Lecture 4

Talk Outline

1. Recap: The Sum-Check Protocol
2. An Interactive Proof for #SAT

The Sum-Check Protocol [LFKN90]

Sum-Check Protocol [LFKN90]

- Input: V given oracle access to a ℓ-variate polynomial g over field $\boldsymbol{F}$.
- Goal: compute the quantity:

$$\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(b_1, \dots, b_\ell).$$

- **Start**: P sends claimed answer C_1. The protocol must check that:

$$C_1 = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(b_1, \dots, b_\ell).$$

- **Start**: P sends claimed answer C_1. The protocol must check that:

$$C_1 = \sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(b_1, \dots, b_\ell).$$

- **Round 1**: P sends **univariate** polynomial $s_1(X_1)$ claimed to equal:

$$H_1(X_1) := \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(X_1, b_2, \dots, b_\ell)$$

- V checks that $C_1 = s_1(0) + s_1(1)$.
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P.
- **Round 2**: They recursively check that $s_1(r_1) = H_1(r_1)$.

  i.e., that $s_1(r_1) = \sum_{b_2 \in \{0,1\}} \dots \sum_{b_\ell \in \{0,1\}} g(r_1, b_2, \dots, b_\ell)$.

- **Round ℓ (Final round):** P sends univariate polynomial $s_\ell(X_\ell)$ claimed to equal

  $H_\ell := g(r_1, \dots, r_{\ell-1}, X_\ell)$.

- V checks that $s_{\ell-1}(r_{\ell-1}) = s_\ell(0) + s_\ell(1)$.
- V picks r_ℓ at random, and needs to check that $s_\ell(r_\ell) = g(r_1, \dots, r_\ell)$.
  - No need for more rounds. V can perform this check with one oracle query.

Example Execution of Sum-Check with Honest Prover

Let $g(X, Y, Z) = X^2 Y^2 Z$

Note: $\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \sum_{b_3 \in \{0,1\}} g(b_1, b_2, b_3) = 1.$

- Recall $g(X, Y, Z) = X^2 Y^2 Z$.
- **Start**: P sends claimed answer $C_1 = 1$.
- **Round 1**: P sends **univariate** polynomial $s_1(X)$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $1 = 0^2 + 1^2$).

- Recall $g(X,Y,Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 1$.
- **Round 1**: P sends **univariate** polynomial $s_1(X)$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $1 = 0^2 + 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y)$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- Recall $g(X, Y, Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 1$.
- **Round 1**: P sends **univariate** polynomial $s_1(X)$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $1 = 0^2 + 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y)$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- V checks that $s_1(3) = s_2(0) + s_2(1)$ (i.e., that $3^2 = 9 \cdot 0^2 + 9 \cdot 1^2$).

- Recall $g(X,Y,Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 1$.
- **Round 1**: P sends **univariate** polynomial $s_1(X)$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $1 = 0^2 + 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y)$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- V checks that $s_1(3) = s_2(0) + s_2(1)$ (i.e., that $3^2 = 9 \cdot 0^2 + 9 \cdot 1^2$).
- V picks r_2 at random from $\boldsymbol{F}$ and sends r_2 to P. Let's say $r_2 = 5$.
- **Round 3**: P sends **univariate** polynomial $s_3(Z)$ claimed to equal:

$$g(3, 5, Z) = 3^2 \cdot 5^2 \cdot Z = 225 \cdot Z.$$

- V checks that $s_2(5) = s_3(0) + s_3(1)$ (i.e., that $9 \cdot 5^2 = 225 \cdot 0^2 + 225 \cdot 1^2$)

- Recall $g(X, Y, Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 1$.
- **Round 1**: P sends **univariate** polynomial $s_1(X)$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $1 = 0^2 + 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y)$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- V checks that $s_1(3) = s_2(0) + s_2(1)$ (i.e., that $3^2 = 9 \cdot 0^2 + 9 \cdot 1^2$).
- V picks r_2 at random from $\boldsymbol{F}$ and sends r_2 to P. Let's say $r_2 = 5$.
- **Round 3**: P sends **univariate** polynomial $s_3(Z)$ claimed to equal:

$$g(3, 5, Z) = 3^2 \cdot 5^2 \cdot Z = 225 \cdot Z.$$

- V checks that $s_2(5) = s_3(0) + s_3(1)$ (i.e., that $9 \cdot 5^2 = 225 \cdot 0^2 + 225 \cdot 1^2$)
- V picks r_3 at random from $\boldsymbol{F}$, say $r_3 = 2$.
- V checks that $s_3(2) = g(3, 5, 2)$ (i.e., that $225 \cdot 2 = 3^2 \cdot 5^2 \cdot 2$).

Example Execution of Sum-Check with Dishonest Prover

Let $g(X, Y, Z) = X^2 Y^2 Z$

Note: $\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \sum_{b_3 \in \{0,1\}} g(b_1, b_2, b_3) = 1.$

- Recall $g(X, Y, Z) = X^2 Y^2 Z$.
- **Start**: P sends claimed answer $C_1 = 2$.

- Recall $g(X,Y,Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 2$.
- **Round 1**: P sends **univariate** polynomial $s_1(X) = 2X$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $2 = 0^2 + 2 \cdot 1^2$).

- Recall $g(X,Y,Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 2$.
- **Round 1**: P sends **univariate** polynomial $s_1(X) = 2X$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $2 = 0^2 + 2 \cdot 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. As long a r_1 is not in $\{0, 2\}$ then $s_1(r_1) \neq H_1(r_1)$. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y) = 6Y$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- Recall $g(X, Y, Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 2$.
- **Round 1**: P sends **univariate** polynomial $s_1(X) = 2X$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $2 = 0^2 + 2 \cdot 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. As long a r_1 is not in $\{0, 2\}$ then $s_1(r_1) \neq H_1(r_1)$. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y) = 6Y$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- V checks that $s_1(3) = s_2(0) + s_2(1)$ (i.e., that $2 \cdot 3 = 6 \cdot 0 + 6 \cdot 1$).

- Recall $g(X, Y, Z) = X^2 Y^2 Z$.
- **Start**: P sends claimed answer $C_1 = 2$.
- **Round 1**: P sends **univariate** polynomial $s_1(X) = 2X$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $2 = 0^2 + 2 \cdot 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. As long a r_1 is not in $\{0, 2\}$ then $s_1(r_1) \neq H_1(r_1)$. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y) = 6Y$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- V checks that $s_1(3) = s_2(0) + s_2(1)$ (i.e., that $2 \cdot 3 = 6 \cdot 0 + 6 \cdot 1$).
- V picks r_2 at random from $\boldsymbol{F}$ and sends r_2 to P. As long a r_2 is not in $\{0, 2 \cdot 3^{-1}\}$ then $s_2(r_2) \neq H_2(r_2)$. Let's say $r_2 = 5$.
- **Round 3**: P sends **univariate** polynomial $s_3(Z) = 30 \cdot Z$ claimed to equal:

$$g(3, 5, Z) = 3^2 \cdot 5^2 \cdot Z = 225 \cdot Z.$$

- Recall $g(X,Y,Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 2$.
- **Round 1**: P sends **univariate** polynomial $s_1(X) = 2X$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $2 = 0^2 + 2 \cdot 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. As long a r_1 is not in $\{0, 2\}$ then $s_1(r_1) \neq H_1(r_1)$. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y) = 6Y$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- V checks that $s_1(3) = s_2(0) + s_2(1)$ (i.e., that $2 \cdot 3 = 6 \cdot 0 + 6 \cdot 1$).
- V picks r_2 at random from $\boldsymbol{F}$ and sends r_2 to P. As long a r_2 is not in $\{0, 2 \cdot 3^{-1}\}$ then $s_2(r_2) \neq H_2(r_2)$. Let's say $r_2 = 5$.
- **Round 3**: P sends **univariate** polynomial $s_3(Z) = 30 \cdot Z$ claimed to equal:

$$g(3, 5, Z) = 3^2 \cdot 5^2 \cdot Z = 225 \cdot Z.$$

- V checks that $s_2(5) = s_3(0) + s_3(1)$ (i.e., that $6 \cdot 5 = 30 \cdot 0 + 30 \cdot 1$.

