

Interactive Proofs

Lecture 17
IP = PSPACE

So far

So far

- IP

So far

- IP
- AM, MA

So far

- IP
- AM, MA
- $GNI \in IP$

So far

- IP
- AM, MA
- $GNI \in IP$
- $GNI \in AM$

So far

- IP
- AM, MA
- $GNI \in IP$
- $GNI \in AM$
 - Using AM protocol for set lower-bound

So far

- IP
- AM, MA
- $\text{GNI} \in \text{IP}$
- $\text{GNI} \in \text{AM}$
 - Using AM protocol for set lower-bound
 - In fact, $\text{IP}[k]$ in $\text{AM}[k+2]$

IP = PSPACE

IP = PSPACE

- Recall, IP means IP[poly]

IP = PSPACE

- Recall, IP means IP[poly]
- $IP \subseteq PSPACE$

IP = PSPACE

- Recall, IP means IP[poly]
- $IP \subseteq PSPACE$
 - Even though prover unbounded, cannot convince poly time verifier of everything

IP = PSPACE

- Recall, IP means IP[poly]
- $IP \subseteq PSPACE$
 - Even though prover unbounded, cannot convince poly time verifier of everything
- $PSPACE \subseteq IP$

IP = PSPACE

- Recall, IP means IP[poly]
- $IP \subseteq PSPACE$
 - Even though prover unbounded, cannot convince poly time verifier of everything
- $PSPACE \subseteq IP$
 - Prover can convince verifier of high complexity statements

$IP \subseteq PSPACE$

$IP \subseteq PSPACE$

- Easier direction!

IP \subseteq PSPACE

- Easier direction!
- Plan: For given input calculate $\Pr[\text{yes}]$ of honest verifier, maximum over all "prover strategies"

IP \subseteq PSPACE

- Easier direction!
- Plan: For given input calculate $\Pr[\text{yes}]$ of honest verifier, maximum over all “prover strategies”
 - Warm-up: public-coins (i.e., AM[poly])

IP \subseteq PSPACE

- Easier direction!
- Plan: For given input calculate $\Pr[\text{yes}]$ of honest verifier, maximum over all “prover strategies”
 - Warm-up: public-coins (i.e., AM[poly])
 - Could then use the “fact” that $\text{IP}[\text{poly}] = \text{AM}[\text{poly}]$

IP \subseteq PSPACE

- Easier direction!
- Plan: For given input calculate $\Pr[\text{yes}]$ of honest verifier, maximum over all “prover strategies”
 - Warm-up: public-coins (i.e., $\text{AM}[\text{poly}]$)
 - Could then use the “fact” that $\text{IP}[\text{poly}] = \text{AM}[\text{poly}]$
 - Or modify the proof (as we’ll do)

$AM[\text{poly}] \subseteq PSPACE$

$AM[\text{poly}] \subseteq PSPACE$

- Plan: For given input calculate $\max \Pr[\text{yes}]$ over all “prover strategies”

$AM[\text{poly}] \subseteq PSPACE$

- Plan: For given input calculate $\max \Pr[\text{yes}]$ over all “prover strategies”
 - Assume for convenience (w.l.o.g) each message is a single bit and P, V alternate

$AM[\text{poly}] \subseteq PSPACE$

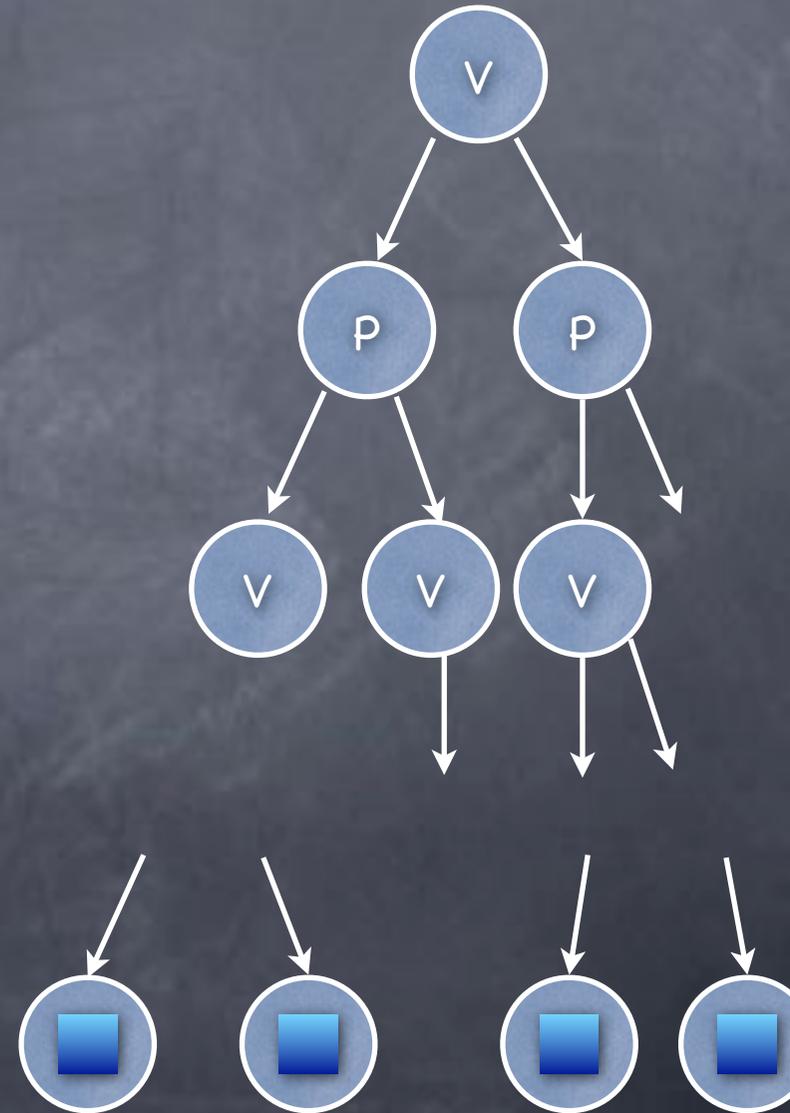
- Plan: For given input calculate $\max \Pr[\text{yes}]$ over all “prover strategies”
 - Assume for convenience (w.l.o.g) each message is a single bit and P, V alternate
 - Since public-coin, V messages are simply uniform random bits

$AM[\text{poly}] \subseteq PSPACE$

- Plan: For given input calculate $\max \Pr[\text{yes}]$ over all “prover strategies”
 - Assume for convenience (w.l.o.g) each message is a single bit and P, V alternate
 - Since public-coin, V messages are simply uniform random bits
 - Protocol’s configuration tree: path to a node corresponds to the transcript so far

