Interactive Proof Systems

Thomas Haines

ANU Logic Summer School 2023

Contents

Chapter 1

Background on Non-Interactive Proof Systems

As the title of this course suggests, we are going to be covering interactive proof systems and specifically the complexity thereof; the suggested background reading was "Computational Complexity - A Conceptual Perspective" by Oded Goldreich.

In the first lecture, or so, I'm going to start by covering some results from complexity theory of non-interactive proof systems.

1.1 $\mathcal{P}$ vs $\mathcal{NP}$

I assume that most of you have heard of the $\mathcal{P}$ vs $\mathcal{NP}$ problem.

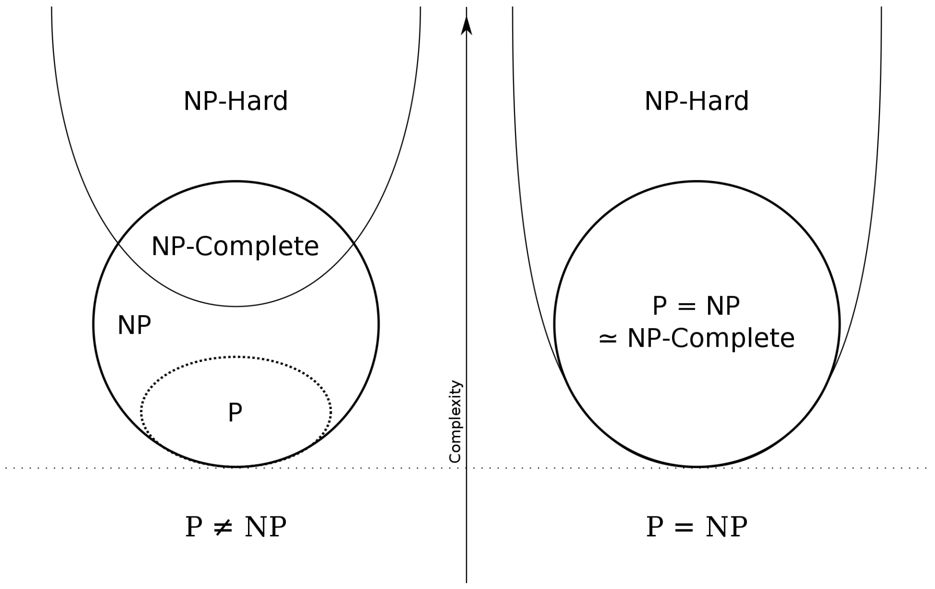


Figure 1.1: By Behnam Esfahbod, CC BY-SA 3.0

Definition 1.1.1: Search Problem

Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ and $R(x) := \{y : (x,y) \in R\}$ denote the set of solutions for the instance x. An algorithm $f : \{0,1\}^* \rightarrow \{0,1\}^* \cup \{\perp\}$ solves the search problem of R if for every x the following holds: if $R(x) \neq \emptyset$ then $f(x) \in R(x)$ otherwise $f(x) =\perp$.

Definition 1.1.2: Decision Problem

Let $S \subseteq \{0,1\}^*$. An algorithm $f : \{0,1\}^* \rightarrow [0,1]$ solves the decision problem of S if for every x it holds that $f(x) = 1$ if and only if $x \in S$.

Definition 1.1.3: Polynomial Time Algorithm

Having fixed a computational model and a specific algorithm A, we can define $t_A : \{0,1\}^* \rightarrow \mathbb{N}$ if for every x, A halts after exactly $t_A(x)$ steps. We can then define $T_A : \mathbb{N} \rightarrow \mathbb{N}$ by $T_A(n) := max_{x\in\{0,1\}^n}\{t_A(x)\}$.
An algorithm is polynomial bounded, or equivalent polynomial-time, if there exists a polynomial p such that for all n, $T_A(n) \leq p(n)$.

Definition 1.1.4: $\mathcal{P}$

- A decision problem $S \subseteq \{0,1\}^*$ is efficiently solvable if there exists a polynomial time algorithm A such that, for every x, it holds that $A(x) = 1$ if and only if $x \in S$.
- We denote by $\mathcal{P}$ the class of decision problem that are efficiently solvable.

The consideration of polynomial bounded is grounded in the following considerations:

Theory: *The set of polynomial algorithms is closed under addition, multiplication, and functional composition. This ensures natural composition of algorithms preserves efficiency.*

Practice: *Past experience shows that every natural problem which can be solved in polynomial time also has a "reasonable efficient" algorithm.*

.

Definition 1.1.5: $\mathcal{NP}$

A decision problem $S \subseteq \{0,1\}^*$ has an efficiently verifiable proof system if there exists a polynomial p and a polynomial-time algorithm V such that the following two conditions hold:

1. Completeness: For ever $x \in S$, there exists y of length at most $p(|x|)$ such that $V(x,y) = 1$
2. Soundness: For every $x \notin S$ and every y, it holds that $V(x,y) = 0$

In such a case, we say that S had an $\mathcal{NP}$-proof system, and refer to V as the verification procedure (or as the proof system itself).

It's clear that if you can solve the search problem then you solve the decision problem; however under the assumption that $\mathcal{P} = \mathcal{NP}$ the decision formulation of the problems in $\mathcal{NP}$ is also equivalent to the search formalisation. I won't give the full proof but basically the decision problem of being a prefix of the witness being effectively solvable allows the polynomial time reconstruction of the witness.

Definition 1.1.6: Reducibility

A problem Π is reducible to a problem Π' if there exits a polynomial-time algorithm A such that for every function f that solves Π' it holds that A^f solves Π.

Definition 1.1.7: $\mathcal{NP}$-Hard

A $S \subseteq \{0,1\}^*$ is $\mathcal{NP}$-Hard if every set in $\mathcal{NP}$ is reducible to S.

Definition 1.1.8: $\mathcal{NP}$-Complete

A $S \subseteq \{0,1\}^*$ is $\mathcal{NP}$-Complete if it is $\mathcal{NP}$-Hard and in $\mathcal{NP}$.

The existence of $\mathcal{NP}$-Complete problems is itself amazing; the fact that many natural problems are $\mathcal{NP}$-Complete is astounding.

Claim: The 6 Million Dollar Problem

A practical algorithm for an $\mathcal{NP}$-Complete problem would allow one not only to claim the prize promised for $\mathcal{P} vs \mathcal{NP}$ but the other five open problems as well since the algorithm could find a proof these problems.

To prove $\mathcal{P} = \mathcal{NP}$ it would suffice to show an efficient algorithm for any $\mathcal{NP}$-Complete.

Conjecture 1.1.9: Church-Turing Thesis

A function can be computed by some Turing machine if and only if it can be computed by some machine of any other "reasonable and general" model of computation.

There are caveats between uniform models of computation and non-uniform which I will not touch upon. Indeed, certain uncomputable (for uniform models) functions are computable in non-uniform models.

Conjecture 1.1.10: Cobham-Edmonds Thesis

A problem has polynomial time complexity in some "reasonable and general" model of computation if and only if it has polynomial time complexity in the model of Turing machines.

Theorem 1.1.11: Scarcity of computable functions

The set of computable functions is countable, whereas the set of all functions (from strings to string) has cardinality $\aleph$.

Proof for Theorem

Since each computable function is computable by a machine that has a finite description, there is a 1-1 correspondence between computable function and set of strings and hence the natural numbers. On the other hands, there is a 1-1 correspondce between the set of Boolean functions and the set of real numbers in $[0, 1)$. This correspondence associates each real $r \in [0, 1)$ to the function $f : \mathbb{N} \to \{0, 1\}$ such that $f(i)$ is the i^{th} bit in the infinite binary expansion of r. ■

1.2 Introducing IP

Interactive proofs relax the verifier in the following ways, the versifier may use randomness and the verifier may interact with $\mathcal{P}$.