- Recall $g(X,Y,Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 2.$
- **Round 1**: P sends **univariate** polynomial $s_1(X) = 2X$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $2 = 0^2 + 2 \cdot 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. As long a r_1 is not in $\{0, 2\}$ then $s_1(r_1) \neq H_1(r_1)$. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y) = 6Y$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- V checks that $s_1(3) = s_2(0) + s_2(1)$ (i.e., that $2 \cdot 3 = 6 \cdot 0 + 6 \cdot 1$).
- V picks r_2 at random from $\boldsymbol{F}$ and sends r_2 to P. As long a r_2 is not in $\{0, 2 \cdot 3^{-1}\}$ then $s_2(r_2) \neq H_2(r_2)$. Let's say $r_2 = 5$.
- **Round 3**: P sends **univariate** polynomial $s_3(Z) = 30 \cdot Z$ claimed to equal:

$$g(3, 5, Z) = 3^2 \cdot 5^2 \cdot Z = 225 \cdot Z.$$

- V checks that $s_2(5) = s_3(0) + s_3(1)$ (i.e., that $6 \cdot 5 = 30 \cdot 0 + 30 \cdot 1$.
- V picks r_3 at random from $\boldsymbol{F}$. As long a $r_3 \neq 0$, then $s_3(r_3) \neq H_3(r_3)$. Let's say $r_3 = 2$.
- V checks that $s_3(2) = g(3,5,2)$ (i.e., that $30 \cdot 2 = 3^2 \cdot 5^2 \cdot 2$). **Check fails.**

- Recall $g(X,Y,Z) = X^2Y^2Z$.
- **Start**: P sends claimed answer $C_1 = 2.$
- **Round 1**: P sends **univariate** polynomial $s_1(X) = 2X$ claimed to equal:

$$H_1(X) := \sum_{b_2 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} g(X, b_2, b_3)$$

$$= X^2 \cdot 0^2 \cdot 0 + X^2 \cdot 0^2 \cdot 1 + X^2 \cdot 1^2 \cdot 0 + X^2 \cdot 1^2 \ \cdot 1 = X^2.$$

- V checks that $C_1 = s_1(0) + s_1(1)$ (i.e., that $2 = 0^2 + 2 \cdot 1^2$).
- V picks r_1 at random from $\boldsymbol{F}$ and sends r_1 to P. As long a r_1 is not in $\{0, 2\}$ then $s_1(r_1) \neq H_1(r_1)$. Let's say $r_1 = 3$.
- **Round 2**: P sends **univariate** polynomial $s_2(Y) = 6Y$ claimed to equal:

$$\sum_{b_3 \in \{0,1\}} g(3, Y, b_3) = 9 \cdot Y^2 \cdot 0 + 9 \cdot Y^2 \cdot 1 = 9 \cdot Y^2.$$

- V checks that $s_1(3) = s_2(0) + s_2(1)$ (i.e., that $2 \cdot 3 = 6 \cdot 0 + 6 \cdot 1$).
- V picks r_2 at random from $\boldsymbol{F}$ and sends r_2 to P. As long a r_2 is not in $\{0, 2 \cdot 3^{-1}\}$ then $s_2(r_2) \neq H_2(r_2)$. Let's say $r_2 = 5$.
- **Round 3**: P sends **univariate** polynomial $s_3(Z) = 30 \cdot Z$ claimed to equal:

$$g(3, 5, Z) = 3^2 \cdot 5^2 \cdot Z = 225 \cdot Z.$$

- V checks that $s_2(5) = s_3(0) + s_3(1)$ (i.e., that $6 \cdot 5 = 30 \cdot 0 + 30 \cdot 1$.
- V picks r_3 at random from $\boldsymbol{F}$. As long a $r_3 \neq 0$, then $s_3(r_3) \neq H_3(r_3)$. Let's say $r_3 = 2$.
- V checks that $s_3(2) = g(3, 5,2)$ (i.e., that $30 \cdot 2 = 3^2 \cdot 5^2 \cdot 2$). **Check fails.**

Costs of the Sum-Check Protocol

- Total communication is $O(d\ell)$ field elements.
  - P sends ℓ messages, each a univariate polynomial of degree at most d. V sends $\ell - 1$ messages, each consisting of one field elements.
- V's runtime is:

$$O(d\ell + [\mathit{time\ required\ to\ evaluate\ g\ at\ one\ point}]).$$

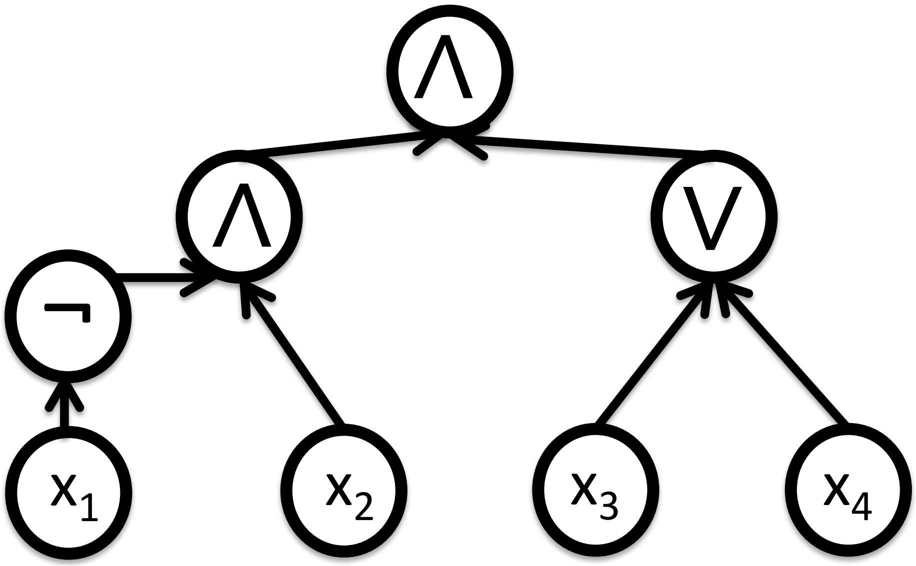
- P's runtime is at most:

$$O\left(d \cdot 2^{\ell} \cdot [\mathit{time\ required\ to\ evaluate\ g\ at\ one\ point}]\right).$$

First Application of Sum-Check: An IP For #SAT [LFKN]

#SAT Problem

- Let φ be a Boolean formula of size S over n variables.

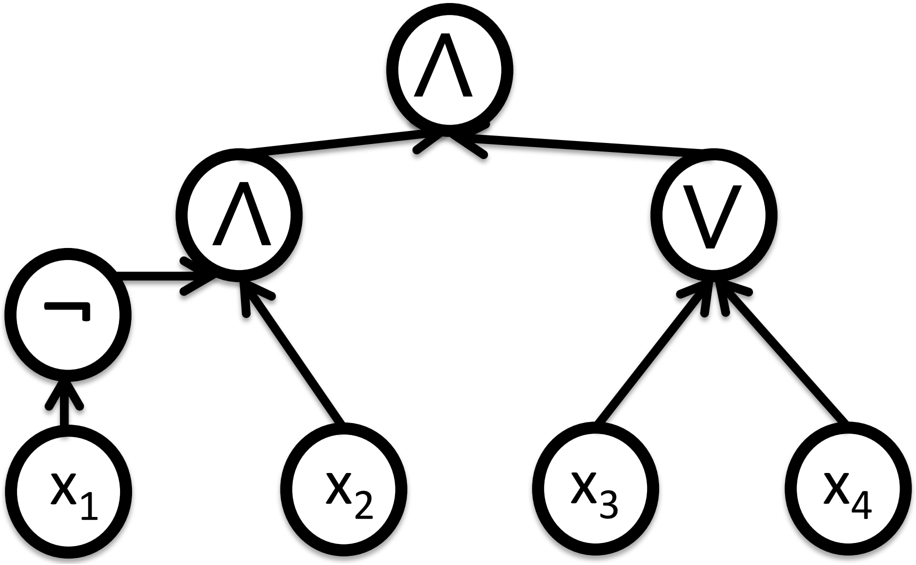


#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: count the number of satisfying assignments of φ.
- i.e., Compute $\sum_{x\in\{0,1\}^n} \varphi(x)$.
- In the sum above, we are viewing φ as a function mapping $\{0,1\}^n \to \{0,1\}$. (0 interpreted as FALSE, 1 as TRUE).

