

COMPUTATION: DAY 8

BURTON ROSENBERG
UNIVERSITY OF MIAMI

CONTENTS

1. Interactive Proofs	1
2. The class IP	2
3. Zero knowledge proof systems	3
3.1. Cheating Verifiers	4

1. INTERACTIVE PROOFS

The NP paradigm has made use of a polynomial time verifier V such that for an $A \subseteq \Sigma^*$, then $a \in A$ if and only if there exists a y and $V(y, a) = T$. This y has various interpretations. It can be a hint, or the answer; it guides the computation to prove that a is in A . The concept of NP is that one does not specify how y is found. It is sufficient that it exists.

In an *interactive proof* two Turing machines converse using a shared tape. One machine is called the Prover, and the other is called the Verifier. The Verifier is a PPT; but the Prover can be any Turing Machine. This generalizes the notion of a hint or proof string, in that the Prover can try to convince the Verifier by providing this string. In fact, the Prover and Verifier can converse in multiple rounds.

Definition 1.1. An *iterative turing machine* is a pair of Turing machines, a probabilistic polynomial time (PPT) verifier V and a possibly computational unbounded prover P . They communicate by a communication tape. Given an input a the prover can help the verifier come to decision, denoted

$$\langle P, V \rangle(\omega, a) \in \{T, F\}$$

where ω is the randomness of V chosen over some probability space Ω . The time is polynomial for V , but the time of P is not generally considered.

Definition 1.2. The class IP is the class of languages $A \subseteq \Sigma^*$ such that there exists a pair of interactive TM's $\langle P, V \rangle$ such that,

- (1) For $a \in A$,

$$Pr_{\Omega}(\langle P, V \rangle(\omega, a) = T) \geq 2/3$$

where the probability is over $\omega \in \Omega$.

- (2) For $a \notin A$, and for any prover P^* ,

$$Pr_{\Omega}(\langle P^*, V \rangle(\omega, a) = T) \leq 1/3$$

where the probability is over $\omega \in \Omega$.

Note the asymmetry. The verifier must verify absolutely. So nothing a “cheating prover” can do will make verifier mistake an $a \notin A$ for one in A , except with limited probability.

Also note, that because of the power of the prover, it is generally not probabilistic. The ω is only used by the verifier. The reason is, since the Prover is so powerful, rather than randomize its actions, it will take the one action that maximizes its advantage, whether that is to convince the verifier to accept or to convince the verifier to reject.

2. THE CLASS IP

Definition 2.1. Graphs $G_1 = \langle V_1, E_1 \rangle$ and $G_2 = \langle V_2, E_2 \rangle$ are isomorphic if,

- (1) There is a bijection on the vertex sets, $\phi : V_1 \rightarrow V_2$, and
- (2) The bijection respects the edge structure if the graph:

$$(v, v') \in E_1 \Leftrightarrow (\phi(v), \phi(v')) \in E_2$$

We write $G_1 \cong G_2$.

Theorem 2.1. The language $ISO = \{ (G_1, G_2) \mid G_1 \cong G_2 \}$ is NP complete.

Proof: It is in NP because given a permutation on the vertices ϕ , it is checkable in polynomial time that $\phi(G_1) \cong G_2$.

To establish that it is NP complete, a reduction such as $3\text{-SAT} \leq_P ISO$ is required. The proof is omitted. \square

Theorem 2.2. The language $NISO = \{ (G_1, G_2) \mid G_1 \not\cong G_2 \}$ is in IP.

Proof: We give an $\langle P, V \rangle$ system that recognizes $NISO$.

- (1) The Verifier V chooses a random bit b and a random permutation ϕ . The verifier V sends the Prover P the graph $H = \phi(G_b)$.
- (2) The Prover figures out for which \tilde{b} does $G_{\tilde{b}} \cong H$, and sends the verifier \tilde{b} . If both or neither, it sends a random \tilde{b} .
- (3) The Verifier accepts if $b = \tilde{b}$.

If the graphics are not isomorphic, then $(G_1, G_2) \in NISO$. Then the Prover can determine for which \tilde{b} that $H \cong G_{\tilde{b}}$, and the verifier accepts with probability 1. If they are isomorphic then $H \cong G_1 \cong G_2$, and it was the verifier's choice, unknown to the prover which of these two isomorphisms the verifier is expecting. A random guess by the prover means the verifier accepts with probability $1/2$. This is greater than the allowed error probability, so two independent trials are made, the the only accepts if both times the prover has guessed its coin. So the probability the verifier accepts is $1/4$. \square

3. ZERO KNOWLEDGE PROOF SYSTEMS

The nature of NP holds surprises. So far, we accept an instance if a proof can be given for it's acceptance. This proof provides us with some knowledge that helps the verifier accept. If we think of an interactive proof system, what is being supplied is something the verifier could not calculate for itself, as otherwise the prover would not be needed.