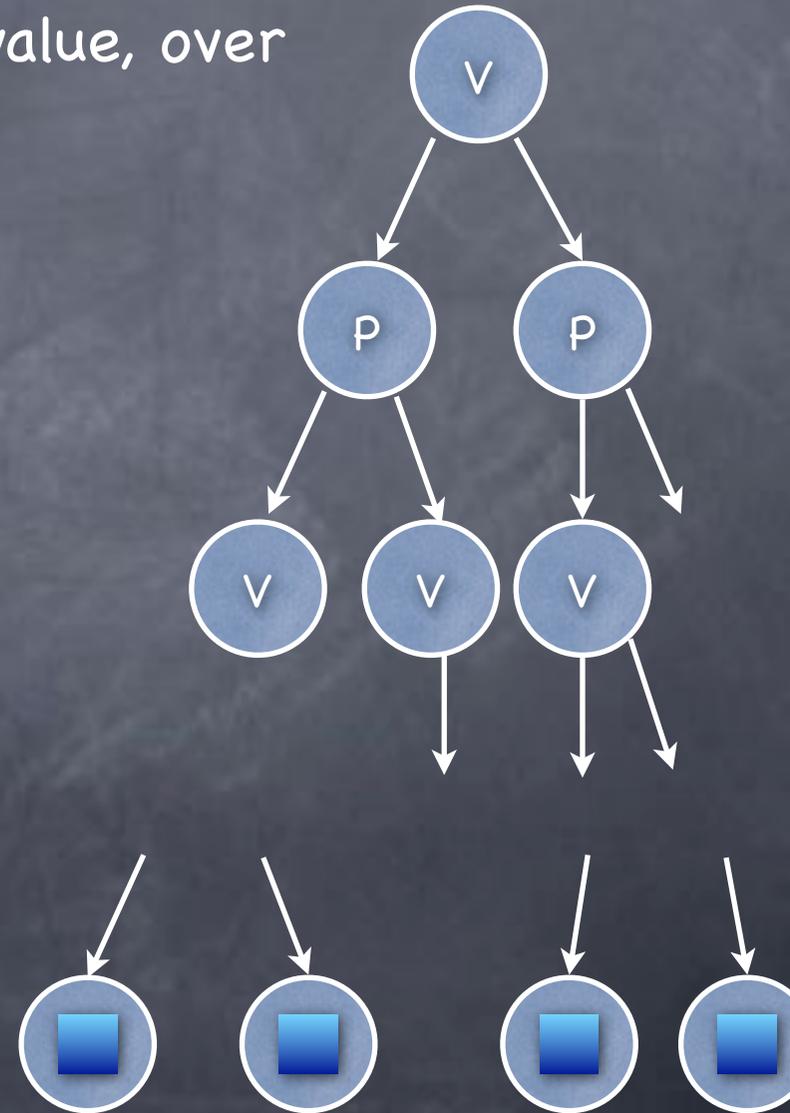
$AM[\text{poly}] \subseteq PSPACE$

- Plan: For given input calculate $\max \Pr[\text{yes}]$ over all “prover strategies”
 - Assume for convenience (w.l.o.g) each message is a single bit and P, V alternate
 - Since public-coin, V messages are simply uniform random bits
 - Protocol’s configuration tree: path to a node corresponds to the transcript so far



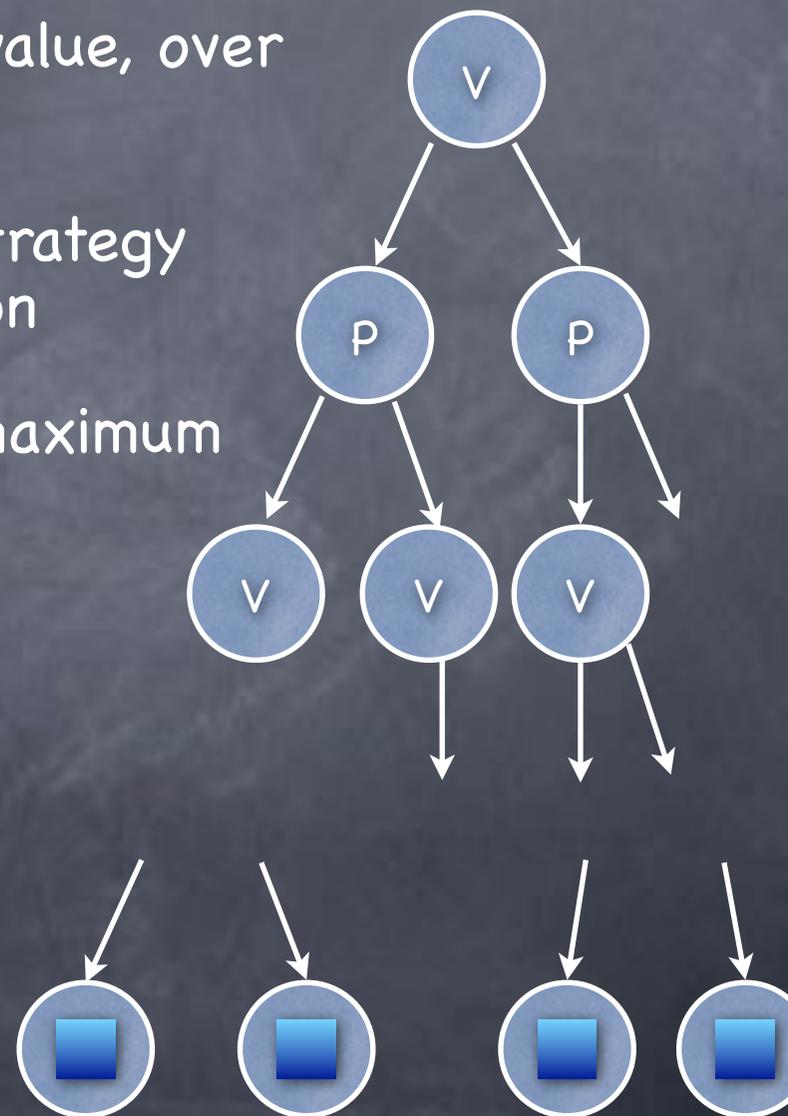
$AM[\text{poly}] \subseteq PSPACE$

- Plan: For given input calculate maximum value, over **all** "prover strategies," of $\Pr[\text{yes}]$



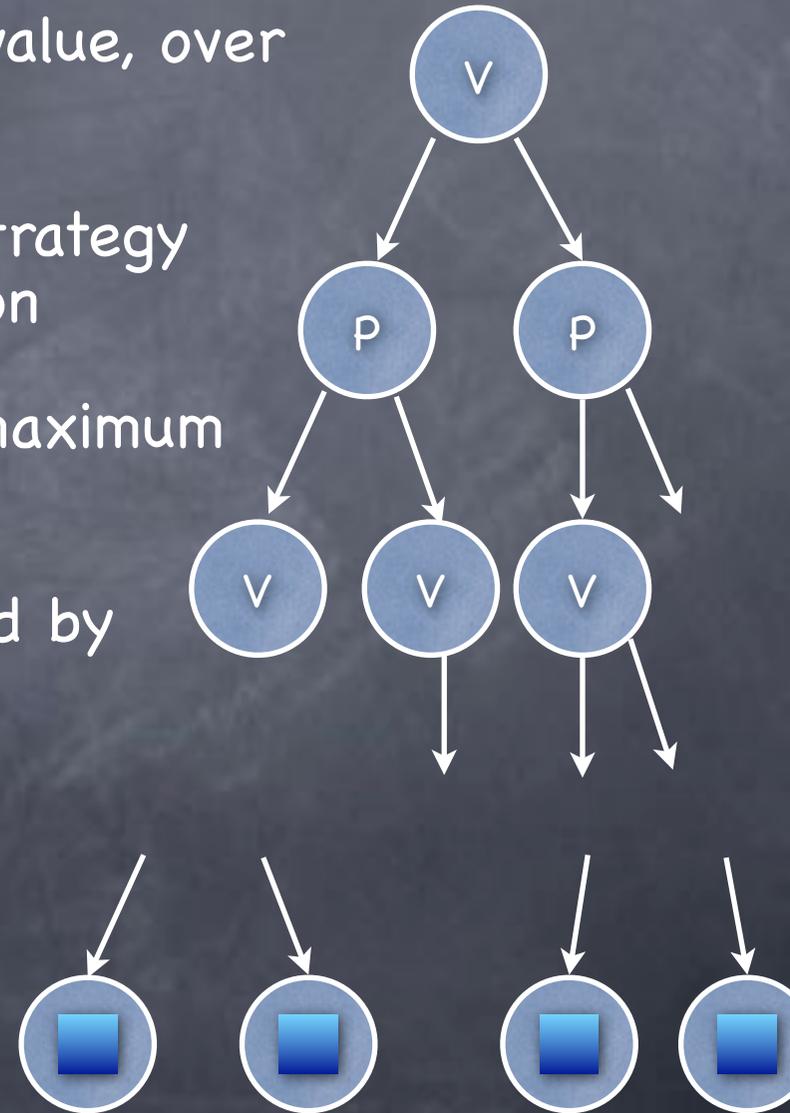
$AM[\text{poly}] \subseteq PSPACE$

- Plan: For given input calculate maximum value, over **all** "prover strategies," of $\Pr[\text{yes}]$
 - Note that finding the honest prover strategy may require super-PSPACE computation
 - Recursively for each node, calculate maximum $\Pr[\text{yes}]$



