Homework 2

Classical computation

Task 1. Prove that a 4-tape Turing machine working in time $T(n)$ for inputs of length n can be simulated by an ordinary Turing machine working in time $O(T^4(n))$. Bonus points if you design more efficient simulation that requires less than O(T^4(n)) time.

Solution.

We can use the solution of Task 1.6 from the book.

First, describe briefly a single-tape machine M1 that simulates a two-tape machine M2, like it is in the solution of Task 1.6 from the book. The machine M1 works in cycles. Each cycle takes $O(s)$ steps of the machine M1, where s is the length of the used portion of the tape. Since $s \leq T(n) + 1$, the machine M1 works in time $O(sT(n)) = O(T^2(n))$.

The last step is to consider that our machine works with 4 tapes. We can first simulate 4-tape Turing machine with 2-tape Turing machine with similar idea, so first tape will be used to simulate tape 1 and 2, and second tape will be used to simulate tape 3 and 4. We get 2-tape Turing machine with time complexity $O(T^2(n))$. After that we simulate our two tapes with 1-tape Turing machine and get time complexity $O(T^4(n))$.

One of the ways to have more efficient simulation is to consider Task 1.7. We can simulate 3 tapes with 2 tapes in time $O(T(n)\ log\ T(n))$. If we try to iterate it from 4 tapes to 3 tapes, then from 3 tapes to 2 tapes, and finally 2 tapes t 1 tape as in solution to Task 1.6, we get time estimate better than $O(T^3(n))$.

Task 2. Recall Task 2.7. from the Book that stated the following: Show that any function can be computed by a circuit of depth ≤3 with gates of type NOT, AND, and OR, if we allow AND- and OR-gates with arbitrary fan-in and fan-out.

Suppose now that we are limited to bases consisting of 2 logic gates. Show that that any function can be computed by a circuit:

a) of depth ≤5 with gates of type NOT and AND if we allow AND-gates with arbitrary fan-in and fan-out.

b) of depth ≤5 with gates of type NOT and OR if we allow OR-gates with arbitrary fan-in and fan-out.

Solution.

In the proof of Task 2.7, we used Disjunctive normal form. When we have {AND, OR, NOT} gates, if we construct a circuit using AND gates and OR gates with arbitrary fan-in (the number of inputs), only three layers are needed: one layer for negations, one for conjunctions, and one for the disjunction.

In a) we do not have OR gate, so we replace OR with sequence of AND and NOT gates according to: $x \lor y = \neg(\neg x \land \neg y)$. So each OR gate is replaced with 3 layers: we apply NOT to each variable, then AND to each result, and finally apply NOT to final outcome. So 1 layer of OR gates is replaced with 3 layers, and initial depth 3 increases to depth 5.

In b) Solution is similar. We do not have AND gate, so we replace AND with sequence of OR and NOT gates according to: $x \land y = \neg(\neg x \lor \neg y)$. So each AND gate is replaced with 3 layers: we apply NOT to each variable, then OR to each result, and finally apply NOT to final outcome. So 1 layer of AND gates is replaced with 3 layers, and initial depth 3 increases to depth 5.

Task 3. Consider the Subset Sum Problem. Given a set of non-negative integers, and a value sum, determine if there is a subset of the given set with sum equal to given sum.

Example: {3,4,5,6,8} sum=20. Here answer should be positive, since 3+4+5+8=20

Example: {2,3,7,8} sum=14. Here answer is negative, we cannot make 14 from numbers that we have.

Show that Subset Sum Problem is in NP.

Solution.

To show that the problem belongs to NP it is enough to show that given a solution to the problem, we can check it in polynomial time. For Subset Sum Problem we have set of integers and number sum. Solution to the instance of the problem (e.g., provided by Merlin) can be checked in the following way:

- We check that each integer of the solution belongs to our set of integers;
- We check that summation of these integers equals to sum.

For example, our set is S, sum=50. We are given the solution: 10,20,15,5. We verify that each integer belongs to S and that 10+20+15+5=50.

Task 4. Consider TQBF (True quantified Boolean formula) problem that we discussed in lecture 7. For each formula, verify whether it is true or false:

a) $\forall x \exists y \exists z((x \lor y) \land z)$
b) $\forall x \exists y((x \lor y) \land (\neg x \lor \neg y))$
c) $\forall x(x)$
d) $\forall x \forall y \exists z((x \land z) \lor y)$

Solution.

a) We consider $\forall x$:

- If $x = 0$, then $\exists y \exists z$ $y = 1$ and $z = 1$ that make formula true
- If $x = 1$, then $\exists y \exists z$ $y = 1$ and $z = 1$ that make formula true

Therefore, $\forall x \exists y \exists z((x \vee y) \wedge z)$ is true.

b) We consider $\forall x$:

- If $x = 0$, then to make $x \vee y$ we need $y = 1$, and then $\neg x \vee \neg y$ is true because $\neg x = 1$.
- If $x = 1$, then $x \vee y$ is true. To make $\neg x \vee \neg y$ true, we need $y = 0$.

Therefore, for each x there exists y such that formula is true, so $\forall x \exists y((x \vee y) \wedge (\neg x \vee \neg y))$ is true.

c) $\forall x(x)$ is false, because when we have $x = 0$ the formula is false.

d) We evaluate $\forall x \forall y$. If we have $x = 0$ and $y = 0$, then $(x \wedge z) \vee y$ is false for any value of z, therefore, $\forall x \forall y \exists z((x \wedge z) \vee y)$ is false.