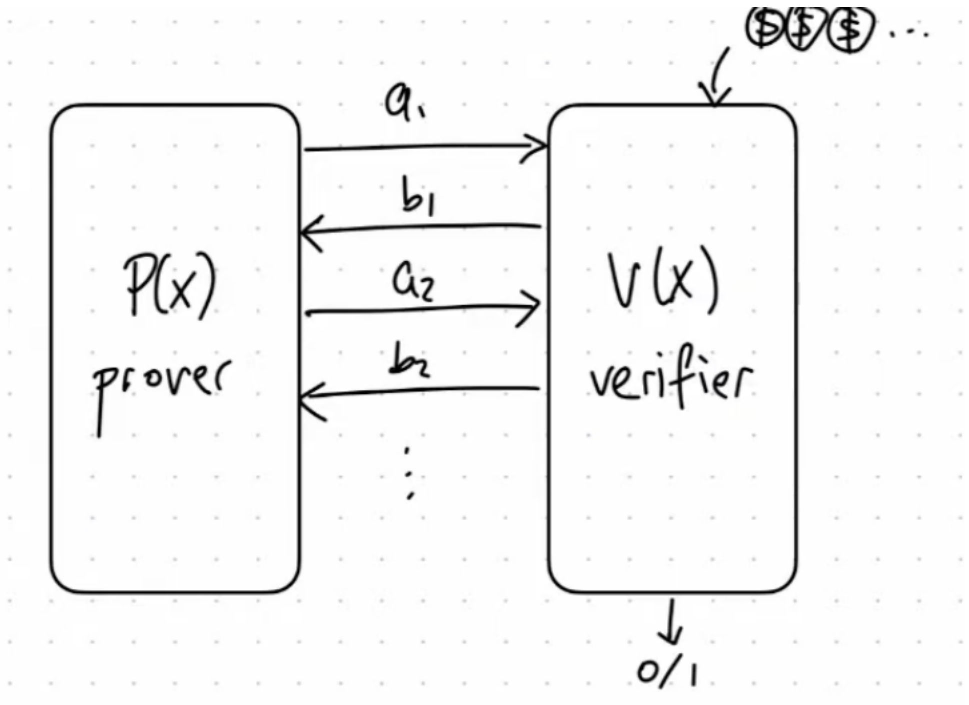


Figure 1.2: By Alessandro Chiesa

		Interaction	
		Y	N
Randomness	Y	$\mathcal{IP}$	$\mathcal{MA}$
	N	$\mathcal{NP}$	$\mathcal{NP}$

Table 1.1: Overview of Interaction and Randomness

Definition 1.2.1: Interactive Proof System

An Interactive Proof System for S is a pair of interactive algorithms (P, V), where P is unbounded and V is polynomial time, s.t.:

1. Completeness: For ever $x \in S$, $Pr[< P(x), V(x;r) >= 1] = 1$
2. Soundness: For every $x \notin S$, for every $\tilde{P}$, $Pr[< \tilde{P}, V(x;r) >= 1] \leq \frac{1}{2}$

Any noticeable gap between the probability of completeness and soundness suffices

Definition 1.2.2: Merlin Arthur $\mathcal{MA}$

Is exactly $\mathcal{NP}$ but allows the verifier to use random coins.

$\mathcal{MA}$ is equal to $\mathcal{NP}$ if Strong Pseudorandom Function exist
Confusingly $\mathcal{AM}$ is also a complexity class

Definition 1.2.3: $\mathcal{IP}$

A decision problem $S \subseteq \{0,1\}^*$ is in $\mathcal{IP}$ if it has an interactive proof system.

So Table 2.1 shows that both interaction and randomness are required to improve expressiveness.

Chapter 2

Interactive Proofs

Last lectrue we covered

- *Definitions of decision and search problems*
- *Definitions of $\mathcal{P}$ and $\mathcal{NP}$*
- *Definition of $\mathcal{IP}$*

2.1 How big is $\mathcal{IP}$?

It is immediate that $\mathcal{NP} \subseteq \mathcal{IP}$ since $\mathcal{IP}$ generalises $\mathcal{NP}$

The question is do we gain any expressive power?

2.1.1 $\mathcal{IP}$ for Graph Non-Isomorphism

Definition 2.1.1: Graph Isomorphism

Let $G_0 = (V, E_0)$ and $G_1 = (V, E_1)$ be two graphs on vertices V.
$G_0 \equiv G_1$ if $\exists$ a permutation $\pi : V \to V$ s.t. $(u, v) \in E_0 \leftrightarrow (\pi(u), \pi(v)) \in E_1$.

We can then write $G_1 = \pi(G_0)$

Definition 2.1.2: GI

GI $:= \{(G_0, G_1) | G_0 \equiv G_1\}$

Definition 2.1.3: GNI

GI $:= \{(G_0, G_1) | G_0 \not\equiv G_1\}$

Fact 2.1.4

GI $\in \mathcal{NP}$, the permutation pi is the witness and its length is quadratic in the number of vertices

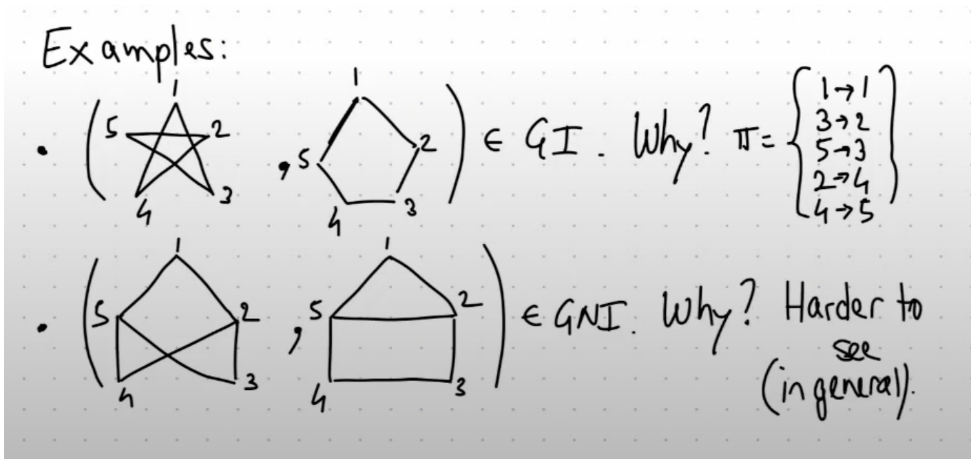

Examples:
$\in GI$. Why? $\pi = \{1\to1, 3\to2, 5\to3, 2\to4, 4\to5\}$
$\in GNI$. Why? Harder to see (in general).
1 2 3 4 5

Figure 2.1: By Alessandro Chiesa

GI ∈ $\mathcal{NP}$, GNI ∈ co$\mathcal{NP}$, not know if in $\mathcal{P}$

Definition 2.1.5: *co$\mathcal{NP}$*

$\{\{0,1\}^*\backslash S : S \in \mathcal{NP}\}$

$\mathcal{NP}$ and co$\mathcal{NP}$ are believed to be different. This aligns with intuition since $\mathcal{NP}$ essentially states the existence of a witnesses which can be checked whereas co$\mathcal{NP}$ says that all witness aren't valid. However, GNI is not believed to be coNP-complete because this would collapse the PH to the 2nd level.

2.1.2 Polynomial Hierarchy

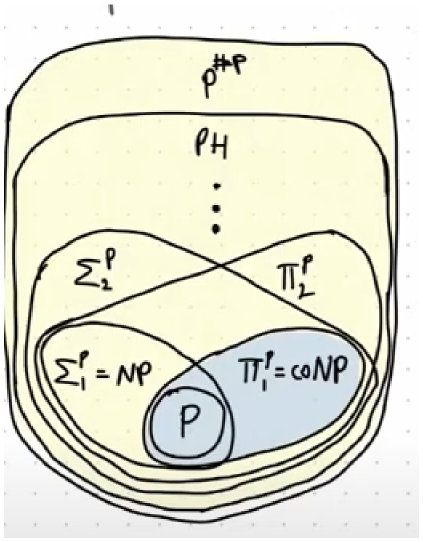


Figure 2.2: By Alessandro Chiesa

Definition 2.1.6: Σ_k

For a natural number k, a decision problem $S \subseteq \{0,1\}^*$ is in Σ_k if there exists a polynomial p and polynomial-time algorithm V such that $x \in S$ iff

$$\exists y_1, \in \{0,1\}^*, \forall y_1 \in \{0,1\}^*, \exists y_3 \in \{0,1\}^* \dots$$

s.t. $V(x, y_1, \dots, y_k) = 1$.