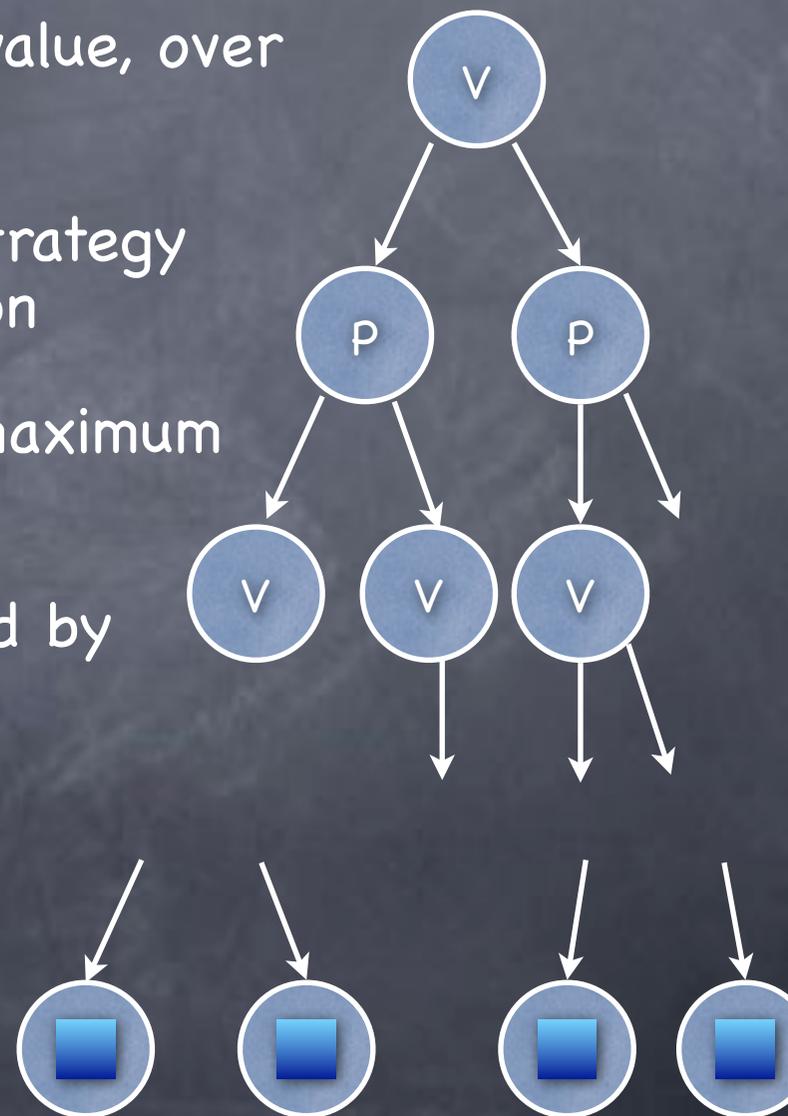
$AM[\text{poly}] \subseteq PSPACE$

- Plan: For given input calculate maximum value, over **all** "prover strategies," of $\Pr[\text{yes}]$
 - Note that finding the honest prover strategy may require super-PSPACE computation
 - Recursively for each node, calculate maximum $\Pr[\text{yes}]$
 - Leaves: $\Pr[\text{yes}] = 0$ or 1 , determined by running verifier's program



$AM[\text{poly}] \subseteq PSPACE$

- Plan: For given input calculate maximum value, over **all** "prover strategies," of $\Pr[\text{yes}]$
 - Note that finding the honest prover strategy may require super-PSPACE computation
 - Recursively for each node, calculate maximum $\Pr[\text{yes}]$
 - Leaves: $\Pr[\text{yes}] = 0$ or 1 , determined by running verifier's program
 - P nodes: max of children
 - V nodes: average of children



$AM[\text{poly}] \subseteq PSPACE$

- Plan: For given input calculate maximum value, over **all** "prover strategies," of $\Pr[\text{yes}]$

- Note that finding the honest prover strategy may require super-PSPACE computation

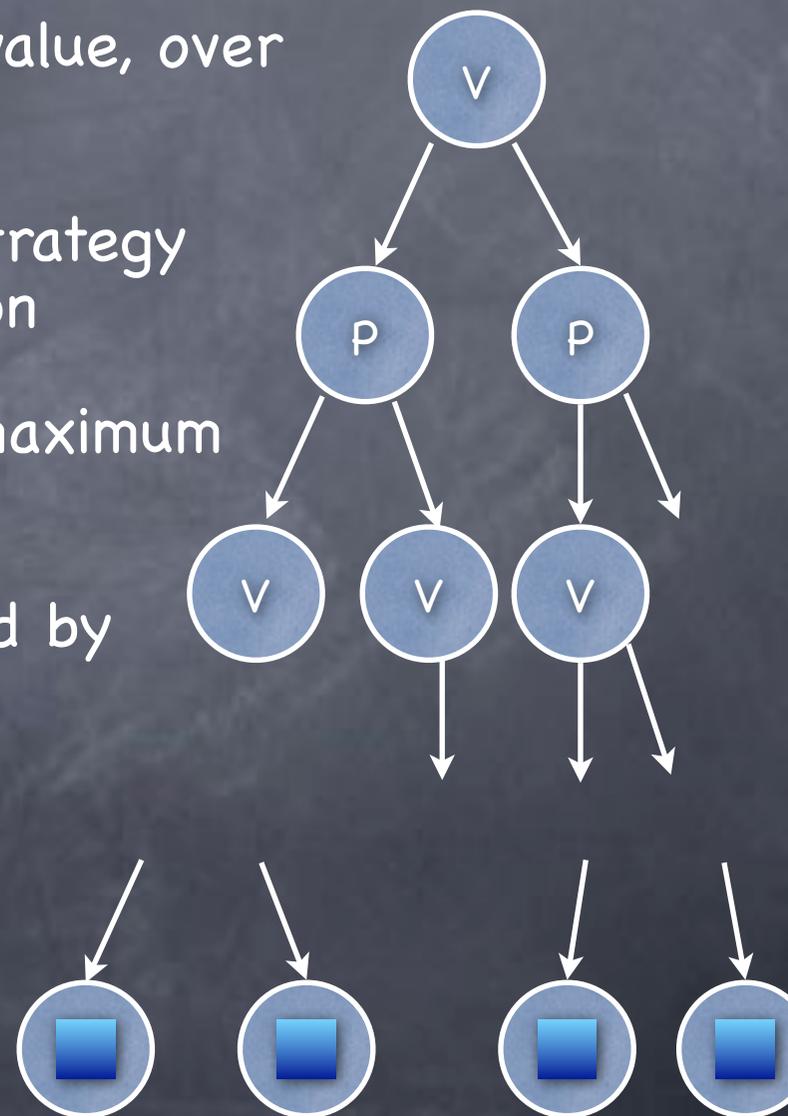
- Recursively for each node, calculate maximum $\Pr[\text{yes}]$

