Exercise 1

Answer yes or no to the following questions and justify your answers:

a) Is it possible to implement an universal TM?

b) Does it make sense to construct a Non-deterministic verifier?

c) Let t(n) be a function where $t(n) \geq n$: for a given NTM N, where $L(N) \in NTIME(t(n))$, does there exist a TM M, where L(N)=L(M), such that $L(M) \in TIME(t(n))$?

Solution:

a) Yes, a universal TM receives as input the description of a TM M and a string w and simulates M on w.

b) Not in the same way as we define verifier for deterministic TM (i.e., by receiving an additional string c which we can use to verify that $w \in L(M)$), because the non-deterministic TM could simply guess c so there is not advantage to provide it in the input.

c) Yes, for some languages (e.g. Σ^*), we can have an NTM that has arbitrarily many steps, where as a TM recognizing the same language runs in $\mathcal{O}(1)$.

Exercise 2

Provide the relation between P, NP, EXPTIME, $\bigcup_k$TIME(n^k), TIME $(\log(n))$, TIME $(n!)$.

Solution:

TIME $(\log(n)) \subseteq \bigcup_k$TIME$(n^k) =$ P $\subseteq$ NP $\subseteq$ TIME$(n!) \subseteq$ EXPTIME

Exercise 3

Explain what it means for a NTM to have a running time of $\mathcal{O}(t(n))$.

Solution:

It means that it is a decider (all branches are guaranteed to terminate in q_{accept} or q_{reject}), and there exist positive integers c and n_0 such that for any input string w of length n ($|w| = n$) such that $n > n_0$, the number of steps in any of the branches is lower or equal than $f(n)$, where $f(n) \leq c \cdot t(n)$.

Exercise 4

Given a NTM N and a TM M. For any of the following staments; say if they are possible or not. Justify your answers.

a) $L(N) = L(M)$

b) $L(N) \neq L(M)$

c) $N \in$ P

d) $M \in$ NP

e) $L(N) = L(M) \wedge L(N) \in TIME(\log(n)) \wedge L(M) \in TIME(\log(n))$

f) Let $f : \mathbb{N} \to \mathbb{R}^+$ and $g : \mathbb{N} \to \mathbb{R}^+ \wedge L(M) \in TIME(f(n)) \wedge L(N) \in TIME(g(n)) \wedge g(n) = \mathcal{O}(f(n))$

g) $L(N)$ is NP-complete $\wedge L(N) \in EXPTIME$

h) $L(N) \leq_p L(M)$

i) $L(N) \leq_m L(M)$

Solution:

a) Yes, in fact this is true for any recognizable language.

b) Yes, of course it is possible to have two TMs that recognize different languages.

c) No, a TM does not belong to P. The class P contains languages, not TMs.

d) No, a TM does not belong to NP. The class NP contains languages, not TMs.

e) Yes, e.g., it is easy to construct deterministic/non-deterministic deciders for Σ^*. Moreover, we have not even restricted the running time of the TMs in any way.

f) Yes, we have not restricted the running time of the TMs in any way.

g) Yes, NP-complete $\subseteq$ NP $\subseteq$ EXPTIME, so this is true for all NP-complete problems.

h) Yes, L(N) and L(M) could be anything, so of course there are languages such that $L(N) \leq_p L(M)$

i) Yes, L(N) and L(M) could be anything, so of course there are languages such that $L(N) \leq_m L(M)$

Exercise 5

Prove that the class NP is closed under union, intersection, concatenation and Kleene star. Is the class NP closed also under complement?

Solution:
It is an open problem whether NP is closed under complement or not. The proofs for the remaining four language operations can go as follows. Assume that $L_1, L_2 \in$ NP. This means that there are nondeterministic deciders M_1 and M_2 such that M_1 decides L_1 in nondeterministic time $O(n^k)$ and M_2 deciders L_2 in nondeterministic time $O(n^\ell)$. We want to show that

1. there is a nondeterministic poly-time decider M such that $L(M) = L_1 \cap L_2$, and
2. there is a nondeterministic poly-time decider M such that $L(M) = L_1 \cup L_2$, and
3. there is a nondeterministic poly-time decider M such that $L(M) = L_1 \circ L_2$, and
4. there is a nondeterministic poly-time decider M such that $L(M) = L_1^*$.

Now we provide the four machines M for the different operations. The constructions are the standard ones, the additional part is the complexity analysis of the running time. Note that we can use the power of nondeterministic choices to make the constructions very simple.

1. **Intersection:**
   M = "On input w:
   1. Run M_1 on w. If M_1 rejected then reject.
   2. Else run M_2 on w. If M_2 rejected then reject.
   3. Else accept."

   Clearly, the longest branch in any computation tree on input w of length n is $O(n^{\max\{k,\ell\}})$. So M is a poly-time nondeterministic decider for $L_1 \cap L_2$.

2. **Union:**
   M = "On input w:
   1. Run M_1 on w. If M_1 accepted then accept.
   2. Else run M_2 on w. If M_2 accepted then accept.
   3. Else reject."

   Clearly, the longest branch in any computation tree on input w of length n is $O(n^{\max\{k,\ell\}})$. So M is a poly-time nondeterministic decider for $L_1 \cup L_2$. Note that in our case, we do not have the run M_1 and M_2 in parallel, as it was necessary e.g. in the proof that recognizable languages are closed under union. Another possible construction would be to nondeterministically choose either M_1 or M_2 and simulate only the selected machine.

3. **Concatenation:**
   M = "On input w:
   1. Nondeterministically split w into w_1, w_2 such that $w = w_1 w_2$.
   2. Run M_1 on w_1. If M_1 rejected then reject.
   3. Else run M_2 on w_2. If M_2 rejected then reject.
   4. Else accept."