Definition 2.1.7: Π_k

$\{\{0,1\}^* \backslash S : S \in \Sigma_k\}$

Definition 2.1.8: $\mathcal{PH}$

$\mathcal{PH} := \cup_k \Sigma_k$

$\mathcal{PH}$ has a simple logical characterisation: it is the set of languages expressible by second-order logic.

Proposition 2.1.9

For every $k \geq 1$, if $\Sigma_k = \Pi_k$, then $\Sigma_{k+1} = \Sigma_k$, which in turn implies $\mathcal{PH} = \Sigma_k$.

Corollary 2.1.10

$\mathcal{NP} = co\mathcal{NP}$ implies $\mathcal{PH} = \mathcal{NP}$ since $\mathcal{NP} = \Sigma_1$ and $co\mathcal{NP} = \Pi_1$

2.1.3 $\mathcal{IP}$ for Graph Non-Isomorphism Resumed

Theorem 2.1.11: $GNI \in \mathcal{IP}$

Proof for Theorem

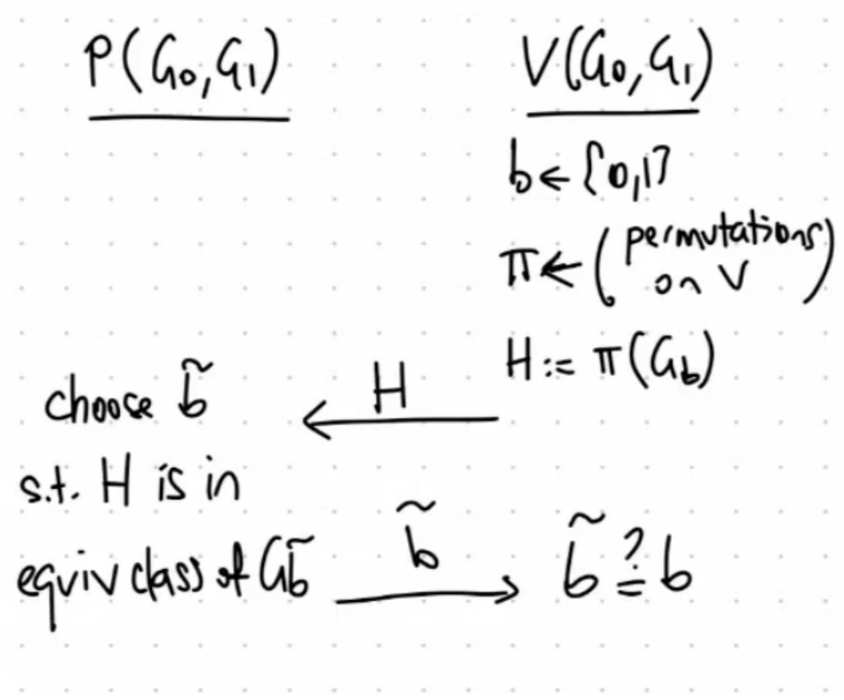


Figure 2.3: By Alessandro Chiesa

Isomorphism is an equivalence relations which partitions the set of graphs into equivalence classes, this suffices for completeness.
π gives a random instance of the equivalence class which leaks nothing about b
This is an example of a private coin protocol since b must remain private. There exists a public coin protocol for GNI which is non-trivial.

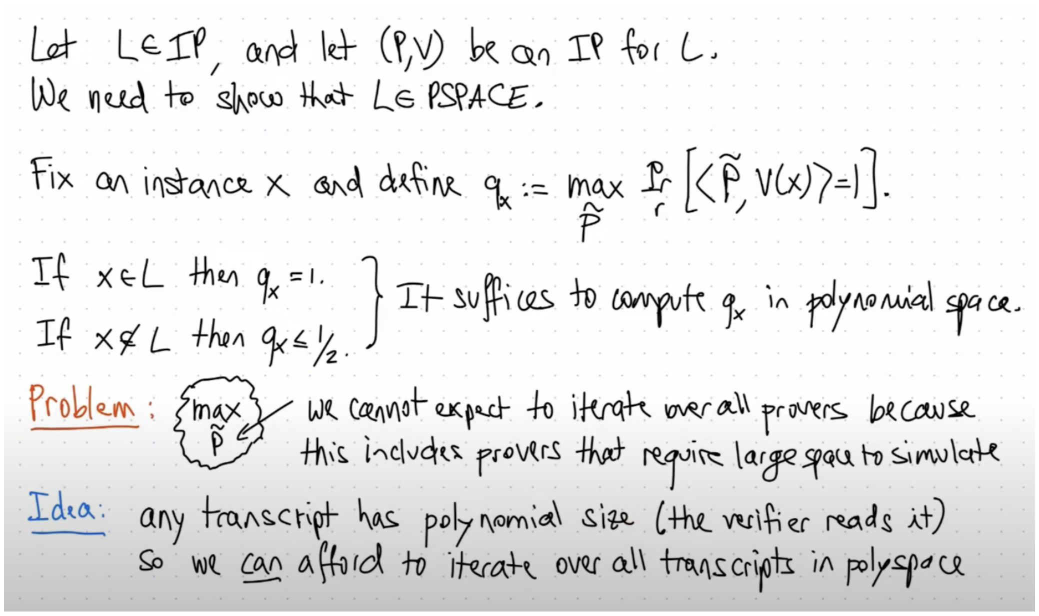
2.1.4 An Upper Bound on $\mathcal{IP}$

Definition 2.1.12: $\mathcal{PSPACE}$

The set of decision problems which can solved in polynomial space.

Theorem 2.1.13: $\mathcal{IP} \subseteq \mathcal{PSPACE}$

Proof for Theorem



Let $L \in IP$, and let (P,V) be an IP for L.
We need to show that $L \in PSPACE$.

Fix an instance x and define $q_x := \max_{\tilde{P}} \Pr_r \left[\langle \tilde{P}, V(x)\rangle = 1\right]$.

If $x \in L$ then $q_x = 1$.
If $x \notin L$ then $q_x \le 1/2$.
$\Big\}$ It suffices to compute q_x in polynomial space.

Problem: $\max_{\tilde{P}}$ we cannot expect to iterate over all provers because this includes provers that require large space to simulate

Idea: any transcript has polynomial size (the verifier reads it)
so we can afford to iterate over all transcripts in polyspace

Figure 2.4: By Alessandro Chiesa

Definition 2.1.14: Optimal prover for one more round

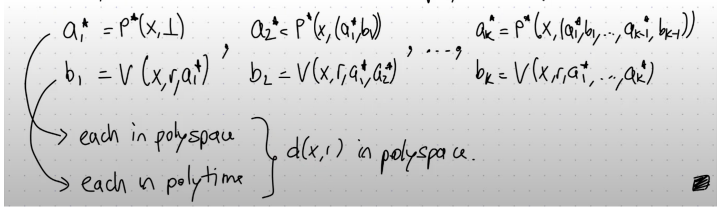
$P^*(x, (a_1, b_1, ..., a_i, b_i))$ outputs a_{i+1} that maximises convincing probability.

Lemma 2.1.15

$P^* \in \mathcal{PSPACE} \to q_x \in \mathcal{PSPACE}$

Proof for Lemma

Since P^* is optimal, we have $q_x = \frac{\sum d(x;r)}{|R|}$ where $d(x;r)$ is the decision of $V(x;r)$ when interacting with P^*.