- Leaves: $\Pr[\text{yes}] = 0$ or 1 , determined by running verifier's program

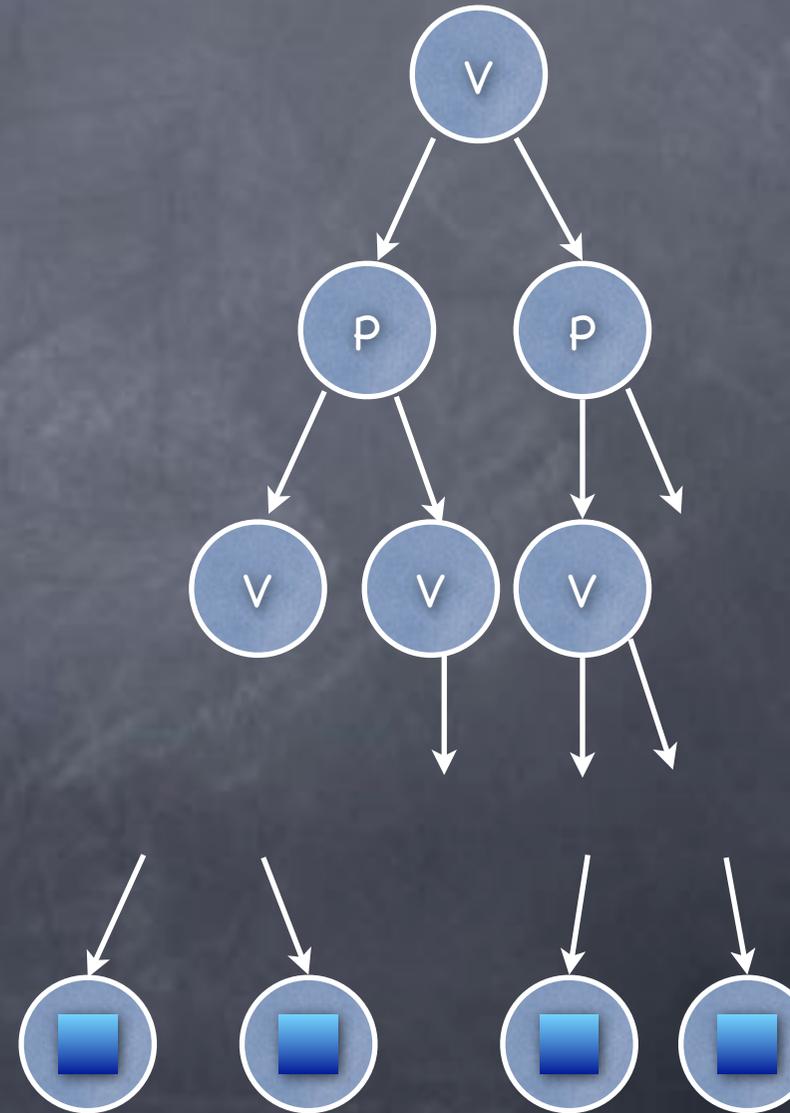
- P nodes: max of children

- V nodes: average of children

- In PSPACE: depth polynomial

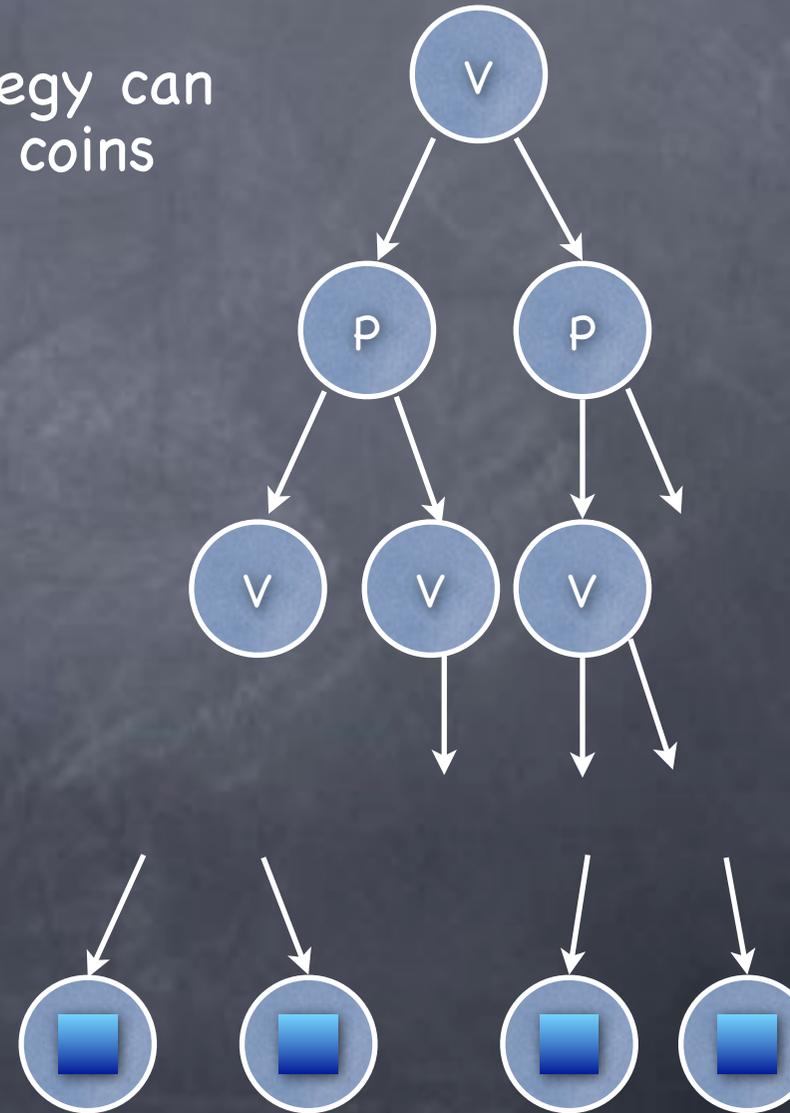


$IP \subseteq PSPACE$



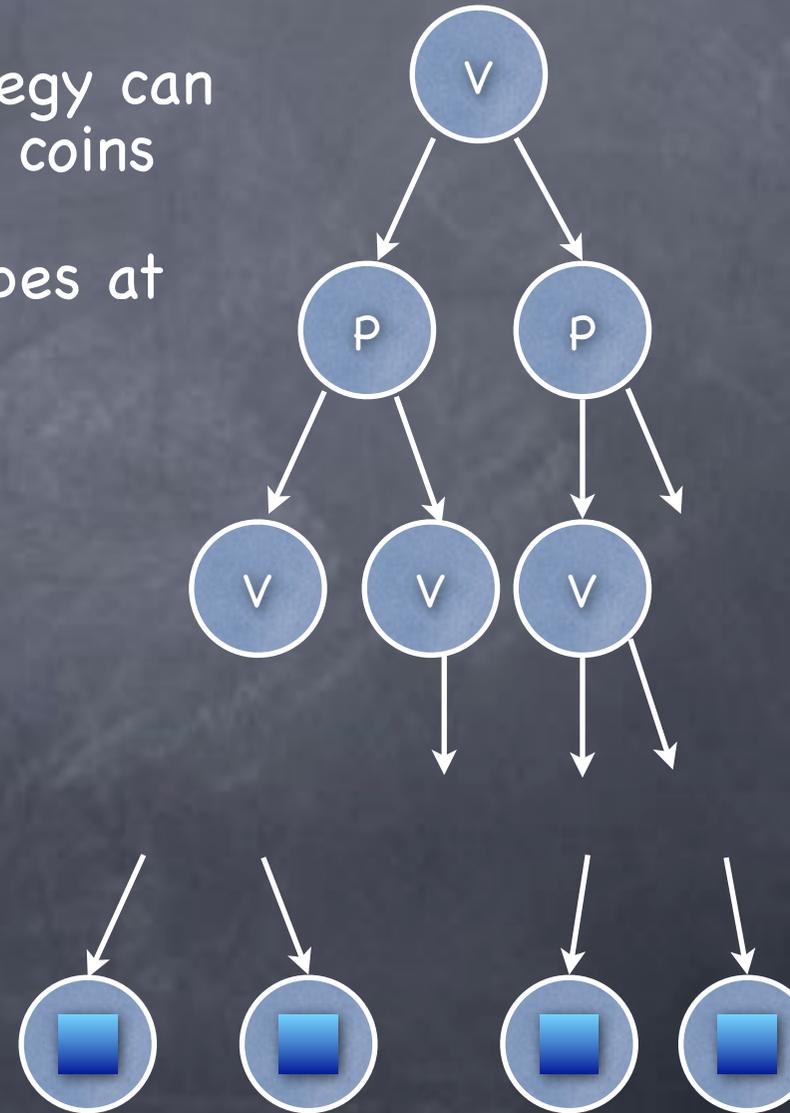
$IP \subseteq PSPACE$

- Calculate $\max \Pr[\text{yes}]$ when prover's strategy can depend only on messages and not private coins



