CoCo - Exam 2016

Johannes Agerskov

Dated: April 9, 2021

Question 1

Part 1.1

Define $B = \{s \mid s \in \Sigma^* \wedge s\text{is balanced}\}$. We show that B is context free, but not regular.

Proof. B is generated by the CFG, G

$$S \to (S) \mid [S] \mid SS \mid \epsilon, \tag{0.1}$$

since this grammar generates any balanced set of parenthises and bracket. Clearly G generates only balanced expressions. On the contrary any balanced s, we may identify the subexpressions of the form () or [], we may replace these by (S) and $[S]$, we can now reverse the derivations and replace $(S) \to S$ and $[S] \to S$ and $SS \to S$, and since s was balanced this can be continued until we are left with S. This shows that any balanced S can be derived from from G. □

Part 1.2

$B_e = \{s \mid s \in B \wedge s$ has the same number of (and [$\}$. We show that B_e is not context free.

Proof. Assume for contradiction that B_e is context free, by the pumping lemma for CFLs, B_e has a pumping length p. Consider the string $s = ({}^p)^p[^p]^p \in B_e$. According to the pumping lemma, this may be split in $s = uvxyz$ such that $|vy| > 0$ and $|vxy| \leq p$ and such that $uv^ixy^iz \in B_e$ for any $i \in \mathbb{N}_0$. Since $|xyz| \leq p$ we have that either vy contains only parenthises *or* only) and [*or* only brackets. In the first and last case, it cannot be pumped, since we would not have an equal number of parenthises and bracket. In the second case, it cannot be pumped, since uv^ixy^iz would not be balanced for $i > 1$ or $i = 0$. Thus we have obtained a contradiction and B_e cannot be context free. □

Part 1.3

Let $P = \{s \mid s \in B \wedge s \in \{(,)\}^*\}$ and $P_3 = \{s \mid s \in P \wedge s$ has nesting depth 3$\}$. We show that P_3 is regular.

Proof. We construct a DFA, that accepts P_3. The DFA consists of 6 states q_0, r, 0, 1, 2, 3, $\dot{1}$, $\dot{2}$, $\dot{3}$, where q_0 is the start state and r is a discard state and $\dot{0}$ is the accepts state. We include then transitions function

δ	(	)
q_0	1	r
1	2	0
2	3	1
3	r	$\dot{2}$
$\dot{2}$	$\dot{3}$	$\dot{1}$
$\dot{3}$	r	$\dot{2}$
$\dot{1}$	$\dot{2}$	$\dot{0}$
$\dot{0}$	$\dot{1}$	r

Clearly if the nesting depth exceeds 3 the DFA rejects, if the nesting depts is below 3 the DFA rejects, and if the parenthesis is not balaced the DFA rejects. Thus it only accepts, for balanced parenthesis with nesting depth 3. □

Part 1.4

P is in L

Proof. To check that s is balanced we construct log-space TM, M, that does the following: "On input s

1. Scan the input, while keeping a (binary) counter on the work tape, that increments 1 for every open parenthesis, and decrement 1 for every closed parenthises.
2. If at any point the counter is 0 and a closed parenthises is read, *reject.*
3. If the input is scanned completely and the counter is 0, *accept*, else, *reject.*

Since the counter requires only log-space, this is clearly log-space, and evidently it accepts exactly P. Clearly, M only accepts balanced inputs, since any closed parenthesis is matched by it closest open parethises on the left. □

Part 1.5

B is in L

Proof. Following the hint, we may construct the machine M' "On input s

1. Run $\tilde{M}$ on s, where $\tilde{M}$ is just M from before, but where it reads () and [] on equal footing.
2. For each $i = 1$ to n if a_i is open, find j such that a_j is the corresponding closing parenthises/bracket. Check that they are of the same kind. If true for all i *accept*, if false for some i *reject.*

Evidently M' accepts s if and only if s is balanced. □

Question 2

Part 2.1

Define d-PCP as PCP but where a match is a collection of dominos such that the top string and bottom string have the same length but differ in at most d symbols.
We show that d-PCP is undecidable.

Proof. Consider the mapping reduction, given an instance of PCP, produce the dPCP instance consisting of the set of dominos, which are identical to the ones in the PCP instance but with all symbols repeated $d+1$ times. This is clearly computable, since it is computed by decider TM that just copies everything d times. Clearly if the PCP instance have a match, the d-PCP instance have a corresponding match, which is in fact a 0-*match* On the other hand since every symbol is $d+1$-fold, if the dPCP has a match, then every symbol must match,*i.e.* it must be a 0-*match*, since a single mismatch would automatically generate at least $d+1$ mismatches. Thus the original PCP instance also has a match. Therefore we see that this mapping reduction reduces PCP to d-PCP and $PCP \leq_m d\text{-}PCP$. It then follows that dPCP is undecidable, since PCP is. □

Part 2.2

We show that L_{erase} is co-Turing-recognizable.