$a_1^* = P^*(x,\perp)$, $a_2^* = P^*(x,(a_1^*,b_1))$, ..., $a_k^* = P^*(x,(a_1^*,b_1,\dots,a_{k-1}^*,b_{k-1}))$
$b_1 = V(x,r,a_1^*)$, $b_2 = V(x,r,a_1^*,a_2^*)$, $b_k = V(x,r,a_1^*,\dots,a_k^*)$

each in polyspace
each in polytime
$\Big\}$ $d(x,r)$ in polyspace.

Figure 2.5: By Alessandro Chiesa

We begin by calling P^* on the empty transcript, then verifier on the a_1^*, and so on and so for. The sum then follows because R is polynomial in size.

Lemma 2.1.16

$P^* \in \mathcal{PSPACE}$

Proof for Lemma

Let $tr = (a_1, b_1, ..., a_i, b_i)$ be a transcript of i rounds.
and let $R[x, tr]$ be the set of random strips consistent with (x, tr).
These set is restricted by $\bigwedge_{j=1}^{i} b_j = V(x, r, a_1, ..., a_j)$
Proof by induction on i:

Base case $i = k - 1$: try over all possible messages to maximise the probability over the randomness of the probability that is consistent so far. Computing this takes exponential time but only polynomial space.

Inductive case Note my inductive hypothesis is that I can compute the answer in polynomial space for all transcripts longer then the current one.
$P^*(x, tr) = argmax_{a_{i+1}} \sum_{r \in R[x,tr]} [V(x, r, a_1, ..., a_{i+1}, a^*_{i+2}, a^*_k)]$ There are many implicit inner loops but they all return to caller within polynomial space.

■

So we have shown that interactive proofs can only decide problems which could be computed in polynomial space.

$\mathcal{PSPACE}$ is at least as big as the $\mathcal{PH}$ but is not larger then $\mathcal{EXPTIME}$ or $\mathcal{EXPSPACE}$.

Chapter 3

Unsatisfiability and Counting Problems

Key concepts in previous chapter:

- *Polynomial Hierarchy*
- *An IPS for $\mathcal{GNI}$*
- $\mathcal{IP} \subseteq \mathcal{PSPACE}$

Key concepts in this chapter: Arithmetisation, sumcheck protocol

3.1 Unsatisfiability

Definition 3.1.1: $\mathcal{UNSAT}$

For a boolean circuit, is it the case that there are not satisfying assignments

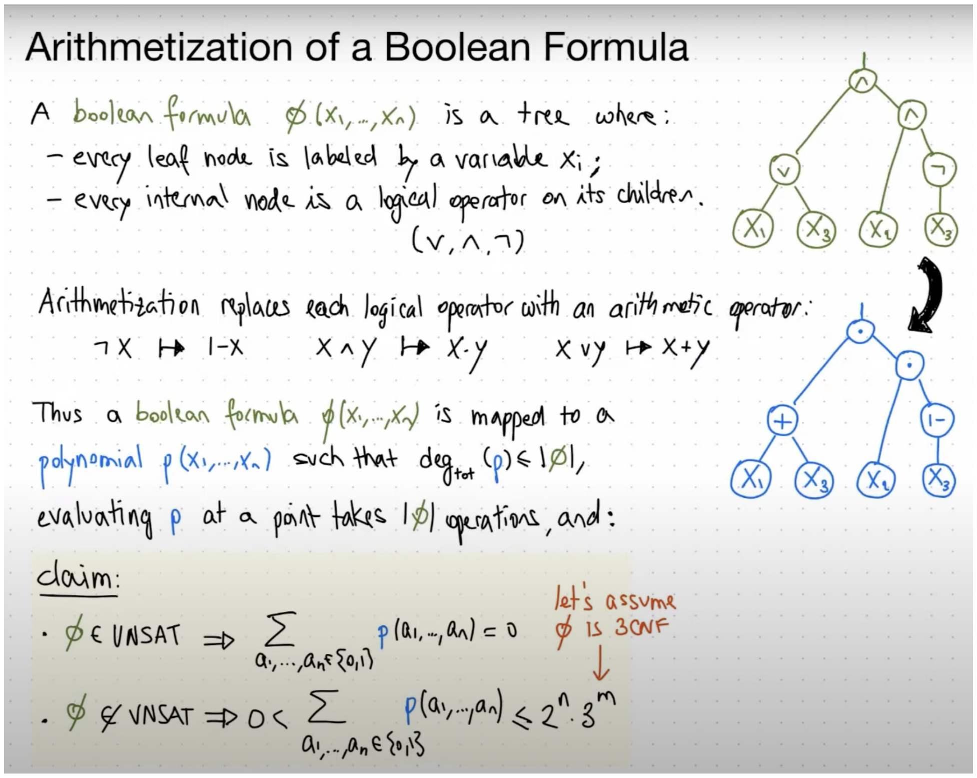
$\mathcal{UNSAT} \in co\mathcal{NP}$*-complete*

Theorem 3.1.2: $\mathcal{UNSAT} \in \mathcal{IP}$

From which it follows that $co\mathcal{NP} \subseteq \mathcal{IP}$

Proof for Theorem

Our poof of GNI leveraged specific properties of graph isomorphism, $\mathcal{UNSAT}$ doesn't appear to have similar properties.



Arithmetization of a Boolean Formula

A boolean formula $\phi(x_1,\dots,x_n)$ is a tree where:
- every leaf node is labeled by a variable x_i;
- every internal node is a logical operator on its children. $(\vee, \wedge, \neg)$

Arithmetization replaces each logical operator with an arithmetic operator:

$\neg x \mapsto 1-x \qquad x \wedge y \mapsto x \cdot y \qquad x \vee y \mapsto x+y$

Thus a boolean formula $\phi(x_1,\dots,x_n)$ is mapped to a polynomial $p(x_1,\dots,x_n)$ such that $\deg_{tot}(p) \leq |\phi|$, evaluating p at a point takes $|\phi|$ operations, and:

claim:

- $\phi \in UNSAT \Rightarrow \sum_{a_1,\dots,a_n \in \{0,1\}} p(a_1,\dots,a_n) = 0$
- $\phi \notin UNSAT \Rightarrow 0 < \sum_{a_1,\dots,a_n \in \{0,1\}} p(a_1,\dots,a_n) \leq 2^n \cdot 3^m$

let's assume ϕ is 3CNF

Figure 3.1: By Alessandro Chiesa

In support of the first claim note that the arithmetisation preserves the zeroness of operators.
3CNF, Conjunctive Normal Form, an And of Ors. No more than three variable per conjunct. The sum has 2^n values it ranges over and each circuit will return at most 3^m. It or clause can be at most three and the number of conjunctions is m.
Not we are assuming that the formula has been adapted so that the not(s) appear at the lowest level.

