IP and PSPACE Notes

Alex Creiner

March 26, 2023

An interactive proof system is a protocol between two machines, a **prover** P and a **verifier** V. have no bounds on it's time or space complexity. Since the prover is unbounded it makes no difference whether or not it is a deterministic machine, a quantum machine, For $\boldsymbol{IP}$, the verifier will operate in polynomial time, while the prover will or any other kind of machine. On the other hand the verifier is probabilistic in the sense that it has the ability to flip coins (which are private to the verifier). The prover and the verifier have a shared input x, and the prover attempts to *convince* the verifier that $x \in L$ (whether it is or not). The two machines send messages back and forth over special designated information channels. Eventually, the verifier decides to accept or reject. A language L is said to be decided by an **interactive proof system** (IPS) if the conditional probability that the verifier V accepts x given that $x \in L$ is at least $\frac{2}{3}$, and the conditional probability that the verifier V rejects x given that $x \notin L$ is also at least $\frac{2}{3}$. If the verifier operates in polynomial time, then the language L is in the class $\boldsymbol{IP}$. Note that $\boldsymbol{NP} \subseteq \boldsymbol{IP}$, since any $\boldsymbol{NP}$ machine can be seen as an interactive proof system in which the verifier doesn't use it's coin flips and the interaction is only the prover P sending a single message to V.

Consider the graph isomorphism problem - given two graphs G and H, are they isomorphic? Clearly this problem is in $\boldsymbol{NP}$ and is therefore also in $\boldsymbol{IP}$. The graph non-isomorphism problem is therefore in $\boldsymbol{co-NP}$ but not necessarily $\boldsymbol{NP}$ itself. But it *is* in $\boldsymbol{IP}$. We describe the $\boldsymbol{IP}$ protocol which decides it.

We will see the input x as a pair of graphs (G, H). As we said, both the prover and verifier have this input available to them. The verifier starts the process off. It uses it's random coin flips to create a random permutation of both graphs G and H. Call them G' and H' respectively. Recall that a permutation of a graph is always isomorphic to the graph itself. Thus $G \cong H$

Theorem 0.1. *For almost all oracles A, $\boldsymbol{IP}^A \neq \boldsymbol{PSPACE}^A$*

Proof. Our test language will be

$$L_A = \{1^n : \text{ the number of strings of length } n \text{ in } A \text{ is odd}\} \tag{1}$$

This is clearly always in $\boldsymbol{PSPACE}^A$. Fix a polynomial time verifier V (not a full protocol) with strict runtime n^c (by strict, we mean that this runtime is exact for *all* oracles A. Let $opt_V(A, x)$ denote the probability that V accepts an input x when paired with an optimal prover P. If V is part of an interactive proof system for some language L, then

$$x \in L \implies opt_V(A, x) > \frac{2}{3} \tag{2}$$

$$x \notin L \implies opt_V(A, x) < \frac{1}{3} \tag{3}$$

Note that the first of these is specific to the prover that V is paired with, while the second is true regardless of the prover. In other words, V has some question in mind to answer for itself regardless of the prover it's paired with, and has the capacity to scrutinize and reject bullshit arguments. It therefore suffices to pair V with the optimal prover P. If V doesn't decide some language L when paired with this optimal prover, then it isn't part of an interactive proof system at all. Conversely if V does form an interactive proof system when paired with an optimal prover, then that will be the same language for all provers which pair with V to make an optimal proof system.

Suppose we were to show that the set of oracles A such that V^A constitutes an interactive proof system for L_A (when paired with an optimal prover) has measure 0. Then the probability that *some* verifier constitutes

an interactive proof system for L_A is bounded below the sum of the probabilities that V^A decides L_A for all V, which is always 0, and therefore the probability that $L_A \in \boldsymbol{IP}^A$ is 0 as well by monotonicity (and the fact that there are countably many V's. Thus this suffices to complete the proof. We proceed to show that the set of oracles A such that V^A constitutes a proof system for L_A under the optimal prover P has measure 0.