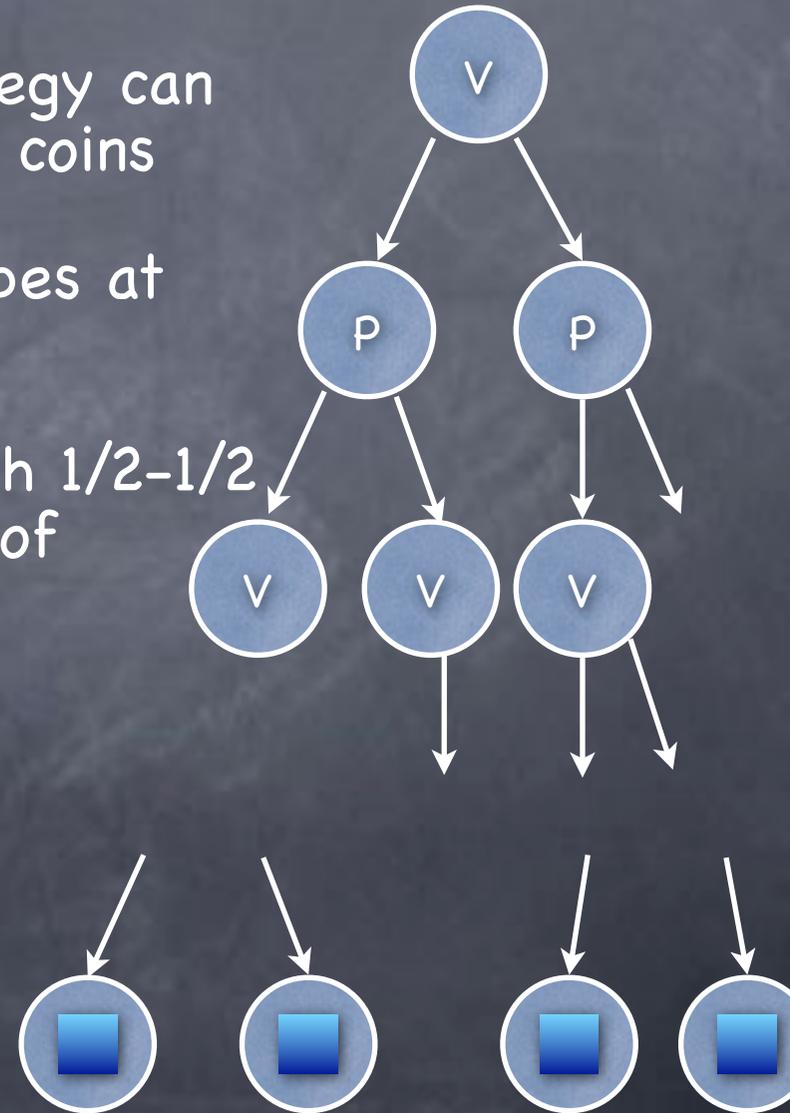
$IP \subseteq PSPACE$

- Calculate $\max \Pr[\text{yes}]$ when prover's strategy can depend only on messages and not private coins
- Maintain the set of consistent random-tapes at each V node



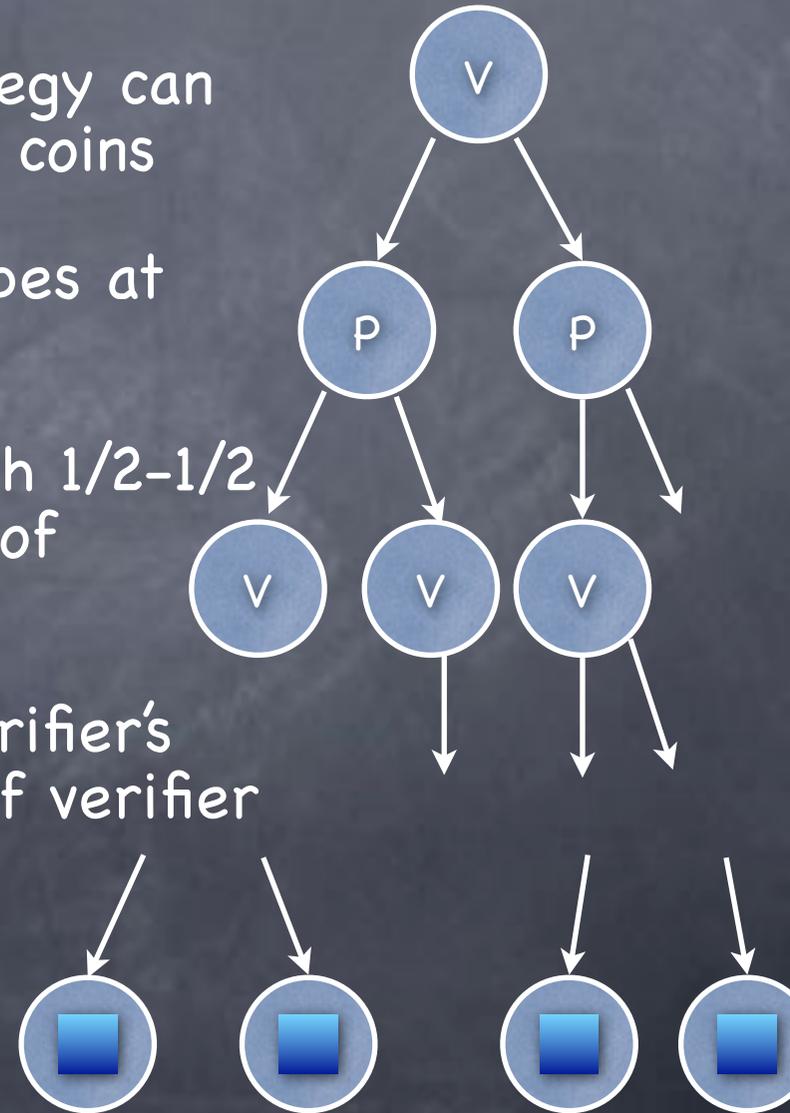
$IP \subseteq PSPACE$

- Calculate $\max \Pr[\text{yes}]$ when prover's strategy can depend only on messages and not private coins
- Maintain the set of consistent random-tapes at each V node
- Children of V node not always chosen with $1/2-1/2$ probability. Instead weighted by fraction of consistent random-tapes



$IP \subseteq PSPACE$

- Calculate $\max \Pr[\text{yes}]$ when prover's strategy can depend only on messages and not private coins
- Maintain the set of consistent random-tapes at each V node
- Children of V node not always chosen with $1/2-1/2$ probability. Instead weighted by fraction of consistent random-tapes
- Leaves: $\Pr[\text{yes}]$ determined by running verifier's program on all consistent random-tapes of verifier
- P nodes: max of children
- V nodes: (weighted) average of children



