

## Final Solutions

Posted: June 3

Chris Umans

1. (a) The procedure that traverses a fan-in 2 depth  $O(\log^i n)$  circuit and outputs a formula runs in  $\mathbf{L}_i$  – this can be done by a recursive depth-first traversal, which only requires 1 bit of information (“left” or ”right”) at each level of recursion. The procedure for FVAL (Lecture 2) runs in log-space, so on a formula of size  $2^{O(\log^i n)}$ , it runs in  $O(\log^i n)$  space. Using space-efficient composition of the logspace procedure that generates the circuit together with these two procedures we obtain a procedure to evaluate an  $\mathbf{NC}_i$  circuit on a given input in only  $O(\log^i n)$  space, as required.
- (b) The configuration graph for an  $\mathbf{NL}_i$  machine on input  $x$  of length  $n$  has at most  $2^{O(\log^i n)}$  nodes. The input  $x$  is accepted if and only if there is a path from the start node  $s$  to the accept node  $t$  in this graph. We can construct the incidence matrix  $A$  of this graph (with ones on the diagonal), and we observe that  $A^* = A^{2^m}$ , for  $m = O(\log^i n)$  has a one in position  $s, t$  if and only if there is a path of length at most  $2^m$  from  $s$  to  $t$  (here we are using Boolean matrix multiplication). We can square matrix  $A$  with a  $O(\log |A|) = O(\log^i n)$  depth circuit. We repeat this squaring  $m$  times, to compute  $A^*$ . The repeated squaring entails  $m$  sequential copies of the squaring circuit, which has depth  $O(\log^i n)$ . The total depth is  $O(\log^{2i} n)$ .
- (c) Suppose we show  $\mathbf{NL}_i \subseteq \mathbf{NC}_{2i}$  for some  $i > 1$ . Then we have

$$\mathbf{L}_i \subseteq \mathbf{NL}_i \subseteq \mathbf{NC}_{2i} \subseteq \mathbf{P}.$$

However, we know by the Space Hierarchy Theorem that  $\mathbf{L}$  is *strictly* contained in  $\mathbf{L}_i$  for  $i > 1$ . Thus we would have proved  $\mathbf{L} \neq \mathbf{P}$ . In fact, we would have proved something stronger: that  $\mathbf{NC}_1 \neq \mathbf{NC}_2$ , since an equality would collapse all of the hierarchy to  $\mathbf{NC}_1$ , including  $\mathbf{NC}_{2i}$  (and then we would have  $\mathbf{NC}_1 = \mathbf{L} = \mathbf{L}_i = \mathbf{NL}_i = \mathbf{NC}_{2i}$ , contradicting the Space Hierarchy Theorem).

2. (a) Fix an  $x \in \{0, 1\}^n$  and a  $y \in \{0, 1\}^k$ . Imagine that we have already chosen  $M$ . In order to have  $h_{M,b}(x) = y$ , we must have  $Mx + b = y$  or equivalently  $y - Mx = b$ . This happens with probability exactly  $2^{-k}$  since  $b$  is chosen uniformly from  $\{0, 1\}^k$ .  
For the second part, we know that  $x_1 \neq x_2$ . Thus there must be a position  $i$  in which they differ. WLOG, assume  $(x_1)_i = 1$  and  $(x_2)_i = 0$ . Imagine that we have already chosen all of  $M$  except for the  $i$ -th column, and denote by  $M'$  the matrix  $M$  with 0s in the  $i$ -th column. Let us denote by  $a \in \{0, 1\}^k$  our choice of the  $i$ -th column of  $M$ . Note that  $h_{M,b}(x_1) = M'x_1 + a + b$  and  $h_{M,b}(x_2) = M'x_2 + b$ . Thus we are interested in the probability that  $a + b = y_1 - M'x_1$  and  $b = y_2 - M'x_2$ . Since each of  $a$  and  $b$  are chosen uniformly and independently from  $\{0, 1\}^k$ , this happens with probability  $2^{-2k}$ . More precisely, there is a  $2^{-k}$  chance of choosing  $b$  equal to the fixed vector  $y_2 - M'x_2$ , and then *given the choice of  $b$* , there is a  $2^{-k}$  chance of choosing  $a$  equal to  $y_1 - M'x_1 - b$ .

- (b) Here is a 2-round **AM** protocol for **LARGESSET**. The common input is  $(C, k)$ .
- Arthur picks a random  $y$  and a random  $k \times n$  matrix  $M$  and  $b \in \{0, 1\}^k$  as above and sends them to Merlin.
  - Merlin replies with an  $x \in \{0, 1\}^n$ .
  - Arthur accepts iff  $C(x) = 1$  and  $h_{M,b}(x) = y$ .