To start, note that since the runtime n^c is strict, for a particular string x of length n, the computation of $V^A(x)$ depends only on strings up to (and not including) length n^c. Let A^{n^c} denote the 'initial segment' of A up to strings of length n^c, and so on. Then if $A^{n^c} = B^{n^c}$ then the computation of $V^B(x)$ and $V^A(x)$ are identical. Let $seg(n)$ denote the set of all subsets of strings of length less than n^c, i.e. the set of all possible A^{n^c}. Let $C(n)$ denote the set of all $\beta \in seg(n)$ such that V^β correctly decides 1^n, specifically. What we are going to essentially show is that $|C(n)| < \frac{2}{3}|seg(n)|$. This is intuitively very close to proving our desired result, since it shows that the probability of a random oracle correctly deciding 1^n is less than $\frac{2}{3}$. Then it appears that the probability of a random oracle correctly deciding *all* 1^n is the measure of the intersection of all of these probabilities, which is very likely 0. The issue, of course, is that these events need not be independent. We must show that given an initial oracle of some length n, that the probability of being in $C(n)$ is minimally dependent on $C(m)$ for $m > n$ at the same time.

Let N be large enough that $2^N > 18N^c$. For $\alpha \subseteq \{0,1\}^n$, let $B(\alpha, n)$ denote the collection of $\beta \in seg(n)$ such that β extends α up to length n^c. We first show the following lemma:

For all $n \geq N$ and for all $\alpha \in \{0,1\}^{<n}$, $|B(\alpha,n) \cap C(n)| \leq \frac{2}{3}|B(\alpha,n)|$. I.e. given *any* fixed α of length n, the set of finite extensions of length n^c which correctly decide $L_A(1^n)$ is $\frac{2}{3}$ of the overall set, as if the α hadn't been fixed at all.

On the input 1^n and with access to an oracle $\beta \in seg(n)$, the verifier V interacts with the optimal prover and makes queries of various strings to β. Let $Q(\beta, q)$ be the probability (over V's coin tosses) that $V(1^n)$ queries q. Since n^c is a strict upper bound on the running time of V, for every n and every sequence of coin tosses made by V^β on input 1^n, the machine makes less than n^c queries. Thus for every $\beta \in seg(n)$ we have

$$\sum_{q \in \{0,1\}^n} Q(\beta, q) = \sum_{q \in \{0,1\}^n} \sum_{\text{paths l}} Q(\beta, q|l)P(l) = \sum_{\text{paths l}} \sum_{q \in \{0,1\}^n} Q(\beta, q|l)P(l) \tag{4}$$

$$= \sum_{\text{paths l}} \sum_{q \text{ queried along l}} Q(\beta, q|l)\frac{1}{2^{n^c}} \tag{5}$$

$$\leq \sum_{\text{paths l}} \sum_{q \text{ queried along l}} 1 * \frac{1}{2^{n^c}} \tag{6}$$

$$\leq \sum_{\text{paths l}} \frac{n^c}{2^{n^c}} \tag{7}$$

$$= 2^{n^c}\frac{n^c}{2^{n^c}} = n^c \tag{8}$$

I don't know why I needed to explain it to myself this way but I did. Now suppose by way of contradiction that for all $q \in \{0,1\}^n$, a larger fraction of the β's in $B(\alpha, n)$ than $\frac{3n^c}{2^n}$ have $Q(\beta, q) \leq \frac{1}{3}$. Then

$$\sum_{\beta \in B(\alpha,n)} \sum_{q \in \{0,1\}^n} Q(\beta, q) \leq \sum_{\beta \in B(\alpha,n)} n^c = |B(\alpha, n)|n^c \tag{9}$$

By what we showed above. But additionally by our hypothesis we have

$$\sum_{\beta\in B(\alpha,n)}\sum_{q\in\{0,1\}^n} Q(\beta,q) = \sum_{q\in\{0,1\}^n}\sum_{\beta\in B(\alpha,n)} Q(\beta,q) \tag{10}$$

$$> \sum_{q\in\{0,1\}^n} \frac{3n^c}{2^n}|B(\alpha,n)|\frac{1}{3} \tag{11}$$

$$= \sum_{q\in\{0,1\}^n} \frac{n^c}{2^n}|B(\alpha,n)| = n^c|B(\alpha,n)| \tag{12}$$

Thus we have a quantity which is simultaneously bigger and smaller than $n^c|B(\alpha,n)|$, a contradiction. Thus we can be sure there exists a particular $q\in\{0,1\}^n$ such that for all but a $\frac{3n^c}{2^n}$ fraction of the β's in $B(\alpha,n)$, $Q(\beta,q)\leq\frac{1}{3}$. Fix such a string q.

