Examen Partiel pour Advanced Complexity (M1)

To be returned to phs@lsv.fr **before Nov. 18th at noon (12AM)**

Concerning a homework test, we expect clear and precise answers. When giving reductions or algorithms, explain the algorithm at a high level of abstraction but do not forget corner cases or use vague notation. 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.

Problem : Arithmetical circuits & formulae

In this problem we consider circuits and formulae representing sets of natural numbers, i.e., subsets of $\mathbb{N}$. Such a circuit is a directed acyclic graph where vertices (called *gates*) carry operations or input values. Figure 1 displays an example : $C_1 = (G_{C_1}, E_{C_1}, l_{C_1})$ is a circuit with 10 gates (here $G_{C_1} = \{g_1, \ldots, g_{10}\}$), with a set $E_{C_1} \subseteq G_{C_1} \times G_{C_1}$ of 14 edges, and where the gates carry labels that are either a natural number or an operation, as indicated by the labeling $l_{C_1} : G_{C_1} \to \mathbb{N} \cup \mathit{Ops}$.

Regarding acyclicity, we require that the gates of C are listed in some order $G_C = g_1, \ldots, g_\ell$ and that each edge (g_i, g_j) in E satisfies $i < j$: therefore it is easy to check that a given input string is indeed an arithmetical circuit.

We write $\mathcal{C}(o_1, \ldots, o_k)$ for the class of number circuits that only carry operations among $\{o_1, \ldots, o_k\} \subseteq \mathit{Ops}$: in our example, $C_1 \in \mathcal{C}(\cup, +, \times)$.

Arithmetical formulae are a special case of arithmetical circuits. Let us write $i_C(g)$ and $o_C(g)$ for the set of input (respectively, output) gates of g. Then an arithmetical circuit is a formula iff $|o_C(g)| \leq 1$ for all $g \in G_C$. Hence C_1 in our example is not a formula since, e.g., $o_C(g_6) = \{g_8, g_9\}$. We write $\mathcal{F}(o_1, \ldots, o_k)$ for the subclass of $\mathcal{C}(o_1, \ldots, o_k)$ restricted to formulae.

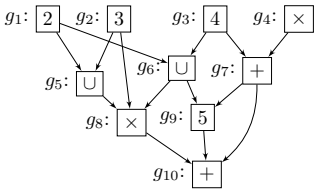


FIGURE 1 – C_1, a number circuit in $\mathcal{C}(\cup, +, \times)$.

In an arithmetical circuit C, every gate evaluates to a subset of $\mathbb{N}$. The evaluation function, $e_C : G_C \to \mathcal{P}(\mathbb{N})$, is defined as expected : gates labeled with a natural evaluate to the corresponding singleton, and gates labeled with an operation o collect the evaluations of their input gates and combine them with o. Formally

$$e_C(g) \stackrel{\text{def}}{=} \begin{cases} \{l_C(g)\} & \text{if } l_C(g) \in \mathbb{N}, \\ o\big(e_C(g_1), \ldots, e_C(g_k)\big) & \text{if } l_C(g) = o \in \mathit{Ops} \text{ and } i_C(g) = \{g_1, \ldots, g_k\}. \end{cases} \tag{D$_1$}$$

We shall only use operations $o \in \mathit{Ops}$ that are commutative and associative, so in the above definition the input gates of g can be listed in any order. In particular the "$+$" and "$\times$" operations denote the usual arithmetic operations *lifted to sets*. Continuing our example one has $e_{C_1}(g_5) = \{2, 3\}$, $e_{C_1}(g_6) = \{2, 4\}$, and

$$e_{C_1}(g_8) = e_{C_1}(g_2) \times e_{C_1}(g_5) \times e_{C_1}(g_6) = \{3\} \times \{2, 3\} \times \{2, 4\} = \{12, 18, 24, 36\}.$$

We also note that the recursive definition in (D$_1$) is well-founded since a circuit is acyclic. In case an op-labeled gate has no input gates, one uses $o() \stackrel{\text{def}}{=} \emptyset$. And one uses $+(E) = \times(E) = E$ when a gate labelled with $+$ or $\times$ has only one input. Thus we also have $e_{c_1}(g_4) = \emptyset$, entailing $e_{c_1}(g_7) = \{4\} + \emptyset = \emptyset$ and $e_{c_1}(g_{10}) = \emptyset$. Note that $e_{c_1}(g_9) = \{5\}$.

