University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
S. Faour

Theoretical Computer Science - Bridging Course
Sample Solution Exercise Sheet 11

Due: Tuesday, 26th of July 2022, 12:00 pm

Exercise 1: Class $\mathcal{P}$

1. Which of the following statements about the class $\mathcal{P}$ are true?

   - $\mathcal{P}$ is the class of all languages that are decidable by deterministic multi-tape Turing machines running in polynomial time.
   - A language L belongs to $\mathcal{P}$ iff there is a constant k and a decider M running in time $O(n^k)$ such that $L = L(M)$.
   - A language L belongs to P iff L is decided by an $O(2^n)$ time DTM .
   - A_{TM} belongs to $\mathcal{P}$.

2. - Show that the following language ($\cong$ decision problem)
     18-DominatingSet := $\{\langle G\rangle \mid G$ has a *dominating set* of size at most 18$\}$ is in the class $\mathcal{P}$.
     Remark: A subset of the nodes of a graph G is a *dominating set* if every other node of G is adjacent to some node in the subset.
   - Is 18-DominatingSet decidable?

Sample Solution

1. - True. Note that any multi-tape TM running in polynomial can be simulated by a single-tape TM running also in polynomial time (it is only quadratically slower).
   - True.
   - False. The implication from left to right of course holds, but the one from right to left does not hold.
   - False. The language A_{TM} does not belong to $\mathcal{P}$ because A_{TM} is an undecidable language and $\mathcal{P}$ contains only decidable languages.

2. - A dominating set of size 18 exists if and only if all nodes are covered by a subset of the nodes $D \subseteq V$ with $|D| = 18$. There are less than n^{18} sets with $|D| = 18$. We iterate through all of the sets (e.g., by using 18 nested for loops iterating over the identifiers of the node set if we number the nodes from 1 to n.). Then for each set we test whether it dominates the whole graph by testing whether every node not in the set has a neighbor in the set. If this is the case for some set we accept. We reject if it's not the case for all sets.
     This test can be done in time $\mathcal{O}(n^2)$ for each other node by scanning the edge set. There are at most $|V \setminus D| \leq n$ nodes outside of D for a D with $|D| = 18$. So to check whether a specific D is a dominating set we only need $\mathcal{O}(n^3)$ time. In total the time complexity is $\mathcal{O}(n^{18} \cdot n^3) = \mathcal{O}(n^{21})$. Therefore 18-DominatingSet $\in \mathcal{P}$.
   - Yes, since 18-DominatingSet $\in \mathcal{P}$ and $\mathcal{P}$ contains only decidable languages.

Exercise 2: Class $\mathcal{NP}$ and $\mathcal{NPC}$

1. Is the following true: A language L is decidable by an $O(\log n)$ time deterministic single tape TM, then L belongs to NP.

2. - Given a set U of n elements ('universe') and a collection $S \subseteq \mathcal{P}(U)$ of subsets of U, a selection $C_1, \ldots, C_k \in S$ of k sets is called a *set cover* of (U, S) of size k if $C_1 \cup \ldots \cup C_k = U$. Show that the problem

     $$\textsc{SetCover} := \{\langle U, S, k\rangle \,|\, U \text{ is a set, } S \subseteq \mathcal{P}(U) \text{ and there is a set cover of } (U, S) \text{ of size } k\}$$

     is NP-complete.
     You may use that

     $$\textsc{DominatingSet} = \{\langle G, k\rangle \mid G \text{ has a dominating set with } k \text{ nodes}\}.$$

     is NP-complete.
   - Why can't we solve DominatingSet in polynomial time the same way we solve 18-DominatingSet?

3. Show $\textsc{DominatingSet} := \{\langle G, k\rangle \,|\, \text{Graph } G \text{ has a dominating set of size at most } k\} \in \mathcal{NPC}$.

   Use that $\textsc{VertexCover} := \{\langle G, k\rangle \mid \text{Graph } G \text{ has a vertex cover of size at most } k\} \in \mathcal{NPC}$.

   *Remark: A **vertex cover** is a subset of nodes of G such that every edge of G is incident to a node in the subset.*

   Hint: Transform a Graph G into a Graph G' such that a vertex cover of G will result in a dominating set G' and vice versa(!). Note that a dominating set is not necessarily a vertex cover ($G = (\{v_1, v_2, v_3, v_4\}, \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_4\}\})$ has the dominating set $\{v_1, v_4\}$ which is not a vertex cover). Also a vertex cover is not necessarily a dominating set (consider isolated notes).

Sample Solution

1. True. Since $O(\log n) \subseteq O(n^k)$ for any constant $k \geq 1$, language L can be solved in polynomial time by a deterministic single tape TM, hence L is in class $\in \mathcal{P}$. Also, $\mathcal{P} \subseteq \mathcal{NP}$, thus L is in $\mathcal{NP}$.

