Examen Partiel pour Advanced Complexity (M1)

Les documents (notes, polycopiés, ..) et calculatrices (téléphone, tablette, ..) ne sont pas autorisés.

Date : 23 oct. 2019 à 10h30 / Durée : 2 heures

General instructions : this exam is not long and most questions admit short answers. Points are awarded for clarity, precision, and completeness, not for guessing or for writing long and confusing arguments that miss the main point. When giving reductions or algorithms, be very precise and do not forget corner cases. When arguing the correctness of an algorithm, make a precise claim about your construction before proving said claim. In the proof, you may abstain from proving trivial observations only if you do not miss the harder parts of the proof.

Exercise 1 : Functions computable in logspace

Let A be some finite alphabet. With a partial function $f : A^* \to A^*$, defined on $dom\, f$, we associate the following language

$$D_f = \{\langle x, i, a\rangle \in A^* \times \mathbb{N} \times A \mid x \in dom\, f \text{ and } 0 < i \leq |f(x)| \text{ and } a \text{ is the } i\text{-th letter in } f(x)\}.$$

For representation purposes, we assume that the number i is written in binary and that D_f uses a new symbol, say $\# \notin A$, to separate the components of a triple. Thus $D_f \subseteq A'^*$ for $A' \stackrel{\text{def}}{=} A \cup \{0, 1, \#\}$.

Recall that f is *logspace computable* iff there exists a deterministic Turing machine M_f that, when started with some $x \in A^*$ on its (bidirectional, read-only) input tape, terminates, produces $f(x)$ on its (unidirectional, write-only) output tape, and only uses $O(\log |x|)$ cells on its work tapes. When $f(x)$ is undefined, M_f terminates by reaching a rejecting/error/.. state.

1. Prove that a *total* function f is logspace computable if, and only if, $D_f \in \mathsf{L}$ and there exists a polynomial p such that $|f(x)| \leq p(|x|)$ for all x in $dom\, f$.

Solution:
($\implies$) As seen in class, if a deterministic TM is logspace it can only, when started on some input x, visit a polynomially-bounded number of configurations before halting, hence $|f(x)|$ is polynomially-bounded. Furthermore, deciding whether $\langle x, i, a\rangle \in D_f$ can be done by simulating M_f on x and, instead of writing $f(x)$ on the output tape, we maintain a counter c that keeps track of how many characters would have been output. When c coincides with i, we can compare a with the character that would be output by M_f. If the simulation terminates before c equals i, we deduce that $|f(x)| < i$ (or $x \notin dom\, f$) and we can reject $\langle x, i, a\rangle$.
In fine, this uses the same work space as M_f plus logspace overhead for c, hence is in L.
($\impliedby$) : Given a logspace Turing machine M deciding D_f, we can produce a machine M' computing f in logspace : M' computes $p(|x|)$ and then loops over all $i = 1, \ldots, p(|x|)$ and all $a \in A$, simulating M on $\langle x, i, a\rangle$. If the triple is accepted, M' outputs a and goes to next i. When there is no $a \in A$ with $\langle x, i, a\rangle \in D_f$, we know that i is larger than $|f(x)|$ hence we have finished outputting $f(x)$ and we can accept. This uses logspace extra workspace (for computing $p(|x|)$, for looping over i and a) on top of what M uses, hence is in logspace.

2. In question 1, can we remove the assumption on p, i.e., do we also have "$D_f \in \mathsf{L}$ iff f is logspace computable" ?

Solution:
The answer is no since $D_f \in \mathsf{L}$ does not imply that $f(x)$ has a polynomially-bounded length. For example consider $A = \{a\}$ and $f(x) = a^{2^{|x|}}$ for all $x \in A^*$. Having exponential length, $f(x)$ is not

computable in polynomial time, nor *a fortiori* in logspace. However D_f is in L : one has

$$\langle x, i, a\rangle \in D_f \text{ iff } 1 \leq i \leq 2^{|x|}$$

and, given some $\langle x, i, a\rangle$, testing whether $i \leq 2^{|x|}$ reduces to comparing the length of (the encoding of) i and the length of x : this can certainly be done in logspace.

3. In question 1, can we remove the assumption that f is a total function ?

Solution:
No. Take any $P \subseteq A^*$ and define f by $f(x) = \epsilon$ if $x \in P$ and $f(x)$ undefined otherwise. Then D_f is empty and $|f(x)| = 0$ for all $x \in \mathit{dom}\, f = P$, hence D_f is in L and $|f(x)|$ is polynomially-bounded. However f is not computable when P is not decidable.

