بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه هجدهم: سیستم اثبات تعاملی

Theory of computation - 002 - S18 - IP

18.404/6.840 Lecture 25

Last time:

- BPP
- Schwartz-Zippel Theorem
- EQROBP $\in$ BPP

Today: (Sipser §10.4)

- Interactive Proof Systems
- The class IP
- Graph isomorphism problem
- coNP $\subseteq$ IP (part 1)

Interactive Proofs – Introduction

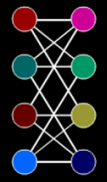
Illustration: Graph isomorphism testing

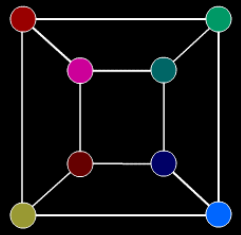
Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.

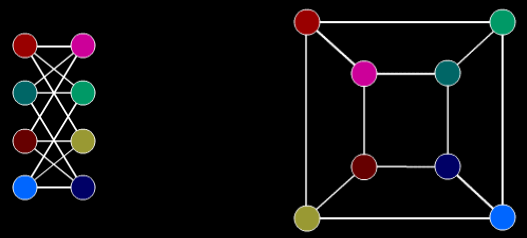




Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

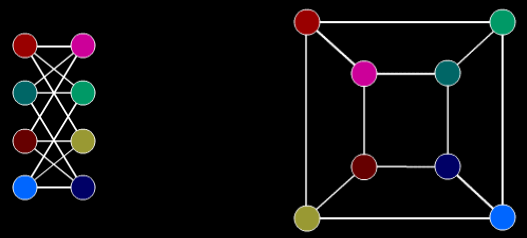
Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.



Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

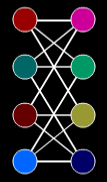
Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.

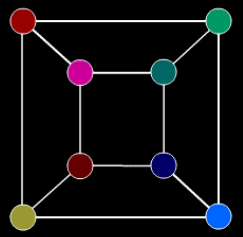


Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.



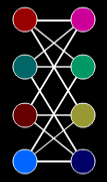


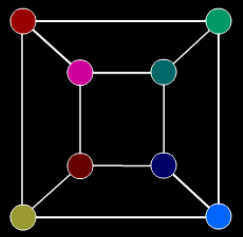
Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





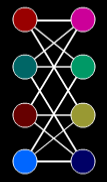
Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

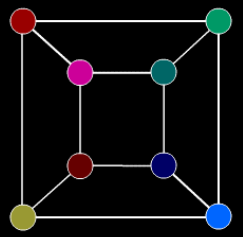
$ISO \in$ NP

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





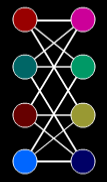
Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

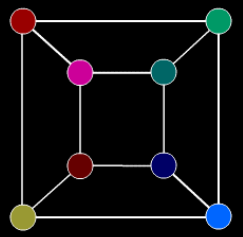
$ISO \in$ NP obvious

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

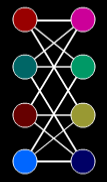
$ISO \in$ NP obvious

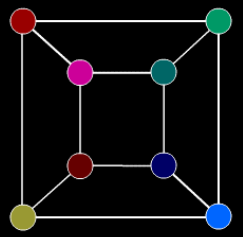
$ISO \in$ P ?

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

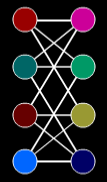
$ISO \in$ NP obvious

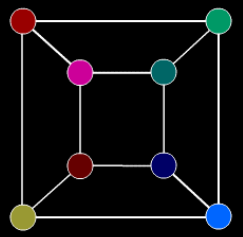
$ISO \in$ P ? unknown

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

$ISO \in$ NP obvious

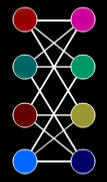
$ISO \in$ P ? unknown

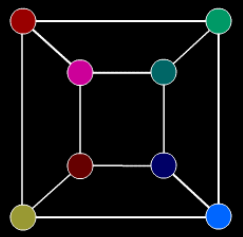
ISO is NP-complete ?

