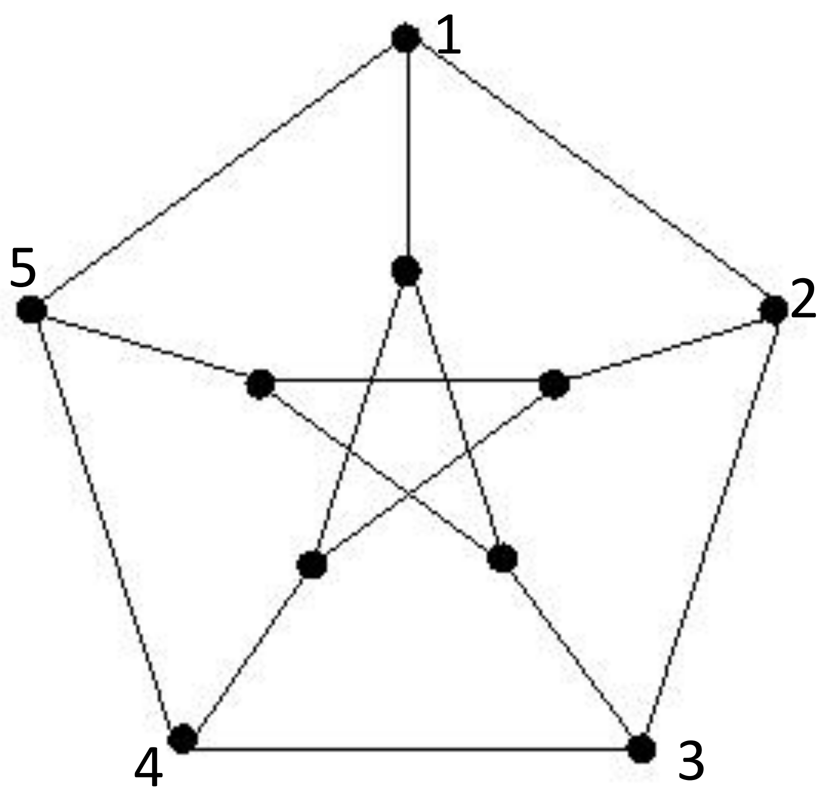
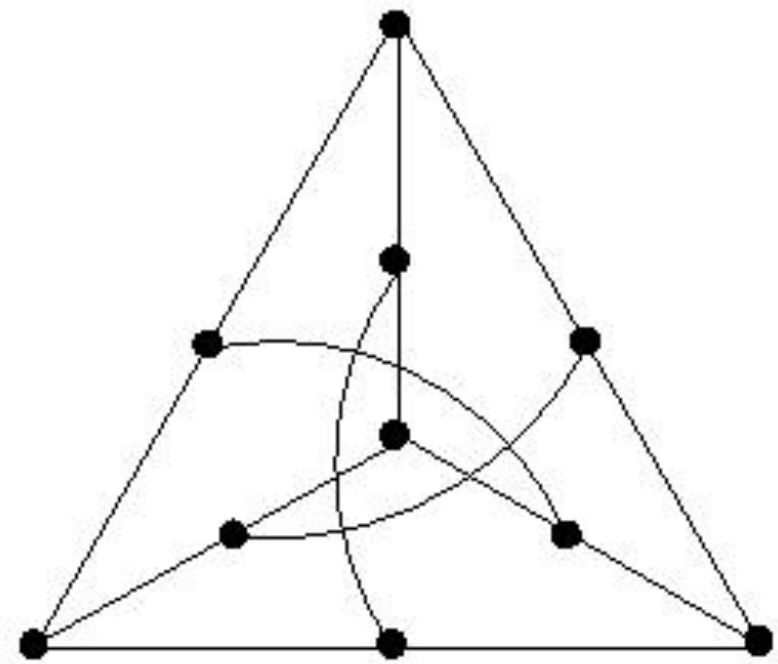
Interactive Proof Systems (IPS's)

Interactive Proof Systems (IPS's)

- NP is the class of languages for which membership can be proved to a deterministic, polynomial-time verifier.
- Can membership in languages outside of NP be proved to a poly-time verifier that can flip coins?