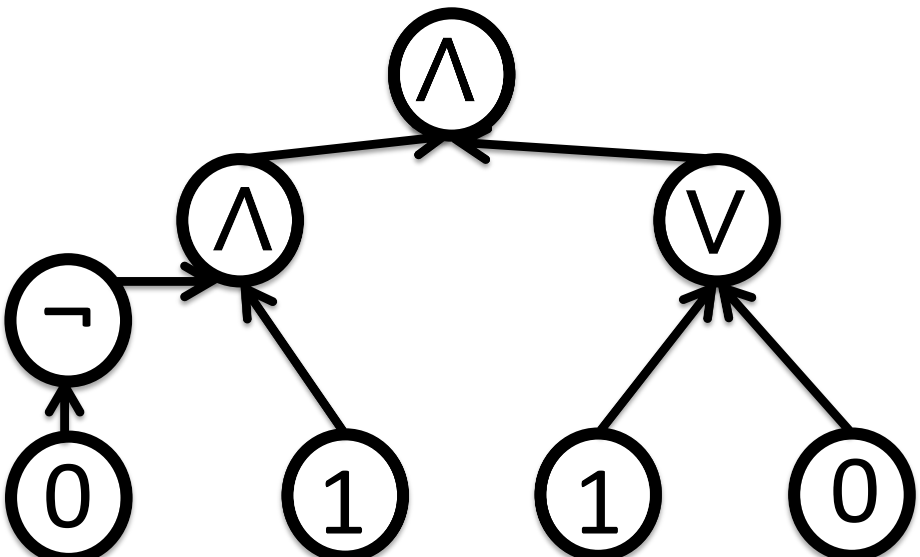
#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: count the number of satisfying assignments of φ.
- i.e., Compute $\sum_{x \in \{0,1\}^n} \varphi(x)$.
- In the sum above, we are viewing φ as a function mapping $\{0,1\}^n \rightarrow \{0, 1\}$. (0 interpreted as FALSE, 1 as TRUE).



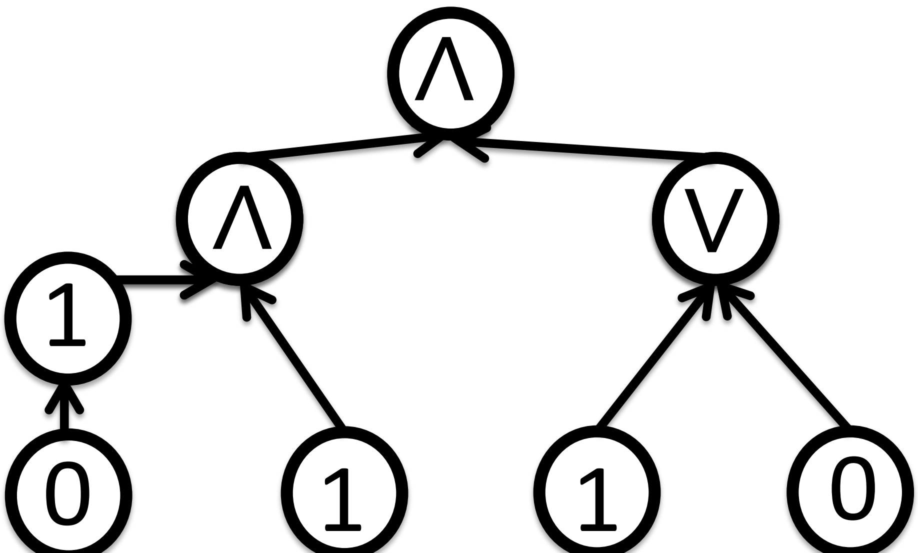
#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: count the number of satisfying assignments of φ.
- i.e., Compute $\sum_{x\in\{0,1\}^n} \varphi(x)$.
- In the sum above, we are viewing φ as a function mapping $\{0,1\}^n \rightarrow \{0,1\}$. (0 interpreted as FALSE, 1 as TRUE).



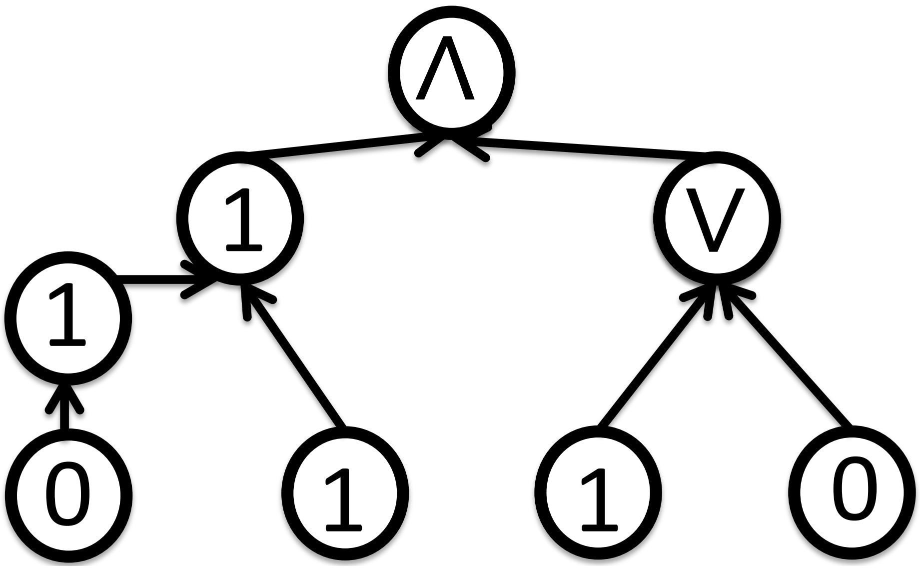
#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: count the number of satisfying assignments of φ.
- i.e., Compute $\sum_{x\in\{0,1\}^n} \varphi(x)$.
- In the sum above, we are viewing φ as a function mapping $\{0,1\}^n \rightarrow \{0,1\}$. (0 interpreted as FALSE, 1 as TRUE).



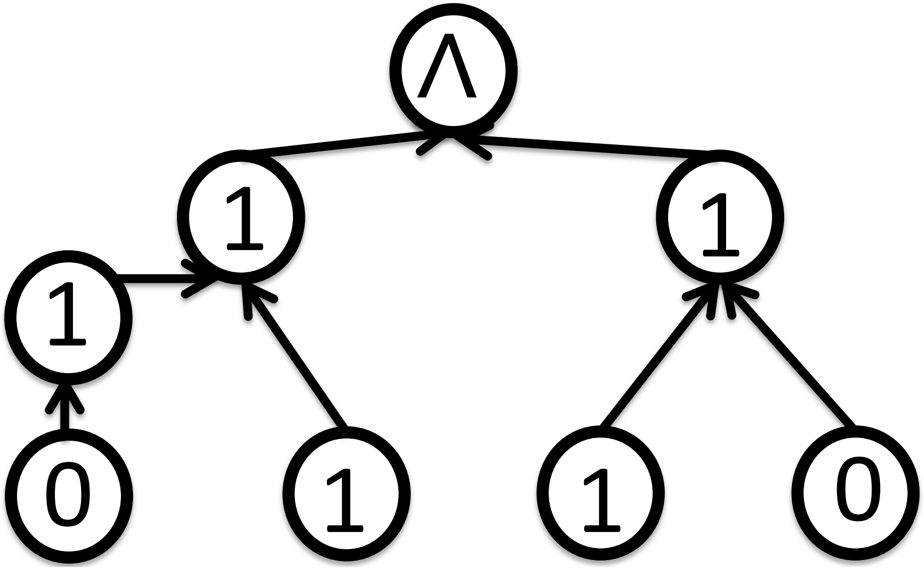
#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: count the number of satisfying assignments of φ.
- i.e., Compute $\sum_{x\in\{0,1\}^n} \varphi(x)$.
- In the sum above, we are viewing φ as a function mapping $\{0,1\}^n \rightarrow \{0,1\}$. (0 interpreted as FALSE, 1 as TRUE).



