

# Computational Complexity Theory

Lecture notes by Henrik Hald Nørgaard

Lecture 13, 11/10 2006

## 1 Randomization and Promise Problems

We define the classes of promise problems  $prRP$  and  $prBPP$  that comprise our notion of randomized computation. In the definitions below  $U$  is the universe of instances for which a promise is kept and  $L$  is the language decided by the promise problems.

**Definition 1 (prRP)** A promise problem  $(U, L)$  is in  $prRP$  iff there exists a language  $L' \in P$  and a polynomial  $p$  such that  $\forall x \in U$ :

$$x \in L \Leftrightarrow Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] \geq \frac{1}{2} \quad (1)$$

$$x \in U - L \Leftrightarrow Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] = 0 \quad (2)$$

**Remark:** For  $x \in U - L$  the condition above may also be phrased as:

$$\forall y \in \{0,1\}^{p(|x|)} : \langle x, y \rangle \in \overline{L'} \quad (3)$$

**Definition 2 (prBPP)** A promise problem  $(U, L)$  is in  $prBPP$  iff there exists a language  $L' \in P$  and a polynomial  $p$  such that  $\forall x \in U$ :

$$x \in L \Leftrightarrow Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] \geq \frac{2}{3} \quad (4)$$

$$x \in U - L \Leftrightarrow Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] \leq \frac{1}{3} \quad (5)$$

### 1.1 Constants

The constants  $\frac{1}{2}$ ,  $\frac{2}{3}$  and  $\frac{1}{3}$  in the above definitions are not important. Any constant may be achieved by repeating the algorithm. For  $prRP$  this is seen by noticing that no matter how many times we repeat the algorithm for  $x \in U - L$ , we will

never get a positive answer (that is, find a  $y$  such that  $\langle x, y \rangle \in L'$ ). And for  $x \in L$  repeating the algorithm e.g. 3 times:

$$(x, y_1 y_2 y_3) \in L'' \Leftrightarrow \exists i : \langle x, y_i \rangle \in L' \quad (6)$$

amplifies the probability of success to:

$$Pr_{y \in \{0,1\}^{3p(|x|)}}[\langle x, y \rangle \in L''] \geq 1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8} \quad (7)$$

In a similar way it can be proved, using a polynomial number of repetitions of the algorithm, that the following bounds on finding a witness for  $x \in L$  gives the same class of promise problems:

$$Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] \geq \frac{1}{|x|^k} \quad (8)$$

$$Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] \geq 1 - 2^{-|x|^k} \quad (9)$$

Similar results are true for *prBPP*. Here the amplified algorithm should return the same result as the majority of the results of a polynomial number of repetitions of the algorithm. Using Chernoff bounds from probability theory, the inequalities in the definition of *prBPP* above can be replaced by either of the following pairs of inequalities:

Bounded away from  $\frac{1}{2}$ :

$$x \in L \Leftrightarrow Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] \geq \frac{1}{2} + \frac{1}{|x|^k} \quad (10)$$

$$x \in U - L \Leftrightarrow Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] \leq \frac{1}{2} - \frac{1}{|x|^k} \quad (11)$$

Bounded away from 0:

$$x \in L \Leftrightarrow Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] \geq 1 - 2^{-|x|^k} \quad (12)$$

$$x \in U - L \Leftrightarrow Pr_{y \in \{0,1\}^{p(|x|)}}[\langle x, y \rangle \in L'] \leq 2^{-|x|^k} \quad (13)$$

## 2 Approximate Integration

The approximate integration problem defined below captures the way randomization is used in practice and it can e.g. be used to solve the problem of Monte Carlo integration. It is an example of a PAC-algorithm (PAC = Probably Approximately Correct).

**Definition 3 (Approximate integration problem (AI))**

Given a circuit  $C : \{0, 1\}^r \rightarrow \{0, 1\}$  and  $1^k$ , estimate

$$\Pr_{x \in \{0,1\}^r}(C(x) = 1) \tag{14}$$

with additive error at most  $\frac{1}{k}$ .