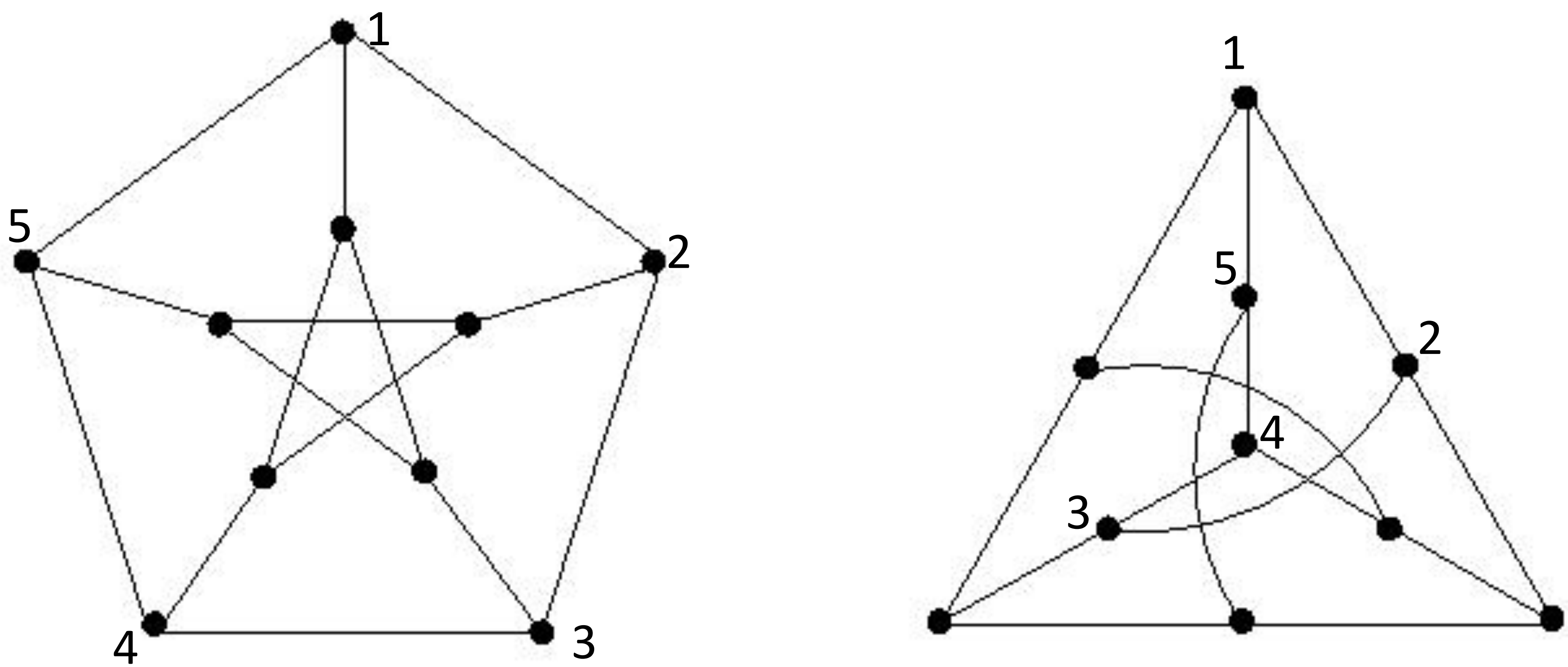
Graph Isomorphism





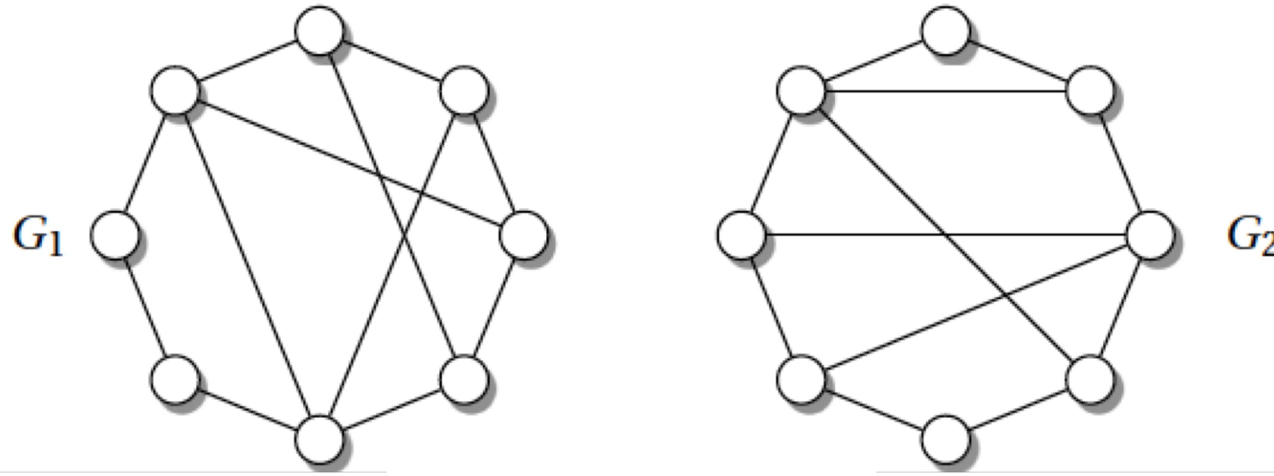
Given two undirected graphs with an equal number of nodes and edges, can the labels of nodes in one graph be permuted to obtain the second graph?

Graph Isomorphism



Given two undirected graphs with an equal number of nodes and edges, can the labels of nodes in one graph be permuted to obtain the second graph?

Graph Non-Isomorphism



How to prove that two graphs are *not* isomporphic?

IPS for Graph Non-Isomorphism

Input: two graphs G_1, G_2

Repeat, say 10 times

- Verifier: Choose one graph, say G_i, at random
  Randomly permute G_i to obtain G'
  Send G' to the prover
- Prover: Send either 1 or 2 to the verifier
- Verifier: Reject if the number sent is not i