$PSPACE \subseteq IP$

$PSPACE \subseteq IP$

- Enough to show an IP protocol for TQBF

PSPACE \subseteq IP

- Enough to show an IP protocol for TQBF
 - For any L in PSPACE, both prover and verifier can first reduce input to a TQBF instance, and then prover proves its membership

PSPACE \subseteq IP

- Enough to show an IP protocol for TQBF
 - For any L in PSPACE, both prover and verifier can first reduce input to a TQBF instance, and then prover proves its membership
- Recall TQBF

PSPACE \subseteq IP

- Enough to show an IP protocol for TQBF
 - For any L in PSPACE, both prover and verifier can first reduce input to a TQBF instance, and then prover proves its membership
- Recall TQBF
 - Decide whether a QBF is true or not

PSPACE \subseteq IP

- Enough to show an IP protocol for TQBF
 - For any L in PSPACE, both prover and verifier can first reduce input to a TQBF instance, and then prover proves its membership
- Recall TQBF
 - Decide whether a QBF is true or not
 - QBF: $Q_1x_1 Q_2x_2 \dots Q_nx_n F(x_1, \dots, x_n)$ for quantifiers Q_i and a formula F on boolean variables

Arithmetization

Arithmetization

- A Boolean formula as a polynomial

Arithmetization

- A Boolean formula as a polynomial
 - Arithmetic over a field (finite, exponentially large characteristic)

Arithmetization

- A Boolean formula as a polynomial
 - Arithmetic over a field (finite, exponentially large characteristic)
 - 0 and 1 (identities of addition and multiplication) instead of False and True

Arithmetization

- A Boolean formula as a polynomial
 - Arithmetic over a field (finite, exponentially large characteristic)
 - 0 and 1 (identities of addition and multiplication) instead of False and True
 - For formula F , polynomial P such that for boolean vector \underline{b} and corresponding 0-1 vector \underline{x} we have $F(\underline{b}) = P(\underline{x})$

Arithmetization

- A Boolean formula as a polynomial
 - Arithmetic over a field (finite, exponentially large characteristic)
 - 0 and 1 (identities of addition and multiplication) instead of False and True
 - For formula F , polynomial P such that for boolean vector \underline{b} and corresponding 0-1 vector \underline{x} we have $F(\underline{b}) = P(\underline{x})$
 - NOT: $(1-x)$; AND: $x.y$

Arithmetization

- A Boolean formula as a polynomial
 - Arithmetic over a field (finite, exponentially large characteristic)
 - 0 and 1 (identities of addition and multiplication) instead of False and True
 - For formula F , polynomial P such that for boolean vector \underline{b} and corresponding 0-1 vector \underline{x} we have $F(\underline{b}) = P(\underline{x})$
 - NOT: $(1-x)$; AND: $x.y$
 - OR (as NOT of AND of NOT): $1 - (1-x).(1-y)$

Arithmetization

- A Boolean formula as a polynomial
 - Arithmetic over a field (finite, exponentially large characteristic)
 - 0 and 1 (identities of addition and multiplication) instead of False and True
 - For formula F , polynomial P such that for boolean vector \underline{b} and corresponding 0-1 vector \underline{x} we have $F(\underline{b}) = P(\underline{x})$
 - NOT: $(1-x)$; AND: $x.y$
 - OR (as NOT of AND of NOT): $1 - (1-x).(1-y)$
 - Alternately, OR: $x+y$. True corresponds to $\neq 0$

Arithmetization

- A Boolean formula as a polynomial
 - Arithmetic over a field (finite, exponentially large characteristic)
 - 0 and 1 (identities of addition and multiplication) instead of False and True
 - For formula F , polynomial P such that for boolean vector \underline{b} and corresponding 0-1 vector \underline{x} we have $F(\underline{b}) = P(\underline{x})$
 - NOT: $(1-x)$; AND: $x.y$
 - OR (as NOT of AND of NOT): $1 - (1-x).(1-y)$
 - Alternately, OR: $x+y$. True corresponds to $\neq 0$
 - Can always use a polynomial linear in each variable since $x^n=x$ for $x=0$ and $x=1$

Arithmetization

Arithmetization

- A QBF as a polynomial

Arithmetization

- A QBF as a polynomial
- TRUE will correspond to $\neq 0$, and FALSE = 0 (when variables are assigned 1/0 for TRUE/FALSE)

Arithmetization

- A QBF as a polynomial
 - TRUE will correspond to $\neq 0$, and FALSE = 0 (when variables are assigned 1/0 for TRUE/FALSE)
 - Suppose for Boolean formula F , polynomial P

Arithmetization

- A QBF as a polynomial

- TRUE will correspond to $\neq 0$, and FALSE = 0 (when variables are assigned 1/0 for TRUE/FALSE)
- Suppose for Boolean formula F , polynomial P
- $\exists x F(x) \rightarrow P(0) + P(1) > 0$ (i.e., $\sum_{x=0,1} P(x) > 0$)

Arithmetization

- A QBF as a polynomial

- TRUE will correspond to $\neq 0$, and FALSE = 0 (when variables are assigned 1/0 for TRUE/FALSE)

- Suppose for Boolean formula F , polynomial P

- $\exists x F(x) \rightarrow P(0) + P(1) > 0$ (i.e., $\sum_{x=0,1} P(x) > 0$)

- $\forall x F(x) \rightarrow P(0) \cdot P(1) > 0$ (i.e., $\prod_{x=0,1} P(x) > 0$)

Arithmetization

- A QBF as a polynomial

- TRUE will correspond to $\neq 0$, and FALSE = 0 (when variables are assigned 1/0 for TRUE/FALSE)

- Suppose for Boolean formula F , polynomial P

- $\exists x F(x) \rightarrow P(0) + P(1) > 0$ (i.e., $\sum_{x=0,1} P(x) > 0$)

- $\forall x F(x) \rightarrow P(0).P(1) > 0$ (i.e., $\prod_{x=0,1} P(x) > 0$)

- Extends to more quantifiers: i.e., if $F(x)$ is a QBF above

Arithmetization

- A QBF as a polynomial

- TRUE will correspond to $\neq 0$, and FALSE = 0 (when variables are assigned 1/0 for TRUE/FALSE)

- Suppose for Boolean formula F , polynomial P

- $\exists x F(x) \rightarrow P(0) + P(1) > 0$ (i.e., $\sum_{x=0,1} P(x) > 0$)

- $\forall x F(x) \rightarrow P(0) \cdot P(1) > 0$ (i.e., $\prod_{x=0,1} P(x) > 0$)

- Extends to more quantifiers: i.e., if $F(x)$ is a QBF above

- So, how do you arithmetize $\exists x \forall y G(x,y)$ and $\forall y \exists x G(x,y)$?

Arithmetization

- A QBF as a polynomial

- TRUE will correspond to $\neq 0$, and FALSE = 0 (when variables are assigned 1/0 for TRUE/FALSE)

- Suppose for Boolean formula F , polynomial P

- $\exists x F(x) \rightarrow P(0) + P(1) > 0$ (i.e., $\sum_{x=0,1} P(x) > 0$)

- $\forall x F(x) \rightarrow P(0) \cdot P(1) > 0$ (i.e., $\prod_{x=0,1} P(x) > 0$)

- Extends to more quantifiers: i.e., if $F(x)$ is a QBF above

- So, how do you arithmetize $\exists x \forall y G(x,y)$ and $\forall y \exists x G(x,y)$?