Corollary 3.1.3

$\forall q > 2^n 3^m, \phi \in UNSAT \leftrightarrow \sum p(a_1,\dots,a_n) = 0 \bmod q$

Sumcheck Protocol $\sum_{\alpha_1,\dots,\alpha_n \in H} p(\alpha_1,\dots,\alpha_n) = \gamma$

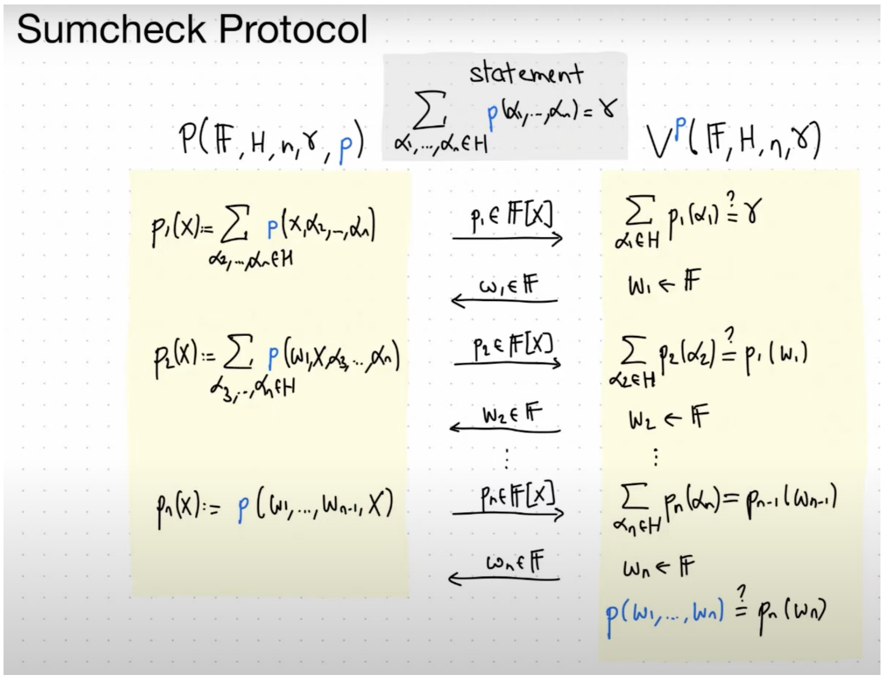

Sumcheck Protocol
statement
$\sum_{\alpha_1,\dots,\alpha_n\in H} p(\alpha_1,\dots,\alpha_n)=\gamma$
$P(\mathbb{F},H,n,\gamma,p)$
$V^p(\mathbb{F},H,n,\gamma)$
$p_1(X):=\sum_{\alpha_2,\dots,\alpha_n\in H} p(X,\alpha_2,\dots,\alpha_n)$
$p_1\in\mathbb{F}[X]$
$\sum_{\alpha_1\in H} p_1(\alpha_1)\overset{?}{=}\gamma$
$\omega_1\in\mathbb{F}$
$w_1\leftarrow\mathbb{F}$
$p_2(X):=\sum_{\alpha_3,\dots,\alpha_n\in H} p(\omega_1,X,\alpha_3,\dots,\alpha_n)$
$p_2\in\mathbb{F}[X]$
$\sum_{\alpha_2\in H} p_2(\alpha_2)\overset{?}{=}p_1(w_1)$
$w_2\in\mathbb{F}$
$w_2\leftarrow\mathbb{F}$
$p_n(X):=p(w_1,\dots,w_{n-1},X)$
$p_n\in\mathbb{F}[X]$
$\sum_{\alpha_n\in H} p_n(\alpha_n)=p_{n-1}(w_{n-1})$
$\omega_n\in\mathbb{F}$
$w_n\leftarrow\mathbb{F}$
$p(w_1,\dots,w_n)\overset{?}{=}p_n(w_n)$

Figure 3.2: By Alessandro Chiesa

We only need to send n low degree polynomials. We need $|H|$ to be polynomial but otherwise the computation is straightforward.

Claim

if $\sum_{\alpha_1,\dots,\alpha_n\in H} p(\alpha_1,\dots,\alpha_n)\neq\gamma$ then $Pr[\text{verifier accepts}]\leq\frac{n*deg(p)}{|\mathbb{F}|}$

A (malicious) prover is characterised by $\tilde{p_1},\dots,\tilde{p_n}\in\mathbb{F}[x]$ where $\tilde{p_i}$ may depend on $w_1,\dots,w_i$

Definition 3.1.4: E_i

event that $\tilde{p_i}=p_i$

Definition 3.1.5: W

event that verifier accepts

Lemma 3.1.6

For $j = n, n-1, ..., 1 : Pr[w] \leq \frac{(n-j+1)*deg(p)}{|\mathbb{F}|} + Pr[w|E_j \wedge ... \wedge E_n]$

Proof for Lemma

This suffices to prove the claim since if $j = 1$ then $Pr[w] \leq \frac{n*deg(p)}{|\mathbb{F}|} + Pr[w|E_1 \wedge ... \wedge E_n]$. By definition the probability of the second term is zero when then statement is false.

Base case $j = n$:
$Pr[W] \leq Pr[W|\tilde{E}_n] + Pr[W|E_n] \leq \frac{deg(p)}{|\mathbb{F}|} + Pr[W|E_n]$

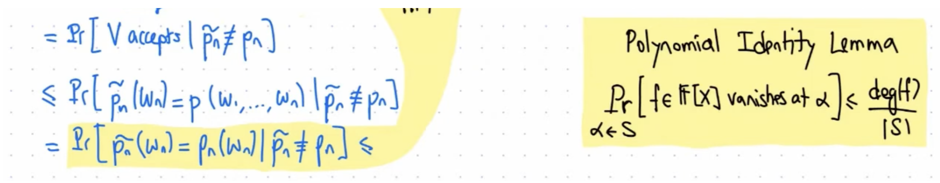

$= Pr[\text{ V accepts } | \tilde{p}_n \neq p_n]$
$\leq Pr[\tilde{p}_n(w_n) = p(w_1, ..., w_n) | \tilde{p}_n \neq p_n]$
$= Pr[\tilde{p}_n(w_n) = p_n(w_n) | \tilde{p}_n \neq p_n] \leq$
Polynomial Identity Lemma
$Pr_{\alpha \in S}[f \in \mathbb{F}[X] \text{ vanishes at } \alpha] \leq \frac{deg(f)}{|S|}$

Figure 3.3: By Alessandro Chiesa

Inductive case: assume it holds for $j \in ..., n$ and prove it holds for $j - 1$
$Pr[W] \leq \frac{(n-j+1)*deg(p)}{|\mathbb{F}|} + Pr[w|E_j \wedge ... \wedge E_n]$
$Pr[W] \leq \frac{(n-j+1)*deg(p)}{|\mathbb{F}|} + Pr[w|\tilde{E}_{j-1} \wedge E_j \wedge ... \wedge E_n] + Pr[w|E_{j-1} \wedge E_j \wedge ... \wedge E_n]$
$Pr[W] \leq \frac{(n-j+1)*deg(p)}{|\mathbb{F}|} + \frac{deg(p)}{|\mathbb{F}|} + Pr[w|E_{j-1} \wedge E_j \wedge ... \wedge E_n]$
$Pr[W] \leq \frac{(n-(j-1)+1)*deg(p)}{|\mathbb{F}|} + Pr[w|E_{j-1} \wedge E_j \wedge ... \wedge E_n]$ ∎

