Introduction to Computation and Complexity

Final Exam

December 2019

Intitulé	EPITA_2020_ING3_S9
Majeure	RDI
Code	COMP
Teacher	Adrien Pommellet
Duration	2 hours
Documents	All non-electronic documents are allowed.

This exam is graded on a scale from 0 to 15 points. Read each one of the four exercises carefully. When asked to prove something, make sure that your answer is detailed and rigorous.

Exercise 1 (3 points)

Consider the language:

$$\mathcal{N} = \{\langle M \rangle \# w \mid M(w) \text{ writes a non-blank symbol on its second tape.}\}$$

where $\langle M \rangle$ stands for the code of the Turing Machine M.

Question 1.

Find a reduction f from the halting problem $\mathcal{H}$ to $\mathcal{N}$.

Hint. Given a Turing machine M and an input x, design a Turing machine N and an input y such that $M(x)$ halts if and only if $N(y)$ writes a non-blank symbol on its tape.

Answer. Consider $f : (\langle M \rangle \# w) \to (\langle M' \rangle \# w)$ where M' is a TM with two tapes that, on an input x, performs the following operations:

1. Simulate $M(x)$ on its first tape.

2. If $M(x)$ accepts, write a non-blank symbol on its second tape then accept.
3. If $M(x)$ doesn't accept, reject.

The function f is obviously a reduction from $\mathcal{H}$ to $\mathcal{N}$.

Question 2.

Is $\mathcal{N}$ decidable? Why?

Answer. $\mathcal{H} \leq_T \mathcal{N}$ and $\mathcal{H}$ is undecidable, thus $\mathcal{N}$ is undecidable as well.

Exercise 2 (4 points)

The radix order $<_{rad}$ is a binary relation on the set of input words Σ^* such that $x \leq_{rad} y$ if and only if $|x| < |y|$ or $|x| = |y|$ and $x \leq_{lex} y$, where $\leq_{lex}$ stands for the lexicographic order.

We want to prove that a language L is decidable if and only if there exists a Turing machine enumerating all the words of L in radix order.

Question 1.

Let M be a Turing machine accepting a decidable language L. Design a Turing machine N enumerating L in radix order.

Hint. N can use multiple tapes.

Answer. Consider a TM N with three tapes that performs the following operations, starting from $i = 0$:

1. Write the i-th word x_i of Σ^* (in the radix order) on its first tape.
2. Simulate $M(x_i)$ on its second tape.
3. If $M(x_i)$ accepts, add $\#x_i$ to its third tape. If it refuses, do nothing.
4. Increment i, scrub the second tape until it is empty, and loop back to the first step.

N enumerates L in the radix order on its third tape.

Question 2.

Let M be a Turing machine enumerating a sequence $(w_i)_{i\geq 0}$ sorted according to the radix order. Prove that the language $L = \{w_i \mid i \geq 0\}$ is decidable.

Hint. Design a Turing machine N with multiple tapes accepting L. Obviously, N should rely on M.

Answer. Consider N with two tapes that, on the input x, performs the following operations:

1. Simulate M on its second tape until it writes a word w. If it can't because the enumeration is over, reject.
2. If $x = w$, accept.
3. If $x < w$, reject.
4. If $x > w$, loop back to the first step and resume the simulation.

If $x \in L$, then x is enumerated by M and N accepts x. Moreover, since M enumerates L in the radix order, x can't be enumerated after a word $w > x$. Thus, $x \notin L$ if M outputs $w > x$ before x. N therefore recognizes L.

Exercise 3 (4 points)

Let `DOUBLE-SAT` $= \{\varphi \mid \varphi$ is a Boolean formula satisfiable at least twice.$\}$.

Question 1.

Prove that `DOUBLE-SAT` is in `NP`.

Hint. Find a non-deterministic polynomial algorithm, or use the certification theorem.

Answer. Consider a NTM N that, on an input φ, performs the following operations:

1. Non-deterministically guess two valuations x and y of φ.
2. Check that $x \neq y$. If it's not the case, reject.
3. Check that $\varphi(x)$ and $\varphi(y)$ are true. If it's not the case, reject. Otherwise, accept.

N recognizes `DOUBLE-SAT` non-deterministically in polynomial time.

Question 2.