For all these circuits we are interested in the *reachability problem* : "given a circuit C, a selected gate $g_s \in G_C$ and a target $t \in \mathbb{N}$, does $t \in e_C(g_s)$?". This problem is denoted $\mathcal{RC}(o_1, \ldots, o_k)$ when we consider arithmetic circuits using operators among $o_1, \ldots, o_k$. With $\mathcal{RF}(o_1, \ldots, o_k)$ we denote the restriction to formulae.

Every gate in an arithmetical circuit has a *height* defined by $h_C(g) \stackrel{\text{def}}{=} \max\{1 + h_C(g_1), \ldots, 1 + h_C(g_m)\}$, where $g_1, .., g_m$ are the input gates of g. In particular $h_C(g) = 0$ if g has no inputs.

A circuit is *layered* is every edge $(g, g') \in E_C$ has $h_C(g') = 1 + h_C(g)$. Thus C_1 in Fig. 1 is not layered as witnessed by its edges (g_2, g_8) and (g_7, g_{10}). We write $\mathcal{RC}_{lay}(o_1, \ldots, o_k)$ for the restriction of $\mathcal{RC}(o_1, \ldots, o_k)$ to layered circuits, and similarly for $\mathcal{RF}_{lay}(\ldots)$.

1. Show that, for any set $\{o_1, \ldots, o_k\} \subseteq \{+, \times, \cup\}$ of operations, $\mathcal{RC}(o_1, \ldots, o_k) \leq \mathcal{RC}_{lay}(o_1, \ldots, o_k)$ and $\mathcal{RF}(o_1, o_2, \ldots, o_k) \leq \mathcal{RF}_{lay}(o_1, o_2, \ldots, o_k)$, where as usual "$\leq$" denotes logspace many-one reducibility. Be careful when justifying the complexity of your reductions.

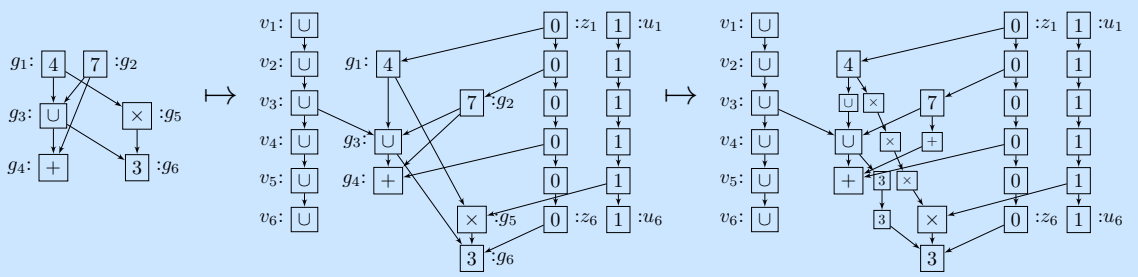
Solution:

One difficulty here is that the reduction cannot compute the height of gates and use that information since computing height is hard (telling whether $h(g) \geq k$ given C, g, k is NL-complete).