**Theorem 1** *AI can be solved by a polynomial time algorithm iff  $prBPP = prP$ .*

**Remark:** The condition  $prBPP = prP$  implies that  $BPP = P$ , but it is not known if  $BPP = P$  can replace  $prBPP = prP$  in the above theorem. Furthermore, there is a result stating, that there exists an oracle  $A$  such that:

$$P^A = BPP^A \wedge prP^A \neq prBPP^A \tag{15}$$

To prove the theorem, we need the following 2 results from the previous lecture:

**Theorem 2** *The following promise problem is complete for  $prRP$ : Given a circuit  $C : \{0, 1\}^r \rightarrow \{0, 1\}$  and a promise that either:*

$$\Pr_{x \in \{0,1\}^r}(C(x) = 1) \geq \frac{1}{2} \tag{16}$$

or

$$\Pr_{x \in \{0,1\}^r}(C(x) = 1) = 0 \tag{17}$$

determine which is the case.

**Remark:** This promise problem is sometimes characterized as 'finding hay in a haystack', because of the many witnesses for the case where

$$\Pr_{x \in \{0,1\}^r}(C(x) = 1) \geq \frac{1}{2} \tag{18}$$

On the other hand problems in  $NP$ , where we are only guaranteed 1 witness for each problem instance, are characterized as 'finding a needle in a haystack'.

**Theorem 3** *The following promise problem is complete for  $prBPP$ : Given a circuit  $C : \{0, 1\}^r \rightarrow \{0, 1\}$  and a promise that either:*

$$\Pr_{x \in \{0,1\}^r}(C(x) = 1) \geq \frac{2}{3} \tag{19}$$

or

$$\Pr_{x \in \{0,1\}^r}(C(x) = 1) \leq \frac{1}{3} \tag{20}$$

determine which is the case.

**Remark:** For both promise problems above the first possibility is called the 'YES'-instance and the second possibility the 'NO'-instance.

**Proof of Theorem 1:**

$\Rightarrow$ : Call the complete problem for  $prBPP$  from theorem 3 for  $X$ . It is enough to show that  $X \in prP$ . Use AI with the circuit from an instance of  $X$  and  $k = 6$ . The polynomial algorithm for AI determines  $P(C(x) = 1)$  within additive error  $\frac{1}{6}$  and this is sufficient to tell whether  $C$  is a 'YES'- or a 'NO'-instance of  $X$ . This proves that  $X \in prP$ .

$\Leftarrow$ : Assume given circuit  $C$  and integer  $k$ . The condition  $prBPP = prP$  is equivalent to  $X \in prP$ . This means that we can express the sought after polynomial algorithm for AI as an algorithm in  $P^X$ .

Define  $\epsilon = \frac{1}{k}$ . Here is the algorithm for AI:

*for*( $\alpha = 0$  to 1 step  $\frac{\epsilon}{2}$ )  
*if*( $Test(C, \alpha, \epsilon)$ ) return  $\alpha$

We still need to specify the subroutine  $Test$ . Here is first a randomized version of the subroutine:

Test:

Throw  $1000(\frac{1}{\epsilon})^2$  darts and return 'YES' iff the fraction of darts for which  $C(\text{dart}) = 1$  is within  $\frac{\epsilon}{2}$  from  $\alpha$ . Here 'throwing a dart' means to choose an  $x \in \{0, 1\}^r$  at random.

Using Chernoff bounds for the sampling above, it can be shown that:

1. If  $\alpha$  is within  $\frac{\epsilon}{2}$  from the correct value of  $Pr_{x \in \{0,1\}^r}(C(x) = 1)$ ,  $Test$  will return 'YES' with high probability.
2. If  $\alpha$  has distance more than  $\epsilon$  from the correct value of  $Pr_{x \in \{0,1\}^r}(C(x) = 1)$ ,  $Test$  will return 'YES' with low probability.

Now turn the randomized procedure for  $Test$  into a circuit with the  $1000(\frac{1}{\epsilon})^2 r$  random bits used above as inputs. This circuit can be constructed in polynomial time in  $|C|$  and  $k$ . An oracle for  $X$  will be able to distinguish the two cases above. The complete algorithm for AI will return an  $\alpha$  by property 1 of  $Test$ , as we move in steps of  $\frac{\epsilon}{2}$ . The answer will be within  $\epsilon$  from the correct answer by the second property of  $Test$ .

### 3 $prRP$ and $prBPP$

In this section we will prove a theorem stating a relation between  $prRP$  and  $prBPP$ .