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

$ISO \in$ NP obvious

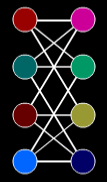
$ISO \in$ P ? unknown

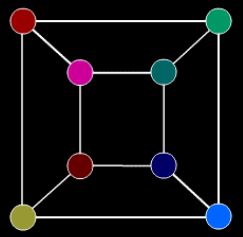
ISO is NP-complete ? unknown

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

$ISO \in$ NP obvious

$ISO \in$ P ? unknown

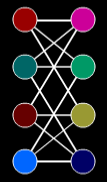
ISO is NP-complete ? unknown

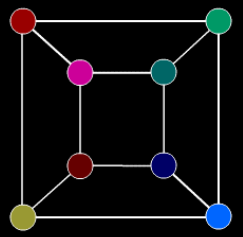
$\overline{ISO} \in$ NP ?

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

$ISO \in$ NP obvious

$ISO \in$ P ? unknown

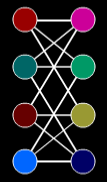
ISO is NP-complete ? unknown

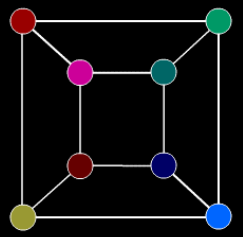
$\overline{ISO} \in$ NP ? unknown

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

$ISO \in$ NP obvious

$ISO \in$ P ? unknown

ISO is NP-complete ? unknown

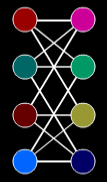
$\overline{ISO} \in$ NP ? unknown

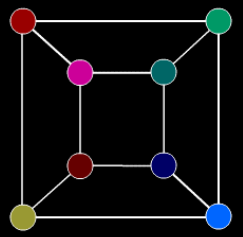
$ISO \in$ NP therefore a Prover can convince a poly-time Verifier that G and H are isomorphic (if true).

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

$ISO \in$ NP obvious

$ISO \in$ P ? unknown

ISO is NP-complete ? unknown

$\overline{ISO} \in$ NP ? unknown

$ISO \in$ NP therefore a Prover can convince a poly-time Verifier that G and H are isomorphic (if true).

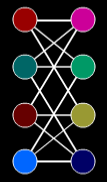
Even though $\overline{ISO} \in$ NP is unknown,

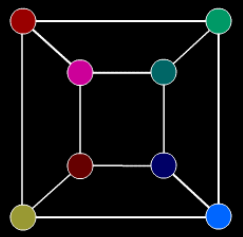
a Prover can still convince a poly-time Verifier that G and H are not isomorphic (if true).

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

$ISO \in$ NP obvious

$ISO \in$ P ? unknown

ISO is NP-complete ? unknown

$\overline{ISO} \in$ NP ? unknown

$ISO \in$ NP therefore a Prover can convince a poly-time Verifier that G and H are isomorphic (if true).

Even though $\overline{ISO} \in$ NP is unknown,

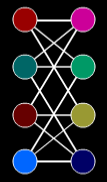
a Prover can still convince a poly-time Verifier that G and H are not isomorphic (if true).

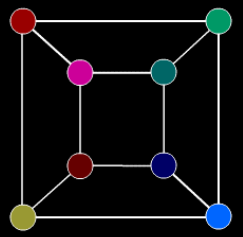
Requires *interaction* and a *probabilistic* Verifier.

Interactive Proofs – Introduction

Illustration: Graph isomorphism testing

Defn: Undirected graphs G and H are isomorphic if they are identical except for a permutation (rearrangement) of the nodes.





Defn: $ISO = \{\langle G, H\rangle \mid G \text{ and } H \text{ are isomorphic graphs}\}$

$ISO \in$ NP obvious

$ISO \in$ P ? unknown

ISO is NP-complete ? unknown

$\overline{ISO} \in$ NP ? unknown

$ISO \in$ NP therefore a Prover can convince a poly-time Verifier that G and H are isomorphic (if true).