   Clearly, the longest branch in any computation tree on input w of length n is still $O(n^{\max\{k,\ell\}})$ because step 1. takes only $O(n)$ steps on e.g. a two tape TM. So M is a poly-time nondeterministic decider for $L_1 \circ L_2$.

4. **Kleene star:**
   M = "On input w:
   1. If $w = \epsilon$ then accept.
   2. Nondeterministically select a number m such that $1 \leq m \leq |w|$.
   3. Nondeterministically split w into m pieces such that $w = w_1 w_2 \dots w_m$.
   4. For all i, $1 \leq i \leq m$: run M_1 on w_i. If M_1 rejected then reject.
   5. Else (M_1 accepted all w_i, $1 \leq i \leq m$), accept."

   Observe that steps 1. and 2. take time $O(n)$, because the size of the number m is bounded by n (the length of the input). Step 3. is also doable in polynomial time (e.g. by nondeterministically inserting m separation symbols $\#$ into the input string w). In step 4. the for loop is run at most n times and every run takes at most $O(n^k)$. So the total running time is $O(n^{k+1})$. This means that M is a poly-time nondeterministic decider for L_1^*.

Exercise 6

For each of these statements provide a proof:

a) SAT $\in$ NP.

b) 3SAT $\in$ EXPTIME.

c) 3SAT $\leq p$ 2SAT $\rightarrow$ P=NP

Exercise 7

Prove by construction the following statements:

a) VERTEX-COVER $\in$ NP.

b) HAMPATH $\in$ NP.

c) 2SAT $\in$ NP.

d) 2SAT $\in$ P.

e) RELPRIME $\in$ NP.

Exercise 8

Mark the correct answer by putting a cross in the corresponding field. Justify your answers.

A language L is in P iff L is decidable in polynomial time on a deterministic 4-tape Turing machine.	☐ True	☐ False
$A_{TM} \in$ NP.	☐ True	☐ False
A language L is in NP iff it has a polynomial time verifier running in time $\mathcal{O}(n^3)$.	☐ True	☐ False
A language L is in co-NP iff there is a constant k such that $\overline{L}$ has a verifier running in time $\mathcal{O}(n^k)$.	☐ True	☐ False
If a language L is in co-NP then it is also in EXPTIME.	☐ True	☐ False
If $L \in$ P then $\overline{L} \in$ P.	☐ True	☐ False

Solution:

A language L is in P iff L is decidable in polynomial time on a deterministic 4-tape Turing machine.	☒ True	☐ False
$A_{TM} \in$ NP.	☐ True	☒ False
A language L is in NP iff it has a polynomial time verifier running in time $\mathcal{O}(n^3)$.	☐ True	☒ False
A language L is in co-NP iff there is a constant k such that $\overline{L}$ has a verifier running in time $\mathcal{O}(n^k)$.	☒ True	☐ False
If a language L is in co-NP then it is also in EXPTIME.	☒ True	☐ False
If $L \in$ P then $\overline{L} \in$ P.	☒ True	☐ False

Exercise 9

Assume a 7-tape nondeterministic TM running in time $\mathcal{O}(n^4)$. What will be the running time of the corresponding single-tape deterministic machine recognizing the same language?

Solution:

The running time of the equivalent deterministic Turing Machine will be: $2^{\mathcal{O}(n^4)}$ (Note: here we are interpreting "equivalent" as using the procedure described in Chapter 7, of course there could be other TMs that recognize the same language with better runtime).

Exercise 10

Are the following SAT formulae satisfiable? Answer yes/no and provide an argument.

$(x \vee y \vee \overline{y}) \wedge (\overline{x} \vee y) \wedge (x \vee \overline{y})$ ☐ Satisfiable ☐ Unsatisfiable

$(x \vee y) \wedge (\overline{x} \vee \overline{x}) \wedge \overline{y}$ ☐ Satisfiable ☐ Unsatisfiable

Solution:

$(x \vee y \vee \overline{y}) \wedge (\overline{x} \vee y) \wedge (x \vee \overline{y})$ ☒ Satisfiable ☐ Unsatisfiable

Argument: $x \to 1$ and $y \to 1$ is a satisfiable assignment.

$(x \vee y) \wedge (\overline{x} \vee \overline{x}) \wedge \overline{y}$ ☐ Satisfiable ☒ Unsatisfiable

Argument: The second clause enforces $x \to 0$ and the third clause enforces $y \to 0$, but then the first clause is false.

Exercise 11

1. Define the language SAT.
2. Is it the case that $SAT \in$ NP?
3. Prove your claim in point 2.

Solution:

1. Define the language SAT.
   $SAT = \{< \varphi > | \varphi$ is satisfiable Boolean formula$\}$
2. Is it the case that $SAT \in$ NP?
   Yes
3. Prove your claim in point 2.
   We give a nondeterministic polynomial-time decider for SAT:
   "On input $< \varphi >$:
   1. Nondeterministically select a truth assignment.
   2. Check if φ evaluates to true. If yes, then accept, else reject."
   Both steps 1 and 2 take only polynomial time, so SAT $\in$ NP

Exercise 12

Consider the following formula ϕ in CNF.

$$(x_1 \vee \overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_4)$$

Using the reduction described in the proof of *CNF-SAT* $\leq_P$ *3SAT* construct a formula ϕ' in 3-CNF such that ϕ is satisfiable if and only if ϕ' is satisfiable.

Solution:
The formula ϕ' is

$$(x_1 \vee \overline{x_2} \vee z) \wedge (\overline{z} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_4 \vee \overline{x_1})$$

where z is a new (fresh) variable.