**Theorem 4 (Buhrmann, Fortnow, '98)**  $prBPP = prRP^{prRP}$

**Proof of theorem:**

$prRP^{prRP} \subseteq prBPP$ : Let  $t$  denote the running time of an  $prRP$ -algorithm using an  $prRP$ -oracle. The oracle can be replaced by an  $prRP$ -algorithm whose amplified error probability is  $< \frac{1}{10t}$  by the results in 1.1. This algorithm will give the correct replies to all the questions to the oracle with probability at least  $\frac{9}{10}$ , as there are at most  $t$  questions to the oracle.

Now run the  $prRP$ -algorithm as if the replacement for the oracle is always correct. Again by the results in section 1.1 we may assume, that the base  $prRP$ -algorithm has an error probability less than  $\frac{1}{10}$ . The total algorithm gives the correct answer as output with probability at least  $1 - 2\frac{1}{10} = \frac{4}{5}$ . The algorithm has not got one-sided errors anymore, as the wrong answers from the simulated oracle may imply wrong answers for the case where the input is from  $U - L$ .

$prBPP \subseteq prRP^{prRP}$ : It is enough to show that the complete problem  $X$  for  $prBPP$  from theorem 3 is in  $prRP^{prRP}$ . By the remarks in section 2.1 we may assume that the probabilities in the definition of  $X$  have been boosted to:

$$Pr_{x \in \{0,1\}^r}(C(x) = 1) \geq 1 - 2^{-n^k} \quad (21)$$

or

$$Pr_{x \in \{0,1\}^r}(C(x) = 1) \leq 2^{-n^k} \quad (22)$$

Here  $k$  depends on the number of random bits used. Using Chernoff bounds it is possible to show, that if the algorithm uses  $r$  random bits and it is repeated  $Cr$  times (where  $C$  is a constant), the error probability drops to  $2^{-r}$ , but now ofcourse the resulting algorithm uses  $O(r^2)$  random bits. Rephrasing this we may assume, that if the algorithm uses  $r$  random bits, the error probability is  $2^{-\sqrt{r}}$  (up to some constants that wont be important below).

We are now in a situation where a problem with either a small or large probability of success (the circuit problem from  $X \in prBPP$ ) should somehow be turned into a problem with either zero or a large probability of success (the kind of problem distinguished by the complete problem for  $prRP$  from Thm. 2, that we will use as our  $prRP$ -oracle). It seems difficult to completely remove the small probability of success and instead we will do the opposite: Turn the instance with a large probability of success into an instance with a probability of success equal to 1 and the instance with a small probability of success into another instance

with small probability of success and then negate. The two types of instances thus change roles when reducing the  $prBPP$ -problem to a  $prRP^{prRP}$ -problem.

Let  $S$  be the subset of  $\{0, 1\}^r$  containing the input sequences to the circuit  $C$  that make  $C$  output 1:

$$x \in S \Leftrightarrow C(x) = 1 \quad (23)$$

If  $S$  is 'big', we will use translations of  $S$  to cover all of  $\{0, 1\}^r$ . If  $S$  is 'small', the translations of  $S$  are still small. Translations in  $\{0, 1\}^r$  are performed using the operator

$$\oplus : \{0, 1\}^r \times \{0, 1\}^r \rightarrow \{0, 1\}^r \quad (24)$$

that performs a componentwise XOR of the bitsequences. This operator satisfies:

$$a \oplus b = c \Leftrightarrow c \oplus a = b \Leftrightarrow a \oplus c = b \quad (25)$$

Thus fixing one component gives us a map on  $\{0, 1\}^r$  ( $a \oplus \cdot$  or  $\cdot \oplus b$ ) that is bijective.

Now pick  $a_1, a_2, \dots, a_r \in \{0, 1\}^r$  at random and define the circuit  $C'$  by:

$$C'(x) = \neg\left(\bigvee_{i=1}^r C(x \oplus a_i)\right) \quad (26)$$

Ask the  $prRP$ -oracle if  $S' := C'^{-1}(1)$  is empty or almost everything. Then return the opposite of the answer from the oracle.

