Midterm Solutions

1. A directed graph $G = (V, E)$ is *strongly connected* if for any two vertices $u, v \in V$ there is a directed path in G from u to v. Let strong-CONN be the problem of deciding whether a given graph is strongly connected.

   (a) Show that strong-CONN is in **NL**.

   (b) Prove the **NL**-completeness of strong-CONN by giving a log-space reduction from ST-CONN to strong-CONN.

Solutions

The NL algorithm just goes through all pairs of vertices u, v and guesses a path from u to v, rejecting if a path is not found in less than $|V|$ steps. If the graph is strongly connected, there is at least an accepting computations that guesses the paths right correctly each time, and if there is an accepting computation, it must be the case that the graph is strongly connected.

In the reduction, given $G = (V, E), s, t$, create $G' = (V, E')$ where E' has all the edges of E and also all edges (v, s) and (t, v). If there is a path from s to t, then from every vertex u we can go to every vertex v by using the edge (u, s), the path from s to t and the edge (t, v), so G' is strongly connected. But if G' is strongly connected, there is a simple path in G' from s to t, and such simple path will not use any of the new edges, so there was already a path from s to t in G.

2. Suppose that there is a deterministic polynomial-time algorithm A that on input (the description of) a circuit C produces a number $A(C)$ such that

$$\mathbf{Pr}_x[C(x) = 1] - \frac{2}{5} \leq A(C) \leq \mathbf{Pr}_x[C(x) = 1] + \frac{2}{5} .$$

   (a) Prove that it follows $\mathbf{P} = \mathbf{BPP}$.

   (b) Prove that there exists a deterministic algorithm A' that, on input a circuit C and a parameter ϵ, runs in time polynomial in the size of C and in $1/\epsilon$ and produces a value $A'(C, \epsilon)$ such that

$$\mathbf{Pr}_x[C(x) = 1] - \epsilon \leq A'(C, \epsilon) \leq \mathbf{Pr}_x[C(x) = 1] + \epsilon .$$

   (c) Prove that there exists a deterministic algorithm A'' that, on input a circuit C computing a function $f : \{0,1\}^n \to \{1, \ldots, k\}$ and a parameter ϵ, runs in time polynomial in the size of C, in $1/\epsilon$ and in k, and produces a value $A''(C, \epsilon)$ such that

$$\mathbf{E}_x[f(x)] - \epsilon \leq A''(C, \epsilon) \leq \mathbf{E}_x[f(x)] + \epsilon .$$

   [For this question, you can think of C as being a circuit with $\log k$ outputs, and the outputs of $C(x)$ are the binary representation of $f(x)$.]

Solutions

Part (2a) was easy: let L be a **BPP** language and A_L the probabilistic polynomial time algorithm that decides L, we may assume that the error probability of L is $\leq 1/10$. Here is a deterministic polynomial time algorithm for L: on input x, construct the circuit C such that $C(r) = 1$ if and only if $A(x, r)$ accepts. Then accept if and only if $A(C) > 1/2$.

Several people said that, once we have $\mathbf{P} = \mathbf{BPP}$, part (2b) can be solved by just giving a probabilistic algorithm that outputs the right answer with high probability. This is not quite right, because $\mathbf{P} = \mathbf{BPP}$ is a statement that concerns only languages, not function computations, and if you try to prove formally that $\mathbf{P} = \mathbf{BPP}$ implies a solution to part (2b) you will run into trouble. In fact, it is a *major* open question to show that $\mathbf{P} = \mathbf{BPP}$ implies part (2b). However, it is known that if $\mathbf{P} = \mathbf{BPP}$ for *promise problems* then part (2b) follows, and the argument above for part (2a) also extends to promise problems. Instead of formulating the solution in terms of promise problems, it seems simpler to just give it directly.

We solve part (2b) by giving a deterministic algorithm *ApxComp* that, on input a circuit C, an accuracy parameter $\epsilon > 0$ and a threshold parameter $0 \leq p \leq 1$, runs in time polynomial in the size of C and in $1/\epsilon$ and:

   (a) If $\mathbf{Pr}_x[C(x) = 1] \geq p + \epsilon/2$, then $ApxComp(C, p, \epsilon)$ accepts;

   (b) If $\mathbf{Pr}_x[C(x) = 1] \leq p - \epsilon/2$, then $ApxComp(C, p, \epsilon)$ rejects;

We run $ApxComp(C, i \cdot \epsilon/2, \epsilon)$ for $i = 0, 1, \ldots, 2/\epsilon$, and we find the smallest i such that $ApxComp(C, (i-1) \cdot \epsilon/2, \epsilon)$ accepts but $ApxComp(C, i \cdot \epsilon/2, \epsilon)$ rejects. Then $\mathbf{Pr}[C(x) = 1] > (i-1) \cdot \epsilon/2 - \epsilon/2$ and $\mathbf{Pr}[C(x) = 1] < i\epsilon/2 + \epsilon/2$, so we can safely output $i\epsilon/2$ as our solution.