- $\sum_{x=0,1} \prod_{y=0,1} P(x,y) > 0$ and $\prod_{y=0,1} \sum_{x=0,1} P(x,y) > 0$

Arithmetization

Arithmetization

- For a protocol for TQBF: Give a protocol for proving that $Q_1(x_1=0,1) Q_2(x_2=0,1) \dots Q_n(x_n=0,1) P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π , and P is a (multi-linear) polynomial

Arithmetization

- For a protocol for TQBF: Give a protocol for proving that $Q_1(x_1=0,1) Q_2(x_2=0,1) \dots Q_n(x_n=0,1) P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π , and P is a (multi-linear) polynomial
- Instead suppose all Q_i are Σ

Arithmetization

- For a protocol for TQBF: Give a protocol for proving that $Q_{1(x_1=0,1)} Q_{2(x_2=0,1)} \dots Q_{n(x_n=0,1)} P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π , and P is a (multi-linear) polynomial
- Instead suppose all Q_i are Σ
 - Counts number of satisfying assignments to an (unquantified) boolean formula F

Arithmetization

- For a protocol for TQBF: Give a protocol for proving that $Q_1(x_1=0,1) Q_2(x_2=0,1) \dots Q_n(x_n=0,1) P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π , and P is a (multi-linear) polynomial
- Instead suppose all Q_i are Σ
 - Counts number of satisfying assignments to an (unquantified) boolean formula F
 - Proving > 0 is trivial

Arithmetization

- For a protocol for TQBF: Give a protocol for proving that $Q_1(x_1=0,1) Q_2(x_2=0,1) \dots Q_n(x_n=0,1) P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π , and P is a (multi-linear) polynomial
- Instead suppose all Q_i are Σ
 - Counts number of satisfying assignments to an (unquantified) boolean formula F
 - Proving > 0 is trivial
 - Consider proving $= K$ (will be useful in the general case)

Sum-check protocol

Sum-check protocol

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P

Sum-check protocol

Verifier has
only oracle
access to p

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P

Sum-check protocol

Verifier has
only oracle
access to p

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Note: to evaluate need to add up 2^n values

Sum-check protocol

Verifier has
only oracle
access to P

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Note: to evaluate need to add up 2^n values
 - Base case: $n=0$. Verifier will simply use oracle access to P .

Sum-check protocol

Verifier has
only oracle
access to P

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Note: to evaluate need to add up 2^n values
 - Base case: $n=0$. Verifier will simply use oracle access to P .
 - For $n>0$: Let $R(x) := \sum_{x_2} \dots \sum_{x_n} P(x, x_2, \dots, x_n)$

Sum-check protocol

Verifier has
only oracle
access to p

- To prove: $\sum_{x_1 \dots x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Note: to evaluate need to add up 2^n values
 - Base case: $n=0$. Verifier will simply use oracle access to P .
 - For $n>0$: Let $R(x) := \sum_{x_2 \dots x_n} P(x, x_2, \dots, x_n)$
 - $\sum_{x_1 \dots x_n} P(x_1, \dots, x_n) = R(0) + R(1)$

Sum-check protocol

Verifier has
only oracle
access to P

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Note: to evaluate need to add up 2^n values
 - Base case: $n=0$. Verifier will simply use oracle access to P .
 - For $n>0$: Let $R(x) := \sum_{x_2} \dots \sum_{x_n} P(x, x_2, \dots, x_n)$
 - $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = R(0) + R(1)$
 - R has only one variable and degree at most d

Sum-check protocol

Verifier has
only oracle
access to P

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Note: to evaluate need to add up 2^n values
 - Base case: $n=0$. Verifier will simply use oracle access to P .
 - For $n>0$: Let $R(x) := \sum_{x_2} \dots \sum_{x_n} P(x, x_2, \dots, x_n)$
 - $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = R(0) + R(1)$
 - R has only one variable and degree at most d

Only Σ , no Π

Sum-check protocol

Verifier has
only oracle
access to P

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Note: to evaluate need to add up 2^n values
 - Base case: $n=0$. Verifier will simply use oracle access to P .
 - For $n>0$: Let $R(x) := \sum_{x_2} \dots \sum_{x_n} P(x, x_2, \dots, x_n)$
 - $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = R(0) + R(1)$
 - R has only one variable and degree at most d
 - Prover sends $T=R$ (as $d+1$ coefficients) to verifier

Only Σ , no Π

Sum-check protocol

Verifier has only oracle access to P

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Note: to evaluate need to add up 2^n values
 - Base case: $n=0$. Verifier will simply use oracle access to P .
 - For $n>0$: Let $R(x) := \sum_{x_2} \dots \sum_{x_n} P(x, x_2, \dots, x_n)$
 - $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = R(0) + R(1)$
 - R has only one variable and degree at most d
 - Prover sends $T=R$ (as $d+1$ coefficients) to verifier

Only Σ , no Π

Needs degree to be small

Sum-check protocol

Verifier has only oracle access to P

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Note: to evaluate need to add up 2^n values
 - Base case: $n=0$. Verifier will simply use oracle access to P .
 - For $n>0$: Let $R(x) := \sum_{x_2} \dots \sum_{x_n} P(x, x_2, \dots, x_n)$
 - $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = R(0) + R(1)$
 - R has only one variable and degree at most d
 - Prover sends $T=R$ (as $d+1$ coefficients) to verifier
 - Verifier checks $K = T(0) + T(1)$. **Still needs to check $T=R$**

Only Σ , no Π

Needs degree to be small

Sum-check protocol

Sum-check protocol

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P

Sum-check protocol

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
- Verifier wants to check $T(X) = R(X) := \sum_{x_2} \dots \sum_{x_n} P(X, x_2, \dots, x_n)$

Sum-check protocol

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Verifier wants to check $T(X) = R(X) := \sum_{x_2} \dots \sum_{x_n} P(X, x_2, \dots, x_n)$
 - Picks random field element a (large enough field)

Sum-check protocol

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Verifier wants to check $T(X) = R(X) := \sum_{x_2} \dots \sum_{x_n} P(X, x_2, \dots, x_n)$
 - Picks random field element a (large enough field)
 - Asks prover to prove that $T(a) = R(a) = \sum_{x_2} \dots \sum_{x_n} P(a, x_2, \dots, x_n)$

Sum-check protocol

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Verifier wants to check $T(X) = R(X) := \sum_{x_2} \dots \sum_{x_n} P(X, x_2, \dots, x_n)$
 - Picks random field element a (large enough field)
 - Asks prover to prove that $T(a) = R(a) = \sum_{x_2} \dots \sum_{x_n} P(a, x_2, \dots, x_n)$
 - Recurse on $P_1(x_2, \dots, x_n) = P(a, x_2, \dots, x_n)$ of one variable less

Sum-check protocol

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Verifier wants to check $T(X) = R(X) := \sum_{x_2} \dots \sum_{x_n} P(X, x_2, \dots, x_n)$
 - Picks random field element a (large enough field)
 - Asks prover to prove that $T(a) = R(a) = \sum_{x_2} \dots \sum_{x_n} P(a, x_2, \dots, x_n)$
 - Recurse on $P_1(x_2, \dots, x_n) = P(a, x_2, \dots, x_n)$ of one variable less
 - i.e., Recurse to prove $\sum_{x_2} \dots \sum_{x_n} P_1(x_2, \dots, x_n) = T(a)$

Sum-check protocol

- To prove: $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$ for some degree d polynomial P
 - Verifier wants to check $T(X) = R(X) := \sum_{x_2} \dots \sum_{x_n} P(X, x_2, \dots, x_n)$
 - Picks random field element a (large enough field)
 - Asks prover to prove that $T(a) = R(a) = \sum_{x_2} \dots \sum_{x_n} P(a, x_2, \dots, x_n)$
 - Recurse on $P_1(x_2, \dots, x_n) = P(a, x_2, \dots, x_n)$ of one variable less
 - i.e., Recurse to prove $\sum_{x_2} \dots \sum_{x_n} P_1(x_2, \dots, x_n) = T(a)$
 - Note: P_1 has degree at most d ; verifier has oracle access to P_1 (as it knows a , and has oracle access to P)

Sum-check protocol

Sum-check protocol

- Why does sum-check protocol work?