Exercise 2 : Closure via operations

We consider the following decision problem :

Problem : BinOpGen

Input : A finite set X, a binary operation $* : X \times X \to X$, a subset $S \subseteq X$ and a target $t \in X$.

Question : Does $t \in \langle S\rangle_*$?

Here $\langle S\rangle_*$ denotes the closure of S, a set of generating elements, by $*$. Formally, we define an increasing sequence $S_{0,*} \subseteq S_{1,*} \subseteq S_{2,*} \subseteq \cdots$ of subsets of X with

$$S_{0,*} \stackrel{\text{def}}{=} S\,, \qquad \forall i \in \mathbb{N} :\ S_{i+1,*} \stackrel{\text{def}}{=} S_{i,*} \cup \{x * y \mid x, y \in S_{i,*}\}\,, \tag{†}$$

and let $\langle S\rangle_* \stackrel{\text{def}}{=} \bigcup_{i\in\mathbb{N}} S_{i,*}$.

For representation purposes, we may assume that X is a finite set of the form $\{1, \ldots, n\}$, so that $*$ can be represented as an $n \times n$ matrix with values in $\{1, \ldots, n\}$.

A related problem is 2BinOpGen, where we consider the closure of S via *two* binary operations on X, say $*_1$ and $*_2$. Formally, we define $\langle S\rangle_{*_1,*_2}$ as $\bigcup_{i\in\mathbb{N}} S_{i,*_1,*_2}$, replacing (†) with

$$S_{i+1,*_1,*_2} = S_{i,*_1,*_2} \cup \{x *_1 y \mid x, y \in S_{i,*_1,*_2}\} \cup \{x *_2 y \mid x, y \in S_{i,*_1,*_2}\} \tag{‡}$$

and again asking whether $t \in \langle S\rangle_{*_1,*_2}$.

4. Show that BinOpGen and 2BinOpGen are inter-reducible, hence "have the same complexity".

Solution:
The reduction BinOpGen $\leq$ 2BinOpGen is easy : we duplicate $*$ and leave X, S, t unchanged.
For 2BinOpGen $\leq$ BinOpGen, a possible solution is to define

$$X, *_1, *_2, S, t \mapsto X', *', S', t'$$

with $X' \stackrel{\text{def}}{=} X \times \{1, 2\}$ and $*', S', t'$ given by

$$(x, k) *' (y, l) \stackrel{\text{def}}{=} (x *_k y, l)\,, \qquad S' \stackrel{\text{def}}{=} S \times \{1, 2\}\,, \qquad t' \stackrel{\text{def}}{=} (t, 1)\,.$$

We claim that $S'_{i,*} = S_{i,*_1,*_2} \times \{1, 2\}$ for all $i \in \mathbb{N}$ and prove this by induction on i (easy proof omitted). Finally, $t \in \langle S\rangle_{*_1,*_2}$ iff $(t, 1) \in \langle S'\rangle_*$, so the reduction is correct. That it is logspace is clear since outputting $*'$ just needs two nested loops on X, and X', S' are even easier to produce.

TernOpGen is a further variant where we are given a *ternary* operation, denoted ϕ, over X, and where one asks, given X, ϕ, S, t, whether $t \in \langle S\rangle_\phi$. Here (†) is replaced by $S_{i+1,\phi} \stackrel{\text{def}}{=} S_{i,\phi} \cup \{\phi(x, y, z) \mid x, y, z \in S_{i,\phi}\}$.

5. Show that BinOpGen and TernOpGen are inter-reducible.

Solution:

The reduction witnessing $\mathsf{BinOpGen} \leq \mathsf{TernOpGen}$ can be as simple as $X, *, S, t \mapsto X, \phi, S, t$ with $\phi(x,y,z) \stackrel{\text{def}}{=} x * y$. We then have $x \in S_{i,*}$ iff $x \in S_{i,\phi}$ for all $x \in X$ and $i \in \mathbb{N}$, hence the reduction is correct.