In order to devise the procedure $ApxComp$, it is enough to show how, given C, ϵ and p, to construct a new circuit C' in time polynomial in the size of C and in $1/\epsilon$ such that:

(a) If $\mathbf{Pr}_x[C(x) = 1] \geq p + \epsilon/2$, then $\mathbf{Pr}_z[C'(z) = 1] \geq 9/10$;

(b) If $\mathbf{Pr}_x[C(x) = 1] \leq p - \epsilon/2$, then $\mathbf{Pr}_z[C'(z) = 1] \leq 1/10$.

Then we just compute $A(C')$ and accept if and only if $A(C') > 1/2$.

The circuit C' has $t = O(1/\epsilon^2)$ inputs $x_1, \ldots, x_t$, where each x_i is an n-bit string (here n is the input length of C). The circuits C' computes $C(x_1), \ldots, C(x_t)$ and accepts iff at least pt of such computations output 1. One can use Chebyshev inequality to prove correctness.

Regarding part (2c), there are various ways of reducing the problem to the problem solved in part (2b). For example, one can compute the probability that each of the $\log k$ wires carries a one (up to an additive error $\epsilon/\log k$), and then use such estimates to compute an estimate of the average up to an additive error ϵ.

Otherwise, given a circuit C with n-bit input and $\log k$ bit output, one can define the circuit C' with $n + \log k$ bit input and one-bit output such that $C'(x, i) = 1$ if and only if $C(x) \geq i$. Then, it should be easy to see that $\mathbf{Pr}_{x,i}[C'(x, i) = 1] = \frac{1}{k}\mathbf{E}_x[C(x)]$.

3. Prove that, for every constant t, $\Sigma_2 \not\subseteq \mathbf{SIZE}(n^t)$.

   [Hint: first prove $\Sigma_4 \not\subseteq \mathbf{SIZE}(n^t)$, which should be easy. Then argue about what happens depending on whether or not $SAT \in \mathbf{SIZE}(n^t)$.]

Solutions

Let us fix an arbitrary t, and see that $\Sigma_4 \not\subseteq \mathbf{SIZE}(n^t)$.

As a first step, let us see that there is a polynomial $p(n)$ such that $\mathbf{SIZE}(p(n)) \not\subseteq \mathbf{SIZE}(n^t)$. This can be proved in many different ways, providing different bounds on $p(n)$. Here are three possible arguments:

(a) We know that every function $f : \{0,1\}^k \to \{0,1\}$ can be computed by a circuit of size $O(2^k)$, and that there are functions $f : \{0,1\}^k \to \{0,1\}$ that cannot be computed by circuit of size $2^k/4k$. Let $k(n)$ be such that $2^{k(n)}/4k(n) > n^t$, for example $k(n) = \log_2(5n^t \log_2 n^t)$ will do for sufficiently large n. Then for every sufficiently large n there is a function $f_{k(n)} : \{0,1\}^{k(n)} \to \{0,1\}$ that cannot be computed by circuits of size n^t but that can be computed by circuits of size $O(2^{k(n)}) = O(n^t \log n^t)$. Define the language L such that $x \in L$ if and only if $f_{k(n)}$ applied to the first $k(n)$ bits of x equals 1, where n is the length of x. Then $L \in \mathbf{SIZE}(O(n^t \log n^t))$ but $L \notin \mathbf{SIZE}(n^t)$.

(b) We know that there are at most $2^{2n^t \log n^t + 5n^t}$ circuits of size n^t. Suppose that we are able to define a family of functions such that each function in the family can be computed by a circuit of size $\leq p(n)$ and such that the family contains more than $2^{2n^t \log n^t + 5n^t}$ distinct functions. Then the family contains a function computable by a circuit of size $p(n)$ but not computable by a circuit of size $\leq n^t$. The above proof can be seen as an application of this method where the family is the family of all functions that depend on only the first $5tn^t \log n$ bits of the input. Another approach would be to fix a set S of $2n^t \log n^t + 5n^t + 1$ elements of $\{0,1\}^n$ and then consider all the functions that are zero outside of S. There are $2^{|S|}$ such function, each computable by a circuit of size $O(|S| \cdot n)$.

(c) Suppose that $\mathbf{SIZE}(n^t + 2n + 1) = \mathbf{SIZE}(n^t)$. We claim that this implies that every function $f : \{0,1\}^n \to B$ can be computed by a circuit of size n^t, which we know to be false (for large enough n). We prove the claim by induction on the number of inputs x such that $f(x) = 1$.

   i. If $f(x) = 0$ for every x, then f can be computed by a circuit of constant size, and, for a stronger reason, by a circuit of size n^t.

   ii. Suppose that we have proved that, for every $g : \{0,1\}^n \to \{0,1\}$ that outputs 1 on $\leq K$ inputs, $g \in \mathbf{SIZE}(n^t)$. Let $f : \{0,1\}^n \to \{0,1\}$ be a funtion that outputs 1 on $K+1$ inputs, and let a be one such input. Define the function g such that $g(x) = f(x)$ for $x \neq a$, and $g(a) = 0$. Then g outputs 1 on K inputs, the induction hypothesis can be applied, and g can be computed by a circuit of size $\leq n^t$. But we can write $f(x) = g(x) \vee (x = a)$, where the

expression $(x = a)$ can be realized by a circuit of size at most $2n$. Then $f \in \mathbf{SIZE}(n^t + 2n + 1)$, and, by the assumption, we also have $f \in \mathbf{SIZE}(n^t)$.

