \epsilon$ and $b_{i_l} \neq \epsilon$ for $1 \leq l \leq m$, $\star t_{i_1} \star t_{i_2} \cdots \star t_{i_m} = \star b_{i_1} \star b_{i_2} \cdots \star b_{i_m}$ and $[\frac{\star t_{i_1}}{\star b_{i_1}}][\frac{\star t_{i_2}}{\star b_{i_2}}] \cdots [\frac{\star t_{i_m}}{\star b_{i_m}}]$ is a match in $P_\star$. So $\langle P_\star\rangle \in PCP_\star$. The reverse is also true. We can get a match in PCP by removing the $\star$'s from the dominos used in any match in $PCP_\star$. Therefore $PCP \leq PCP_\star$ and $PCP_\star$ is undecidable.

Note: We explicitely excluded the use of empty strings in dominos. (And our solution would *not* work for those cases.) So our proof is correct but incomplete. However it is know that for every $P = \{[\frac{t_1}{b_1}], \ldots, [\frac{t_k}{b_k}]\}$ in standard PCP, there is a $P' = \{[\frac{t'_1}{b'_1}], \ldots, [\frac{t'_l}{b'_l}]\}$ in PCP such that t'_j and b'_j are not empty for all j and P has a match if and only if P' has a match. Besides, P' can be constructed efficiently. So this would fill the gap in our proof. The detailed proof is somewhat tedious and therefore will not be included here. But the idea is to first reduce standard PCP to MPCP, then from MPCP to the restricted PCP where empty startings are not allowed.

(b) PCP_1 is decidable.

Let $P = \{[\frac{t_1}{b_1}], \ldots, [\frac{t_n}{b_n}]\}$ in PCP_1 over Σ. To be consistent, we ask that either t_i or b_i be a nonempty string for $1 \leq i \leq n$. Claim that P has a match iff there is a sequence $i_i, \ldots, i_m$ with $m \leq n$ such that $t_{i_1} \cdots t_{i_m} = b_{i_1} \cdots b_{i_m}$. If the claim is true, we only need to test all the possible sequences of dominos whose length is no greater than n to be able to decide whether a match exists. Clearly there are only finite number of such sequences, so the problem is decidable.

To prove our claim is true: If there is i such that $t_i = b_i$, then the sequence i is already a solution (i.e., domino $[\frac{t_i}{b_i}]$ is a match) and the claim is true. Assume that $t_i \neq b_i$ for all i. Let $i_1, \ldots, i_m$ be the shortest sequence such that $t_{i_1} \cdots t_{i_m} = b_{i_1} \cdots b_{i_m}$ and $m > n$. Since $t_i \neq b_i$ for all i, either $t_{i_1} = \epsilon$ or $b_{i_1} = \epsilon$. For convenience, let $b_{i_1} = \epsilon$. Then $t_{i_m} = \epsilon$ and $|t_{i_2} \cdots t_{i_{m-1}}| = |b_{i_2} \cdots b_{i_{m-1}}|$. If there is $l < m$ such that $|t_{i_1} \cdots t_{i_l}| \leq |b_{i_1} \cdots b_{i_l}|$, then there is a solution with length shorter than m (see Note 1). Accordingly, assume that $|t_{i_1} \cdots t_{i_l}| > |b_{i_1} \cdots b_{i_l}|$ for each $l < m$. Since $m > n$, there are $j, k \in \{1, \ldots, m\}$ and $j < k$ such that $i_j = i_k$. Now, we want to subtract i_j from the sequence (without invalidate the match, of course). Firstly, there are three possible cases, $|b_{i_j}| = 0$, $|t_{i_j}| = 0$, and $|t_{i_j}| = |b_{i_j}| = 1$. Since case $|b_{i_j}| = 0$ and $|t_{i_j}| = 0$ are symmetric and $|t_{i_j}| = |b_{i_j}| = 1$ can be seen as the combination of the $|b_{i_j}| = 0$ and $|t_{i_j}| = 0$, we only discuss case $|b_{i_j}| = 0$ here.
If $|b_{i_j}| = 0$, $|t_{i_j}| = 1$, and α is the index such that $b_{i_\alpha} \neq \epsilon$ and $|t_{i_1} \cdots t_{i_j}| = |b_{i_1} \cdots b_{i_\alpha}|$ (i.e., $t_{i_1} \cdots t_{i_j} = b_{i_1} \cdots b_{i_\alpha}$ and moreover $t_{i_j} = b_{i_\alpha}$), then $\alpha > j$ and $\alpha \neq k$. We subtract i_α from the sequence and the process is completing; otherwise, use the same logic, identy the domino whose bottom symbol corresponds to the top symbol of t_{i_α} and subtract it, continue until the top of the domino removed is ϵ. Since $|t_{i_1} \cdots t_{i_l}| > |b_{i_1} \cdots b_{i_l}|$ for each $l < m$, after the subtracting processes, the sequence is still nonempty and remains a match. Finally, we have a sequence that contains no dupilicate dominos and therefore $m \leq n$. The claim is true.
If $t_{i_\alpha} = \epsilon$, then (i_j, i_α) is a pair where the t_{i_j} corresponds to b_{i_α} in the match so removing both of them keeps the resulting sequence as a valid match. otherwise continue this procedure until we reach

Note 1: If $|t_{i_1} \cdots t_{i_l}| \leq |b_{i_1} \cdots b_{i_l}|$, since $b_{i_1} = \epsilon$, there must be at least one ϵ in $t_{i_1} \cdots t_{i_l}$. By repeatedly removing the trailing sequence starting with the last ϵ on the top, one can find another sequence with $|t_{i_1} \cdots t_{i_{l'}}| = |b_{i_1} \cdots b_{i_{l'}}|$ which means $i_1, \ldots, i_{l'}$ is a solution with length (l') shorter than m.