For the other reduction, witnessing $\mathsf{TernOpGen} \leq \mathsf{BinOpGen}$, one possible solution is to define

$$X, \phi, S, t \mapsto X', *, S, t$$

with $X' = X \cup X^2$. In other words, $*$ operates on elements from X, but also on pairs from X^2 (assuming X and X^2 are disjoint). For all $x, y, z, u \in X$, we let

$$x * y \stackrel{\text{def}}{=} (x,y) \ , \qquad x * (y,z) \stackrel{\text{def}}{=} \phi(x,y,z) \ , \qquad (x,y) * z \stackrel{\text{def}}{=} \phi(x,y,z) \ , \qquad (x,y) * (z,u) \stackrel{\text{def}}{=} (x,y) \ .$$

We claim that $S_{i,\phi} \subseteq S_{2i,*}$ for all $i \in \mathbb{N}$, and also that $S_{i,*} \subseteq S_{i,\phi} \cup (S_{i,\phi} \times S_{i,\phi})$ for all $i \in \mathbb{N}$. Both claims are easily proven by induction on i. They entail $t \in \langle S \rangle_*$ iff $t \in \langle S \rangle_\phi$, hence the reduction is correct. It is clearly logspace.

Exercise 3 : Complexity of some closure operations

We let $\mathsf{BinOpGen}_{\text{assoc}}$ be the restriction of $\mathsf{BinOpGen}$ to the case where $*$ is associative, i.e., satisfies $x * (y * z) = (x * y) * z$ for all $x, y, z \in X$.

6. Show that $GAP \leq \mathsf{BinOpGen}_{\text{assoc}}$, where GAP is the Graph Accessibility Problem seen in class.

Solution:
We consider a reduction

$$G = (N, E), s, t \mapsto X, *, S, (s,t)$$

where $X \stackrel{\text{def}}{=} N^2 \cup \{\bot\}$ and with $*$ given by, for any $x \in X$ and any $n_1, n_2, n_3, n_4 \in N$:

$$(n_1, n_2) * (n_3, n_4) = \begin{cases} (n_1, n_4) & \text{if } n_2 = n_3, \\ \bot & \text{otherwise,} \end{cases} \qquad\qquad x * \bot = \bot * x = \bot \ .$$

We observe that, as required, this indeed defines a transitive $*$ (where $\bot$ is absorbing). We further let $S \stackrel{\text{def}}{=} E \cup Id_N$. We now claim that, for all $n, n' \in N$ and $i \in \mathbb{N}$, $(n, n') \in S_i$ iff G has a path $n \xrightarrow{*} n'$ of length $\leq 2^i$. This is easily proven by induction on i. As a consequence, $(s,t) \in \langle S \rangle_*$ iff G has a path $s \xrightarrow{*} t$, hence the reduction is correct.

The reduction is logspace since producing $*$ only requires four nested loops on N, while X and S are even easier to produce.

7. Show that $\mathsf{BinOpGen}_{\text{assoc}}$ is NL-complete.

Solution:
$\mathsf{BinOpGen}_{\text{assoc}}$ is NL-hard as shown in the previous question, and there only remains to show that the problem can be solved in NL. For this, and thanks to associativity, we observe that, given an instance $\langle X, *, S, t \rangle$, it is enough to guess a sequence $n_1, \ldots, n_k$ of elements of S and check that $t = n_1 * n_2 \cdots * n_k$.

Let us now prove that when such a sequence exists, the shortest one has length $k \leq |X|$: given a sequence $n_1, \ldots, n_k$ generating t, we write m_ℓ for the partial product $n_1 * n_2 * \cdots * n_\ell$. If $k > |X|$ then two partial products coincide, say $m_i = m_j$ for some $1 \leq i < j \leq k$. Then $t = (n_1 * \cdots * n_i) * (n_{j+1} * n_{j+2} * \cdots * n_k)$ and there is a shorter way of generating t.

It is now easy to solve $\mathsf{BinOpGen}_{\text{assoc}}$ in NL since we can guess a sequence of length at most $|X|$ and check that its product equals t. We note that we only store in memory the current partial product (and a counter $i = 1, \ldots, k$) while we guess the sequence. We also need to check the associativity of $*$ before accepting an instance $\langle X, *, S, t \rangle$ but this only uses three nested loops on X.

We recall that $\mathsf{MonotoneCircuitValue}$ is defined as follows.

Problem : $\mathsf{MonotoneCircuitValue}$

Input : A circuit C and one of its nodes n_f.

Question : Does $v_C(n_f) = \mathit{true}$?

