

# Interactive Proofs

Lecture 17  
 $IP = PSPACE$

# So far

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

# IP = PSPACE

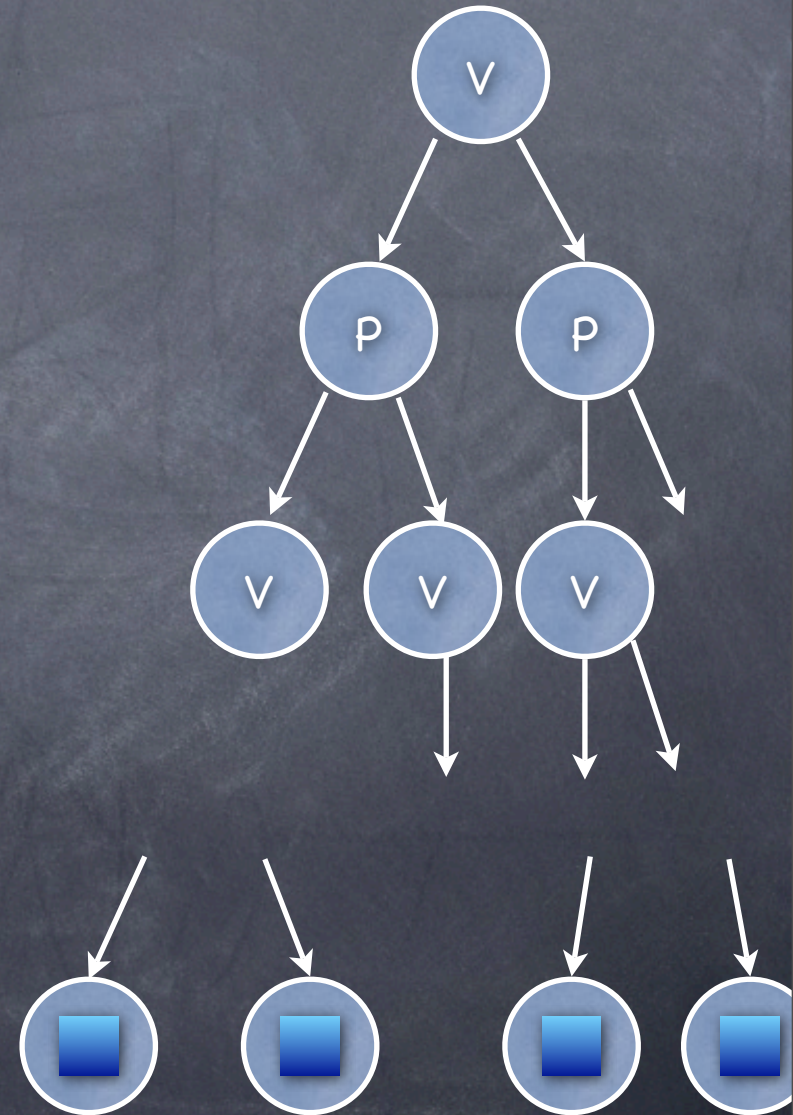
- Recall, IP means IP[poly]
- $IP \subseteq PSPACE$ 
  - Even though prover unbounded, cannot convince poly time verifier of everything
- $PSPACE \subseteq IP$ 
  - Prover can convince verifier of high complexity statements

# IP $\subseteq$ PSPACE

- 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  $\text{IP}[\text{poly}] = \text{AM}[\text{poly}]$ 
    - Or modify the proof (as we'll do)