We propose a transformation τ in two steps. First we introduce extra gates and edges ensuring that the n-th gate has height n. Informally, if C has n gates, we add a column of n $\cup$-gates $v_1, \ldots, v_n$, a column of n 0-gates $z_1, \ldots, z_n$ and a column of n 1-gates $u_1, \ldots, u_n$ such that $h(v_i) = h(z_i) = h(u_i) = i - 1$ and $e(v_i) = \emptyset$, $e(z_i) = \{0\}$, $e(u_i) = \{1\}$. We further add edges from the new gates to the original ones : if $l(g_i) = \cup$, we add the edge (v_i, g_i) ; if $l(g_i) = \times$, we add the edge (u_i, g_i) ; otherwise we add the edge (z_i, g_i). This ensures that gate g_i has height i in the new circuit and that its evaluation is unchanged. (NB : In order to ensure that the resulting circuit is in the same class $\mathcal{C}(o_1, \ldots, o_k)$ we only add the v_i's if the original circuit contained at least one $\cup$-gate.)

The second step considers each edge (g', g) and inserts intermediary gates whenever $h(g) \neq h(g') + 1$. (Note that in the new circuit we know the height of each gate without having to compute it from the graph structure.) Formally, and writing $\delta(g', g)$ for $h(g) - h(g') - 1$, we replace any edge (g', g) with $\delta > 0$ by new edges $(g', h_1), (h_1, h_2), \ldots, (h_\delta, g)$ where $h_1, h_2, \ldots, h_\delta$ are new gates with $l(h_i) = l(g')$ (ensuring that we don't introduce ops that were not in C).

The 2-step transformation is illustrated below :



The essential properties of this transformation are :

1. It is logspace computable since each half clearly is.
2. It produces a new circuit $C' = \tau(C)$ that is layered, contains all the original gates, and satisfies $e_{C'}(h) = e_C(h)$ if $h \in G_C$, and $e_{C'}(h) = e_C(g')$ if h was introduced when replacing $(g', g) \in E_C$. This correctness property is proven by induction on $h_{C'}(h)$.
3. C and $\tau(C)$ use the same set of operations $o_1, \ldots, o_k$.

We can thus reduce $\mathcal{RC}(o_1, \ldots, o_k)$ to $\mathcal{RC}_{lay}(o_1, \ldots, o_k)$ by the reduction $r : C, g, n \mapsto \tau(C), g, n$.

We note that $\tau(C)$ is not necessarily an arithmetical formula when C is, so the same idea does not reduce $\mathcal{RF}(o_1, \ldots, o_k)$ to $\mathcal{RF}_{lay}(o_1, \ldots, o_k)$. There are several ways to solve the problem : either we only keep the second half of the transformation (since for arithmetical formulae it is possible to compute the heights of all the gates in logspace), or we adapt τ so that the first step produces a formula (one possibility would be to create one column $(u_i)_i$ or $(v_i)_i$ or $(z_i)_i$ for each gate of C, cutting it at the required depth).

2. For complexity reasoning, we define the size $|C|$ of an arithmetic circuit as $|G_C|+|E_C|+\sum_{g\in G_C} |l_C(g)|$ where $|o| \stackrel{\text{def}}{=} 1$ for $o \in Ops$, and $|a| \stackrel{\text{def}}{=} \lceil \log_2 \max(2, a) \rceil$ for $a \in \mathbb{N}$.
Give a polynomial p such that $n \in e_C(g) \implies n \leq 2^{p(|C|)}$ for any arithmetical formula $C \in \mathcal{F}(\cup, +, \times)$ and any gate $g \in G_C$. Justify your answer.

Solution:
Since C is a formula, any reachable number is obtained by using *at most once* the constant labels $a_1, \ldots, a_\ell$ of C, so necessarily $n \in e(g)$ implies $\log_2(n) \leq \log_2(a_1) + \cdots + \log_2(a_\ell) \leq |C|$, or $n \leq 2^{|C|}$.

3. Show that for any polynomial p, there exist circuits in $\mathcal{C}(\cup, +, \times)$ with gates such that $e_C(g)$ contain numbers larger than $2^{p(|C|)}$.
Give a function f such that $n \in e_C(g) \implies n \leq f(|C|)$ for any circuit $C \in \mathcal{C}(\cup, +, \times)$ and any gate $g \in G_C$. Justify your answer.