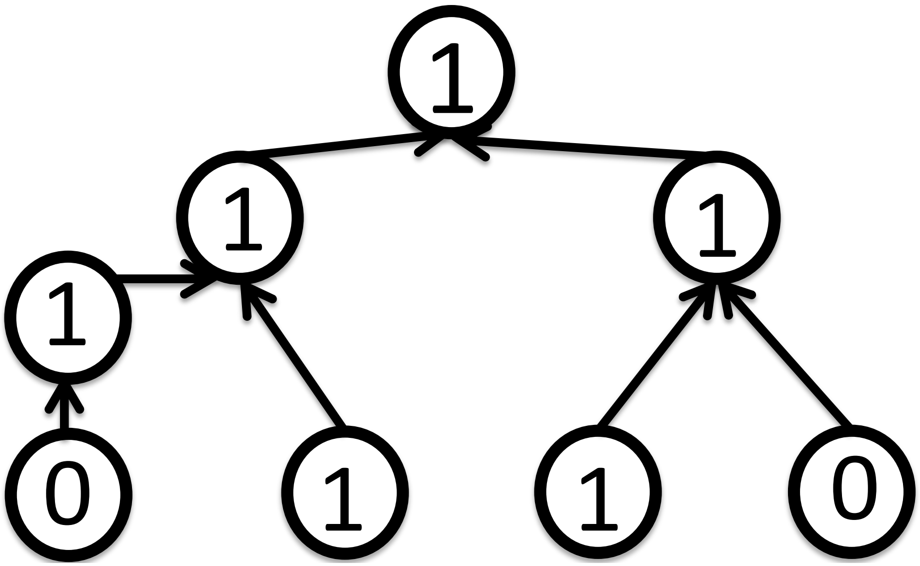
#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: count the number of satisfying assignments of φ.
- i.e., Compute $\sum_{x\in\{0,1\}^n} \varphi(x)$.
- In the sum above, we are viewing φ as a function mapping $\{0,1\}^n \rightarrow \{0,1\}$. (0 interpreted as FALSE, 1 as TRUE).



#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: count the number of satisfying assignments of φ.
- i.e., Compute $\sum_{x\in\{0,1\}^n} \varphi(x)$.
- In the sum above, we are viewing φ as a function mapping $\{0,1\}^n \rightarrow \{0,1\}$. (0 interpreted as FALSE, 1 as TRUE).



#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: Compute $\sum_{x \in \{0,1\}^n} \varphi(x)$.

#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: Compute $\sum_{x \in \{0,1\}^n} \varphi(x)$.

- Protocol:
- Let g be an extension polynomial of φ.
- Apply the sum-check protocol to compute $\sum_{x \in \{0,1\}^n} g(x)$.

#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: Compute $\sum_{x\in\{0,1\}^n}\varphi(x)$.

- Protocol:
- Let g be an extension polynomial of $\boldsymbol{\varphi}$.
- Apply the sum-check protocol to compute $\sum_{x\in\{0,1\}^n} g(x)$.
  - Note: in final round of sum-check, V needs to compute $g(r)$ for some randomly chosen r in $\boldsymbol{F^n}$.
    - To control V's runtime, we need this to be fast.

#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: Compute $\sum_{x \in \{0,1\}^n} \varphi(x)$.

- Protocol:
- Let g be an extension polynomial of φ.
- Apply the sum-check protocol to compute $\sum_{x \in \{0,1\}^n} g(x)$.
  - Note: in final round of sum-check, V needs to compute $g(r)$ for some randomly chosen r in $\boldsymbol{F^n}$.
    - To control V's runtime, we need this to be fast.
  - To control communication and P and V's runtime, we need g to be "low-degree".

#SAT Problem

- Let φ be a Boolean formula of size S over n variables.
- Goal: Compute $\sum_{x\in\{0,1\}^n} \varphi(x)$.

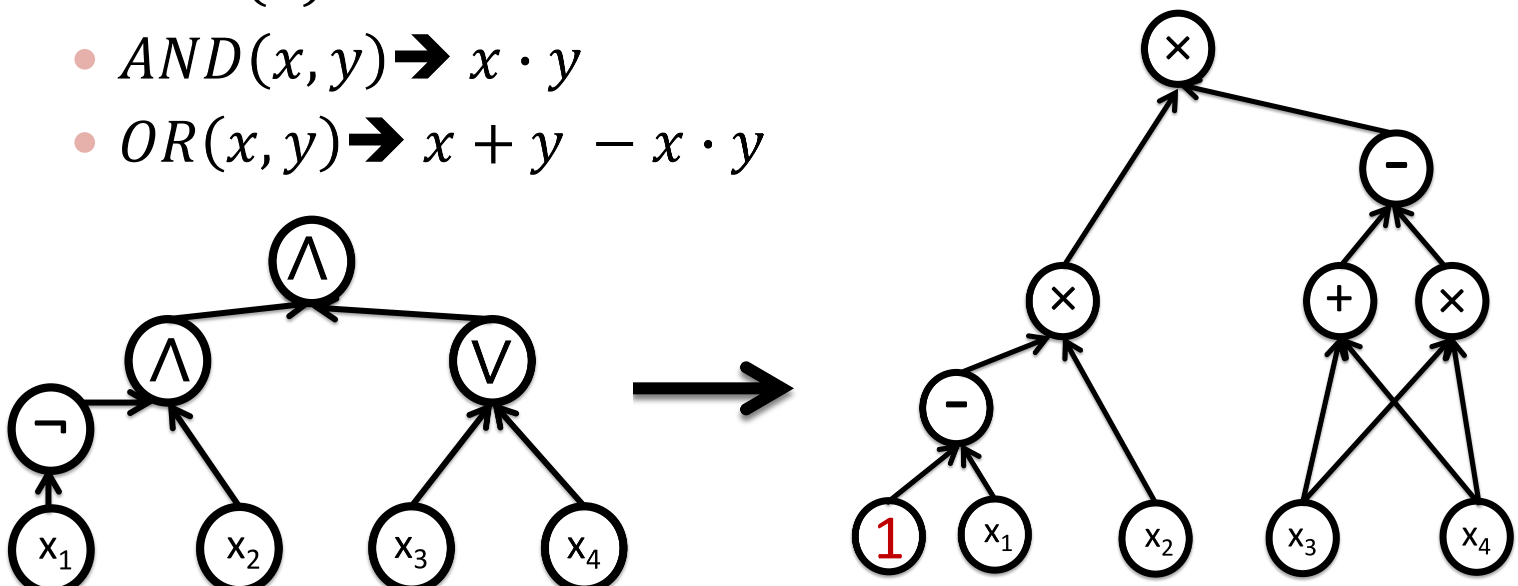
- Protocol:
- Let g be an extension polynomial of φ.
- Apply the sum-check protocol to compute $\sum_{x\in\{0,1\}^n} g(x)$.
  - Note: in final round of sum-check, V needs to compute $g(r)$ for some randomly chosen r in $\boldsymbol{F^n}$.
    - To control V's runtime, we need this to be fast.
  - To control communication and P and V's runtime, we need g to be "low-degree".
  - Key question: how to construct the extension polynomial g?

Arithmetization

- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x) \rightarrow 1 - x$
    - $AND(x, y) \rightarrow x \cdot y$
    - $OR(x, y) \rightarrow x + y - x \cdot y$