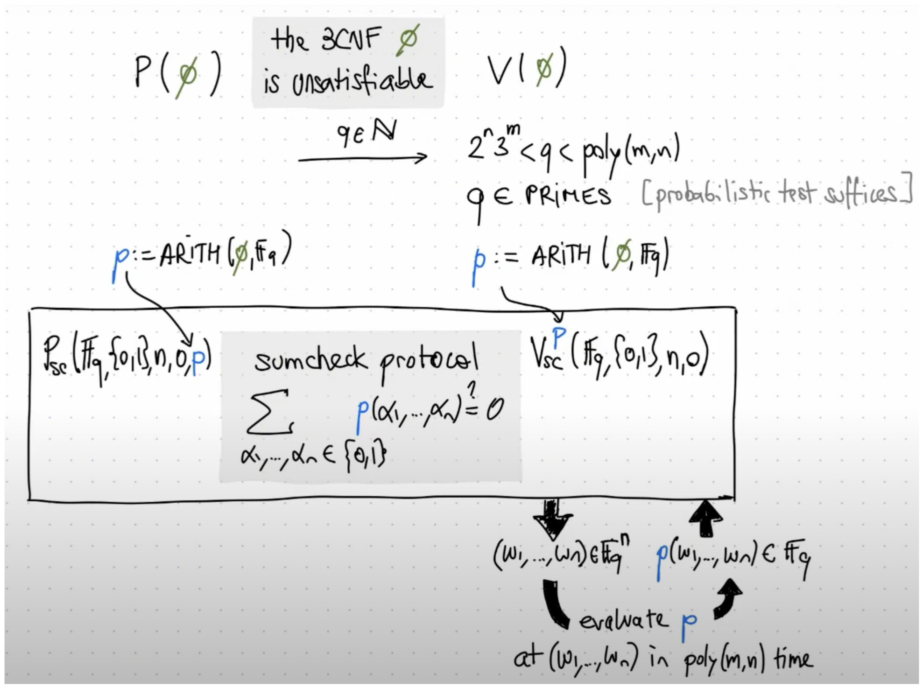

$P(\phi)$
the 3CNF ϕ is unsatisfiable
$V(\phi)$
$q \in \mathbb{N}$
$2^n 3^m < q < poly(m,n)$
$q \in$ PRIMES [probabilistic test suffices]
$p := ARITH(\phi, \mathbb{F}_q)$
$p := ARITH(\phi, \mathbb{F}_q)$
$P_{SC}(\mathbb{F}_q, \{0,1\}, n, 0, p)$
sumcheck protocol
$\sum_{\alpha_1,\dots,\alpha_n \in \{0,1\}} p(\alpha_1,\dots,\alpha_n) \overset{?}{=} 0$
$V_{SC}^{p}(\mathbb{F}_q, \{0,1\}, n, 0)$
$(\omega_1,\dots,\omega_n) \in \mathbb{F}_q^n$
$p(\omega_1,\dots,\omega_n) \in \mathbb{F}_q$
evaluate p at $(\omega_1,\dots,\omega_n)$ in $poly(m,n)$ time

Figure 3.4: By Alessandro Chiesa

$q < 2^{poly(m,n)}$

3.2 $\mathcal{P}^{\#\mathcal{P}} \subseteq \mathcal{IP}$

Definition 3.2.1: $\mathcal{P}^{\#\mathcal{P}}$

The complexity class of polynomial machines which have access to an oracle which solves the counting problems associated with $\mathcal{NP}$

$\mathcal{PH} \subseteq \mathcal{P}^{\#\mathcal{P}}$

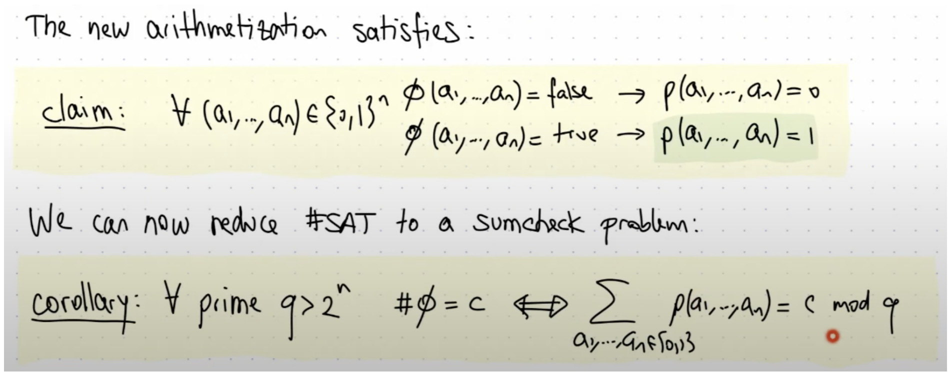
Theorem 3.2.2: $\#\mathcal{SAT} \in \mathcal{IP}$

From which it follows that $\mathcal{P}^{\#\mathcal{P}} \subseteq \mathcal{IP}$

Proof for Theorem

Our previous arithmetisation was too course. It only told us if a satisfaction assignment existed or not.

$\neg x \to 1 - x$ $\qquad x \wedge y \to x * y$ $\qquad x \vee y \to x + y - x * y$



The new arithmetization satisfies:

claim: $\forall (a_1,\dots,a_n) \in \{0,1\}^n$ $\phi(a_1,\dots,a_n) = \text{false} \rightarrow p(a_1,\dots,a_n) = 0$

$\phi(a_1,\dots,a_n) = \text{true} \rightarrow p(a_1,\dots,a_n) = 1$

We can now reduce #SAT to a sumcheck problem:

corollary: $\forall$ prime $q > 2^n$ $\quad \#\phi = c \Longleftrightarrow \sum_{a_1,\dots,a_n \in \{0,1\}} p(a_1,\dots,a_n) = c \mod q$

Figure 3.5: By Alessandro Chiesa

This arithmetisation does result in higher, but still polynomial, degree polynomials
Let $L \in \mathcal{P}^{\#\mathcal{P}}$ and let M be a machine that decides L with a $\#SAT$ oracle.

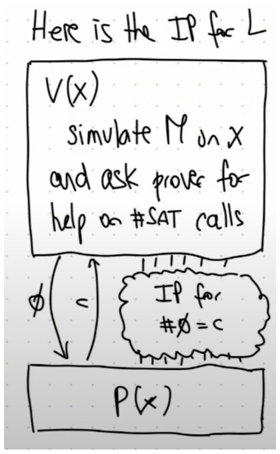


Figure 3.6: By Alessandro Chiesa

3.3 $\mathcal{IP} = \mathcal{PSPACE}$

Claim: $\mathcal{IP} = \mathcal{PSPACE}$

Chapter 4

Applications in Cryptography

4.1 Why I want to hide a witness

In many cases I would like to convince of the truth of the statement but I would rather not show you the witness. Let me motivate this with an example from secure election voting

Definition 4.1.1: Integer Encryption Scheme

An Integer Encryption Scheme is a tuple of polynomial time algorithms ($\texttt{KeyGen}$,$\texttt{Enc}$,$\texttt{Dec}$) s.t.

- $\texttt{KeyGen}$ takes a natural number λ and produces a public key y and a secret key x
- $\texttt{Enc}$ takes a public key pk, and an integer m, some randomness r and produces a ciphertext c
- $\texttt{Dec}$ takes a secret key sk, and a ciphertext c and produces a number m'

Generally with are after two properties:

Correctness: $\forall \lambda, m, r, 0 \leq m < 2^\lambda \Rightarrow Pr[\texttt{Dec}(x, \texttt{Enc}(y, m; r)) = m : (y, x) \in \texttt{KeyGen}(\lambda)] = 1$

Hiding: Informally, without knowledge of the secret key no polynomial time algorithm can learn anything about m with non-negligible probability.

Correctness forces $\texttt{Enc}$ to be injective

Definition 4.1.2: Homomorphic Natural Encryption Scheme

As above with the additional property that their exists an algorithm Com which takes two ciphertext and returns a ciphertext such that:

$$\forall \lambda, m, m', r, r', 0 \leq m + m' < 2^{\lambda} \Rightarrow$$

$$Pr[\texttt{Dec}(x, \texttt{Comb}(\texttt{Enc}(y, m; r')\texttt{Enc}(y, m'; r))) = m + m' : (y, x) \in \texttt{KeyGen}(\lambda)] = 1$$

Definition 4.1.3: A Voting Protocol for Referendums

Our voting protocol will consist of one authority A and n voters $V_1, ..., V_n$

1. A runs `KeyGen` and sends public key y to all voters and keeps x private
2. Each voter V_i encrypts their yes, encoded as 1, or no vote, encoded as 0, $\texttt{Enc}(y, v_i; r_i)$ and send the resulting ciphertext c_i to A
3. A computes $c_t := \texttt{Comb}_{i=1}^{n} c_i$ and published the tally $\texttt{Dec}(x, c_t)$

The protocol above protocol has several problems from an integrity standpoint:

1. *A dishonest authority can publish an incorrect tally*
2. *A dishonest voter could encrypt an integer other than zero or one*

Both of these problems could be addressed by proving the correct computation of `Enc` and `Dec` respectively by revealing the inputs to the functions. However, to reveal the input to `Enc` would break the privacy of the voter whereas to reveal the secret key used in `Dec` would break the privacy of all voters.
What I want to do is to prove that a satisfactory pre-image of the function exists without revealing what it is.