Verifier: Accept

IPS for Graph Non-Isomorphism

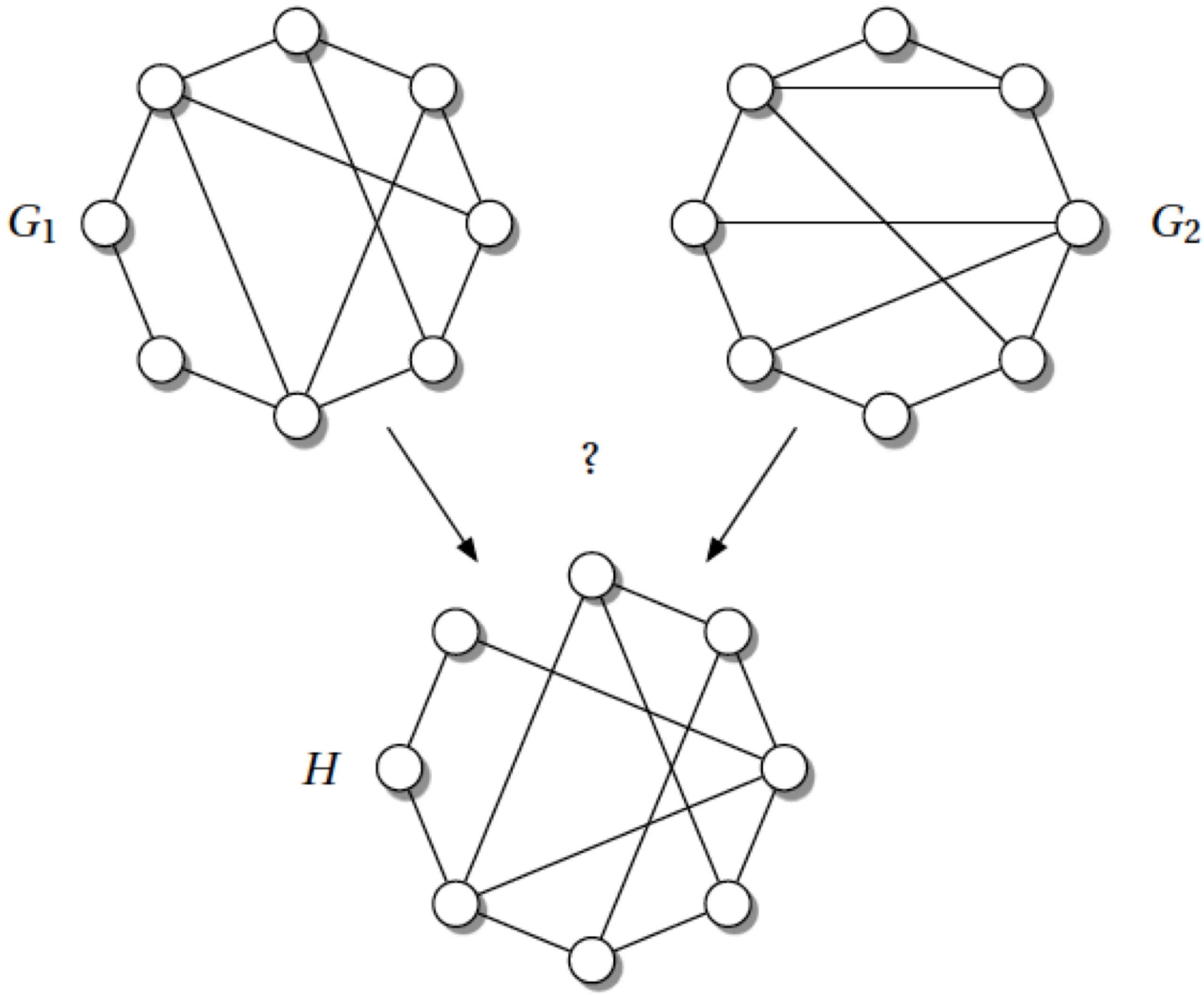


FIGURE 11.3: Arthur generates H by applying a random permutation to either G_1 or G_2. If Merlin can consistently tell which graph Arthur started with (can you?) then G_1 and G_2 are nonisomorphic.

IPS for Graph Non-Isomorphism

- In the graph non-isomorphism protocol, the verifier uses *private coins*, i.e. the prover does not see the verifier's random bits
- The protocol would not be correct if the coins were *public*

- It turns out, however, that there is a public-coin interactive proof for graph non-isomorphism that uses public coins
- In fact, all languages in PSPACE have public-coin interactive proofs

Interactive Proof System Definition

- An interactive proof system (IPS) is a Turing machine whose non-halting states are partitioned into two types: existential/guessing and coin-flipping
- There are exactly two possible next steps from each non-halting state

Interactive Proof System Definition