Even though $\overline{ISO} \in$ NP is unknown,

a Prover can still convince a poly-time Verifier that G and H are not isomorphic (if true).

Requires *interaction* and a *probabilistic* Verifier.

Interactive Proofs – informal model

Interactive Proofs – informal model

Professor

Interactive Proofs – informal model

Professor

Probabilistic
polynomial time TM

Interactive Proofs – informal model

Professor

Probabilistic
polynomial time TM

Interactive Proofs – informal model

Professor

Probabilistic
polynomial time TM

Graduate Students

Interactive Proofs – informal model

Professor

Probabilistic
polynomial time TM

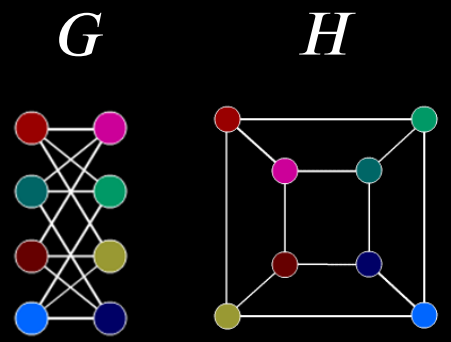
Graduate Students

Unlimited
computation

Interactive Proofs – informal model

Professor

Probabilistic
polynomial time TM



Professor wants to know if graphs G and H are isomorphic.

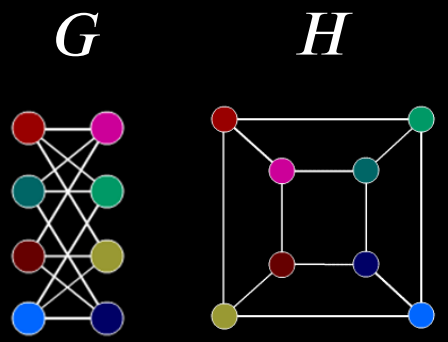
Graduate Students

Unlimited
computation

Interactive Proofs – informal model

Professor

Probabilistic
polynomial time TM



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.

Graduate Students

Unlimited
computation

Interactive Proofs – informal model

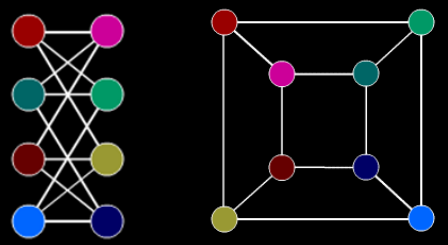
Professor

Probabilistic
polynomial time TM

Graduate Students

Unlimited
computation

G H



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

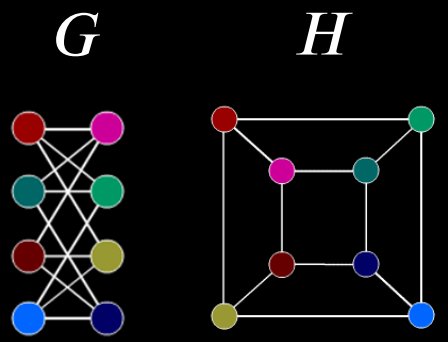
Interactive Proofs – informal model

Professor

Probabilistic
polynomial time TM

Graduate Students

Unlimited
computation



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

If the Students claim that G and H are isomorphic,
they can give the isomorphism and convince him.

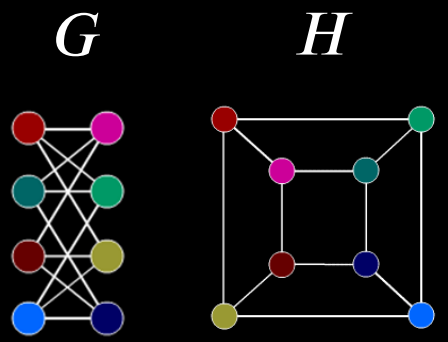
Interactive Proofs – informal model

Professor

Probabilistic
polynomial time TM

Graduate Students

Unlimited
computation



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

If the Students claim that G and H are isomorphic,
they can give the isomorphism and convince him.