4.2 Zero-knowledge proofs

Definition 4.2.1: One way functions

A polynomial time algorithm f is one way if for all polynomial time algorithm F:

$$\Pr[f(F(f(x))) = f(x)] < n^{-c}$$

Claim

Assuming the existence of one way functions there exists Zero-Knowledge proofs for $\mathcal{IP} = \mathcal{PSPACE}$

Intuitively the zero-knowledge property means the verifier learns nothing about the witness

Definition 4.2.2: An alternative IPS for $\mathcal{GI}$

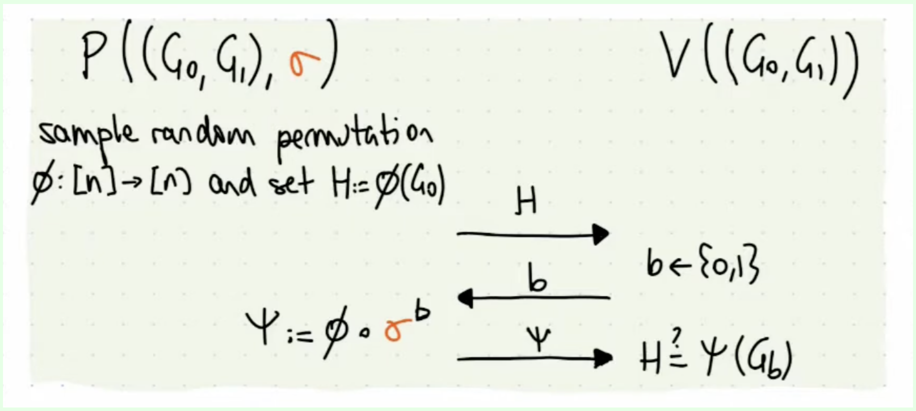


Figure 4.1: By Alessandro Chiesa

Completeness is immediate from the definition of $\mathcal{GI}$ and equivalence classes.
Soundness: if G_0 and G_1 are not isomorphic then H can be isomorphic to at most one of them. If the corresponding b is sampled P will be caught.

Definition 4.2.3: Honest Verifier Zero-Knowledge

An interactive proof (P, V) for L is honest verifier zero-knowledge if there exists a polynomial-time simulator S such that

$$\forall x \in L, \quad S(x) \equiv View_V(< P, V > (x))$$

This essentially says the verifier could simulate the entire conversation without talking to the honest prover.

Definition 4.2.4: Zero-Knowledge

An interactive proof (P, V) for L is (malicious-verifier) zero-knowledge if there exists a polynomial-time simulator S such that

$$\forall x \in L, \quad \forall \texttt{ppt}\tilde{V} \quad S(\tilde{V}, x) \equiv View_{\tilde{V}}(< P, \tilde{V} > (x))$$

Zero-knowledge is a really weird property and is justified by how it used in high level protocols.
Returning to our voting example, privacy for such a protocol would commonly be defined as the adversary's view in the honest protocol being indistinguishable from a protocol where each of the voters encrypted nonsense. The security of the encryption schemes allows me to swap the ciphertexts for non-sense and zero-knowledge allows me to simulate the zero-knowledge proofs.

Claim

There exists a generic transform from Honest Verifier Zero-Knowledge to Zero-Knowledge (under some computational assumptions)

4.3 Digital Signatures

This material is shamelessly copied with slight modifications from Ivan Damgård's introduction "On Σ-protocols."

Definition 4.3.1: $\mathcal{DL}$

For a given (cyclic) group G and generator g the language of discrete logs $\{(x, w) | x = (g, h), h = g^w\}$

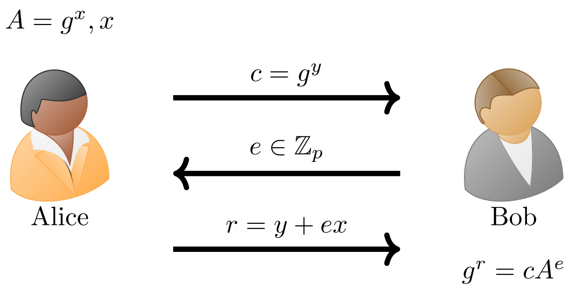
The best know algorithm for computing discrete logs for a generic group takes exponential time

Sigma protocols are particularly simple and efficient zero knowledge proof of knowledge.

Definition 4.3.2: Sigma protocols

A protocol $\mathcal{P}$ is said to be a Σ-protocol for relation R if:

- $\mathcal{P}$ is a 3-move form, and we have completeness: if P, V follow the protocol on input x and private input w to P where $(x, w) \in R$, the verifier always accepts.
- From any x and any pair of accepting conversations on input x, $(c, e, r)(c, e', r')$ where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$. (Special soundness)
- There exists a polynomial-time simulator M, which on input x and a random e outputs an accepting conversation of the form (c, e, r) with the same probability distribution as conversation between the honest P, V on input x. (Honest-verifier zero-knowledge)



Binding: After Alice commits we choose a random challenge

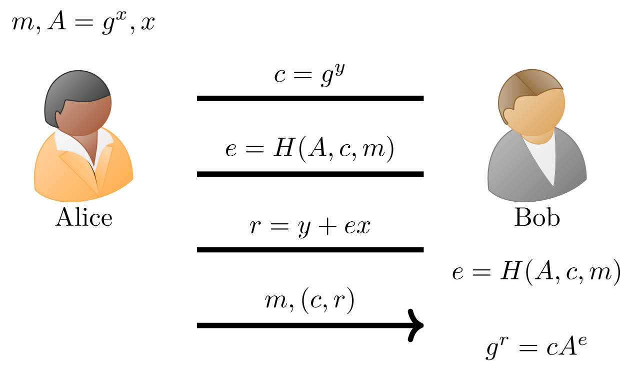
If she can answer correctly for two separate challenges then we have (c,e,r)(c,e',r').
$x = (r - r')/(e - e')$

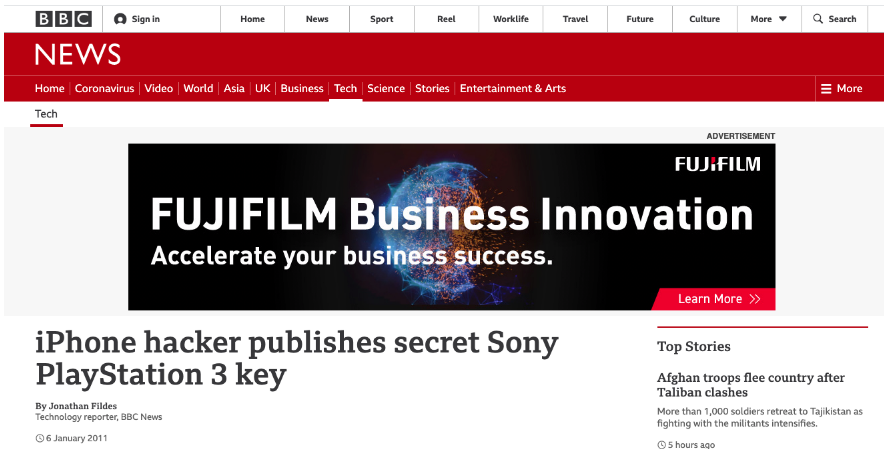
Hiding: Observe that the only part of the protocol which depends upon the secret is r

r is uniformly distributed by y

y is only otherwise revealed by $c = g^y$ where it is hard to find by the discrete log problem

4.4 Interaction is such a pain




BBC
Sign in
Home
News
Sport
Reel
Worklife
Travel
Future
Culture
More
Search
NEWS
Home | Coronavirus | Video | World | Asia | UK | Business | Tech | Science | Stories | Entertainment & Arts
More
Tech
ADVERTISEMENT
FUJIFILM
FUJIFILM Business Innovation
Accelerate your business success.
Learn More

iPhone hacker publishes secret Sony PlayStation 3 key

By Jonathan Fildes
Technology reporter, BBC News

6 January 2011

Top Stories

Afghan troops flee country after Taliban clashes

More than 1,000 soldiers retreat to Tajikistan as fighting with the militants intensifies.