We have to show completeness and soundness for this protocol. For completeness, set  $A = \{x : C(x) = 1\}$ , and observe that if  $|A| \geq 3(2^k)$  then the inequality from the problem statement gives us:

$$\Pr_{M,b,y} [\exists x \in A \ h_{M,b}(x) = y] \geq 1 - \frac{2^k}{|A|} \geq \frac{2}{3}.$$

Thus given a YES instance, with probability at least  $2/3$ , Merlin has a reply that will cause Arthur to accept.

For soundness, again set  $A = \{x : C(x) = 1\}$ , and observe that if  $|A| \leq (2^k)/3$  then from part (a), we have that for each fixed  $x \in A$ ,

$$\Pr_{M,b,y} [h_{M,b}(x) = y] = 2^{-k}.$$

Taking a union bound over all  $x \in A$ , we get

$$\Pr_{M,b,y} [\exists x \in A \ h_{M,b}(x) = y] \leq 2^{-k}|A| \leq \frac{1}{3}.$$

Thus given a NO instance, the probability that Merlin has a reply that will cause Arthur to accept is at most  $1/3$ .

Finally, apply the transformation from Problem Set 6, Problem 3, to this protocol to achieve perfect completeness.

3. Let  $L$  be a language in **PSPACE**, and let  $x$  be an input of length  $n$ . Using the given fact, together with the assumption that **PSPACE** has polynomial-size circuits, there is a polynomial size circuit  $C$  that computes the (honest) prover's messages as a function of  $x$  and the messages seen so far, in the **IP** protocol for  $L$ .

We need to describe a **MA** protocol for  $L$ . We have Merlin send the circuit  $C$  in the first round. Then Arthur simulates the **IP** protocol for  $L$  with input  $x$ , evaluating  $C$  to determine the prover's messages at each step. This entails flipping polynomially many coins, and evaluating the circuit  $C$  polynomially many times. In the end Arthur accepts if the Verifier he is simulating would have accepted.

Now, if  $x$  is in  $L$ , then there exists a Merlin message that will cause Arthur to accept with probability at least  $2/3$  – namely, the circuit that correctly computes the Prover messages in the **IP** protocol for  $L$ . On the other hand, if  $x \notin L$ , then no matter what  $C$  is sent in the first round, Arthur will reject with probability at least  $2/3$ , because of the soundness guarantee for the **IP** protocol. I.e., the evaluations of any circuit  $C$  correspond to *some* (possibly dishonest) prover, and we know that when  $x \notin L$ , no prover can cause the Verifier to accept with more than  $1/3$  probability.

This shows that **PSPACE**  $\subseteq$  **MA**. We know that **MA**  $\subseteq$  **PSPACE** unconditionally, so we conclude that under the assumption **PSPACE**  $\subseteq$  **P/poly**, we have **PSPACE** = **MA**.

4. (a) For a language  $L \in \mathbf{S}_2^P$ , we have

$$\begin{aligned} x \in L &\Rightarrow \exists y \forall z (x, y, z) \in R \\ x \notin L &\Rightarrow \exists z \forall y (x, y, z) \notin R \Rightarrow \forall y \exists z (x, y, z) \notin R \end{aligned}$$

Thus  $L \in \Sigma_2^P$ . We also have:

$$\begin{aligned} x \in L &\Rightarrow \exists y \forall z (x, y, z) \in R \Rightarrow \forall z \exists y (x, y, z) \in R \\ x \notin L &\Rightarrow \exists z \forall y (x, y, z) \notin R \end{aligned}$$

and so  $L \in \Pi_2^P$ . We conclude that  $\mathbf{S}_2^P \subseteq (\Sigma_2^P \cap \Pi_2^P)$ .

- (b) Let  $L$  be an arbitrary language in  $\mathbf{P}^{\mathbf{NP}}$  and let  $M$  be an oracle Turing Machine that decides  $L$  in time  $n^c$  for some constant  $c$ . Fix an input  $x$ . Without loss of generality we standardize  $M$  so that its oracle is SAT, and all of its oracle queries are 3-CNF formulas with  $m$  variables.