Solution:
Regarding lower bounds, we can consider a circuit C_n with $n + 1$ gates $g_0, \ldots, g_n$ with $l(g_0) = l(g_1) = 2$ and, for any $i > 1$, $l(g_i) = \times$ and $i(g_i) = \{g_j \mid j < i\}$. One has $|G| = n + 1$, $|E| \leq n^2$ and $\max_g |l(g)| = 1$. On the other hand, $e(g_i) = 2^{2^{i-1}}$ when $i > 0$ so $e(g_n) = 2^{2^{n-1}}$ which dominates any $2^{p(|C_n|)}$ when n is large enough.

Let $C \in \mathcal{C}(\cup, +, \times)$ and let $A = \max\{a_1, \ldots, a_m\}$ where $a_1, \ldots, a_m$ are the number appearing as labels on non-op gates. Write simply G for $|G|$. We claim that $n \in e(g) \implies n \leq A^{G^{h(g)}}$ for any $g \in G$ and prove this by induction on $h(g)$.
If g is a non-op gate, then n is one of the a_i's and $n \leq A$.
If g is a $\cup$-gate, then $n \in e(g')$ for some $g' \in i(g)$, hence $n \leq A^{G^{h(g')}}$ by induction hypothesis, which proves our claim since $h(g') < h(g)$.
If g is a $+$ or $\times$-gate with inputs $g_1, \ldots, g_m$, then $m > 0$ since $e(g) \neq \emptyset$ and $n = n_1 + \cdots + n_m$ or $n = n_1 \times \cdots \times n_m$, for some tuple $n_1, \ldots, n_m$ such that $n_i \in e(g_i)$ for all i. The induction hypothesis gives $n_i \leq A^{G^{h(g_i)}}$, hence $n_i \leq A^{G^{h(g)-1}}$ since $h(g_i) < h(g)$. We obtain $n \leq m \cdot A^{G^{h(g)-1}}$ in the case of a $+$-gate, and $n \leq \left(A^{G^{h(g)-1}}\right)^m$ in the case of a $\times$-gate. Since $m < G$ one may conclude $n \leq A^{G^{h(g)}}$ in both cases.
Finally, and since $d(g) < G \leq |C|$ and $\log_2 A \leq \max_{g\in G_c} |l_C(g)| < |C|$, we obtain $n \leq 2^{\log_2 A \cdot |C|^{|C|-1}} \leq 2^{|C|^{|C|}}$. Therefore $f(|C|) = 2^{|C|^{|C|}}$ works.

4. Prove that $\mathcal{RF}(\cup, \times, +) \in \mathsf{NP}$.

Solution:
The intuition is that it is enough to "nondeterministically guess a value for each gate" but let us explain how this can be handled more formally, taking care of the gates where $e_C(g)$ is empty.
We say that a *partial* function $w : G_C \to \mathbb{N}$ is a *sampling* for C if w satisfies the following consistency property for every $g \in \operatorname{dom} w$:

1. if $l_C(g) = n \in \mathbb{N}$ then $w_C(g) = n$,
2. if $l_C(g) = \cup$ then $w(g) = w(g')$ for some $g' \in i_C(g)$,

3. if $l_C(g) = +$ (or $\times$) then $i_C(g) = \{g_1, \ldots, g_m\}$ has size $m > 0$ and $w(g) = w(g_1) + \cdots + w(g_m)$ (or $w(g) = w(g_1) \times \cdots \times w(g_m)$).

