CoCo - Assignment 5

Johannes Agerskov, Mads Friis Frand-Madsen, and Sifan Huang

Dated: March 12, 2021

7.7

We show that NP is closed under union and concatination.

Proof. We start by considering union. Let $A, B \in \text{NP}$ and consider $L = A \cup B$. We construct a polynomial time verifier for L. We know that there exist polynomial time verifiers V_A and V_B for A and B. Now Consider the verifier V that on input $\langle w, c\rangle$, run V_A and V_B on $\langle w, c\rangle$ and if one of them accepts, V accepts, otherwise it rejects. This is clearly polynomial in time, since it is just two polynomial time verifiers in series. Also V accepts w for some c if and only if $w \in A$ or $w \in B$ or equivalently if and only if $w \in L = A \cup B$.
For the concatenation, let $L' = AB$. We construct polynomial time verifier V' such that V' verifies L'. A certificate, c, in this case, is an encoding of three things: Where to split the input w, a certificate for V_A on the left part, and a certificate for V_B on the right part. Given such a $c = (\langle k\rangle, c_A, c_B)$, where $\langle k\rangle$ denotes a suitable encoding of the number k, and input $w = w_1...w_n$, V' acts as following on input $\langle w, c\rangle$: V' splits the input, w, at the kth position, according to the certificate, c. It then runs V_A on $w_1...w_k$ with certificate c_A and V_B on $w_{k+1}...w_n$ with c_B. If both accept V' accepts. Clearly, V' accepts w for some c if $w \in L' = AB$. On the other hand, if V' accepts w for some c there exist a splitting of w such that the left part is in A and the right part is in B or equivalently, $w \in L' = AB$ □

7.9

We show that $TRAINGLE = \{\langle G\rangle \mid G \text{ is an undirected graph that contains a 3-clique}\}$ is in P.

Proof. We consider the following algorithm. $D =$"On input $\langle G\rangle$

1. check that $\langle G\rangle$ encodes an udirected graph. (say with vertices $v_1, ..., v_n$)
2. For $i = 1$ to $n - 2$ select v_i and:
3.   For $j = i + 1$ to $n - 1$ select v_j and:
4.   For $k = j + 1$ to n select v_k and:

5. Check that $\{v_i, v_j, v_k\}$ forms a 3-clique. If true accepts. If false and $i = n-2$, $j = n-1$ and $k = n$, reject."

Since there are $\frac{n(n-1)(n-3)}{6} = O(n^3)$ ways to choose 3 vertices out of n and each selection process if polynomial in time, this algorithm is clearly polynomial in time. It is also clear that the algorithm accepts if and only if G contains a 3-clique (Triangle).
Notice that we use that for any reasonable encoding of a $\langle G \rangle$ the input length is polynomial in the number of vertices. □

7.10

We show that $ALL_{DFA} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \Sigma^*\}$ is in P.

Proof. We show this, by noting that ALL_{DFA} can be solved with the TM, M from the proof that $PATH$ is in P. Simply construct the TM, $M' =$"On input $\langle A \rangle$

1. Construct the state diagram, G of A, viewed as a directed graph.
2. For all states $q \in Q \setminus F$ (F is the set of accepts states of A), run M on $\langle G, q_{\text{start}}, q \rangle$, if it accepts for some q, *reject* if it rejects for all q, *accept.*

Evidently, M' accepts exactly those DFAs that can never reach a non-accepts state, *i.e.* those that accept the language Σ^*. Since $PATH$ is in P, M runs in polynomial time, $\langle G, q_{\text{start}}, q \rangle$ is polynomial in the input length, $|\langle A \rangle|$, and since the set Q is polynomial in the input length, we conclude that M' is a polynomial time TM that accepts ALL_{DFA}. □

7.42

We show that P is closed under the star operation.