We describe the behavior of two machines  $M_1$  and  $M_2$  that run in polynomial time; these are then converted into the circuits  $C_1$  and  $C_2$  that the reduction produces from  $x$ . Machine  $M_1$  simulates machine  $M$  on input  $x$ , until  $M$  makes an oracle query:  $\phi \in \text{SAT}?$ . At this point  $M_1$  consults its input  $y$ , and reads  $m + 1$  bits of  $y$ . If the first bit is 0, it checks if the remaining  $m$  bits are a satisfying assignment to  $\phi$ ; if they are it continues simulating  $M$  as if  $M$  had received a “yes” answer to its query, otherwise it rejects. If the first bit is 1, it discards the remaining  $m$  bits, and continues simulating  $M$  as if  $M$  had received a “no” answer to its query. We continue in this fashion, reading successive  $(m - 1)$ -bit segments of  $y$  as (our simulation of)  $M$  encounters successive oracle queries. We stop when  $M_1$  has simulated  $|x|^c$  steps of  $M$ , at which point it accepts.

Machine  $M_2$  does exactly the same thing as  $M_1$ , except that it accepts at the end iff  $M$  would have accepted at this point. Note that depending on  $y$ , this may or may not agree with what  $M^{\text{SAT}}$  actually does on input  $x$ . However, we claim that the *lexicographically first*  $y$  that  $M_1$  accepts causes  $M_2$  to correctly simulate  $M^{\text{SAT}}$  on input  $x$ . This is true because at each query  $\phi$ , the lexicographically first  $m + 1$  bits that will cause  $M_1$  to continue its simulation are either (1) 0 followed by the lexicographically first satisfying assignment to  $\phi$  if  $\phi \in \text{SAT}$ , or (2) 1 followed by all zeros if  $\phi \notin \text{SAT}$ . In case (1) our simulation proceeds as if it received a “yes” answer to the query and in case (2) our simulation proceeds as if it received a “no” answer; in both cases this correctly simulates  $M^{\text{SAT}}$ .

We conclude that  $M_2$  accepts the lexicographically first  $y$  accepted by  $M_1$  iff  $M^{\text{SAT}}$  accepts  $x$ , as required.

We also should argue that the problem is in  $\mathbf{P}^{\mathbf{NP}}$ , but this is easy, because we can do a binary search (using the NP oracle) to identify the lexicographically first  $y$  accepted by  $C_1$ , and then plug it into  $C_2$ .

- (c) We argue that LEX-FIRST-ACCEPTANCE is in  $\mathbf{S}_2^P$ . Let  $(C_1, C_2)$  be an instance of LEX-FIRST-ACCEPTANCE. Define the function  $f(y, y')$  to be  $C_2(y_{\min})$  where  $y_{\min}$  is the lexicographically first among  $y, y'$  that  $C_1$  accepts; or 0 if  $C_1(y) = C_1(y') = 0$ . We claim that

$$\begin{aligned} (C_1, C_2) \in \text{LEX-FIRST-ACCEPTANCE} &\Rightarrow \exists y \forall y' f(y, y') = 1 \\ (C_1, C_2) \notin \text{LEX-FIRST-ACCEPTANCE} &\Rightarrow \exists y' \forall y f(y, y') = 0 \end{aligned}$$

This is easily seen by taking  $y$  to be the lexicographically first string accepted by  $C_1$  in the first case, and  $y'$  to be the lexicographically first string accepted by  $C_1$  in the second case. Since LEX-FIRST-ACCEPTANCE is  $\mathbf{P}^{\mathbf{NP}}$ -complete, we conclude that  $\mathbf{P}^{\mathbf{NP}} \subseteq \mathbf{S}_2^{\mathbf{P}}$ .

- (d) By error reduction, we may assume that for every language  $L$  in  $\mathbf{MA}$  there is a language  $R$  in  $\mathbf{P}$  for which

$$\begin{aligned} x \in L &\Rightarrow \exists y \Pr_z[(x, y, z) \in R] = 1 \\ x \notin L &\Rightarrow \forall y \Pr_z[(x, y, z) \in R] < 2^{-|y|}. \end{aligned}$$

We claim that

$$\begin{aligned} x \in L &\Rightarrow \exists y \forall z (x, y, z) \in R \\ x \notin L &\Rightarrow \exists z \forall y (x, y, z) \notin R, \end{aligned}$$

which implies that  $L \in \mathbf{S}_2^{\mathbf{P}}$  as required. The first part is obvious from the definitions. For the second part, observe that

$$\forall y \Pr_z[(x, y, z) \in R] < 2^{-|y|}$$

implies (by the union bound)

$$\Pr_z[\exists y (x, y, z) \in R] < 2^{|y|} 2^{-|y|} = 1. \quad (0.1)$$

This implies  $\exists z \forall y (x, y, z) \notin R$  as required.