- Let M be an IPS that always halts, and let C be a configuration of M. C is either an existential, coin-flipping, accepting, or rejecting configuration depending on its state.
- Let $\text{Prob}_a[C]$ denote the probability of reaching an accepting configuration from C

Interactive Proof System Definition

$\text{Prob}_a[C]$ can be defined recursively as follows:

- C is rejecting: $\text{Prob}_a[C] = 0$
- C is accepting: $\text{Prob}_a[C] = 1$

Otherwise let C′ and C″ be the two configurations reachable from C

- C is existential: $\text{Prob}_a[C] = \max \{ \text{Prob}_a[C'], \text{Prob}_a[C''] \}$
- C is coin-flipping: $\text{Prob}_a[C] = (\text{Prob}_a[C'] + \text{Prob}_a[C''])/2$

Interactive Proof System Definition

Let Prob[M accepts x] be $Prob_a[C_0]$, where C_0 is the initial configuration of M on x. We say that the IPS M accepts language L with bounded error if:

- for all $x \in L$, Prob[M accepts x] $\geq 2/3$, and
- for all $x \notin L$, Prob[M accepts x] $\leq 1/3$

- IP is the class of languages accepted by polynomial time bounded IPS's

An IPS for TQBF

TQBF: given a quantified Boolean formula (QBF) φ, is it valid?