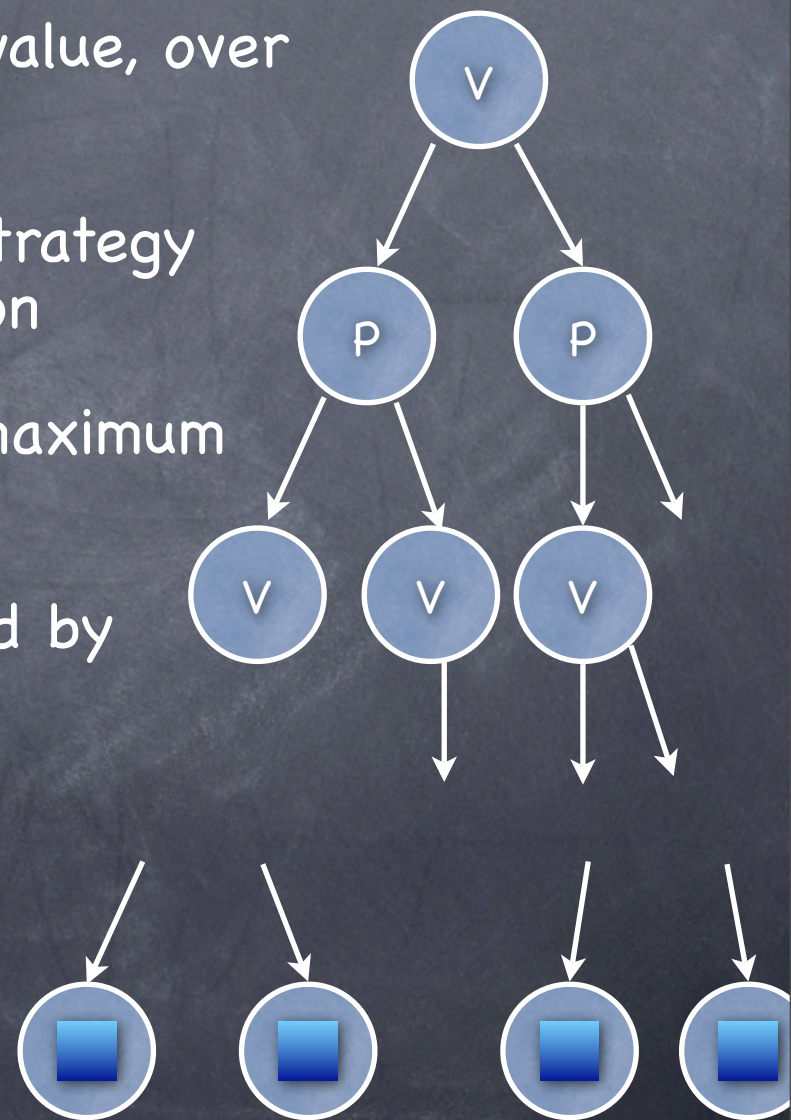
# $AM[\text{poly}] \subseteq PSPACE$

- 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
  - Protocol’s configuration tree: path to a node corresponds to the transcript so far