2. - SetCover is in NP: Guess a collection $C_1, \ldots, C_k \in S$ of k sets from S. Go through all elements of U and check if it is in one of the C_i. This takes polynomial time.

     SetCover is NP-hard: We reduce DominatingSet to SetCover. Let $G = (V, E)$ be a graph and k an integer. We define a SetCover instance in the following way: We choose V to be the universe, i.e., $U = V$ and $S := \{\Gamma_G(v) \mid v \in V\}$ where $\Gamma_G(v)$ consists of the vertex v and all vertices adjacent to v in G. This conversion takes polynomial time. Then $\Gamma_G(v_1), \ldots, \Gamma_G(v_k)$ is a set cover of (U, S) iff $v_1, \cdots, v_k$ is a dominating set of G. Hence, $\langle U, S, k\rangle \in$ SetCover iff $\langle G, k\rangle \in$ DominatingSet.
   - To prove DominatingSet in $\mathcal{P}$, we would need to find a constant c, and an associated $O(n^c)$ algorithm, which would decide on änyïnstance (G, k) , whatever k is However, if we use the same brute force algorithm in exercise 1 to solve DominatingSet, once we take the instance (G, k) to be checked, then it will run on $O(n^c)$ for some constant c, but here we are choosing c after we have seen k, for arbitrary k. So as k increases it approaches to $n/2$, then using the same brute force algorithm will yield in an exponential of order $n/2$ time complexity, and this does not prove that DominatingSet in $\mathcal{P}$.

3. **Guess and Check:** we show that DominatingSet $\in \mathcal{NP}$.

   Consider the following verifier for DominatingSet on input $\langle\langle G, k\rangle, D\rangle$, that verifies in polynomial time that G has a dominating set of size at most k, where the idea of the certificate D is the dominating set. Let $G = (V, E)$, where $n := |V|, m := |E|$.
   The verifier first tests if D has at most k different nodes from G with $O(|D| + |D| \cdot n + |D|^2)$ comparisons (similar to the Clique problem in a prev. sheet), then it tests if all nodes are in D or adjacent to a node in D in $O(n(|D| + m))$ comparisons (or you can say $O(nm)$ comparisons, since to do this second test $|D| \leq k \leq n$). If both these tests pass, accept; else reject. Since the certificate has polynomial length in the input size, therefore the total running time is polynomial in the input size. So DominatingSet has a polynomial time verifier. Therefore, DominatingSet is in $\mathcal{NP}$.

   Polynomial reduction of VertexCover **to** DominatingSet**:** we show that DominatingSet is $\mathcal{NP}$-hard.
   We define a function f that can be computed in polynomial time and transforms an instance $\langle G, k\rangle$ of DominatingSet into an instance $f(\langle G, k\rangle) = \langle G', k\rangle$, such that G has a vertex cover of size at most k, iff G' has a vertex cover of size at most k, i.e.,

$$\langle G, k\rangle \in \text{VertexCover} \iff f(\langle G, k\rangle) = \langle G', k\rangle \in \text{DominatingSet}.$$

   For $G = (V, E)$ we construct $G' = (V', E')$ as follows. Initially, we set $V' := V, E' := E$. For each edge $\{u, v\} \in E$ we add an additional node w to V' and add the edges $\{u, w\}, \{v, w\}$ to E' (i.e., G' has a triangle with nodes u, v, w). Furthermore we remove all isolated nodes from V'. The construction of G' can be accomplished, by generating V' and E', in $O((m+n)+nm+m)$ (i.e. $\mathcal{O}(nm)$) comparisons. Indeed, it takes $O((m + n) + nm)$ comparisons to generate V', since we add $\mathcal{O}(m)$ new nodes alongside the old ones and remove at most $\mathcal{O}(n)$ nodes (for each node in V we can check if it is isolated in $O(m)$ comparisons); moreover, we can generate E' in $O(m)$ comparisons as we add at most $\mathcal{O}(m)$ new edges (for each corresponding edge in E', we add 2 new edges). It remains to prove the equivalency stated above.

   $\Longrightarrow$**:** Let (G, k) be such, that G has a vertex cover C of size at most k. Let $D := C \cap V'$ which corresponds to C but without isolated nodes. We have $|D| \leq |C| \leq k$ since D is a subset of C.

   It remains to show that D is a dominating set. We know that for every edge $\{u, v\} \in E$ either $u \in C$ or $v \in C$ (or both). Therefore, every node w that was added to V' during the construction due to an edge $\{u, v\} \in E$ is adjacent to either $u \in D$ or $v \in D$. All other nodes in $v \in V'$ have an incident edge $\{u, v\} \in E \subset E'$ since we removed isolated nodes from V'. Therefore $v \in C$ (and thus $v \in D$), or v is adjacent to a node u in C (hence dominated by one in D).