But what if they claim that G and H are not isomorphic?

Interactive Proofs – informal model

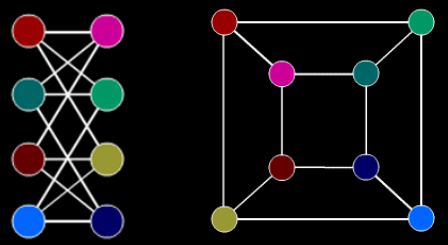
Professor

Probabilistic
polynomial time TM

Graduate Students

Unlimited
computation

G H



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

If the Students claim that G and H are isomorphic,
they can give the isomorphism and convince him.

But what if they claim that G and H are not isomorphic?
- The Professor randomly and secretly picks G or H and permutes it, then sends the result to the Students.

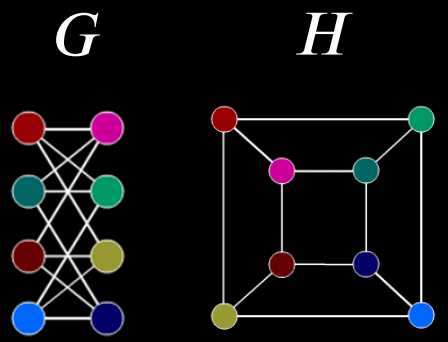
Interactive Proofs – informal model

Professor

Probabilistic
polynomial time TM

Graduate Students

Unlimited
computation



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

If the Students claim that G and H are isomorphic,
they can give the isomorphism and convince him.

But what if they claim that G and H are not isomorphic?
- The Professor randomly and secretly picks G or H and permutes it, then sends the result to the Students.
- If Students can identify which graph the Professor picked reliably (repeat this 100 times), then he's convinced.

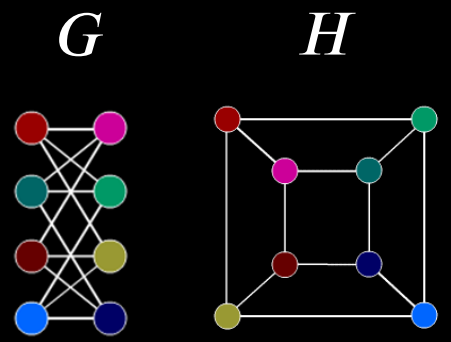
Interactive Proofs – informal model

Probabilistic
polynomial time TM

Professor = Verifier (V)

Unlimited
computation

Graduate Students



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

If the Students claim that G and H are isomorphic,
they can give the isomorphism and convince him.

But what if they claim that G and H are <u>not</u> isomorphic?
- The Professor randomly and secretly picks G or H and permutes it, then sends the result to the Students.
- If Students can identify which graph the Professor picked reliably (repeat this 100 times), then he's convinced.

Interactive Proofs – informal model

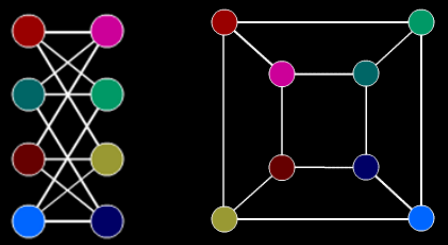
Probabilistic
polynomial time TM

Professor = Verifier (V)

Unlimited
computation

Graduate Students = Prover (P)

G H



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

If the Students claim that G and H are isomorphic,
they can give the isomorphism and convince him.

But what if they claim that G and H are <u>not</u> isomorphic?
- The Professor randomly and secretly picks G or H and permutes it, then sends the result to the Students.
- If Students can identify which graph the Professor picked reliably (repeat this 100 times), then he's convinced.