Proof. Let A be any language in P. We follow the hint, and construct a polynomial time TM that accepts A^*. Let D be the polynomial time TM that accepts A, and let M be the polynomial TM that accepts $PATH$. Consider then, $M' =$"On input $y = y_1...y_n$

1. Build $n \times n$-table, with entries $T_{i,j}$, for $i, j = 1, ..., n$, by the following procedure:
   For $i = 1$ to n
2. For $j = i$ to n
3. Run D on $y_i...y_j$, if it accepts, set $T_{ij} = 1$, if it rejects set $T_{ij} = 0$. For $i > j$ set $T_{ij} = 0$.
4. Let G be the directed graph with adjacency matrix T, *i.e.* view $\langle T_{ij} \rangle_{i,j=1}^n$ as an encoding of the graph G (we label the vertices of G by $1, ..., n$).
5. Run M on $\langle \langle T_{i,j} \rangle_{i,j=1}^n, 1, n \rangle$, if it accepts, *accept*, if it rejects, *reject.*

By design, we that M' accepts y if and only if there is a path $1 \to i_1 \to i_2 \to ... \to i_l \to n$ such that $T_{1,i_1} = T_{i_j,i_{j+1}} = T_{i_l,n} = 1$ for all $j = 1,...,l$, which is equivalent to $y_1...y_{i_1} \in A$, $y_{i_j}...y_{i_j+1} \in A$, and $y_{i_l}...y_n \in A$ for all $j = 1,...,l$, or equivalently that $y \in A^*$. Furthermore, we see that all steps are polynomial time the input length, and the number of steps is also polynomial in the input length from which we conclude that M' is a polynomial time TM that accepts A^*. $\square$

7.44

Let *UNARY-SSUM* be the subset sum problem where all numbers are represented in unary numbers.

Proof. We describe an algorithm that accepts *UNARY-SUM* in polynomial time. Let M be an algorithm that solves the SUBSET-SUM problem in exponential time, which is known to exist, since $SUBSET-SUM$ is in NP by Theorem 7.25, and by Theorem 7.20 there then exist a non-deterministic TM that decides it in polynomial time $O(n^m)$ for some $m \geq 1$, and then by Theorem 7.11 there exist a deterministic single tape TM that desides it in exponential time ($2^{O(n^m)}$). Consider $D =$" on input $\langle\{i_1, ..., i_n\}, t\rangle$ where $i_1, ..., i_n, t$ are unary numbers

1. For all $j = 1$ to n: Encode i_j as a binary number, b_j, by straightforwardly starting with $b_j = 0$ and then updating b_j while reading across i_j. *e.g.* $i_j = 1111$ we would have $(b_j = 0,\ 1111) \to (b_j = 1,\ \dot{1}111) \to (b_j = 10,\ 1\dot{1}11) \to (b_j = 11,\ 11\dot{1}1) \to (b_j = 100,\ 111\dot{1})$, where the dot, symbolises the current position of, say the tape head in a TM.

2. Encode t as a binary $\tilde{t}$ by same proceedure as in step 1.

3. For each subset of $\{b_1, ...b_n\}$, I, calculate the sum of I, and check if it equals $\tilde{t}$.

Notice that the input length of M in the above algorithm $\left|\langle\{b_1, ..., b_n\}, \tilde{t}\rangle\right| = O(\log(|\langle\{i_1, ..., i_n\}, t\rangle|))$. Thus M will run in time $2^{O(\log(n)^m)} \leq 2^{c\log(n)^m} <= n^c$ for some c, thus it runs in polynomial time. Clearly converting unary to binary is also polynomial in time. Thus D runs in polynomial time, and accepts exactly *UNARY-SSUM*. $\square$

Proof. We describe an algorithm that accepts *UNARY-SUM* in polynomial time. Let A be any language in P, and let M_A^* be the algorithm from 7.42, that desides A^* in polynomial time. Now given an input $\langle\{i_1, ..., i_n\}, t\rangle$ where $i_1, ..., i_n, t$ are unary, we notice that $\{i_1, ..., i_n\}$ is trivially a language in P, *e.g.* because membership can be checked by trial an error or because it is context free. Now consider the algorithm $D =$" on input $\langle A, t\rangle$ where $A = \{i_1, ..., i_n\}$, and $i_1, ..., i_n, t$ are unary

1. Run M_{A^*} on t, if it accepts, *accept*, if it rejects, *reject.*

Clearly, D accepts $\langle\{i_1, ..., i_n\}, t\rangle$ if and only if $t = i$ $\square$

Let $UNARY\text{-}SSUM$ be the subset sum problem where all numbers are represented in unary numbers. We show that $UNARY\text{-}SSUM$ is in P.