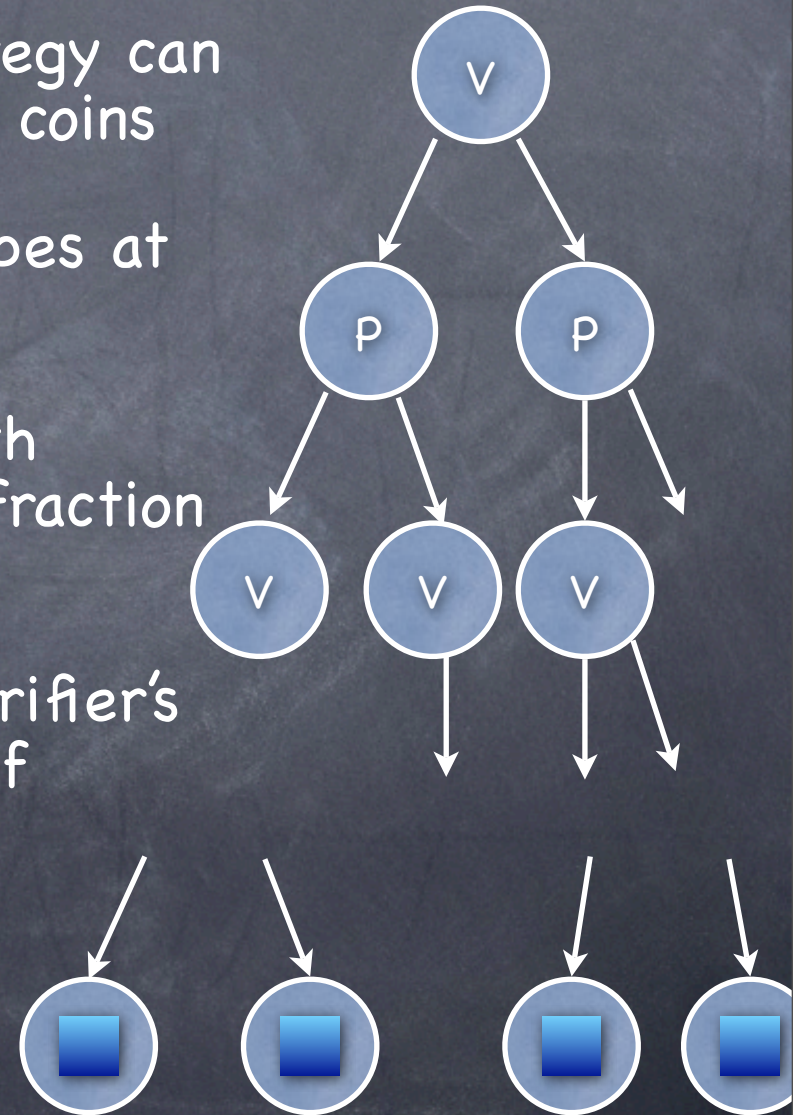
# $AM[\text{poly}] \subseteq PSPACE$

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



# $IP \subseteq PSPACE$

- 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

- 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

- A Boolean formula as a polynomial
  - Arithmetic over a (finite, exponentially large) field
  - 0 and 1 (identities of addition and multiplication) instead of True and False
    - 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:  $x \cdot y$
    - OR (as NOT of AND of NOT):  $1 - (1-x) \cdot (1-y)$
    - Exercise: Arithmetize  $x=y$  (now!). Degree? Size?
      - Can always use a polynomial linear in each variable since  $x^n=x$  for  $x=0$  and  $x=1$

# Arithmetization

- A QBF as a polynomial

- TRUE will correspond to  $\neq 0$ , and FALSE,  $= 0$

- Suppose for Boolean formula  $F$ , polynomial  $P$

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

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

- Even if  $F(x)$  is a QBF and  $P(x)$  arithmetization as above

- So, how do you arithmetize  $\exists x \forall y x=y$  and  $\forall y \exists x x=y$ ?

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

# Arithmetization

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

# Sum-check protocol

- To prove:  $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = K$  for some degree  $d$  polynomial  $P$ 
  - Note: to evaluate need to add up  $2^n$  values
  - Base case:  $n=0$ . (note: verifier will use oracle access to  $P$ )
  - For  $n>0$ : Let  $P'(x) := \sum_{x_2} \dots \sum_{x_n} P(x, x_2, \dots, x_n)$ 
    - $\sum_{x_1} \dots \sum_{x_n} P(x_1, \dots, x_n) = P'(0) + P'(1)$
    - $P'$  has only one variable and degree at most  $d$
    - Prover sends  $T=P'$  (as  $d+1$  coefficients) to verifier
    - Verifier checks  $K = T(0) + T(1)$ . **Still needs to check  $T=P'$**

Only  $\Sigma$ , no  $\Pi$

Needs degree to be small

# 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  $T(X) = P'(X) := \sum_{x_2} \dots \sum_{x_n} P(X, x_2, \dots, x_n)$
  - Picks random field element  $a$  (large enough field)
  - Asks prover to prove that  $T(a) = P'(a) = \sum_{x_2} \dots \sum_{x_n} P(a, x_2, \dots, x_n)$ 
    - 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

- Why does sum-check protocol work?
  - Instead of checking  $T(X) = P'(X)$ , simply checks (recursively) if  $T(a) = P'(a)$  for a random  $a$  in the field
    - Completeness is obvious
    - Soundness: Since  $T(X)$  and  $P'(X)$  are of degree  $d$ , if  $T \neq P'$ , at most  $d$  points where they agree
      - Error probability  $\leq d/p$ , where field is of size  $p$
      - Also possible error in recursive step
        - At most  $nd/p$  if  $n$  variables. Can have  $p$  exponential.

# IP Protocol for TQBF

- For a protocol for TQBF: Give a protocol for proving that  $Q_{1(x_1=0,1)} Q_{2(x_2=0,1)} \dots Q_{n(x_n=0,1)} P(x_1, \dots, x_n) \neq 0$ , where  $Q_i$  are  $\Sigma$  or  $\Pi$  and  $P$  a multi-linear polynomial
  - Using protocol to prove:  $Q_{1 x_1} \dots Q_{n x_n} P(x_1, \dots, x_n) = K$ , with
- Problem with generalizing sum-check protocol: the univariate poly  $P'(X) := Q_{2 x_2} \dots Q_{n x_n} P(X, x_2, \dots, x_n)$  has exponential degree. Verifier can't read  $T(X)=P'(X)$
- To reduce degree: instead of  $P'$ , work with "linearization" of  $P'$ . Define:  $L_x P(x,y) := (1-x) P(0,y) + x P(1,y)$ 
  - $L_x P(x,y)$  is linear in  $x$  and matches  $P(x,y)$  for  $(x,y)$  in  $\{0,1\}^2$
  - Modify  $\sum_{x=0,1} \prod_{y=0,1} P(x,y)$  as  $\sum_{x=0,1} L_x \prod_{y=0,1} L_y P(x,y)$
  - Same protocol as before for sum-check works, but with verifier checks appropriately modified

# IP Protocol for TQBF

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