Samplings have the following properties :

1. if $w(g) = n$ for a sampling w then $n \in e_C(g)$. This is proven for any circuit C and any gate in G_C by induction on the depth of the gate, using (D_1) and the induction hypothesis to prove that, indeed, $w(g) \in e_C(g)$.
2. if $n \in e_C(g)$ and C is a formula, then there exists a sampling w with $w(g) = n$ and with $\operatorname{dom} w \subseteq i^*_C(g)$, where $i^*_C(g) \stackrel{\text{def}}{=} \{g\} \cup \bigcup_{g' \in i(g)} i^*_C(g')$ collects all gates from which one can reach g. Again the proof is by induction on the depth of g. One crucial argument is that, *since C is a formula*, the set $i^*(g'_1)$ and $i^*(g'_2)$ are disjoint when g'_1, g'_2 are any two different inputs of some g. Therefore we can always combine a witness for g'_1 and a witness for g'_2 when building a witness for g.

There remains to prove that a sampling necessarily has size polynomial in $|C|$, in particular that the values are not too large, but since C is a formula, this is a direct consequence of Question 2.

Therefore, to decide $n \in e_C(g)$, it is enough to guess some w of polynomial size and verify that it is indeed a sampling with $w(g) = n$. (Verifying a sampling is easily done in polynomial time.)

5. Let $f : \mathbb{N} \to \mathbb{N}$ be any proper [1] function with $f(x) \geq x$:
Prove that $\mathcal{RC}(o_1, \ldots, o_k) \in \mathsf{TIME}\big(f(2^{n^{O(1)}})\big)$ if $\mathcal{RF}(o_1, \ldots, o_n) \in \mathsf{TIME}(f(n))$.
Prove that $\mathcal{RC}(o_1, \ldots, o_k) \in \mathsf{NTIME}\big(f(2^{n^{O(1)}})\big)$ if $\mathcal{RF}(o_1, \ldots, o_n) \in \mathsf{NTIME}(f(n))$.
Use this to prove that $\mathcal{RC}(\cup, +, \times) \in \mathsf{NEXPTIME}$.

Solution:
One may reduce $\mathcal{RC}(\ldots)$ to $\mathcal{RF}(\ldots)$ by unfolding the circuit into a formula. Formally, we would let $H = \max_{g \in G} h(g)$ be the maximal depth of a gate and replace every gate g_i with $|G|^{H-h(g_i)}$ copies $g_i^1, \ldots, g_i^{H-h(g_i)}$. Then an edge $(g_i, g_j) \in E$ such that $h(g_j) = h(g_i) + \Delta$ is replaced with all the edges $(g_i^{j+a\cdot|G|^\Delta}, g_j^a)$ for $a = 1, \ldots, |G|^{h(g_j)}$.
The transformation is essentially making copies of C's components via loops. If we let $n = |C|$, the resulting formula has size bounded by $n \times |G|^H$ (since $|G|^H$ bounds the number of copies), which is $\leq n \times n^n$, i.e., in $2^{O(n \log n)}$. Once C has been transformed into an equivalent (but exponential-sized) formula, we can use the $\mathsf{TIME}(f)$ algorithm, or the $\mathsf{NTIME}(f)$ algorithm, and obtain the desired result. Since $f(x) \geq x$, the cost of the transformation step can be absorbed in the cost of the solving step.

Finally we obtain an $\mathsf{NEXPTIME}$ upper bound by relying on Question 4 : $\mathcal{RF}(\cup, +\times)$ is in $\mathsf{NTIME}(p(n))$ for some polynomial p, hence $\mathcal{RC}(\cup, +\times)$ is in $\mathsf{NTIME}\big(2^{p(n)^k}\big)$ for some k, i.e. in $\mathsf{NTIME}\big(2^{n^{k'}}\big)$ for some k', hence in $\mathsf{NEXPTIME}$.

6. Show that $\mathcal{RC}(\cup, +)$ is NP-hard. For this one may use results seen in class or during exercises showing how several versions of the $\mathsf{KnapSack}$ problems, e.g., $\mathsf{SubsetSum}$, are NP-complete.