5 hours ago

Chapter 5

Probabilistically Checkable Proofs

Many $\mathcal{NP}$-hard problems can be approximated. That is, there exists polynomial algorithms which return results within a known bound of the optimal solution. For some problems, such as the famous knapsack problem, these errors margins can be arbitrarily small.

Definition 5.0.1: Knapsack Problem

For a given set $\{v_i\}_{i=1}^n$ with associated weights $\{w_i \in \mathbb{N}\}_{i=1}^n$ and a capacity $W \in \mathbb{N}$:

$$(max \sum_{i=1}^{n} v_i x_i) \leq W$$

when $x_i \in \{0, 1\}$

This leads to the natural question if all $\mathcal{NP}$-hard problems be approximated with arbitrary precision.

Bizarrely the answer to this question from the interactive proofs we already seen.

I won't give a full explanation here, interested readers are recommend to read "The Tale of the PCP Theorem" by Dana Moshkovitz `https://www.cs.utexas.edu/~danama/XRDS.pdf`. The basic gist is:

- For some bound x, the hardness of x-approximation is related to the hardness of deciding if two solutions are x apart.
- The hardness of deciding if two solutions are x apart is related to the length of the mathematical proof for the statement.

5.1 Definition of $\mathcal{PCP}$

PCPs still have randomness but rather than interaction have oracle access to the proof

Definition 5.1.1: $\mathcal{PCP}$

A PCP System for S is a pair of algorithms (P, V), where P is unbounded and V is a polynomial time oracle algorithm, with completeness error ϵ_c and soundness error ϵ_s s.t.:

1. Completeness: For ever $x \in S$, for $\pi := P(x)$, $Pr[V^{\pi}(x; r) = 1] \geq 1 - \epsilon_c$
2. Soundness: For every $x \notin S$, for every $\tilde{\pi}$, $Pr[V^{\tilde{\pi}}(x; r) = 1] \leq \epsilon_s$

Things we care about include:

Σ**:** *proof alphabet*

l**:** *proof length*

q**:** *verifier query complexity*

r**:** *verifier randomness complexity*

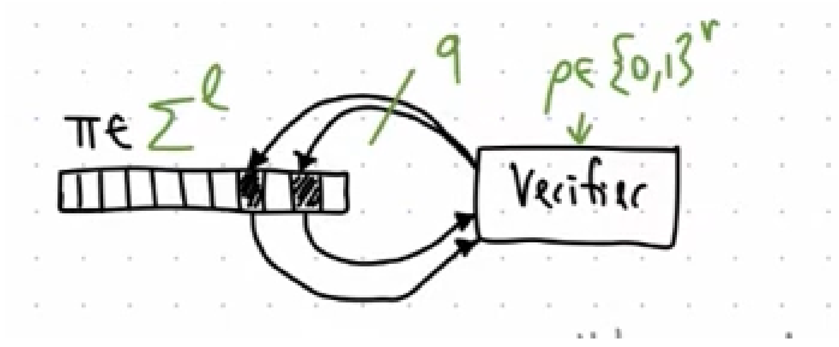


Figure 5.1: By Alessandro Chiesa

Claim: $\mathcal{PCP}[r = 0, q = 0] = \mathcal{P}$

Claim: $\mathcal{PCP}[r = O(\log n)), q = 0] = \mathcal{P}$

Claim: $\mathcal{PCP}[r = poly(n), q = 0] = \mathcal{BPP}$

Claim: $\mathcal{PCP}[r = 0, q = poly(n)] = \mathcal{NP}$

Theorem 5.1.2: $\mathcal{IP} \subseteq \mathcal{PCP}$

Proof for Theorem

Suppose that (P, V) is a public-coin IPS for S. (Public coin is WLOG.)
Consider the union of all possible transcripts. $\Pi = \{a_{r_1}\}_{r_1} \cup ... \cup \{a_{r_1,...,r_k}\}_{r_1,...,r_k}$
The verifier samples $r_1, .., r_k$ and accepts if the IPS verifier accepts:

$$V(x, a_{r_1}, a_{r_1,r_2}, ...; r_1, ..., r_k) = 1$$

Completeness follows immediately from the IP completeness.
The proof is a complete commitment to a proof strategy.
Soundness then follows from the soundness of the IP verifier. ■

5.2 $\mathcal{PCP}$ Theorem

Theorem 5.2.1: $\mathcal{PCP}$ Theorem

$\mathcal{PCP}[r = O(\log n), q = O(1)] = \mathcal{NP}$

Proof for Theorem

That is for all problems in $\mathcal{NP}$ there exists a PCP system for the problem which uses a logarithmic amount of randomness and looks at a constant amount of the proof.
This theorem is the result of a line of work by Arora, Feige, Goldwasser, Lund, Lovász, Motwani, Safra, Sudan, and Szeged; the margins of these lecture notes is not large enough for the proof.

The intuition is that we encode the existing proof in an error correcting code, that is a polynomial.
It then suffices to check that the polynomial is zero using the sumcheck protocol we saw before.

This could be interpreted as using a circuit to catch my verification routine and getting the prover to show that there exists a satisfying proof. ■

This theorem results in a weird situation where reading the statement is most of the work

Amazingly we have PCPs[1] *with efficient provers which can be used in practice.*

One of my honours students recently did some work where for a referendum of 2^{24} voters (roughly the number of eligible voters in the Australian 2023 Voice referendum) we reduced the verification time from the naive verification lower bound of approximately 77 ***core-days*** *to a projected time of 2.57* ***core-minutes*** *for the proposed protocol, and reduces the sizes of the inputs from the naive 37.92 GB to a total proof size of 1.54 GB.*

[1]well sort of PCPs, formally interactive arguments

Chapter 6

Bonus: Average-Case Complexity

I want to spend a little time discussing how complexity theory affects practical computation and cryptography.

Recommend Reading: Russell Impagliazzo's A Personal View of Average-Case Complexity (Sections 1 and 2)

- The paper is from 1995 some of the comments on cryptography are outdated

6.1 Average-case complexity

Average-case complexity:

$$Pr_{x \in_R D_n}[t_A(x) \geq t] \leq \frac{p(n)}{t^\epsilon}$$

- A "hard" (NP-complete) problem may be easily decidable for almost all problems
- In cryptography we want to be able to efficiently generate hard instances

6.2 Five worlds

Russell Impagliazzo describes five possible worlds which he describes in terms of Professor Grouse (the teach of the young Gauss) attempting to humiliate the young Gauss by inventing problems Gauss couldn't solve. (Historically this result in Grouse being committed to a lunatic asylum.)

Definition 6.2.1: Algorithmica

- P = NP (or equivalent)
- It's as easy to recognise a solution as to produce it (at least asymptotically)
- Would solve all kinds of optimisation problems
- No cryptography based on computational hard problems (only information-theoretic constructions)

Definition 6.2.2: Heuristica

- NP problems are intractable in the worst case but feasible on average for any samplable distribution
- There are hard problems but they are hard to find
- Are the problems we want to solve hard to find?
- No cryptography based on computational hard problems (only information-theoretic constructions)

Definition 6.2.3: Pessiland

- Hard average case problems but no one-way functions (PRP, PRG, hash functions)
- It's easy to generate hard instances but not hard solved instances
- No cryptography based on computational hard problems (only information-theoretic constructions)

Definition 6.2.4: Minicrypt

- One-way functions exist but not public key encryption
- We don't actually know what the necessary and sufficient condition for public key cryptography is

Definition 6.2.5: Cryptomania

- (Clearly the world we are in even if not for the reason Russell meant)
- Public key encryption is possible