Technische Universität München (I7) **Summer term 2019**
Prof. J. Křetínský / Dr. M.Raskin **6 May 2016**

Computational Complexity – Homework 4

Discussed on 8 May 2016.

Exercise 4.1

Let us denote $\mathbf{DP} = \{L \mid \exists M, N \in \mathbf{NP} : L = M \setminus N\}$ the class of languages that are diferences of two NP languages.

(a) Show that $C = \{\langle G_1, k_1, G_2, k_2\rangle \mid G_1$ has a k_1-clique and G_2 does not have any k_2-clique$\}$ is DP-complete.

(b) Show that $MAX - CLIQUE = \{\langle G, k\rangle \mid$ the largest clique of G is of size exactly $k\}$ is DP-complete.

(c) It is unknown whether $MAX - CLIQUE$ is in **NP**. Show that if **P** = **NP** then $MAX - CLIQUE$ is in **NP** and a largest clique can be found in polynomial time.

Exercise 4.2

(a) Assume that **P**=**NP**. Show that then **EXP**=**NEXP**.

Remark: Assume that L is decided by some TM running in time $T(n)$ with $T(n)$ time-constructible and $T(n) \in \mathcal{O}(2^{n^c})$ for some $c \geq 1$. Show that then

$$L_{\text{pad}} := \{x10^{T(|x|)}1 \mid x \in L\} \in \mathbf{NP}.$$

*(b) Show that also **EXP**=**NEXP** if only *every unary* **NP**-language is also in **P**.

Remark: For $x \in \{0,1\}^*$ let $\langle x\rangle$ be the natural number represented by x assuming lsbf. Given a language L which is decided in time $T(n)$ (with $T(n)$ time-constructable) show that

$$L_{\text{upad}} = \{1^{\langle x10^{|T(n)|}1\rangle} \mid x \in L\} \in \mathbf{NP}$$

with $|T(n)|$ ($\approx \lceil \log T(n)\rceil$) the length of the lsbf representation of $T(n)$.

Exercise 4.3

Is there a language in $\mathbf{DSPACE}(2^{2^{2^{\mathcal{O}(n)}}})$ that is not in **EXPSPACE** and not **NP**-hard (assuming $\mathbf{P} \neq \mathbf{NP}$)?

Exercise 4.4

- Is the following problem in $\mathbf{DTIME}(2^{\mathcal{O}(n)})$?

  A function $f : \{1, \ldots, n\} \times \{1, \ldots, n\} \to \{1, \ldots, n\}$ is given as a table of values. Is there a sequence of n values $x_1, \ldots, x_n \in \{1, \ldots, n\}$ such that $f(x_1, f(x_2, \ldots f(x_{n-1}, x_n) \ldots)) = n$?

- Is the following problem in $\mathbf{DTIME}(2^{\mathcal{O}(n)})$?

  Multiplying an $n \times m$ matrix by an $m \times l$ matrix yields an $n \times l$ matrix and is implemented with time complexity $C \times m \times n \times l$ (the constant C is known).

  Given the number of steps T and the number of matrices k (both written in unary), determine whether there are k matrices that can be multiplied in T steps but not $0.9T$ steps.

Exercise 4.5

Say that A is *linear-time reducible* to B if there is function f computable in time $\mathcal{O}(n)$ such that $x \in A \Leftrightarrow f(x) \in B$.

- Show that there is no P-complete problem w.r.t. linear-time reductions.

Hint: Use the time hierarchy theorem for **DTIME**.

PSPACE-related questions

Note that these topics were not yet discussed during the lectures. The exercises 4c and 5 are included but will not be discussed in class on the 8th May.

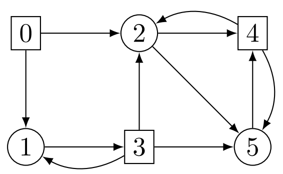
Exercise 4.6

A *two-person game* consists of a directed graph $G = (V_0, V_1, E)$ (called the *game graph*) whose nodes $V := V_0 \cup V_1$ are partitioned into two sets and a *winning condition.* We assume that every node $v \in V$ has a successor. The two players are called for simplicity player 0 and player 1. A *play* of the two is any finite or infinite path $v_1 v_2 \ldots$ in G where v_1 is the starting node. If the play is currently in node v_i and $v_i \in V_0$, then we assume that it is the turn of player 0 to choose v_{i+1} from the successors of v_i; if $v_i \in V_1$, player 1 determines the next move. The winning condition defines when a play is won by player 0. E.g.:

- In a *reachability game* the winning condition is simply defined by a subset $T \subseteq V_0 \cup V_1$ (*targets*) of the nodes of G, and a play is won by player 0 if it visits T within $n-1$ moves (where n is the total number of nodes of G). Hence, player 1 wins a play if he can avoid visiting T for at least $n-1$ moves.
- In a *revisiting game* player 0 wins a play $v_1 v_2 \ldots$ if the first node v_i which is visited a second time belongs to player 0, i.e., $v_i \in V_0$; otherwise player 1 wins the play.