Proof. We do this be mapping reduction from $\overline{A_{TM}}$. Given $\langle M, w\rangle$ construct the TM, M' that ignores its input and simply print a special character, say ℵ on the left most position on the tape. It then simulates M on input w on the rest of the tape, which is clearly possible in spite of the character on the leftmost position. Notice that this clearly is construction is computable. If M accept, M' accepts, if M rejects, M' erase the entire tape and moves its head to the leftmost position (and then reject). Clearly if $M \in \overline{A_{TM}}$ then M either rejects or runs forever. In either case, M' runs forever, or erases the entire tape and moves its head to the leftmost position. Thus M' is in L_{erase}. On the contrary if M does accept w then M' will halt and never have an empty tape, so M' is not in L_{erase}. Therefore, this constitutes a mapping reduction, $\overline{A_{TM}} \leq_m L_{erase}$. Thus L_{erase} is co-Turing-recognizable. THIS SHOWS THAT L_{erase} IS NOT TURING-RECOGNIZABLE. □

Proof. Given $\langle M\rangle$, construct M' that simulates M on all strings in parallel, if ever the tape is empty and the head is on the leftmost position in any of the simulations, M' terminates that simulation. If some simulation halts without being terminated, M' accepts. Clearly M' accepts $\langle M\rangle$ if and only if there exist a string, w such that M halts on w without the tape have ever been empty witht he tapehead on the leftmost, positon. Thus M' accepts $\overline{L_{erase}}$ and L_{erase} is co-Turing-recognizable. □

Part 2.3

Let $\langle M \rangle \in L_{erase}$ and define $S = \{s \in \Sigma^* \mid \exists w \in \Sigma^*, \text{such that } M \text{ halts on input } w \text{ with } s \text{ on its tape}\}$. We show that $|S| \leq |Q|$, where Q is the set of states of M.

Proof. Given any $s \in S$ there exist $w \in \Sigma^*$ and a computations history $C_1, ..., C_n$ of M on w such that s is tape content in C_n. Furthermore, there exist $k \leq n$ such that $C_k = q$ for some $q \in Q$. Now let w be the first string in lexicographical ordering, such that M halts on w with s on the tape. Let k_{min} be the minimal k such that $C_{k_{min}} = q$ for some $q \in Q$. Then the map $s \mapsto q$ is injective, since $C_k = q$ fixes remaining computation history uniquely and therefore s is fixed, *i.e.* given a q we may find s, if it exists, by simulating M on the empty string, starting in state q. Thus each q can be reached from at most one s, and $|S| \leq |Q|$. Notice each q need not give an s, as M need not halt on ϵ starting in state q. □

Question 3

Part 3.1

Let k be a natural number. We define

$$k\text{-}HAMPATH = \{\langle G, s, t\rangle \mid G \text{ is a directed graph and there is a set of paths } (\gamma_i)_{i=1}^{n}, \text{ with } n \leq k \text{ in } G, \text{ such that one of the paths start at } s \text{ and ends at } t, \text{ each path visit any vertex in } G \text{ at most once, and any vertex in } G \text{ is visited by at least one path}\}$$

We show that kHAMPATH belongs to NP.

Proof. We construct a polynomial time verifier for $k - HAMPATH$. Consider the verifier $V =$"On input $(\langle G, s, t\rangle, c)$

1. Verify that c is a collection of k or less paths in G (each of length less than or equal to the number of vertices in G).
2. Verfy that one of the paths start at s and end at t.
3. Verify that any of vertex of G is in one of the paths.
4. Verify that all the paths visit any vertex in G at most once.
5. If all the above are yes instances, *accept*, else *reject*.

Step 1 runs in polynomial time, since it needs only check that there are k or less elements in c each of which consistist of valid paths in G that have length less than $n = |V(G)|$ Notice that there are at most n^2 edges in G to check for each edge in the path, and since we can reject of there are more than n edges, this can be done in at most $O(n^3)$ time. Step two is clealy

polynomial in time. Step 3 is also clearly polynomial in time, as we can scan c for each vertex and c has at most lenght kn. Step 4 is also a simple scanning procedure, so polynomial in time for each vertex. Thus this shows that k-$HAMPATH$ is in NP. □

Part 3.2

We show that k-$HAMPATH$ is NP-complete.

Proof. We proceed by polynomial time reducing from $HAMPATH$, which is known to be NP-complete. Given input $\langle G, s, t\rangle$, construct the graph $\tilde{G}$ consisting of G with aditional $k-1$ vertices that are disconnected from the rest of $\tilde{G}$. This can clearly be done by a polynomial time tranducer. Notice now that $\langle \tilde{G}, s, t\rangle$ is in k-$HAMPATH$ if $\langle G, s, t\rangle$ is in $HAMPATH$, since the Hamiltionian path together with the collections of trivial paths that visit each of the extra vertices seperately, *i.e.* paths that visit one and only one vertex each, constitutes the desired collection of at most k paths. On the contrary if $\langle \tilde{G}, s, t\rangle$ is in k-$HAMPATH$, then there must be $k-1$ trivial paths that visit one and only one node, and thus the last path must be a Hamiltonian path from s to t of G, so that $\langle G, s, t\rangle$ is in $HAMPATH$. Thereby, it has been shown that $HAMPATH \leq_P k$-$HAMPATH$, and we conclude that k-$HAMPATH$ is NP-hard. By Part 3.1 it follows that k-$HAMPATH$ is NP-complete. □

Part 3.3

Let $LADDER_{DFA}$ be the language of strings $\langle M, s, t\rangle$ such that M is a DFA and $L(M)$ contains a ladder of strings starting with string s and ending with string t. We show that $LADDER_{DFA}$ is in PSPACE.

Proof. By Savitch's theorem, PSAPCE=NPSPACE, so we proceed by construction a polynomial space NTM that decides $LADDER_{DFA}$. Simply consider $N =$"On input $\langle M, s, t\rangle$

1. Non-deterministically change a single character in s, and run M on the new string while keeping a counter of how many times this step has been done on the branch. If M rejects, *reject* that branch. If the new string is t, *accept*.
2. If the counter exceeds $|\Sigma|^n$ where $n = |s|$, then *reject*.

Clearly we can reject after $|\Sigma|^n$ changes, since there are at most $|\Sigma|^n$ different string of length n, and therefore, if we do more changes than this, we are bound to loop on that branch. Clearly all steps uses polynomial space (DFA uses no space), and N is a decider since it is bound to reject at all branches if one of them have not accepted before step 2. Thus $LADDER_{DFA}$ is in NPSPACE=PSPACE. □