Homework 23

Joe Baker, Brett Schreiber, Brian Knotten

March 19, 2018

39

a

The interactive protocol to show that coNP $\subseteq$ IP has a problem when a boolean formula ϕ with variables $x_1, ..., x_n$ is arithmetized and products are expanded. When ϕ is arithmetized to create function f, expanding the product will multiply the variables in such a way that the exponent of each variable can double in size, making the size of f exponential. Since the verifier runs in polytime, it may not be able to read the function f.

To solve this problem, assume that ϕ has the property that its variables $x_1, ..., x_n$ occur in order in ϕ and that for each x_i there is at most a single universal quantifier for variable x_j appearing before the last occurrence of x_i. Now when ϕ is arithmetized, the resulting function f will have a maximum degree of size $O(n)$.

To see how the maximum degree is $O(n)$, consider how the f will have the product for each variable expanded. At any point in the expansion we will have one of the two following scenarios:

1. $...\Pi^1_{x_i=0} f_{x_i} ... \Pi^1_{x_j=0} f_{x_j} ...$
2. $...\Pi^1_{x_{j'}=0} f_{x_{j'}} ... \Pi^1_{x_i=0} f_{x_i} ... \Pi^1_{x_j=0} f x_j ...$

In case 1, when each product is expanded from right-to-left we will have:

$$\left(...\Pi^1_{x_i=0} f_{x_i} ...\right) f_{x_j=0} f_{x_j=1}$$

and the part where $f_{x_j=0}$ will prevent the degrees of the variables from doubling for x_j. Additionally, any variables x_k with $k > j$ will not be doubled in future expansions because they are not in future f by definition (with the exception mentioned in case 2). By similar logic, variables x_h with $h < i$ will not have been doubled in previous expansions.

In case 2, we will have the same situation, but there will be one extra occurrence of the variable x_j, so the degree of x_j will increase once from 1 to 2.

Finally, each of the n variables will have a max degree of 2, so the degree of f will be $O(n)$ and the verifier can validate the prover's response.

b

Every size n boolean formula ϕ can be transformed into a boolean formula ψ with the sorted-variable property mentioned above and size $O(n^2)$.

To transform ϕ into ψ apply the following procedure from right to left until ψ meets the sorted-variable property:

Find the rightmost "fragment" of ψ of the form: $\forall x_i ... \forall x_j ... \forall x_{i'} f(x_i, ...)$ where $i' > j > i$
Replace the fragment with the following: $\forall x_i \exists y_i$ s.t. $(y_i = x_i) \land ... \forall x_j ... \forall x_{i'} f(y_i, ...)$

Each of the n variables can be out of place out of place (meaning there is a fragment) less than n times, so there will be at most $O(n^2)$ fragments to fix. Since fixing each fragment adds 1 new variable (without adding a new fragment, because a fixed fragment adds an $\exists$ and a variable in correct order) , the resulting formula ψ will have a maximum size of $O(n^2)$.

40

Alice and Bob both know a secret key k, a one time pad, which is the same bitlength as the message m. Carol does not know any information about this secret key, and her best course of action for determining a random bit of the key is to guess. So Carol has probability $1/2$ of guessing any bit of the key using a randomized polynomial algorithm A.

The xor operation has the property such that $m \oplus k = c$, where c is the ciphertext, and $c \oplus k = m$. Moreover, the xor function is one-to-one, meaning that no other k can derive c from m and vice versa.

Alice can encrypt her message m using a one time pad k to get c using xor. Bob can similarly decrypt c into m using xor. Since no other k can derive m, and since Carol cannot determine any bit of k with probability greater than $1/2$, it follows that Carol cannot derive any bit of m with the same probability greater than $1/2$. So Alice and Bob have computational security on m.

41

Assume $P = NP$, let $f : \{0,1\}^* \to \{0,1\}^*$ be a one-way function, and let y be an output of f.
Let A be a non-deterministic poly-time Turing Machine that tries every possible x until $f(x) = y$ and then returns x i.e. A solves the problem of inverting f.

Because A solves the problem of inverting a one-way function in poly-time, the problem is in NP. By our assumption $P = NP$, so there the problem is also in P and exists an efficient algorithm for inverting one-way functions. Therefore one-way functions do not exist.