Interactive Proofs, **MA**; **RL** and **NL**

All written documents allowed. No access to Internet, no mobile phones allowed.

Questions are graded according to their difficulty. I have given an indication of the number of lines my solution has for each question, between square brackets. But beware that difficulty and length may be uncorrelated. My solution to the whole exam is *[102 lines]* long.

A final note: it may (and will) happen that the key trick in solving one question was introduced in one of the previous questions.

1 Interactive Proofs and MA

[35 lines total for this part]

1. *[2 lines]* Let p be a prime number, and $P(X)$ be a polynomial in one variable X over $\mathbb{Z}/p\mathbb{Z}$, of degree $d \ll p$. Assume $P(X)$ is given implicitly: all we have is an oracle $\mathcal{O}$ that computes $P(v)$ for any input $v \in \mathbb{Z}/p\mathbb{Z}$. Show that we can compute the coefficients a_k of X^k in $P(X)$, $0 \leq X \leq d$, in time polynomial in d and $\log_2 p$ on a deterministic Turing machine with oracle $\mathcal{O}$.

2. *[10 lines]* Reexamine the Shen protocol (or the Shamir protocol, if you prefer), and show that if Merlin can convince Arthur that the input QBF formula is true, then he can do so in in polynomial space. In other words, show that we can implement the Merlin function M in the definition of the **IP** protocol for QBF by a polynomial-space computable function. (Note that the "obvious" way of producing the polynomial that Merlin submits to Arthur at each step is not polynomial space.)

3. *[20 lines]* If **PSPACE** $\subseteq$ **P**/*poly*, then the above shows that we can replace Merlin, in an interactive proof of QBF, by a polynomial-sized circuit. Make this precise, and show that, under this assumption, QBF is solvable in **MA**.

4. *[3 lines]* Conclude that if **PSPACE** $\subseteq$ **P**/*poly*, then **PSPACE** = **MA**.

2 RL and NL

[67 lines total for this part]

RL (randomized logspace) is the class of all languages L such that there is a randomized Turing machine $\mathcal{M}$, working in space $O(\log n)$, that terminates with probability 1, and such that:

- If $x \in L$, then $Pr_r[\mathcal{M}(x,r) \text{ accepts}] \geq \frac{1}{2}$;
- If $x \notin L$, then $\mathcal{M}(x,r)$ accepts for no r (it may reject, or fail to terminate—with zero probability).

As for **ZPP**, we do not require $\mathcal{M}$ to terminate on all inputs, just on a subset of measure 1. We also have the same measure-theoretic concernes as for **ZPP**, but will assume them solved.

Beware of a counter-intuitive fact: it will become apparent from the last questions below that we cannot require $\mathcal{M}$ to terminate in much less than *exponential* average time, while the analogy with **NL** would lead us to think that polynomial time would be enough. (The status of the corresponding class **RLP** of all problems solvable in logspace and polynomial time is not known.)

One can replace the $\frac{1}{2}$ above by any constant ϵ ($0 < \epsilon < 1$), or by any function of the form $1 - \frac{1}{2^{p(n)}}$, where p is a non-negative polynomial. As with **RP**, this is by repeating experiments. In the second case, the counter used takes only logarithmic space.

5. *[9 lines]* Show that **RL** $\subseteq$ **NL**.

6. *[7 lines]* Let $G = (V, E)$ be a directed graph, s and t be two vertices of G (i.e., $s, t \in V$). Let N be the number $\#V$ of vertices of G. Show that the following *repeated random walk* algorithm $\mathcal{A}$, taking G as input, has the following properties:

   P1 $\mathcal{A}$ runs in $O(\log n)$ space, where n is the size of G;

   P2 If t is reachable from s in G, then $\mathcal{A}$ accepts with probability 1;

   P3 If t is not reachable from s in G, then $\mathcal{A}$ does not terminate.

```
1   v := s;
2   for i:=0 to N {
3       if v = t then accept;
4       else if v has no successor then break; // exit the loop (goto 10)
5       else {
6               pick a successor vertex v' of v in G at random uniformly;
7               v := v';
8           }
9       }
10  goto 1 // start again from s
```

   For now, you shall assume that: ($*$) there is a way of picking a successor vertex v' of v in G at random uniformly in finite average time and using $O(\log n)$ space. We shall return to this in Question 7.

7. *[5 lines]* Why is assumption $(*)$ valid? I.e., propose a algorithmic way of picking v' at random so that $(*)$ holds. If you don't see the difficulty, train yourself on the following question first: how do you pick a element from a set of 3 possible successors, knowing that a randomized Turing machine only draws *bits* at random?

8. *[14 lines]* Modify Algorithm $\mathcal{A}$ by adding a counter, so that the resulting algorithm $\mathcal{A}'$ satisfies:

   P1mod $\mathcal{A}'$ runs in space $O(\log n)$ plus the size of the counter;

   P1bis The space used by the counter is bounded by some polynomial $p(n)$ (yes, this means the counter is too big for our final purpose, but you should not be able to do better at this point);

   P2mod If t is reachable from s in G, then $\mathcal{A}$ accepts with probability at least $\frac{1}{2}$;

   P3mod If t is not reachable from s in G, then $\mathcal{A}$ rejects.

9. *[6 lines]* Imagine we wish to repeat some task T $2^{p(n)}$ times. One possibility is to maintain a $p(n)$-bit counter. If we have access to a source of randomness, we may instead repeat the following:

   (a) Draw $p(n) + 2$ bits at random uniformly.
   (b) If all of them are 0, then stop.
   (c) Otherwise, do T and go back to (a)

   Show that this will execute task T at least $2^{p(n)}$ times, with probability at least $\frac{3}{4}$. (We assume $p(n)$ tends to infinity when $n \to +\infty$, and require this asymptotically, i.e., for n large enough. You may need the numerical estimate $e^{-1/4} \approx 0.7788$.)

10. *[20 lines]* Use the previous observation to produce an **RL** algorithm deciding whether t is reachable from s in G.

11. *[6 lines]* Conclude that $\mathbf{RL} = \mathbf{NL}$.