Find a polynomial reduction f from `SAT` to `DOUBLE-SAT`.

Hint. Given a formula φ, design in polynomial time a formula ψ such that φ admits at least one solution if and only ψ admits at least two solutions.

Answer. Consider $f : \varphi \to \varphi'$ such that, if φ has n variables $x_1, \ldots, x_n$, then φ' has $n+1$ variables $x_0, \ldots, x_n$ and $\varphi' = \varphi \vee \varphi[x_1 \leftarrow x_0]$.

If $\varphi(y_1, \ldots, y_n) = 1$, then $\varphi'(\neg y_1, y_1, \ldots, y_n) = 1$ and $\varphi'(y_1, y_1, \ldots, y_n) = 1$. And if $\varphi'(y_0, \ldots, y_n) = 1$, then $\varphi(y_1, \ldots, y_n) = 1$ or $\varphi(y_0, y_2, , \ldots, y_n) = 1$. Thus, f is indeed a polynomial reduction from `SAT` to `DOUBLE-SAT`.

Question 3.

Is DOUBLE-SAT NP-complete? Why?

Answer. SAT $\leq_T^P$ DOUBLE-SAT and SAT is NP-hard, thus DOUBLE-SAT as well. Moreover, DOUBLE-SAT is in NP. Therefore, DOUBLE-SAT is NP-complete.

Exercise 4 (4 points)

We want to prove that P is closed under Kleene star. To this end, we consider a language $L \subseteq \Sigma^*$ recognized by an algorithm A running in polynomial time.

There is no need to write proofs featuring Turing machines in this exercise.

Question 1.

Let $x = x_1 \dots x_n$ be a non-empty word in Σ^*. $\forall i, j \in \{1, \dots, n\}$, we define:

$$l_{i,j} = \begin{cases} 1 \text{ if } i \leq j \text{ and } x_i \dots x_j \in L \\ 0 \text{ otherwise.} \end{cases}$$

Design an algorithm computing the matrix $(l_{i,j})_{i,j \in \{1,\dots,n\}}$ in polynomial time.

Answer. $\forall i, j \in \{1, \dots, n\}$, $i \leq j$, we run the algorithm A on the word $x_i \dots x_j$. If A accepts, we fill the cell $l_{i,j}$ with a 1, and if it refuses, with a 0 instead. The algorithm A runs in polynomial time and is called on $\mathcal{O}(n^2)$ words of length $\leq n$; thus, the whole process runs in polynomial time w.r.t. n.

Question 2.

Let G be a directed graph with $n+1$ vertices $X_1, \dots, X_{n+1}$ such that $X_i \to X_j$ if and only if $l_{i,j-1} = 1$. Prove that $x \in L^*$ if and only if there exists a path from X_1 to X_{n+1} in G.

Hint. Prove both directions of the equivalence.

Answer. If $x \in L^*$, then there exists an increasing sequence $i_1, \dots, i_k$ of indices such that $i_1 = 1$, $i_k = n+1$, and $\forall j \in \{1, \dots, k-1\}$, $x_{i_j} \dots x_{i_{j+1}-1} \in L$, hence, $l_{i_j, i_{j+1}-1} = 1$. Thus, by definition of G, there is a path $X_1 \to X_{i_2} \to \dots \to X_{i_{k-1}} \to X_{n+1}$ in G.

If there is a path $X_1 \to X_{i_2} \to \dots \to X_{i_{k-1}} \to X_{n+1}$ in G, consider the sequence $i_1, \dots, i_k$ of indices where $i_1 = 1$ and $i_k = n+1$. By definition of G, $\forall j \in \{1, \dots, k-1\}$, $l_{i_j, i_{j+1}-1} = 1$, thus $x_{i_j} \dots x_{i_{j+1}-1} \in L$ and $x \in L^*$.

Question 3.

Prove that P is closed under Kleene star.

Hint. Design an algorithm B recognizing L^* in polynomial time.

Answer. Given an input word x, compute the matrix l, then the graph G, and find a path from X_1 to X_{n+1} in G using a depth-first search. Such a path exists if and only if $x \in L^*$.

These three operations can be performed in polynomial time, hence the whole algorithm as well. L^* can therefore be recognized in polynomial time, thus $L \in \mathsf{P}$.