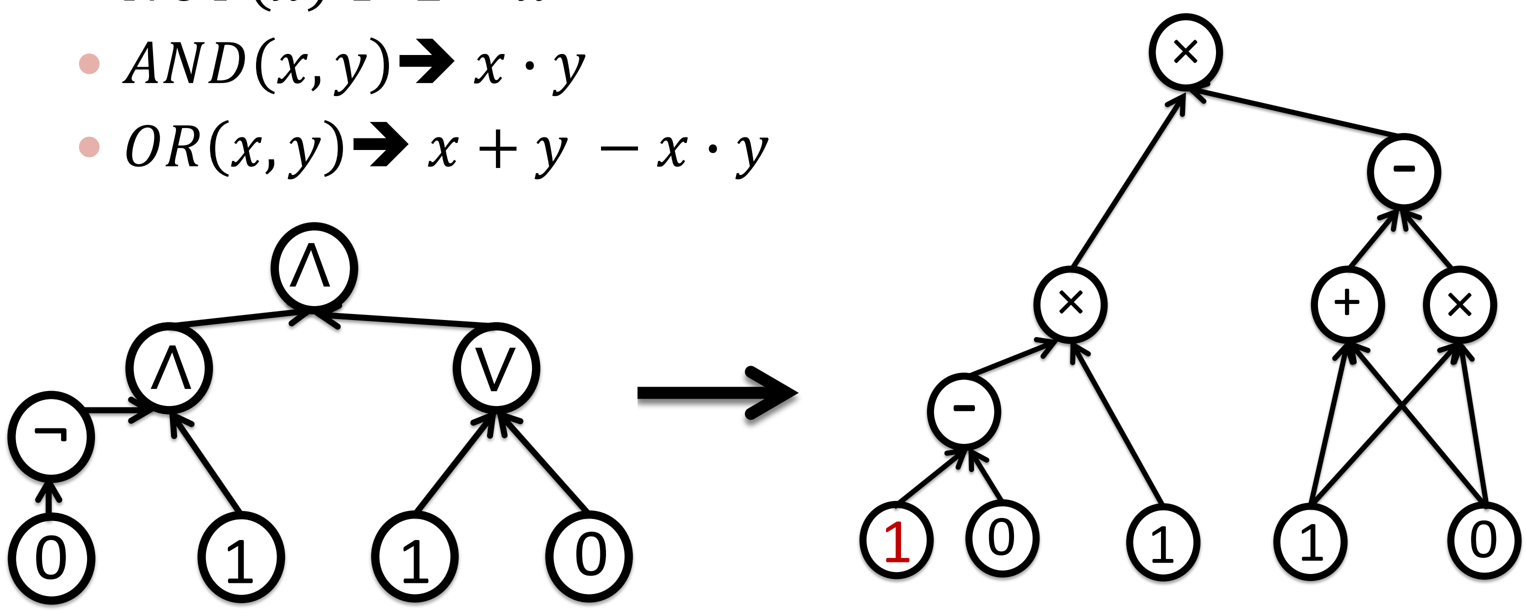
Arithmetization

- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x) \rightarrow 1 - x$
    - $AND(x, y) \rightarrow x \cdot y$
    - $OR(x, y) \rightarrow x + y - x \cdot y$



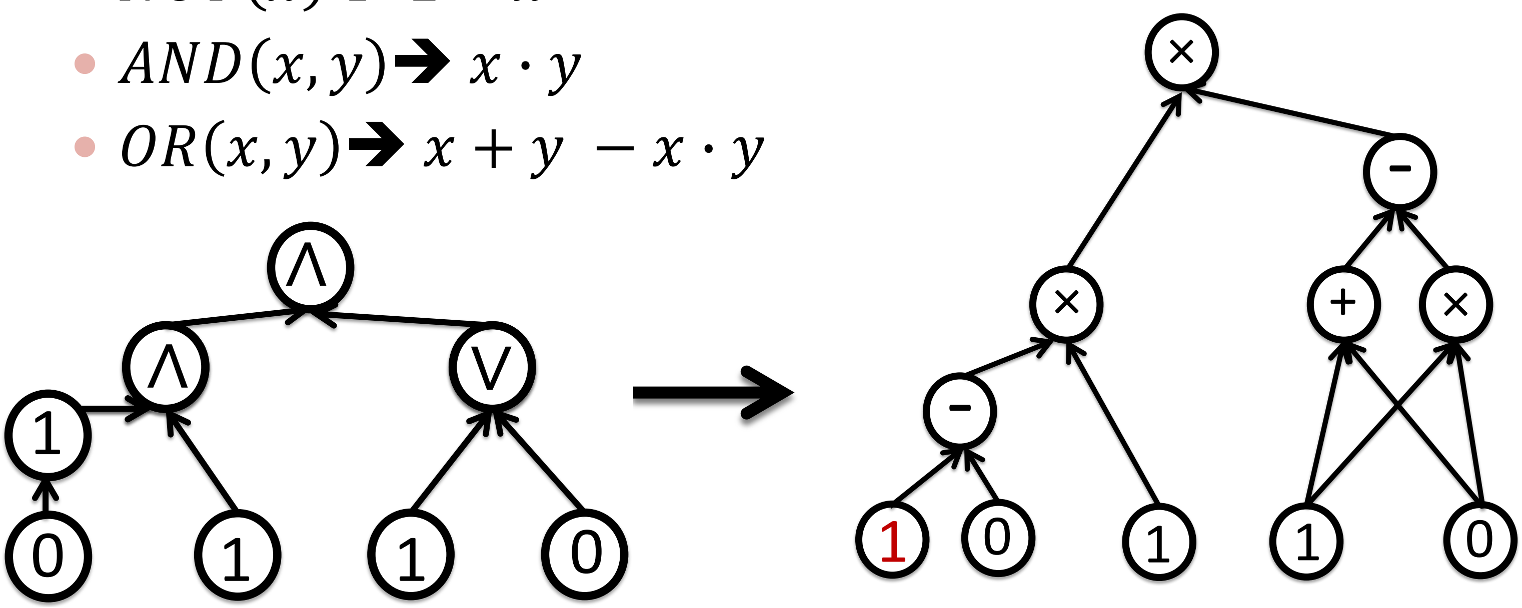
Arithmetization

- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x)$ ➔ $1 - x$
    - $AND(x, y)$ ➔ $x \cdot y$
    - $OR(x, y)$ ➔ $x + y - x \cdot y$



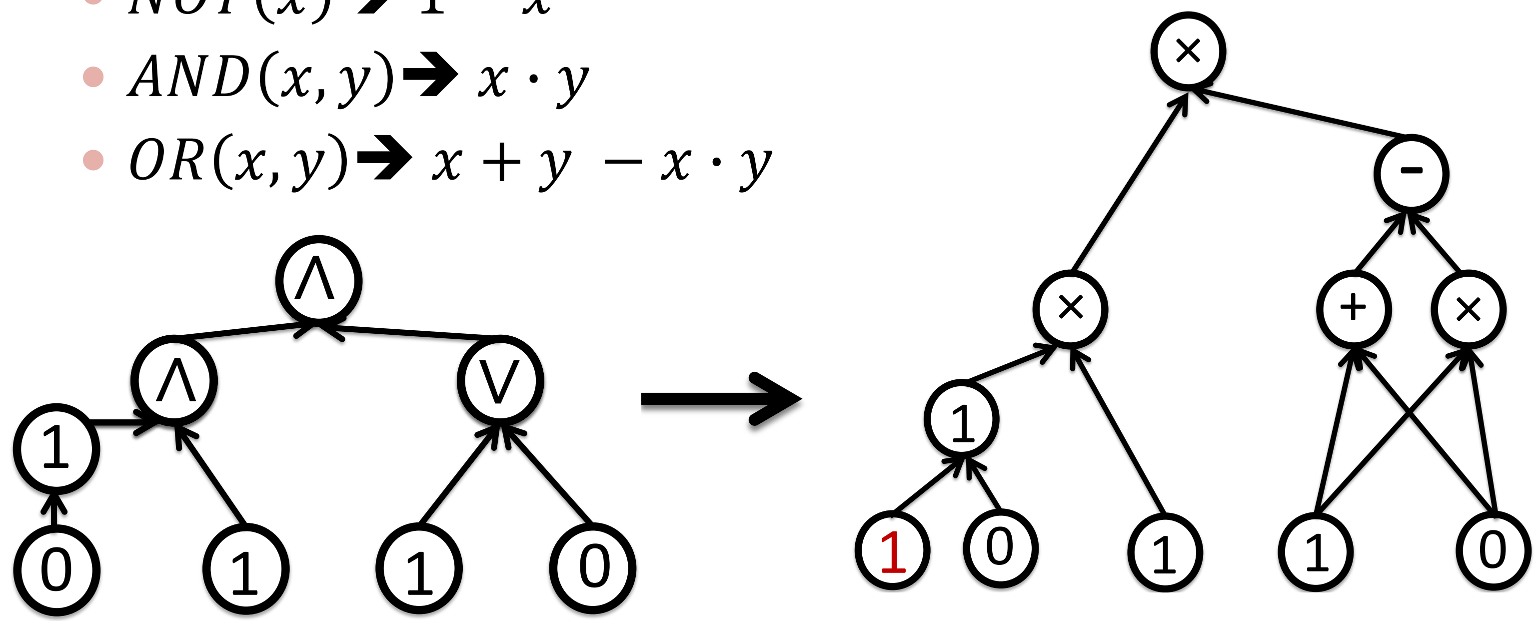
Arithmetization

- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x) \rightarrow 1 - x$
    - $AND(x, y) \rightarrow x \cdot y$
    - $OR(x, y) \rightarrow x + y \; - x \cdot y$