Idea: *Arithmetize* φ to obtain an arithmetic expression $A\varphi$, and test if $A_{\varphi} > 0$

Arithmetizing QBFs

- Replace each $\vee$ by $+$, $\wedge$ by $\cdot$ (times)
- Replace $\neg x$ by $(1-x)$
- Replace $\exists x$ by $\sum_{x\in\{0,1\}}$ and $\forall x$ by $\prod_{x\in\{0,1\}}$
- Replace true or false with 1 or 0

Arithmetizing QBFs

- Replace each $\vee$ by $+$, $\wedge$ by $\cdot$ (times)
- Replace $\neg x$ by $(1-x)$
- Replace $\exists x$ by $\sum_{x\in\{0,1\}}$ and $\forall x$ by $\prod_{x\in\{0,1\}}$
- Replace true or false with 1 or 0

$$\phi = \forall x \exists y[(x \vee y) \wedge \forall z[(x \wedge z) \vee (y \wedge \bar{z}) \vee \exists w(z \vee (y \wedge \bar{w}))]]$$

$$A_\phi = \prod_{x=0}^{1} \sum_{y=0}^{1} [(x+y) \cdot \prod_{z=0}^{1} [(x \cdot z + y \cdot (1-z)) + \sum_{w=0}^{1} (z + y \cdot (1-w))]]$$

Arithmetizing QBFs

- Replace each $\vee$ by $+$, $\wedge$ by $\cdot$ (times)
- Replace $\neg x$ by $(1-x)$
- Replace $\exists x$ by $\sum_{x\in\{0,1\}}$ and $\forall x$ by $\prod_{x\in\{0,1\}}$
- Replace true or false with 1 or 0

- Claim: If A_φ is the arithmetization of φ, then φ is valid if and only if $A_\varphi > 0$.

Arithmetizing QBFs

- Replace each $\vee$ by $+$, $\wedge$ by $\cdot$ (times)
- Replace $\neg x$ by $(1-x)$
- Replace $\exists x$ by $\sum_{x\in\{0,1\}}$ and $\forall x$ by $\prod_{x\in\{0,1\}}$
- Replace true or false with 1 or 0

- Claim: If A_φ is the arithmetization of φ, then φ is valid if and only if $A_\varphi > 0$.

  Proof: By induction over the number of $\sum$, $\prod$, $\vee$, and $\wedge$ in the formula A_φ

An IPS for TQBF

TQBF: given a quantified Boolean formula (QBF) φ, is it valid?

Idea:

- *Arithmetize* φ to obtain an arithmetic expression $A\varphi$, and test if $A_\varphi > 0$
- Testing if $A_\varphi > 0$ sounds no easier than evaluating φ
- The IPS will leverage nice properties of polynomials

An IPS to test if $A_\varphi > 0$

(very rough sketch)

$$A_\phi = \prod_{x=0}^{1} \sum_{y=0}^{1} [(x+y) \cdot \prod_{z=0}^{1} [(x \cdot z + y \cdot (1-z)) + \sum_{w=0}^{1} (z + y \cdot (1-w))]]$$

Prover: "$A_\varphi = 96$"

Verifier: Let $A_\varphi = \prod_{x\in\{0,1\}} A_1(x)$. What is $A_1(x)$?

Prover: "$A_1(x)$ is $\alpha_1(x) = 2x^2 + 8x + 6$"

Verifier:

An IPS to test if $A_\varphi > 0$

(very rough sketch)

$$A_\phi = \prod_{x=0}^{1} \sum_{y=0}^{1} [(x+y) \cdot \prod_{z=0}^{1} [(x \cdot z + y \cdot (1-z)) + \sum_{w=0}^{1} (z + y \cdot (1-w))]]$$

Prover: "$A_\varphi = 96$"

Verifier: Let $A_\varphi = \prod_{x \in \{0,1\}} A_1(x)$. What is $A_1(x)$?