Interactive Proofs – informal model

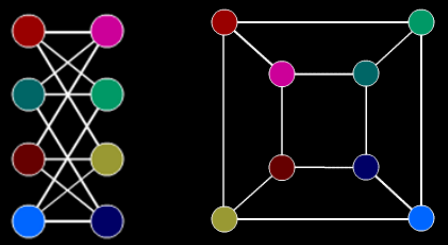
Probabilistic
polynomial time TM

Professor = Verifier (V)

Unlimited
computation

Graduate Students = Prover (P)

G H



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

If the Students claim that G and H are isomorphic,
they can give the isomorphism and convince him.

But what if they claim that G and H are not isomorphic?
- The Professor randomly and secretly picks G or H and permutes it, then sends the result to the Students.
- If Students can identify which graph the Professor picked reliably (repeat this 100 times), then he's convinced.

Interactive Proofs – informal model

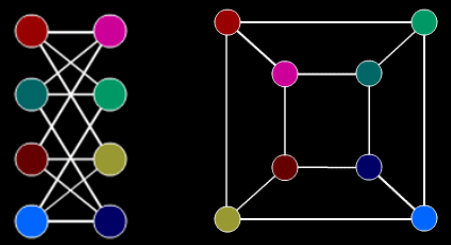
Probabilistic
polynomial time TM

Professor = Verifier (V)

Unlimited
computation

Graduate Students = Prover (P)

G H



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

If the Students claim that G and H are isomorphic, they can give the isomorphism and convince him.

But what if they claim that G and H are not isomorphic?
- The Professor randomly and secretly picks G or H and permutes it, then sends the result to the Students.
- If Students can identify which graph the Professor picked reliably (repeat this 100 times), then he's convinced.

Interactive Proofs – informal model

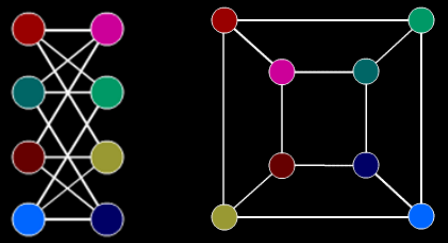
Probabilistic
polynomial time TM

Professor = Verifier (V)

Unlimited
computation

Graduate Students = Prover (P)

G H



Professor wants to know if graphs G and H are isomorphic.
- He asks his Students to figure out the answer.
- But he doesn't trust their answer. He must be convinced.

If the Students claim that G and H are isomorphic, they can give the isomorphism and convince him.

But what if they claim that G and H are not isomorphic?
- The Professor randomly and secretly picks G or H and permutes it, then sends the result to the Students.
- If Students can identify which graph the Professor picked reliably (repeat this 100 times), then he's convinced.

Interactive Proofs – formal model

Two interacting parties

Verifier (V): Probabilistic polynomial time TM

Prover (P): Unlimited computational power

Interactive Proofs – formal model

Two interacting parties

Verifier (V): Probabilistic polynomial time TM

Prover (P): Unlimited computational power

Both P and V see input w.

Interactive Proofs – formal model

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.

Interactive Proofs – formal model

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Interactive Proofs – formal model

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Interactive Proofs – formal model

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{A \mid$ for some V and P (This P is an "honest" prover)

Interactive Proofs – formal model

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{A \mid$ for some V and P (This P is an "honest" prover)

$$w \in A \rightarrow \text{Pr}\,[\,(\text{V} \leftrightarrow \text{P}) \text{ accepts } w\,] \geq \frac{2}{3}$$

Interactive Proofs – formal model

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{A \mid$ for some V and P (This P is an "honest" prover)

$$w \in A \rightarrow \text{Pr}\,[\,(\text{V} \leftrightarrow \text{P}) \text{ accepts } w\,] \geq \frac{2}{3}$$

$$w \notin A \rightarrow \text{for any prover } \tilde{\text{P}} \quad \text{Pr}\,[\,(\text{V} \leftrightarrow \tilde{\text{P}}) \text{ accepts } w\,] \leq \frac{1}{3}$$

Interactive Proofs – formal model

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{A \mid$ for some V and P (This P is an "honest" prover)

$$w \in A \rightarrow \Pr[\,(V \leftrightarrow P) \text{ accepts } w\,] \geq \frac{2}{3}$$

$$w \notin A \rightarrow \text{for any prover } \tilde{P} \quad \Pr[\,(V \leftrightarrow \tilde{P}) \text{ accepts } w\,] \leq \frac{1}{3}$$

Think of $\tilde{P}$ as a "crooked" prover trying to make V accept when it shouldn't.