- (e) Given a language  $L \in \mathbf{BPP}$ , we can use strong error reduction to produce a probabilistic polynomial time TM  $M$  for which:

$$\begin{aligned} x \in L &\Rightarrow \Pr_y[M(x, y) = 1] \geq 1 - \frac{2^{|y|^{1/3}}}{2^{|y|}} \\ x \notin L &\Rightarrow \Pr_y[M(x, y) = 0] \geq 1 - \frac{2^{|y|^{1/3}}}{2^{|y|}}. \end{aligned}$$

We split  $y$  into two equal-length substrings  $y = u \circ v$ . Our predicate  $R$  is simply  $R(x, u, v) = M(x, u \circ v)$ .

Now, if  $x \in L$ , then it must be that  $\exists u \forall v R(x, u, v) = 1$ , for if not, then  $\forall u \exists v R(x, u, v) = 0$  which implies that  $M(x, y) = 0$  for at least  $2^{|y|/2} \gg 2^{|y|^{1/3}}$  values of  $y$ , a contradiction. Similarly, if  $x \notin L$ , then it must be that  $\exists v \forall u R(x, u, v) = 0$ , for if not, then  $\forall v \exists u R(x, u, v) = 1$  which implies that  $M(x, y) = 1$  for at least  $2^{|y|/2} \gg 2^{|y|^{1/3}}$  values of  $y$ , a contradiction.

We conclude that  $L \in \mathbf{S}_2^{\mathbf{P}}$  and therefore  $\mathbf{BPP} \subseteq \mathbf{S}_2^{\mathbf{P}}$  as required.

Another solution is to observe that  $\mathbf{BPP}$  is contained in (2-sided error)  $\mathbf{MA}$  and apply the previous part!

- (f) The following notation will be useful: given a circuit  $C$  with a single Boolean output, let  $\tilde{C}$  be the circuit derived from  $C$  that uses  $C$  as if it were a circuit for SAT to actually find

a satisfying assignment (via the self-reducibility of SAT). If at any point in the repeated applications of  $C$ , there is an inconsistent answer,  $\tilde{C}$  outputs some fixed string, say, the all-zeros string. So,  $\tilde{C}$  has as many outputs as inputs, and  $|\tilde{C}| \leq \text{poly}(|C|)$ , and if  $C$  is a circuit correctly computing SAT, then  $\tilde{C}$  will correctly output a satisfying assignment if there is one.

Let  $L$  be a language in  $\Pi_2^P$ , so we have

$$\begin{aligned} x \in L &\Rightarrow \forall y \exists z (x, y, z) \in R \\ x \notin L &\Rightarrow \exists y \forall z (x, y, z) \notin R \end{aligned}$$

for some language  $R \in \mathbf{P}$ . Observe that the language  $L' = \{(x, y) : \exists z (x, y, z) \in R\}$  is in  $\mathbf{NP}$ , and so given a pair  $(x, y)$  we can use a procedure that solves SAT and actually returns a satisfying assignment if there is one to find  $z$  for which  $(x, y, z) \in R$  if such a  $z$  exists.

Define  $R'$  to be the language consisting of exactly the triples  $(x, C, y)$  for which using  $\tilde{C}$ , we obtain a  $z$  for which  $(x, y, z) \in R$ . Notice that  $R'$  can be evaluated in polynomial time.

We are assuming that SAT has polynomial-size circuits. If  $x \in L$ , then there exists a circuit  $C$  (the one that computes SAT) for which for all  $y$ ,  $\tilde{C}$  will successfully find a  $z$  that causes  $R'$  to accept. Thus  $x \in L \Rightarrow \exists C \forall y (x, C, y) \in R'$ .

If  $x \notin L$ , then there is some  $y^*$  for which  $\forall z (x, y^*, z) \notin R$ . Thus for all  $C$ ,  $(x, C, y^*) \notin R'$ , because no matter what  $z$  we find using  $\tilde{C}$ , it will not be the case that  $(x, y^*, z) \in R$ . Therefore  $x \notin L \Rightarrow \exists y \forall C (x, C, y) \notin R'$ . We conclude that  $L \in \mathbf{S}_2^P$ .

We have shown that  $\Pi_2^P \subseteq \mathbf{S}_2^P$ . Since  $\mathbf{S}_2^P$  is closed under complement, we also have that  $\Sigma_2^P \subseteq \mathbf{S}_2^P$ . Using part (a), we have  $\Pi_2^P = \Sigma_2^P = \mathbf{S}_2^P$ , and so the PH collapses to  $\Sigma_2^P = \mathbf{S}_2^P$  as required.