- *Polynomial time calculations do not increase knowledge. They reveal aspects of the same knowledge.*

Oddly, the knowledge of say a graph isomorphism is not intrinsic in the statement. Given two isomorphic graphics, finding the isomorphism is hard, but it is not hard to great tow isomorphic graphs. This asymmetry can be exploited to create zero-knowledge proofs — proofs that reveal nothing beyond the truth of the statement. For graph isomorphism, the proof proves the graphs are isomorphic without giving the isomorphism. Or providing for any other increase of knowledge.

Theorem 3.1. Graph isomorphism can be proved in zero-knowledge.

Proof: An interactive proof system $\langle P, V \rangle$ decides *ISO* by this algorithm,

- (1) The prover chooses a random permutation ϕ of the vertices of G_1 and sends the verifier $H = \phi(G_1)$.
- (2) The verifier chooses are random bit b and sends it to the prover.
- (3) The prover sends the isomorphism ϕ' such that $H = \phi'(G_b)$.

This is an interactive proof system. If the graphs are isomorphic, the prover can always provide ϕ' and the verifier accepts with probability 1. If the graphs are not isomorphic, no prover can convince the verifier with probability greater than $1/2$, because half the time the H it provided is not isomorphic to the G_b the verifier choose.

To show it is zero-knowledge, we need to show that for the case of isomorphic graphs nothing the verifier experiences in the course of the computation is anything it could not have computed itself. It if does receive knowledge, it does so from the

communication with the prover. In this algorithm that communication is a triple drawn from a probability space of triples,

$$X = \{ (H, b, \phi) \mid H = \phi(G_b) \},$$

where H is a graph, G_1 and G_2 are the given graphs, b chosen randomly, and ϕ is a random permutation of vertices on G_b .

We think of the random sampler that delivers one of this triples. In the algorithm, it is drawn by first the Prover providing H , then the Verifier providing b and the the Prover responding with the only correct ϕ that satisfies the constraint for the triples. Because of this solving an instance of ISO, this sampler is NP hard.

However, a sampler with the same distribution could start with a random ϕ and b and set H to satisfy the constraint. This sampler, called *the simulator* can be a PPT.

A simple case is where the verifier flips a fair coin for b . In which case without changing any computation it can let the simulator flip the coin. In this case, that of *honest verifier zero-knowledge*, we have our proof. The Prover-Verifier combination is replaced with this Simulator-Verifier combination for a PPT computation.

The prover might also bias the choice of H (equivalently, ϕ). Since we prove zero-knowledge in the positive case of accepting the input, we can choose our prover, and we choose one that does not do this. The verifier does not have the flip a fair coin. It can react to H as it cares to, thereby shaping the distribution X . The simulator must match this distribution. The formal definition of zero-knowledge does require that the proof hold for any verifier.

The formal definition of zero-knowledge only asks for a proof in the accepting case, and in this case the simulator is working fine. Fortunately, nothing has to be proven in the rejecting case, since in this case the simulator cannot simulate the prover. The prover will recognize that in has been given an impossible task and modify its answers. The simulator however must continue to produce answers the same as always, where $H = \phi(G_b)$, and therefore incapable of rejecting an input. The algorithm will become completely unsound.

3.1. Cheating Verifiers. The simulation of X when the protocol is instantiated with an arbitrary V must by necessity use V to mold the distribution. A copy of V is made, that references the same randomness string, and the $x = (H, b, \phi(G_b))$ is tested against the V . Specifically, V is run up to its decision on b , and if it agrees with that in the drawn x , the simulation outputs that x . Else it retries. If at last it never succeeds, it marks its failure outputting the value \perp . The repetition limit is set to satisfy an error bound to be determined later.

We claim that this stochastic process of generating a uniform distribution underlying X , then selecting, gives the distribution on X of the original protocol. However,

what we mean is when conditioned on $x \neq \perp$. Left to consider is how the case of indeterminacy $x = \perp$ is to be handled.

We can choose our prover in the case of accept, and therefore a verifier must reject on \perp . Else a cheating prover can send \perp in place of valid arguments and cause a false accept. This will mean that completeness will not be perfect. That is, with an (exponentially) small probability, isomorphic graphs will be mis-classified.

The non-zero knowledge protocol had perfect completeness. This approach to zero-knowledge sacrifices perfect completeness. And with that I leave you to ponder.