Proof. Consider the CFG, G:

$$
\begin{aligned}
S &\to A|B \\
A &\to 1A1|\#B|\$ \\
B &\to 1B|\#B|\#A|\$
\end{aligned}
$$

By inspection, it is clear that this CFG can produce strings of the form $u_1\#...\#u_m\$u$ where for some $1 \leq i_1 < ... < i_k \leq m$, we have $u_{i_1}...u_{i_k} = u$, *e.g.* 111#11#111111#1\$1111. This is easily seen from the fact that we can produce a 1 to the right of \$ only by producing one to the left of \$ as well, furthermore, one can only stop producing 1s to the right of \$ by putting down a # on the left. However, this language is clearly equivalent to $UNARY\text{-}SSUM$, by the bijective the map $\langle\{u_1, ..., u_m\}, u\rangle \mapsto \langle u_1\#...\#u_m\$u\rangle$ which is clearly both polynomial time comptable and its inverse is also polynomial time computable. Thus we conclude that $UNARY\text{-}SSUM \leq_P L(G)$, but by theorem 7.16 $L(G)$ is in P, and thus by Theorem 7.31 $UNARY\text{-}SSUM$ is in P. $\square$

Exam 2019, Question 3

3.1

$$
i\text{-}RSP = \{\langle G\rangle \mid \text{G is a graph and there exist tree subgraph } T \subset G \text{ such that } V(T) = V(G), \text{ and for any vertex in } T \text{ the degree is 0, 1, or } i\}
$$

3.2

We show that 3-RSP is NP-complete.

Proof. We do this by reducing the NP-complete problem $UHAMPATH$ to 3-RSP. Consider the algorithm, $D =$"On input $\langle G, s, t\rangle$ where G is a graph and s, t are vertices in G

1. Construct the graph G' such that $G \subset G'$ by adding one vertex b_i to G for every vertex $v_i \in G \setminus s, t$ where we identify $V(G) = \{v_1, ..., v_m, s, t\}$, and add one line between each v_i and b_i.

this is clearly polynomial in time. Furthermore, notice that if G has a Hamiltonian path from s to t, then G' has a 3-regular spanning tree, since the path from s to t go through all v_i exactly once, and by adding the lines from v_i to b_i we see that the vertices $\{v_1, ..., v_m, b_1, ..., b_m, s, t\}$ with the lines given by the Hamiltonian path from s to t and the lines from v_i to b_i forms a 3-regular spanning tree, such that v_i has degree 3 for all $i = 1, ..., m$ and $\{s, t, b_1, ..., b_m\}$ are the leaves of degree 1. On the contrary, if G' has a 3-regular spanning tree, we see that $\{b_1, ..., b_m\}$ must be leaves since they have degree 1. Hence $\{v_1, ..., v_m\}$ has degree 3, but since s, t only connects to $\{v_1, ..., v_n\}$ they must themselves be leaves. Notice then that s and t actually must be at the bottom of the tree, since any v_i exept the bottom one can have at most 1 leaf, since they have degree 3. Thus a Hamiltonian path from s to t exist by going from s up the tree all the way to the root, and down the other branch all the way down to t at the bottom. This path goes through every vertex $v_1, .., v_m$ exactly once. Therefore, we see that D maps $UHAMPATH$ to 3RSP and $\overline{UHAMPATH}$ to $\overline{3\text{-}RSP}$. Thus the map $G \mapsto G'$ is a polynomial time computable function, and we conclude that $UHAMPATH \leq_P 3\text{-}RSP$ by Theorem 7.36 that 3-RSP is NP-complete. $\square$