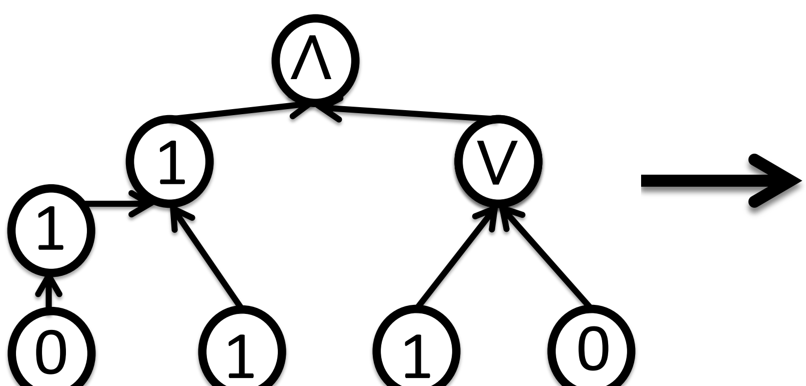
Arithmetization

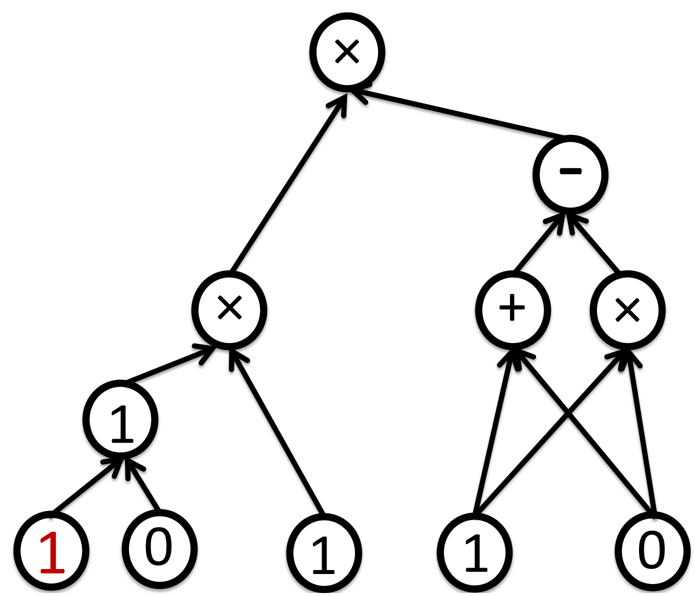
- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x)$ ➔ $1 - x$
    - $AND(x, y)$ ➔ $x \cdot y$
    - $OR(x, y)$ ➔ $x + y - x \cdot y$



Arithmetization

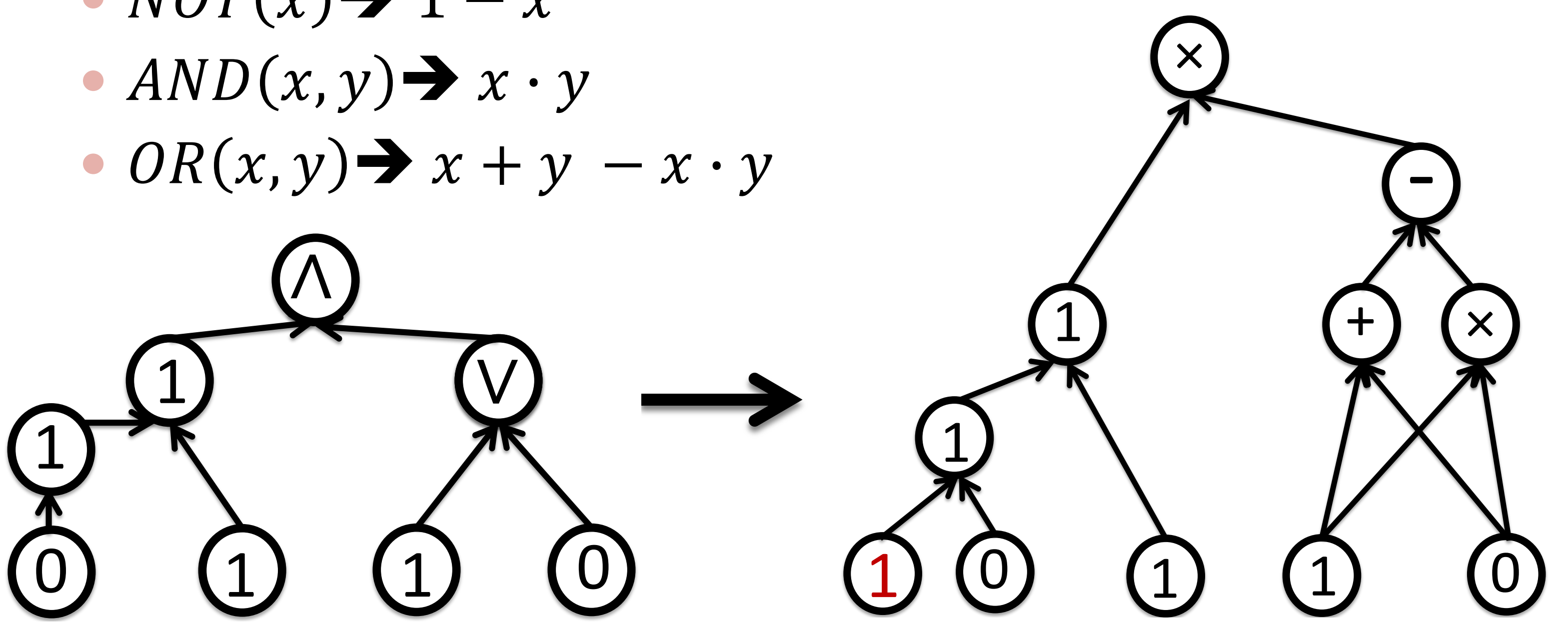
- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x)$ ➔ $1 - x$
    - $AND(x, y)$ ➔ $x \cdot y$
    - $OR(x, y)$ ➔ $x + y - x \cdot y$





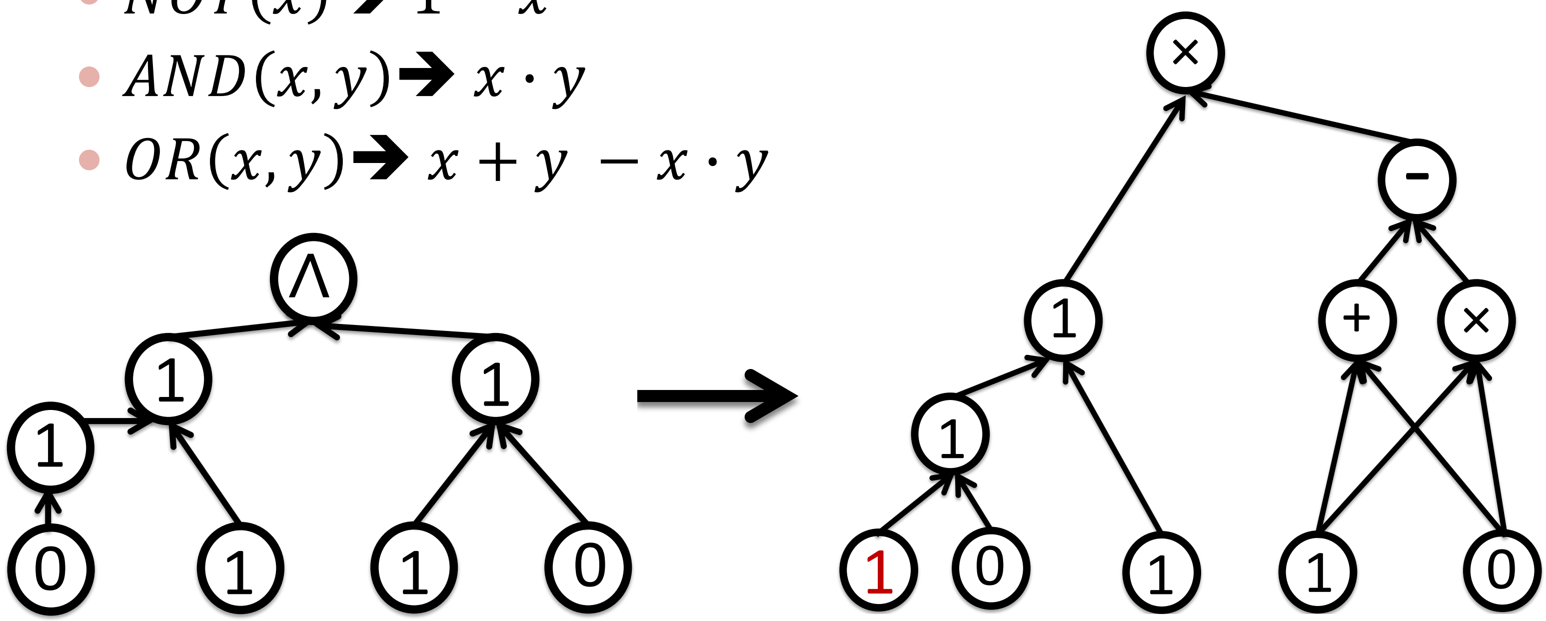
Arithmetization

- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x)$ ➔ $1 - x$
    - $AND(x, y)$ ➔ $x \cdot y$
    - $OR(x, y)$ ➔ $x + y - x \cdot y$



