End Semester Examination, Autumn 2003-04

Solutions

---

1. Well! This is the integer factoring problem with the inputs given in the unary representation. We know that the unary representation is exponentially long compared to the binary representation. Any reasonable algorithm that solves the integer factoring problem in time exponential in the binary size of the operands takes time polynomial in the unary size. For example, one may first compute the binary representations of m and k from the input and then try dividing m in succession by $2, 3, \ldots, k$. If any of the divisions results in zero remainder, the algorithm accepts. If all divisions give non-zero remainders, the algorithm rejects.

2. Let $(C_0, C_1, C_2, \ldots)$ be a uniform family of NC^j circuits deciding the language L. Inverting the output of each C_n (using a single NOT gate) gives a new circuit family that decides $\overline{L}$. This addition increases the size and depth of each C_n by 1 and so, in particular, does not add to the asymptotic size or depth complexity of the original family. Moreover, the log-space transducer that can manufacture $(C_0, C_1, C_2, \ldots)$ knows the output gate of each C_n and so can add another NOT gate to it using no more than an additional logarithmic space.

3. Suppose that f is one-way. Then, among other things, f is injective, i.e., $f(\alpha) = \alpha$ for every $\alpha \in \Sigma^*$. But then $f^{-1}(\alpha) = \alpha$ for every α, i.e., f^{-1} is trivially in FP, a contradiction.

4. We know that if any NP-hard language is in coNP, or symmetrically, if any coNP-hard language is in NP, then $\mathrm{NP} = \mathrm{coNP}$. Thus we can take $A = \mathrm{SAT}$ and $B = \mathrm{UNSAT} = \overline{\mathrm{SAT}}$.

5. We have seen how an AC^0 circuit can add two n-bit (or $2n$-bit) integers given in binary representation. Since $\mathrm{AC}^0 \subseteq \mathrm{NC}^1$, addition can be done also by NC^1 circuits. For multiplication of two integers $a := (a_{n-1} \ldots a_1 a_0)_2$ and $b := (b_{n-1} \ldots b_1 b_0)_2$ one first computes $2^i a b_i$ for $i = 0, 1, \ldots, n-1$ in parallel. Since $2^i a b_i$ is either all zeros or a shifted version of a depending on whether $b_i = 0$ or $b_i = 1$, a constant-depth poly-size circuit can produce all these values. Now we view each $2^i a b_i$ as a $2n$-bit number and employ a height-balanced binary tree of $2n$-bit adders to compute ab. Each such adder is itself of logarithmic depth and polynomial size. It follows that the overall multiplier circuit is of log-square depth and polynomial size.

6. Suppose that $\mathrm{TIME}(n) = \mathrm{NL}$. I then claim that $\mathrm{TIME}(n^2) \subseteq \mathrm{NL}$. This claim implies that $\mathrm{TIME}(n^2) \subseteq \mathrm{TIME}(n)$, a contradiction to $\mathrm{TIME}(n) \subsetneqq \mathrm{TIME}(n^2)$, a fact that follows from the time hierarchy theorem. For the proof of the claim take $L \in \mathrm{TIME}(n^2)$ and consider the language

$$L' := \{\alpha \underline{\sqcup}^{|\alpha|^2 - |\alpha|} \mid \alpha \in L\},$$

where $\underline{\sqcup}$ is a quasiblank symbol. Since $L \in \mathrm{TIME}(n^2)$, we have $L' \in \mathrm{TIME}(n)$ and so $L' \in \mathrm{NL}$. But then an NL machine N' for L' can be converted to an NL machine N for L as follows. N first appends the requisite number of quasiblanks to its input α and then simulates N' on this padded input α'. If $|\alpha| = n$, $|\alpha'| = n^2$ and so the non-deterministic space complexity of N is $\mathrm{O}(\log(n^2))$, which is again $\mathrm{O}(\log n)$, i.e., $L \in \mathrm{NL}$.

7. Let N_1 and N_2 be PP machines respectively deciding the languages L_1 and L_2 (over the same alphabet Σ). Consider the following algorithm N:

Input: $\alpha \in \Sigma^*$.

Stages:
1. Simulate N_1 on α.
2. Simulate N_2 on α.
3. If N_1 and N_2 differ about the acceptance of α, *accept*, else *reject*.

$\alpha \notin L_2$, $\Pr[N_2 \text{ rejects } \alpha] = \frac{1}{2} + \delta_2$ for some $0 < \delta_2 \leqslant \frac{1}{2}$. (Note that the biases δ_1 and δ_2 may, in general, be dependent on α, but can be (uniquely) obtained from the computation trees of N_1 and N_2 on α. In fact, a PSPACE machine can compute them. However, we don't have to know their exact values; all we require is that they are both positive.) We then have:

$$\begin{aligned}\Pr[N \text{ accepts } \alpha] &= \Pr[N_1 \text{ accepts } \alpha] \times \Pr[N_2 \text{ rejects } \alpha] + \Pr[N_1 \text{ rejects } \alpha] \times \Pr[N_2 \text{ accepts } \alpha] \\ &= \left(\frac{1}{2} + \delta_1\right)\left(\frac{1}{2} + \delta_2\right) + \left(\frac{1}{2} - \delta_1\right)\left(\frac{1}{2} - \delta_2\right) = \frac{1}{2} + 2\delta_1\delta_2 > \frac{1}{2}\,.\end{aligned}$$

Symmetrically N accepts $\alpha \in L_2 \setminus L_1$ with probability $> \frac{1}{2}$.