This algorithm above is in  $prRP^{prRP}$  and it will answer correctly given the promises for the circuit  $C$  given in the problem  $X$ :

First assume that  $P(C(x) = 1) \leq 2^{-\sqrt{r}}$ : Observe that

$$C'^{-1}(0) = S \oplus a_1 \cup S \oplus a_2 \cup \dots \cup S \oplus a_r \quad (27)$$

Thus

$$P(C'(x) = 0) \leq \sum_{i=1}^r P(x \in S \oplus a_i) = \sum_{i=1}^r P(x \in S) = \sum_{i=1}^r P(C(x) = 1) \leq r2^{-\sqrt{r}} \quad (28)$$

where we use, that  $\cdot \oplus a_i$  is a bijection on  $\{0, 1\}^r$ . This means that

$$P(C'(x) = 1) \geq 1 - r2^{-\sqrt{r}} > \frac{1}{2} \quad (29)$$

So  $C'$  can be identified by the  $prRP$ -oracle as a 'YES'-instance and thus we answer 'NO', which is always correct, independently of the choice of  $a_i$ 's in  $\{0, 1\}^r$ .

Now suppose  $P(C(x) = 1) \geq 1 - 2^{-\sqrt{r}}$ . We will show that with high probability (for the random choice of  $a_1, a_2, \dots, a_r$ ),  $C'$  has no satisfying assignment:

$$P_a(\forall x : C'(x) = 0) = P_a(\forall x \in \{0, 1\}^r \quad \exists i : x \oplus a_i \in S) \quad (30)$$

$$= 1 - P_a(\exists x \in \{0, 1\}^r \quad \forall i : x \oplus a_i \in \overline{S}) \quad (31)$$

$$\geq 1 - 2^r P_a(\forall i : x \oplus a_i \in \overline{S}) \quad (32)$$

$$= 1 - 2^r P_a(\forall i : a_i \in \overline{x \oplus S}) \quad (33)$$

$$= 1 - 2^r (2^{-\sqrt{r}})^r \quad (34)$$

$$= 1 - 2^{r - \frac{3}{2} + r} > \frac{1}{2} \quad (35)$$

Thus with probability at least  $\frac{1}{2}$ ,  $C'$  can be identified by the  $prRP$ -oracle as a 'NO'-instance, in which case we answer 'YES'.

**Remark:** The above proof was originally used by Sipser and Lautemann to prove that  $BPP \subseteq \Sigma_2^p$ . This result is a corollary to our theorem: As  $RP \subseteq NP$ , we have that  $prRP \subseteq prNP$ . Thus by the above theorem:  $prBPP \leq prNP^{prNP}$  and in particular we have Sipser and Lautemanns result.

**Corollary 1**  $prRP = prP \Rightarrow prBPP = prP$

**Proof:** If  $prRP = prP$ ,  $prRP^{prRP} = prP^{prP} = prP$ .

**Remark:** It is not known if

$$P = RP \Rightarrow P = BPP \quad (36)$$

but there exists an oracle  $A$  such that:

$$P^A = RP^A \wedge P^A \neq BPP^A \quad (37)$$

**Remark:** Notice that

$$prBPP^{prBPP} = prBPP \quad (38)$$

This also holds without promises:

$$BPP^{BPP} = BPP \quad (39)$$

To see this use the same idea as in the first part of the proof of theorem 4, replacing an  $prBPP$ -oracle with an  $prBPP$ -approximation depending on running time of the base algorithm.

### 3.1 Search versions of promise problems

We have considered the following levels of problems for randomized algorithms:

1. Identity testing
2. Monte Carlo integration
3. Approximation algorithms and heuristics

To derandomize problems from 1. it is enough that  $P = RP$ . To derandomize problems from 2, it is enough that  $prP = prRP$  and this implies  $P = RP$ . To derandomize problems from 3 (that is: find a corresponding algorithm that doesn't use randomization and that gives the same approximation bounds or that works well on the same instances), it is enough to be able to solve the following problem:

*H*: Given a circuit  $C : \{0,1\}^r \rightarrow \{0,1\}$  with  $P(C(x) = 1) \geq \frac{1}{2}$ , find  $x$  such that  $C(x) = 1$  efficiently.

There is a parallel between the difference between solving decision problems and search problems in  $NP$  on one side and the question of  $prP = prRP$  and solving *H* on the other side. But the method of using binary search to solve the search problem with a solution to the decision problem does not work in the latter case, as we can't be sure that the promise will be kept during the binary search.