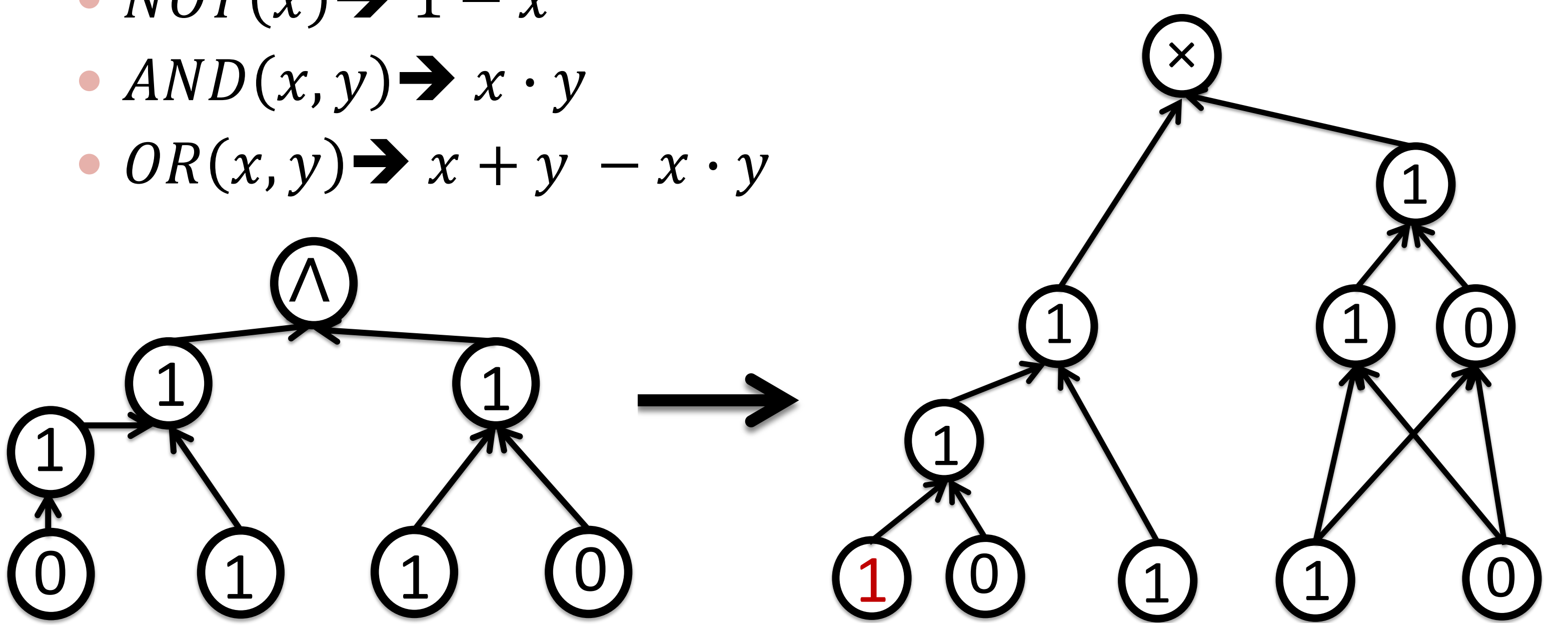
Arithmetization

- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x)$ ➔ $1 - x$
    - $AND(x, y)$ ➔ $x \cdot y$
    - $OR(x, y)$ ➔ $x + y - x \cdot y$



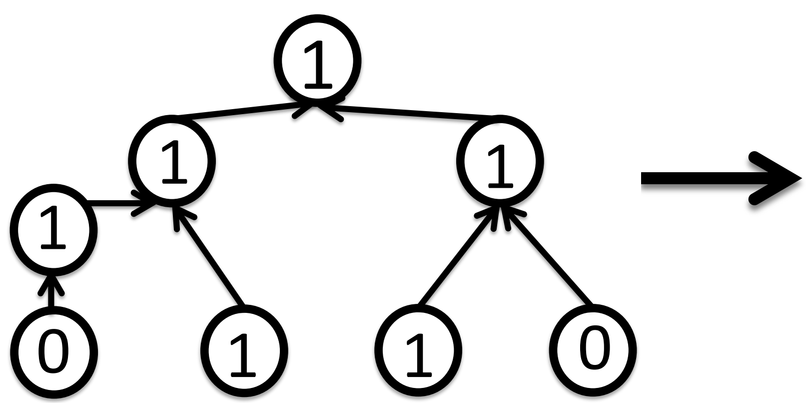
Arithmetization

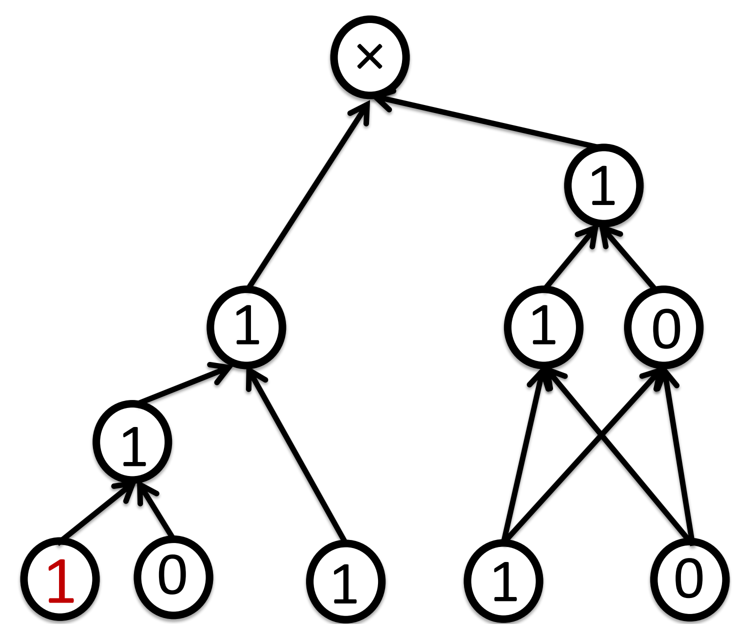
- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x)$ ➔ $1 - x$
    - $AND(x, y)$ ➔ $x \cdot y$
    - $OR(x, y)$ ➔ $x + y - x \cdot y$