Let us use the last solution. So we know that is a circuit C with n inputs of size $n^t + n + 1$ that computes a function that cannot be computed by circuits of size n^t. We define a language L in Σ_4 that, for every input length n, agrees with the output of the lexicographically first circuit like that. Let $\mathcal{C}_s$ be the set of circuits of size s, and let $<_L$ denote the lexicographic ordering, then we can define L formally as follows. For every $x \in \{0,1\}^n$, we have

$$\begin{aligned} x \in L \text{ if and only if } \exists C \in \mathcal{C}_{n^t+n+1}. \quad & (\forall C' \in \mathcal{C}_{n^t}. \exists y \in \{0,1\}^n. C(y) \neq C'(y)) \\ \wedge \quad & (\forall C'' \in \mathcal{C}_{n^t+n+1} : C'' <_L C' \\ & \qquad \exists C''' \in \mathcal{C}_{n^t}. \forall z. C''(z) = C'''(z)) \\ \wedge \quad & C(x) = 1 \end{aligned}$$

Which is logically equivalent to the Σ_4 formulation

$$\begin{aligned} x \in L \text{ if and only if } \quad & \exists C \in \mathcal{C}_{n^t+n+1}. \\ & \forall C' \in \mathcal{C}_{n^t}, C'' \in \mathcal{C}_{n^t+n+1}. \\ & \exists y \in \{0,1\}^n, C''' \in \mathcal{C}_{n^t}. \\ & \forall z. \\ & C(y) \neq C'(y) \wedge ((C'' <_L C) \Rightarrow (C''(z) = C'''(z))) \wedge C(x) = 1 \end{aligned}$$

And so we have $\Sigma_4 \not\subseteq \mathbf{SIZE}(n^t)$. Now, if $\mathbf{NP} \subseteq \mathbf{SIZE}(n^t)$, then, by Karp-Lipton, we have $\Sigma_2 = \Sigma_4 \not\subseteq \mathbf{SIZE}(n^t)$; if $\mathbf{NP} \not\subseteq \mathbf{SIZE}(n^t)$, then, for a stronger reason, $\Sigma_2 \not\subseteq \mathbf{SIZE}(n^t)$.

4. Let f be a one-way permutation and g be a polynomial time computable permutation. Show that $g(f(\cdot))$ and $f(g(\cdot))$ are one-way permutations.

   [Ideally, do the proof in the finite setting: show that if f is (S, ϵ)-one way and g can be computed by a circuit of size t, then $f(g(\cdot))$ and $g(f(\cdot))$ are $(S - t, \epsilon)$-one way. Then derive the asymptotic result from the finite one. Be as detailed as you can in the analysis.]

Solutions

Let g be computable by a circuit of size $\leq t$, and suppose that $f(g())$ is not $(S - t, \epsilon)$ one way. Then there is a function A computable by a circuit of size $\leq S - t$ such that

$$\mathbf{Pr}[A(f(g(x)) = x] \geq \epsilon$$

which also implies that

$$\mathbf{Pr}[g(A(f(g(x)) = g(x)] \geq \epsilon$$

Let us do the change of variable $y \leftarrow g(x)$, notice that the uniform distribution over x corresponds to the uniform distribution over y, call $B() := g(A())$, and we have

$$\mathbf{Pr}[B(f(y)) = y] \geq \epsilon$$

where B is computable by a circuit of size $\leq S$. That is, if $f(g())$ is not $(S - t, \epsilon)$ one way, then f is not (S, ϵ) one way.

Suppose now that $g(f())$ is not $(S - t, \epsilon)$ one way, so that there is A computable in size $S - t$ such that

$$\mathbf{Pr}[A(g(f(x))) = x] \geq \epsilon \ .$$

Define $B() := A(g())$, and note that B can be computed in size $\leq S$ and

$$\mathbf{Pr}[B(f(x)) = x] \geq \epsilon \ ,$$

proving that f is not (S, ϵ) one way. That is, if $f(g())$ is not $(S - t, \epsilon)$ one way, then f is not (S, ϵ) one way.

Asymptotically, let $t(n)$ be a polynomial such that g is computable in time $t(n)$.

Suppose that $f(g())$ is not one-way, then there are polynomials p, q such that, for infinitely many n, $f(g())$ is not $(p(n), 1/q(n))$-one way on inputs of length n, and so f is not $(p(n) + t(n), 1/q(n))$ one way on inputs of length n. This contradicts the assumption that f is one-way.

Suppose that $g(f())$ is not one-way, then there are polynomials p, q such that, for infinitely many n, $g(f())$ is not $(p(n), 1/q(n))$-one way on inputs of length n, and so f is not $(p(n) + t(n), 1/q(n))$ one way on inputs of length n. This contradicts the assumption that f is one-way.