A circuit is defined as an acyclic directed graph $C = (N, E)$ with two kind of nodes : conjunctive and disjunctive (we write $N = N_\wedge \cup N_\vee$). The boolean value of a node $n \in N$, written $v_C(n)$, is defined by

$$v_C(n) \stackrel{\text{def}}{=} \begin{cases} \bigwedge\{v_C(m) \mid (m,n) \in E\} & \text{if } n \in N_\wedge, \\ \bigvee\{v_C(m) \mid (m,n) \in E\} & \text{if } n \in N_\vee. \end{cases}$$

Since the circuit is acyclic, the above definition is well-founded.
Observe that, for nodes with no inputs, the definition relies on $\bigvee \emptyset = \mathit{false}$ and $\bigwedge \emptyset = \mathit{true}$. We further recall that it is easy to reduce $\mathsf{MonotoneCircuitValue}$ to a restricted version, called $\mathsf{MonotoneCircuitValue}_{\mathrm{deg2}}$, where we only allow circuits where every node has only 2 (or 0) inputs.

8. Give a reduction witnessing $\mathsf{MonotoneCircuitValue}_{\mathrm{deg2}} \leq L$ where L is either $\mathsf{BinOpGen}$, $\mathsf{TernOpGen}$ and $\mathsf{2BinOpGen}$.

   Indication : Choose the problem that makes the reduction cleanest and easiest to prove correct. (There is no loss of generality since, as per Exercise 2, the three versions are equivalent.)

   Solution:
   It will be useful to define inductively a height $h(n) \in \mathbb{N}$ for each node in N via $h(n) = \max\{1 + h(n') \mid (n', n) \in E\}$ (which ensures $h(n) = 0$ if n is a leaf node with no inputs).
   We define a reduction witnessing $\mathsf{MonotoneCircuitValue}_{\mathrm{deg2}} \leq \mathsf{BinOpGen}$

   $$N_\wedge, N_\vee, E, n_f \mapsto X, *, S, t$$

   with $X \stackrel{\text{def}}{=} (N \cup E) \times \{\top, ?\}$. Here X has two copies of each node n and of each edge $e = (n', n'')$ of C. We define $*$ via

   $$(n, \top) * ((n', n''), ?) \stackrel{\text{def}}{=} ((n', n''), \top) \text{ if } n = n', \tag{R1}$$
   $$((n', n''), \top) * (n, ?) \stackrel{\text{def}}{=} (n, \top) \text{ if } n'' = n \text{ and } n \in N_\vee, \tag{R2}$$
   $$((n_1, n_2), \top) * ((n_3, n_4), \top) \stackrel{\text{def}}{=} (n_2, \top) \text{ if } n_2 = n_4 \in N_\wedge \text{ and } n_1 \neq n_3, \tag{R3}$$
   $$x * y \stackrel{\text{def}}{=} x \text{ if none of the above applies.} \tag{R4}$$

   The above definition read informally as
   R1 : if we know that $v_C(n) = \top$ and e is an edge leaving from n then we know that e carries $\top$.
   R2 : if we know that edge e carries $\top$ and is input to a disjunctive node n, then $v_C(n) = \top$.
   R3 : if we know that edges e_1 and e_2 carry $\top$ and they are the input of a conjunctive node n then $v_C(n) = \top$.
   We further set $S \stackrel{\text{def}}{=} (N \cup E) \times \{?\} \cup \{(n, \top) \mid n \in N_\wedge \text{ and } h(n) = 0\}$ and claim that, for all $i \in \mathbb{N}$:

   $$(n, \top) \in S_{i,*} \text{ iff } v_C(n) = \mathit{true} \text{ and } i \geq 2 \cdot h(n)\,,$$
   $$((n, n'), \top) \in S_{i,*} \text{ iff } v_C(n) = \mathit{true} \text{ and } i \geq 2 \cdot h(n) + 1\,.$$

   This is easily proven by induction on i (proof omitted). Finally, we set $t \stackrel{\text{def}}{=} (n_f, \top)$ and have $t \in \langle S \rangle_*$ iff $v_C(n_f) = \mathit{true}$ and the reduction is correct. It is easily seen to be in logspace.

9. Show that $\mathsf{BinOpGen}$ is PTIME-complete.

   Solution:
   After the previous question, there only remains to show that $\mathsf{BinOpGen}$ is decidable in PTIME. For this we note that each $S_{i,*}$ has size at most $|X|$ and can be computed in quadratic time from $S_{i-1,*}$ and $*$. Finally the fixpoint $\langle S \rangle_* = \bigcup_{i \in \mathbb{N}} S_{i,*}$ is reached after polynomially-many iterations since $\langle S \rangle_* = S_{\ell,*}$ for $\ell = |X| - 1$.