Next take $\alpha \in L_1 \cap L_2$. We again have positive biases δ_1 and δ_2 such that $\Pr[N_1 \text{ accepts } \alpha] = \frac{1}{2} + \delta_1$ and $\Pr[N_2 \text{ accepts } \alpha] = \frac{1}{2} + \delta_2$. But then

$$\begin{aligned}\Pr[N \text{ accepts } \alpha] &= \Pr[N_1 \text{ accepts } \alpha] \times \Pr[N_2 \text{ rejects } \alpha] + \Pr[N_1 \text{ rejects } \alpha] \times \Pr[N_2 \text{ accepts } \alpha] \\ &= \left(\frac{1}{2} + \delta_1\right)\left(\frac{1}{2} - \delta_2\right) + \left(\frac{1}{2} - \delta_1\right)\left(\frac{1}{2} + \delta_2\right) = \frac{1}{2} - 2\delta_1\delta_2 < \frac{1}{2}\,.\end{aligned}$$

Similarly, N accepts $\alpha \in \overline{L_1} \cap \overline{L_2}$ with probability $< \frac{1}{2}$.

To sum up, N decides $L_1 \triangle L_2$ in probabilistic polynomial time.

8. Since the reduction from MAJSAT to THRESHOLD-SAT is obvious (convert $\langle\phi\rangle$ to $\langle\phi, 2^{m-1}\rangle$, where m is the number of variables in ϕ), it suffices to show that THRESHOLD-SAT $\in$ PP. Consider the following algorithm N:

Input: $\langle\phi, k\rangle$, where ϕ is a Boolean formula and $k \in \mathbb{N}_0$.

Stages:

1. Compute the number m of variables in ϕ.
2. If $k \geqslant 2^m$, *reject*.
3. Make a coin toss.
4. If the toss outcome is 'Head'
5. Evaluate ϕ at a (uniformly) random truth assignment of the variables.
6. If the evaluation result is 1, *accept*, else *reject*.
7. else
8. Make m coin tosses and treat the toss outcomes as an m-bit number t, $0 \leqslant t < 2^m$.
9. If $0 \leqslant t < k$, *reject*, else *accept*.

I now establish that the above algorithm N decides THRESHOLD-SAT in PP. First assume that ϕ has more than k satisfying truth assignments. Then $\Pr[N \text{ accepts } \langle\phi, k\rangle] > \frac{1}{2} \times \frac{k}{2^m} + \frac{1}{2} \times (\frac{2^m - k}{2^m}) = \frac{1}{2}$. On the other hand, if ϕ has $\leqslant k$ satisfying truth assignments $\Pr[N \text{ accepts } \langle\phi, k\rangle] \leqslant \frac{1}{2} \times \frac{k}{2^m} + \frac{1}{2} \times (\frac{2^m - k}{2^m}) = \frac{1}{2}$. It follows that THRESHOLD-SAT $\in \text{PP}' = \text{PP}$.

9. Clearly, NAND-CIRCUIT-VALUE is in P. In order to show its P-hardness I reduce CIRCUIT-VALUE to NAND-CIRCUIT-VALUE (in log-space). Let $\langle C, \alpha\rangle$ be an instance for CIRCUIT-VALUE. I want to build a NAND circuit C' such that $C'(\alpha') = 1$ if and only if $C(\alpha) = 1$, where $\alpha' := \alpha 1$. To that end I convert each gate g of C to an equivalent circuit consisting of NAND gates only. Note that the constant bit 1 is available in α'.

[g is a NOT gate] Let x be the input of g. Then its output is $\overline{x} = \overline{x} \vee 0 = \overline{x} \vee \overline{1} = x \mathbin{\overline{\wedge}} 1$.

[g is an AND gate] With inputs x and y the gate g computes $x \wedge y = \overline{x \mathbin{\overline{\wedge}} y} = (x \mathbin{\overline{\wedge}} y) \mathbin{\overline{\wedge}} 1$.

[g is an OR gate] Let the inputs of g be x and y. Then its output is $x \vee y = \overline{x} \mathbin{\overline{\wedge}} \overline{y} = (x \mathbin{\overline{\wedge}} 1) \mathbin{\overline{\wedge}} (y \mathbin{\overline{\wedge}} 1)$.

With these replacements the number of gates in C' becomes at most three times that in C, whereas the depth of C' is at most twice that of C. But these figures are not much relevant in this context. It is necessary to argue that a log-space transducer can convert $\langle C, \alpha\rangle$ to $\langle C', \alpha'\rangle$. But this is evident from the fact that while converting each gate g of C to a NAND circuit, it suffices to store only the pointers to the inputs to g (input variables or outputs of other gates in C) and to the output of g. These pointers occupy space logarithmic in the size of $\langle C, \alpha\rangle$.

$$L_f := \{\langle x, y\rangle \mid f(z) = y \text{ for some } z \leqslant x\}$$

is in $\mathrm{UP} \setminus \mathrm{P}$. Since $\mathrm{UP} \subseteq \mathrm{NP}$, we have $L_f \in \mathrm{NP} \setminus \mathrm{P}$. What remains is to prove that if f is bijective, then $L_f \in \mathrm{coNP}$, i.e., $\overline{L_f} \in \mathrm{NP}$. The bijectivity of f implies that $f^{-1}(y)$ exists for all y, i.e.,

$$\overline{L_f} = \{\langle x, y\rangle \mid f^{-1}(y) > x\}.$$

Since f is a one-way function, it suffices to search over all candidates z with $|y|^{1/k} \leqslant |z| \leqslant |y|^k$ for some constant k and accept if and only if one such z corresponds to both $f(z) = y$ and $z > x$. Clearly, this can be achieved in non-deterministic polynomial time. In fact, (since f is injective), we have proved a stronger result, namely, $L_f \in (\mathrm{UP} \cap \mathrm{coUP}) \setminus \mathrm{P}$.