Prover: "$A_1(x)$ is $\alpha_1(x) = 2x^2 + 8x + 6$"

Verifier:

- Check that $\alpha_1(0) . \alpha_1(1) = 96$
- Check that indeed $A_1(x) = \alpha_1(x)$, i.e., that the prover isn't cheating, by plugging in a random number r for x and using recursion

An IPS to test if $A_\varphi > 0$

Problems with this IPS idea:

- The degree of polynomial $\alpha_1(x)$ (or equivalently, $A_1(x)$) could be exponential in $|A_\varphi|$
- The value of A_φ could be double exponential in $|A_\varphi|$

An IPS to test if $A_\varphi > 0$

Problems with this IPS idea:

- The degree of polynomial $\alpha_1(x)$ (or equivalently, $A_1(x)$) could be exponential in $|A_\varphi|$

$$\forall x_1 \forall x_2 \ldots \forall x_n \exists y (y \vee x_1)$$

- The value of A_φ could be double exponential in $|A_\varphi|$

$$\forall x_1 \forall x_2 \ldots \forall x_n \exists y (y \vee \bar{y})$$

An IPS to test if $A_\varphi > 0$

Problems with this IPS idea:

- The degree of polynomial $\alpha_1(x)$ (or equivalently, $A_1(x)$) could be exponential in $|A_\varphi|$

- The value of A_φ could be double exponential in $|A_\varphi|$

An IPS to test if $A_\varphi > 0$

Problems with this IPS idea:

- The degree of polynomial $\alpha_1(x)$ (or equivalently, $A_1(x)$) could be exponential in $|A_\varphi|$
  - *Solution*: convert φ to an equivalent *simple* qbf before arithmetizing
- The value of A_φ could be double exponential in $|A_\varphi|$

An IPS to test if $A_\varphi > 0$

Problems with this IPS idea:

- The degree of polynomial $\alpha_1(x)$ (or equivalently, $A_1(x)$) could be exponential in $|A_\varphi|$
  - *Solution*: convert φ to an equivalent *simple* qbf before arithmetizing
- The value of A_φ could be double exponential in $|A_\varphi|$
  - *Solution*: perform polynomial evaluations modulo a small prime

Simple QBFs

- Call a variable x of ϕ *simple* if x is in the scope of at most one ∀ quantifier that is within the scope of the quantifier of ϕ to which x is bound
- A qbf ϕ is simple if all variables of ϕ are simple

- *Exercise*: which of these is simple?

$$\forall x[\ \forall y((x \vee y) \wedge (\forall z(x \wedge z)))\ ]$$

$$\forall x[\ (\forall y(x \vee y)) \wedge (\forall z(x \wedge z))\ ]$$

Simple QBFs

- Call a variable x of ϕ *simple* if x is in the scope of at most one $\forall$ quantifier that is within the scope of the quantifier of ϕ to which x is bound
- A qbf ϕ is simple if all variables of ϕ are simple

- *Simplification Lemma*: Given an instance φ' of TQBF, we can convert φ' into a simple instance φ in polynomial time, such that φ' is valid if and only if φ is valid

Simple QBFs

- Call a variable x of ϕ *simple* if x is in the scope of at most one ∀ quantifier that is within the scope of the quantifier of ϕ to which x is bound
- A qbf ϕ is simple if all variables of ϕ are simple

- *Low-Degree Lemma*: Let φ be simple, and let $A_\varphi = Qx\ \alpha(x)$. Then the degree of $\alpha(x)$ is at most $2|\alpha|$.

Summary

- Our goal is to show that TQBF is in IP
- Ideas so far:
  - Prover will help verifier evaluate an arithmetization of the TQBF instance
  - WLOG, work with *simple* qbf instances
  - Arithmetizations of simple qbf's can be expressed as low-degree polynomials
  - Polynomial evaluation can be done modulo primes to avoid working with large values
- To be continued…