We say that *player i wins node s* if he can choose his moves in such a way that he wins any play starting in s.

Example: Consider the following game graph where nodes of V_0 (V_1) are of circular (rectangular) shape:



In the reachability game with $T = \{5\}$ player 0 can win node 4: if player 1 moves from 4 to 5, player 0 immediately wins; if player 1 moves from 4 to 2, then player 0 can win again by moving from 2 to 5. On the other hand, player 1 can win node 0 by choosing to always play from 0 to 1 and from 3 to 1.

In the revisiting game played on the same game graph, player 0 can win node 2: he moves from 2 to 5 and then on to 4; no matter how player 1 then chooses to move, the play will end in an already visited node which belongs to player 0. Player 1 can e.g. win node 3 by simply moving to node 1.

(a) Consider a reachability game:

Show that one can decide in time polynomial in $\langle G, s, T \rangle$ if player 0 can win node s.

Hint: Starting in T compute the set of nodes from which player 0 can always reach T no matter how player 1 chooses his moves.

(b) Consider a revisiting game and the decision problem: for a given game graph G and node s determine whether player 0 can win s.

Show that this decision problem is in **PSPACE**.

(c) Show that this decision problem is **PSPACE**-complete.

Remarks:

- A game is called *determined* if every node if won by one of the two players.

  Are reachability, resp. revisiting games determined?

- Assume that we change the definition of reachability game by dropping the restriction on the number of moves, i.e., player 0 wins a play if the play eventually reaches a state in T.

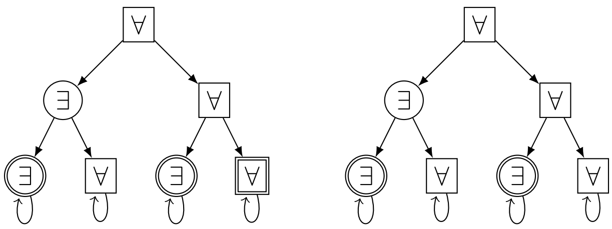
  Does this change the nodes player 0 can win for a given game graph?

Exercise 4.7

An *alternating Turing machine* (ATM) $M = (\Gamma, Q_\forall, Q_\exists, \delta_0, \delta_1)$ is an NDTM $(\Gamma, Q_\forall \cup Q_\exists, \delta_0, \delta_1)$ except that (i) the control states are partitioned into sets $Q_\forall$ and $Q_\exists$ and (ii) the acceptance condition is defined as follows:

> Consider the configuration graph $G(M, x)$. We extend the partition of the control states to the configurations (nodes) of $G_{M;x}$: a configuration is in V_0 if its control state is in $Q_\exists$; otherwise it is in V_1. We then can consider the reachability game played on $G(M, x)$ by the players 0 and 1 where the target set is the set of accepting configurations. M accepts x iff player 0 wins the initial configuration in this reachability game. (For the sake of completeness, assume that every halting/accepting configuration is its unique successor.)

Example: Consider the following configuration graphs where accepting configurations have a second circle/rectangle drawn around them. In the left graph the corresponding ATM accepts the input while it rejects the input in the right example:



A language is decided by an ATM M if M accepts every $x \in L$ and rejects any $x \notin L$. The time and space required by an ATM is the time and space required by the underlying NDTM.

The class **AP** consists of all languages L which are decided by an ATM M running in time $T(n) \in \mathcal{O}(n^k)$ for some $k \geq 1$.

(a) An *existential* (*universal*) ATM is an ATM with $Q_\forall = \emptyset$ ($Q_\exists = \emptyset$).

Show that any language $L \in$ **AP** which is decided by an existential (universal) ATM is in **NP** (co**NP**).

(b) Define co**AP** as usual: $L \in$ co**AP** iff $\overline{L} \in$ **AP**.

Show or disprove that **AP** = co**AP**.

(c) Show that QBF is in **AP**.

(d) Show that any $L \in$ **AP** is in **PSPACE**.

Remark: Adapt the recursive decision procedure for QBF $\in$ **PSPACE** you have seen in the lecture.

Remark: Similarly as **AP**=**PSPACE**, one can show **APSPACE**=**EXPTIME**.