Tutorial 12

Exercise 1 (compulsory)

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 2 (compulsory)

Consider the language $A = \{a^n b^n \mid n \geq 0\}$, which is a language in P. We will now try to show that *VERTEX-COVER* $\leq_P A$. The reduction is

$$f(\langle G, k \rangle) = \begin{cases} aabb & \text{if } G \text{ has a vertex cover of size } k \\ aab & \text{otherwise} \end{cases}$$

Since *VERTEX-COVER* is NP-complete, *VERTEX-COVER* $\leq_P A$ and $A \in$ P, we get that P=NP.

Explain carefully what is the flaw is in this "proof".

Solution:
The problem with this "proof" is that the reduction is not known to be computable in polynomial time. For this to be the case, we would have to prove that it is polynomial-time decidable if G has a vertex cover of size k. This is an open problem but many scientists think that it is indeed not the case (no such poly-time algorithm exists), though there is no proof of this statement.

Exercise 3 (compulsory)

A Boolean formula ϕ is a *tautology* if every truth assignment will cause ϕ to evaluate to true. Consider the problem

"Given a formula ϕ, is it the case that ϕ is *not* a tautology?"

1. Express this problem as a language called *NOTA*.
2. Show that *NOTA* is NP-complete.

Solution:
1. The language is:

   $$\textit{NOTA} = \{\langle \phi \rangle \mid \phi \text{ is a Boolean formula which is not a tautology}\}$$

2. First note that a formula is not tautology exactly when some truth assignment causes it to evaluate to false. We now show that *NOTA* $\in$ NP and that $SAT \leq_P$ *NOTA*.

   - Here is a polynomial-time NTM deciding *NOTA*:

     "On input $\langle \phi \rangle$:

1. Guess a truth assignment.
2. Evaluate ϕ with this truth assignment.
3. If ϕ evaluates to false, then accept, else reject."

Let $n = |\phi|$. This means that ϕ has at most n distinct variables; guessing truth values thus requires at most $O(n)$ steps. Evaluating ϕ can be done by scanning repeatedly ϕ from left to right. This, too, requires only polynomially many steps. Consequently, the whole decider runs in nondeterministic polynomial time.

- We show that $SAT \leq_P$ *NOTA* by giving the following reduction:

  "On input $\langle\phi\rangle$:

  1. Output $\langle\neg\phi\rangle$."

  Clearly, this reduction is computable in polynomial time and ϕ is not a tautology if and only if $\neg\phi$ is satisfiable.

Exercise 4 (compulsory)

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.

Exercise 5 (compulsory)

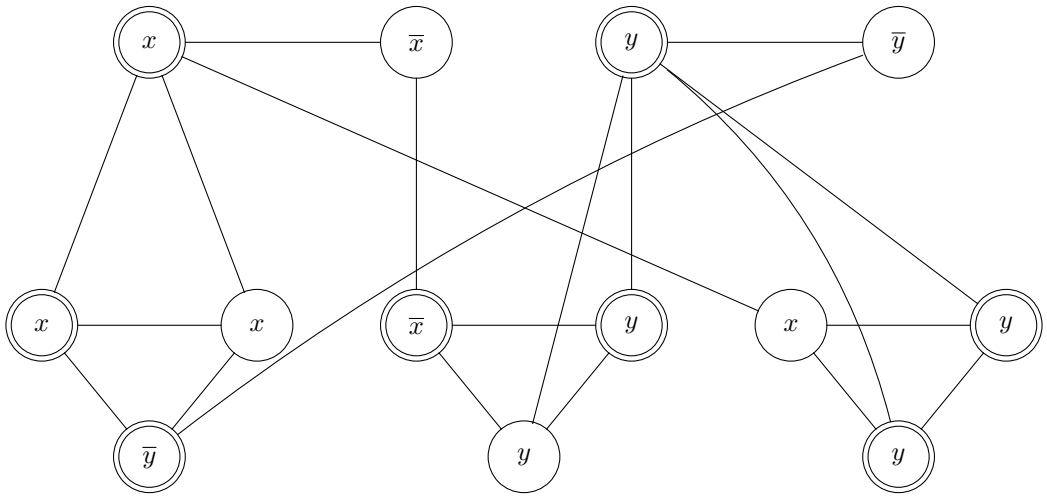
Consider the following formula ϕ in 3-cnf.

$$(x \vee x \vee \overline{y}) \wedge (\overline{x} \vee y \vee y) \wedge (x \vee y \vee y)$$

Using the reduction described in the proof of *3SAT* $\leq_P$ *VERTEX-COVER* construct an undirected graph G and a number k such that G has k-vertex cover if and only if ϕ is satisfiable. List at least one k-vertex cover of the graph and find a corresponding satisfying truth assignment of the formula ϕ.

Solution:
Let k be twice the number of clauses plus the number of variables, so $k = 8$ in our case. The graph G

looks as follows:



The nodes marked by the double circle form an 8-vertex cover (note that there are more 8-vertex covers). The corresponding satisfying assignment is $x \mapsto 1$, $y \mapsto 1$.

Exercise 6 (optional)

Consider the language *SUBSET-SUM*. In its variant discussed in the book, we are given a multiset S of numbers (that means that some of the numbers in S can repeat several times) and we try to select some of the numbers from S that add up to a given number t. We know that this problem is NP-complete. Show that a slight variant of *SUBSET-SUM* where S is given as a set of numbers (which means that numbers cannot repeat) is also NP-complete.