Sum-check protocol

- Why does sum-check protocol work?
 - Instead of checking $T(X) = R(X)$, simply checks (recursively) if $T(a)=R(a)$ for a single random a in the field

Sum-check protocol

Can't afford
more than
one check

- Why does sum-check protocol work?
 - Instead of checking $T(X) = R(X)$, simply checks (recursively) if $T(a) = R(a)$ for a single random a in the field

Sum-check protocol

Can't afford
more than
one check

- Why does sum-check protocol work?
 - Instead of checking $T(X) = R(X)$, simply checks (recursively) if $T(a)=R(a)$ for a single random a in the field
 - Completeness is obvious

Sum-check protocol

Can't afford more than one check

- Why does sum-check protocol work?
 - Instead of checking $T(X) = R(X)$, simply checks (recursively) if $T(a)=R(a)$ for a single random a in the field
 - Completeness is obvious
 - Soundness: Since $T(X)$ and $R(X)$ are of degree d , if $T \neq R$, at most d points where they agree

Sum-check protocol

Can't afford more than one check

- Why does sum-check protocol work?
 - Instead of checking $T(X) = R(X)$, simply checks (recursively) if $T(a)=R(a)$ for a single random a in the field
 - Completeness is obvious
 - Soundness: Since $T(X)$ and $R(X)$ are of degree d , if $T \neq R$, at most d points where they agree
 - Error (picking a bad a), with probability $\leq d/p$, where field is of size p

Sum-check protocol

Can't afford more than one check

- Why does sum-check protocol work?
 - Instead of checking $T(X) = R(X)$, simply checks (recursively) if $T(a)=R(a)$ for a single random a in the field
 - Completeness is obvious
 - Soundness: Since $T(X)$ and $R(X)$ are of degree d , if $T \neq R$, at most d points where they agree
 - Error (picking a bad a), with probability $\leq d/p$, where field is of size p
 - Also possible error in recursive step (despite good a)

Sum-check protocol

Can't afford more than one check

- Why does sum-check protocol work?
 - Instead of checking $T(X) = R(X)$, simply checks (recursively) if $T(a)=R(a)$ for a single random a in the field
 - Completeness is obvious
 - Soundness: Since $T(X)$ and $R(X)$ are of degree d , if $T \neq R$, at most d points where they agree
 - Error (picking a bad a), with probability $\leq d/p$, where field is of size p
 - Also possible error in recursive step (despite good a)
 - At most nd/p if n variables. Can take p exponential.

IP Protocol for TQBF

IP Protocol for TQBF

- For a protocol for TQBF: Give a protocol for proving that $Q_1(x_1=0,1) Q_2(x_2=0,1) \dots Q_n(x_n=0,1) P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π and P is a multi-linear polynomial

IP Protocol for TQBF

- For a protocol for TQBF: Give a protocol for proving that $Q_{1(x_1=0,1)} Q_{2(x_2=0,1)} \dots Q_{n(x_n=0,1)} P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π and P is a multi-linear polynomial
- In fact a protocol to prove: $Q_{1 x_1} \dots Q_{n x_n} P(x_1, \dots, x_n) = K$

IP Protocol for TQBF

- For a protocol for TQBF: Give a protocol for proving that $Q_{1(x_1=0,1)} Q_{2(x_2=0,1)} \dots Q_{n(x_n=0,1)} P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π and P is a multi-linear polynomial
 - In fact a protocol to prove: $Q_{1 x_1} \dots Q_{n x_n} P(x_1, \dots, x_n) = K$
- Problem with generalizing sum-check protocol: the univariate poly $R(X) := Q_{2 x_2} \dots Q_{n x_n} P(X, x_2, \dots, x_n)$ has exponential degree. Verifier can't read $T(X)=R(X)$

IP Protocol for TQBF

- For a protocol for TQBF: Give a protocol for proving that $Q_{1(x_1=0,1)} Q_{2(x_2=0,1)} \dots Q_{n(x_n=0,1)} P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π and P is a multi-linear polynomial
 - In fact a protocol to prove: $Q_{1 x_1} \dots Q_{n x_n} P(x_1, \dots, x_n) = K$
- Problem with generalizing sum-check protocol: the univariate poly $R(X) := Q_{2 x_2} \dots Q_{n x_n} P(X, x_2, \dots, x_n)$ has exponential degree. Verifier can't read $T(X)=R(X)$
- Instead of T , can work with "linearization" of T .

IP Protocol for TQBF

- For a protocol for TQBF: Give a protocol for proving that $Q_{1(x_1=0,1)} Q_{2(x_2=0,1)} \dots Q_{n(x_n=0,1)} P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π and P is a multi-linear polynomial
 - In fact a protocol to prove: $Q_{1 x_1} \dots Q_{n x_n} P(x_1, \dots, x_n) = K$
- Problem with generalizing sum-check protocol: the univariate poly $R(X) := Q_{2 x_2} \dots Q_{n x_n} P(X, x_2, \dots, x_n)$ has exponential degree. Verifier can't read $T(X)=R(X)$
- Instead of T , can work with "linearization" of T .
 - Prover sends $L(X) = (T(1)-T(0)) X + T(0)$

IP Protocol for TQBF

- For a protocol for TQBF: Give a protocol for proving that $Q_{1(x_1=0,1)} Q_{2(x_2=0,1)} \dots Q_{n(x_n=0,1)} P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π and P is a multi-linear polynomial
 - In fact a protocol to prove: $Q_{1 x_1} \dots Q_{n x_n} P(x_1, \dots, x_n) = K$
- Problem with generalizing sum-check protocol: the univariate poly $R(X) := Q_{2 x_2} \dots Q_{n x_n} P(X, x_2, \dots, x_n)$ has exponential degree. Verifier can't read $T(X)=R(X)$
- Instead of T , can work with "linearization" of T .
 - Prover sends $L(X) = (T(1)-T(0)) X + T(0)$
 - Verifier checks (as appropriate) $L(1).L(0) = K$ or $L(1)+L(0) = K$

IP Protocol for TQBF

- For a protocol for TQBF: Give a protocol for proving that $Q_{1(x_1=0,1)} Q_{2(x_2=0,1)} \dots Q_{n(x_n=0,1)} P(x_1, \dots, x_n) > 0$, where Q_i are Σ or Π and P is a multi-linear polynomial
 - In fact a protocol to prove: $Q_{1 x_1} \dots Q_{n x_n} P(x_1, \dots, x_n) = K$
- Problem with generalizing sum-check protocol: the univariate poly $R(X) := Q_{2 x_2} \dots Q_{n x_n} P(X, x_2, \dots, x_n)$ has exponential degree. Verifier can't read $T(X)=R(X)$
- Instead of T , can work with "linearization" of T .
 - Prover sends $L(X) = (T(1)-T(0)) X + T(0)$
 - Verifier checks (as appropriate) $L(1).L(0) = K$ or $L(1)+L(0) = K$
 - Verifier picks random a , and asks prover to show $R'(a) = L(a)$

linearization of $R(X)$

IP Protocol for TQBF

IP Protocol for TQBF

- IP = PSPACE

IP Protocol for TQBF

- $IP = PSPACE$
- Protocol is public-coin

IP Protocol for TQBF

- $IP = PSPACE$
- Protocol is public-coin
 - $IP = AM[poly] = PSPACE$

IP Protocol for TQBF

- $IP = PSPACE$
- Protocol is public-coin
 - $IP = AM[poly] = PSPACE$
- Protocol has perfect completeness