Now, let β be an oracle in $B(\alpha,n)$ such that $1^n\in L_\beta$. and let $\beta^{(q)}$ denote the oracle which contains all strings except for q. We claim that

$$opt_V(\beta^{(q)},1^n)\geq opt_V(\beta,1^n)-Q(\beta,q) \tag{13}$$

It is clear that the difference should be small since by definition of q it is mostly irrelevent for most of the β's, but to understand this exact statement, consider the prover P' which uses the same strategy on $V^\beta(1^n)$ as whatever the optimal prover P would use for $V^{\beta^{(q)}}(1^n)$. On the computational paths in which $V^{\beta^{(q)}}(1^n)$ would never query q, P' would behave identically to the actual optimal prover for V^β. But we know that only a $Q(\beta,q)$ fraction of the paths ask about q by definition.

$$opt_V(\beta^{(q)},1^n) = P(V^{\beta^{(q)}}\text{ accepts}) \tag{14}$$

$$\geq P(V^{\beta^{(q)}}(1^n)\text{ accepts under optimal prover and doesn't query q}) \tag{15}$$

$$= P(V^{\beta^{(q)}}(1^n)\text{ accepts under optimal prover}) - P(V^{\beta^{(q)}}\text{ accepts under optimal prover and does query q}) \tag{16}$$

$$\geq P(V^{\beta}(1^n)\text{ accepts under optimal prover}) - P(V^{\beta^{(q)}}\text{queries q}) \tag{17}$$

$$= opt_V(\beta,1^n)-Q(\beta,q) \tag{18}$$

Now note that if $1^n\in L_\beta$ then $1^n\notin L_{\beta^{(q)}}$ by definition of our test language. We now exploit this. We claim that whenever $Q(\beta,q)<\frac{1}{3}$, the verifier V must be incorrect in determining membership for either L_β or $L_{\beta^{(q)}}$. To see this, assume such a q and suppose that $V^\beta(1^n)$ accepts and $1^n\in L_\beta$. Then $opt_V(\beta,1^n)>\frac{2}{3}$. But then by our identity we have that $opt_V(\beta^{(q)},1^n)>\frac{1}{3}$, meaning that $V^{\beta^{(q)}}$ is failing to properly discern that 1^n is not in $L_{\beta^{(q)}}$. Assume $V^\beta(1^n)$ rejects and $1^n\notin L_\beta$. For both V^β and $V^{\beta^{(q)}}$ to be correct, we must have then that $V^{\beta^{(q)}}(1^n)$ accepts, i.e. has a probability of acceptance greater than $\frac{2}{3}$. But then by an identical argument we get that the probability of V^β itself accepting is too high, so that it is failing to discern non-membership just as $V^{\beta^{(q)}}$ was in the previous case. Thus regardless of the situation, one of these machines is making an error.

By our choice of q, $Q(\beta,q)<\frac{1}{3}$ for at least a $1-\frac{3n^c}{2^n}$ fraction of the $B(\alpha,n)$. In the worst case, every pair $(\beta,\beta^{(q)})$ will be jointly have this property, halving our lower bound on the set of pairs $(\beta,\beta^{(q)})$ which have that property. So at least a $1-2\frac{3n^c}{2^n}$ fraction of the pairs have the property that $Q(\beta,q)<\frac{1}{3}$. Note that by our choice of n such that $2^n>18n^c$, we have

$$1-\frac{6n}{2^n}>1-\frac{6}{18}=\frac{2}{3} \tag{19}$$

So this is at least $\frac{2}{3}$ of the pairs. Thus when grouping the elements of $B(\alpha,n)$ into pairs, V fails on one or the other of the elements, and thus fails on $\frac{1}{2}\frac{2}{3}=\frac{1}{3}$ of all elements of $B(\alpha,n)$ itself.