Solution:
We can transform a $\mathsf{SubsetSum}$ instance $\langle n; a_1, \ldots, a_m \rangle$ into an arithmetical formula $(a_1 \cup 0) + \cdots + (a_m \cup 0)$, showing that already $\mathcal{RF}(\cup, +)$ is NP-hard.

7. Show that $\mathcal{RC}(\cup, \times)$ is NP-hard.

Solution:
NB : Here we cannot simply reduce $\mathcal{RC}(\cup, +)$ to $\mathcal{RC}(\cup, \times)$ by replacing $+$-gates by $\times$-gates, and n-gates by corresponding 2^n gates : this would not be a logspace (or even a polynomial-time) reduction since 2^n is too large. Similarly, replacing 2^n by a product of n 2-gates uses too many gates. Replacing 2^n by a nested product like in Question 3 would work but the reduction still has to produces a new target 2^t and this requires exponential time. So we have to propose another reduction... (NB : Some of you searched through the literature, managed to find papers showing that SUBSET-PRODUCT is NP-complete, and gave adequate references : this is a perfectly acceptable solution for this exam at home.)

1. Cf. "polycopié du cours", page 11.

Let $\phi = \bigwedge_{j=1}^{m} H_j$ be a 3SAT instance using variables $x_1, \ldots, x_n$. With each clause H_j we associate the j-th prime number p_j. Then, for all $1 \leq i \leq n$, we define the numbers $a_i \stackrel{\text{def}}{=} \prod_{x_i \in H_j} p_j$ and $b_i \stackrel{\text{def}}{=} \prod_{\neg x_i \in H_j} p_j$ as the product of the prime numbers corresponding to the clauses where the variable x_i appears positively (resp. negatively). We claim that ϕ is satisfiable iff the $\mathcal{F}(\cup, \times)$ formula

$$(a_1 \cup b_1) \times \cdots \times (a_n \cup b_n) \times (1 \cup p_1 \cup p_1^2) \times \cdots \times (1 \cup p_m \cup p_m^2)$$

can reach $P \stackrel{\text{def}}{=} \prod_{j=1}^{m} p_j^3$.

Proof of the claim. With a valuation $\nu : \{x_i\}_{1 \leq i \leq n} \to \{\text{true}, \text{false}\}$, we associate the number $K_\nu = \prod_{i=1}^{n} c_i$ where $c_i = a_i$ if $\nu(x_i) = \text{true}$ and $c_i = b_i$ otherwise. First, note that K_ν always divides P (written $K_\nu \mid P$) since the only prime numbers appearing in K_ν are the p_j for $1 \leq j \leq m$ and any p_j appears at most thrice in K_ν since we consider a 3SAT formula.

It is now clear that $\nu \models \phi$ iff $p_j \mid K_\nu$ for all $1 \leq j \leq m$. Furthermore, $p_j \mid K_\nu$ for all $1 \leq j \leq m$ iff the formula

$$(1 \cup p_1 \cup p_1^2) \times \cdots \times (1 \cup p_m \cup p_m^2)$$

can reach $\frac{P}{K_\nu}$ since $P = \prod_{j=1}^{m} p_j^3$. By combining, these two observations, our claim follows.

It remains to show that the first prime numbers $p_1, \ldots, p_m$, the a_i's and b_i's, etc., and the product P can be constructed in logspace and this is the main challenging question at the end of this homework exam. Note that m, the number of clauses, is at least the size of the input. Equivalently, we want to generate the p_j's, etc. from m written in unary. The Prime Number Theorem tells us that p_j is $O(j \log j)$, hence has $O(\log j)$ digits and can be found in logspace e.g. by naive loops trying all divisors.

Producing $P = \prod_j p_j^3$ from $p_1, \ldots, p_m$ in logspace is trickier and we won't explain how to do it here. The problem is called IteratedMultiplication in the literature and interested readers are welcome to look up for solutions on the web. Naturally, answers just giving references for the result on IteratedMultiplication will be accepted.