   $\Longleftarrow$**:** Let the transformed instance $f(\langle G, k\rangle) = \langle G', k'\rangle$ be such that G' has a dominating set D with $|D| \leq k$. We show that we can construct a vertex cover C of size at most k in the original graph G from D. Let $\{u, v\}$ be an arbitrary edge of G. Due to the way G' was constructed, it has a triangle formed by the nodes u, v, w where w is only connected to u and v.

   This means that at least one of the three cases holds: w is dominated by $u \in D$ or by $v \in D$ or it holds that $w \in D$. In the first two cases we add u or v respectively to C (whichever was in D). In the third case we simply add one of the two nodes u or v instead. In all cases $\{u, v\}$ is covered. Since we add at most $|D| \leq k$ nodes to C it holds that $|C| \leq k$.

   In summary: DominatingSet $\in \mathcal{NP}$ and $\mathcal{NP}$-hard, thus DominatingSet $\in \mathcal{NPC}$.

Exercise 3: Regular and Context Free Languages

1. Is the language $L := \{w \in \text{DominatingSet} \mid |w| \leq 2021\}$ regular? Is it decidable?

2. Using the pumping lemma for regular languages, show that the following language $L_1 = \{0^m \mid m \text{ is a prime}\}$ is not regular.

3. Using the Pumping Lemma for CFL, show also that L_1 is not a CFL.

4. Let $L = \{w \in \Sigma^* \mid \text{number of 1s equals number of 2s, and number of 3s equals number of 4s in } w\}$. Here, $\Sigma = \{1, 2, 3, 4\}$. Show that L is not context-free.

Sample Solution

1. We know that all words in the language have length at most 2021, hence L can only contain a finite number of strings, thus L is a finite language. And finite languages are regular since they can be easily described by a DFA or regular expressions, hence decidable.

2. L_1 can be proven, as follows, non regular using the pumping lemma for regular languages. Let p be the pumping length. Let $t > p$ be a prime. Then, let $x = 0^i$, $y = 0^j$ and $z = 0^k$ such that $x + y + z = t$. Based on Pumping Lemma, for all $\ell \geq 0$, $xy^\ell z$ must also be in L_2. However, for $\ell = t + 1$, $xy^\ell z = 0^i 0^{j(t+1)} 0^k = 0^{(i+j+k)} 0^{tj} = 0^{t(j+1)}$, where both t and $j + 1$ are greater than 1. Therefore, $t(j + 1)$ is not a prime, and hence $xy^{t+1}z$ is not in L_2.

3. To prove that L_1 is not a context free language, let p be the pumping length by considering the Pumping Lemma for context free languages. Let $t > p$ be a prime. Since a^t is in L_1, let $uvxyz$ be the decomposition of s^t, regarding the Pumping Lemma. Let $v = 0^i$ and $y = 0^j$ Note that $|vy| > 0$ and $|vxy| \leq p$. Hence, $i + j > 0$ It must hold that $uv^{t+1}xy^{t+1}z \in L_1$. However, $uv^{t+1}xy^{t+1}z = uvxyz \cdot v^t y^t = 0^{t(1+i+j)}$. Since both t and $1 + i + j$ are greater than 1, $t(1 + i + j)$ is not a prime. Hence, $uv^{t+1}xy^{t+1}z$ is not in L_1.

4. We use the pumping lemma for context-free languages to prove that L is not context-free. Assume L is a context free language, hence the property described by the pumping lemma holds on L. Let $p > 0$ be the pumping length. Consider the (bad) string $s = 1^p 3^p 2^p 4^p \in L$, where p is the pumping length. Now we check all possible partitions of s as $uvxyz$ such that the conditions 2 and 3 in the pumping lemma are satisfied (we try to do it in a generic way) and prove that for each such partition condition 1 in the pumping lemma is not satisfied. First, we notice that since the condition 3 of the pumping lemma states that $|vxy| \leq p$, vy can contain at most two different types of symbols, and only symbols that are in consecutive order in s. We consider two scenarios. In the first scenario, vxy contains only one kind of symbol. In this case, by pumping up, the resulting string will have more of that symbol than of all other three symbols, thus the resulting string is not in L. In the second scenario, vxy contains two kinds of symbols. In this case, by pumping up, either the requirement that 1 and 2 have same number is violated, or the requirement that 3 and 4 have same number is violated. Thus, the resulting string is not in L. This means that all three conditions 1,2, and 3 can't be satisfied at the same time for "any" partition of s as $uvxyz$. This is a contradiction to the property of pumping lemma. Hence, L is not context-free.