Arithmetization

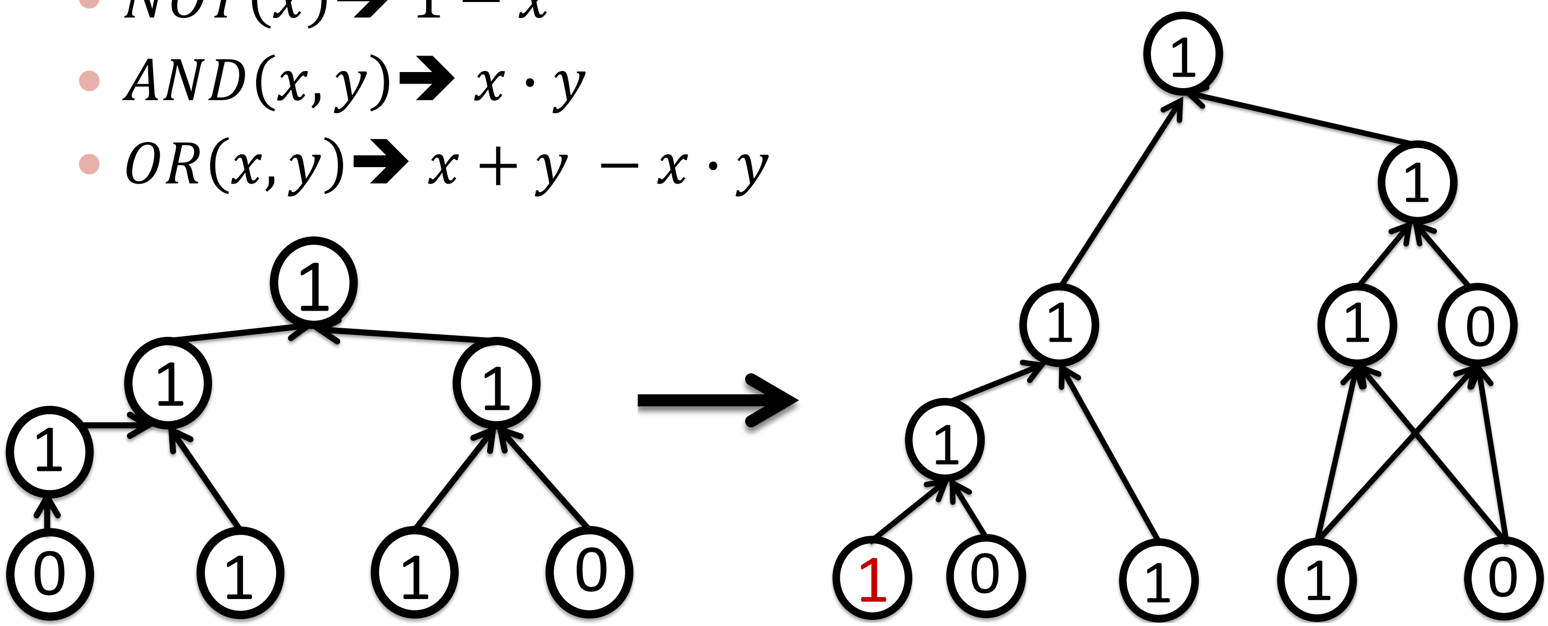
- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x)$ ➔ $1 - x$
    - $AND(x, y)$ ➔ $x \cdot y$
    - $OR(x, y)$ ➔ $x + y - x \cdot y$



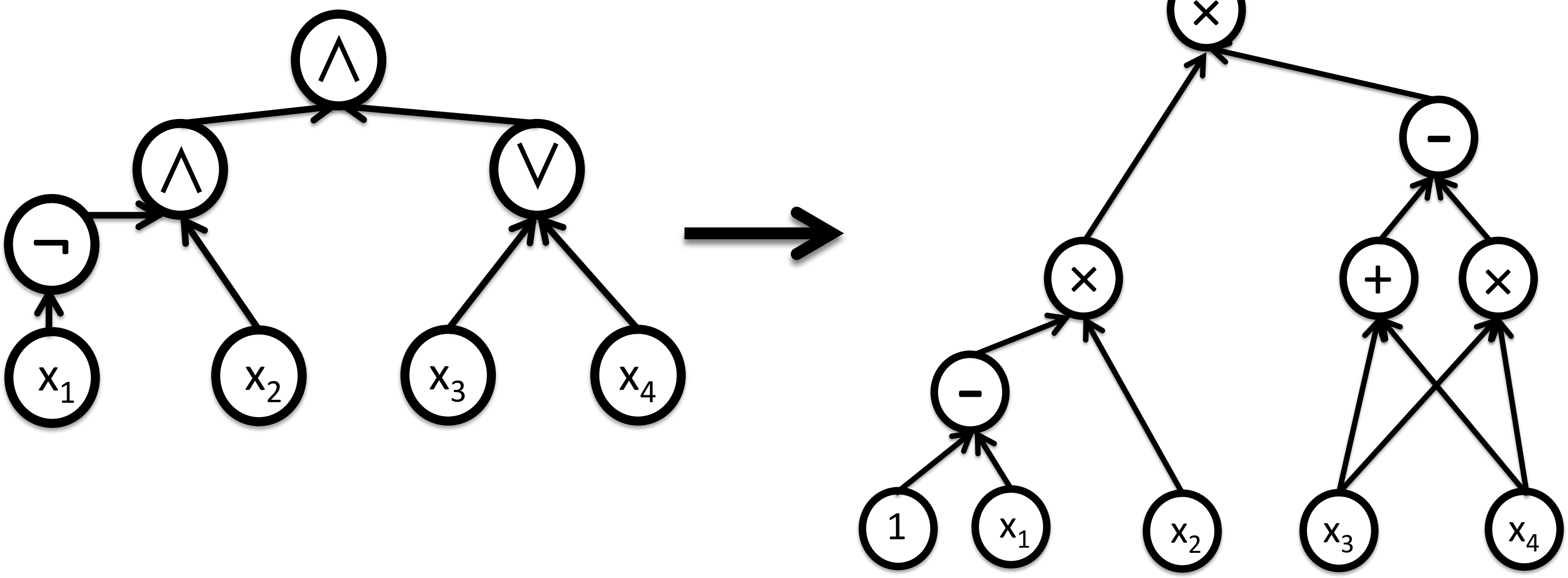


Arithmetization

- Key question: how to construct the extension polynomial g?
- Answer: Arithmetize φ
  - i.e., replace φ with an **arithmetic** circuit computing extension g
    - Go gate-by-gate through φ, replacing each gate with the gate's multilinear extension.
    - $NOT(x) \rightarrow 1 - x$
    - $AND(x, y) \rightarrow x \cdot y$
    - $OR(x, y) \rightarrow x + y - x \cdot y$



Summary of Arithmetization



Transforming a Boolean formula φ of size S into an arithmetic circuit computing an extension g of φ.

Note: $\deg(g) \leq S$, and g can be evaluated at any input, gate by gate, in time $O(S)$.

Costs of #SAT Protocol Applied to g

- Let φ be a Boolean formula of size S over n variables, g the extension obtained by arithmetizing φ.

Rounds	Communication	V Time	P Time
n	P sends a degree S polynomial in reach round, V sends one field element in each round $\Longrightarrow$ $O(S \cdot n)$ field elements sent in total.	•$O(S)$ time to process each of the n messages of P •$O(S)$ time to evaluate $g(r)$ $\Longrightarrow$ $O(S \cdot n)$ time total	P evaluates g at $O(S \cdot 2^n)$ points to determine each message $\Longrightarrow$ $O(S \cdot n \cdot 2^n)$ time in total.

IP=PSPACE

- #SAT is a **#P**-complete problem.
  - Hence, the protocol we just saw implies **every** problem in **#P** has an interactive proof with a polynomial time verifier.
- It is not much harder to show that this in fact holds for every problem in **PSPACE** [LFKN, Shamir].

IP=PSPACE

- #SAT is a **#P**-complete problem.
  - Hence, the protocol we just saw implies **every** problem in **#P** has an interactive proof with a polynomial time verifier.
- It is not much harder to show that this in fact holds for every problem in **PSPACE** [LFKN, Shamir].
- But is this a **practical** result?

IP=PSPACE

- #SAT is a **#P**-complete problem.
  - Hence, the protocol we just saw implies **every** problem in **#P** has an interactive proof with a polynomial time verifier.
- It is not much harder to show that this in fact holds for every problem in **PSPACE** [LFKN, Shamir].
- But is this a **practical** result?
  - No. The main reason: P's runtime.

IP=PSPACE

- #SAT is a **#P**-complete problem.
  - Hence, the protocol we just saw implies **every** problem in **#P** has an interactive proof with a polynomial time verifier.
- It is not much harder to show that this in fact holds for every problem in **PSPACE** [LFKN, Shamir].
- But is this a **practical** result?
  - No. The main reason: P's runtime.
  - When applying the protocols of [LFKN, Shamir] even to very simple problems, the honest prover would require **superpolynomial** time.

IP=PSPACE

- #SAT is a **#P**-complete problem.
  - Hence, the protocol we just saw implies **every** problem in **#P** has an interactive proof with a polynomial time verifier.
- It is not much harder to show that this in fact holds for every problem in **PSPACE** [LFKN, Shamir].
- But is this a **practical** result?
  - No. The main reason: P's runtime.
  - When applying the protocols of [LFKN, Shamir] even to very simple problems, the honest prover would require **superpolynomial** time.
  - The #SAT prover took time at least 2^n.
    - This seems unavoidable for #SAT, since we don't know how to even solve the problem in less than 2^n time.
    - But we can hope to solve "easier" problems without turning those problems into #SAT instances.