This proves the lemma and thus the bulk of the proof. We just need to send it off. The following lemma allows us to cascade our results downward as we need to:

Let $n_i = N^{c^i}$ for some i (with N and c as we've already defined them), and let R_i be the collection of finite sets $\beta \in seg(n_i)$ such that for all $r \leq n_i$, V^β correctly determines whether 1^r is in L_β. Then for all $i \geq 0$, $|R_i| \leq \left(\frac{2}{3}\right)^i |seg(n_i)|$

test □

Lemma 0.1. *Let A be a tame oracle. Then $\boldsymbol{NP}^A \subseteq \boldsymbol{PSPACE}^A \subseteq \boldsymbol{IP}^A$.*

Proof. Our first task will be to show that the operation of any relativized machine under these circumstances are still both space bounded in a certain sense. In the case of $\boldsymbol{PSPACE}^A$ this is obviously true in any sense by definition. In the case of $\boldsymbol{IP}^A$, consider a protocol defined by a verifier V paired with it's optimal prover. This verifier is a nondeterministic machine operating in polynomial time. In polynomial time it can only query polynomial length strings, and so remains space bounded in it's operation. Thus $\boldsymbol{IP}^A$ can only query strings up to it's runtime, say n^k. This will be important.

Now, let M^A be a $\boldsymbol{PSPACE}^A$ machine. Note that the operation of this machine can be broken down into the operation of a relativized $\boldsymbol{NP}^A$ machine followed by that of a non-relativized $\boldsymbol{PSPACE}$ machine. Let M^A operate in space n^k. Then it can only query strings of length up to n^k. A relativized $\boldsymbol{NP}^A$ machine can create a lookup table by simply answering the question 'does there exist a string of length n in A' by guessing a string in the oracle for each length between 1 and n^k. If it can't guess a string, then there are no strings in the oracle of that length. If it can guess the right string, then it can add that to the lookup table and be content that this is all of the strings in A of that length since A is tame. (Note that tameness is crucial here. If we didn't have this then it would not be clear that a lookup table could be created without access to a $\boldsymbol{coNP}^A$ machine, and $\boldsymbol{coNP}^A \subseteq \boldsymbol{IP}^A$ is famously a non-relativizing result!) Now the operation of the non-relativized machine which decides the same language as M^A using the lookup table can be also decided by a non-relativized $\boldsymbol{IP}$ protocol, since $\boldsymbol{PSPACE} \subseteq \boldsymbol{IP}$. Moreover $\boldsymbol{NP}^A \subseteq \boldsymbol{IP}^A$ as well since $\boldsymbol{NP}^A$ is simply the special case of $\boldsymbol{IP}$ where the prover and verifier only communicate once. Thus there is an $\boldsymbol{IP}^A$ protocol for creating the same lookup table. We thus have an $\boldsymbol{IP}^A$ protocol for creating the lookup table and an $\boldsymbol{IP}$ protocol for using that lookup table to arrive at the same answers as M^A. Thus $L(M^A) \in \boldsymbol{IP}^A$, and we have that $\boldsymbol{PSPACE}^A \subseteq \boldsymbol{IP}^A$. □

Now let

$$\boldsymbol{I} = \{A : \boldsymbol{PSPACE}^A \subseteq \boldsymbol{IP}^A\}$$

It immediately follows from the above result that $\boldsymbol{I}$ is Ramsey positive. Since $\boldsymbol{I}$ is clearly Borel like the others, any infinite oracle has to contain a suboracle for either $\boldsymbol{I}$ or it's complement. Consider, therefore, any infinite oracle A. Every infinite oracle has a tame suboracle B. Moreover all subsets of a tame oracle are tame. Thus this B has to be homogeneous for $\boldsymbol{I}$. We can go further than this however. Let $[a, A]$ be a basic open Ellentuck set. The body A clearly has a tame sub-body B, while the strings represented by the finite initial stem a can simply be hardcoded into any machine. Thus $[a, B]$ is homogeneous for $\boldsymbol{I}$. It follows that $\boldsymbol{I}$ is Ramsey 1.

Theorem 0.2. *$\boldsymbol{I}$ is Ramsey* 1*.*

Lemma 0.2. *Let A be a tame oracle, and $\boldsymbol{NTIME}(f(n)) \subseteq \boldsymbol{C} \subseteq \boldsymbol{SPACE}(f(n))$ for some complexity class $\boldsymbol{C}$. Then $\boldsymbol{NTIME}(f(n))^A \subseteq \boldsymbol{C}^A$. Moreover the actual relationship between two classes both containing $\boldsymbol{NTIME}(f(n))$ and contained within $\boldsymbol{SPACE}(f(n))$ will remain the same relativized to A.*

Proof. Let □

Proof. What we were showing with the above example is just a special case of this. To have a tame oracle is to have an oracle which can't be 'squeezed' for usefulness □

How to interpret these results? Consider the above construction, in which we took a complexity class which always contained $\boldsymbol{NP}^A$. In this situation