Interactive Proofs – formal model

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{A \mid$ for some V and P (This P is an "honest" prover)

$$w \in A \rightarrow \Pr[(V \leftrightarrow P) \text{ accepts } w] \geq \frac{2}{3}$$

$$w \notin A \rightarrow \text{for any prover } \tilde{P} \quad \Pr[(V \leftrightarrow \tilde{P}) \text{ accepts } w] \leq \frac{1}{3}$$

Think of $\tilde{P}$ as a "crooked" prover trying to make V accept when it shouldn't.

An amplification lemma can improve the error probability from $\frac{1}{3}$ to $\frac{1}{2^{\text{poly}(n)}}$

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:

2) V→P Randomly choose G or H and permute to get K, then send K

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:
2) V→P Randomly choose G or H and permute to get K, then send K
3) P→V Compare K with G and H. Send "G" or "H" (V's choice in step 2)

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:
2) V→P Randomly choose G or H and permute to get K, then send K
3) P→V Compare K with G and H. Send "G" or "H" (V's choice in step 2)
4) V *accepts* if P was correct both times. Otherwise V *rejects*.

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:
2) V→P Randomly choose G or H and permute to get K, then send K
3) P→V Compare K with G and H. Send "G" or "H" (V's choice in step 2)
4) V *accepts* if P was correct both times. Otherwise V *rejects*.

If G and H are not isomorphic then P can determine which graph V chose randomly.

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:
2) V→P Randomly choose G or H and permute to get K, then send K
3) P→V Compare K with G and H. Send "G" or "H" (V's choice in step 2)
4) V *accepts* if P was correct both times. Otherwise V *rejects*.

If G and H are not isomorphic then P can determine which graph V chose randomly.

Thus Pr [(V $\leftrightarrow$ P) accepts $\langle G, H \rangle$] $= 1 \geq \frac{2}{3}$

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:
2) V→P Randomly choose G or H and permute to get K, then send K
3) P→V Compare K with G and H. Send "G" or "H" (V's choice in step 2)
4) V *accepts* if P was correct both times. Otherwise V *rejects*.

If G and H are not isomorphic then P can determine which graph V chose randomly.

Thus Pr [(V $\leftrightarrow$ P) accepts $\langle G, H \rangle$] $= 1 \geq \frac{2}{3}$

If G and H are isomorphic then any $\tilde{P}$ has no way to tell which graph V chose randomly.

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:
2) V→P Randomly choose G or H and permute to get K, then send K
3) P→V Compare K with G and H. Send "G" or "H" (V's choice in step 2)
4) V *accepts* if P was correct both times. Otherwise V *rejects*.

If G and H are not isomorphic then P can determine which graph V chose randomly.

Thus Pr [(V $\leftrightarrow$ P) accepts $\langle G, H \rangle$] $= 1 \geq \frac{2}{3}$

If G and H are isomorphic then any $\tilde{P}$ has no way to tell which graph V chose randomly.

So $\tilde{P}$ has a 50% chance of answering correctly each time and a 25% chance of being correct twice.

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:
2) V→P Randomly choose G or H and permute to get K, then send K
3) P→V Compare K with G and H. Send "G" or "H" (V's choice in step 2)
4) V *accepts* if P was correct both times. Otherwise V *rejects*.

If G and H are not isomorphic then P can determine which graph V chose randomly.

Thus Pr [(V $\leftrightarrow$ P) accepts $\langle G, H \rangle$] $= 1 \geq \frac{2}{3}$

If G and H are isomorphic then any $\tilde{P}$ has no way to tell which graph V chose randomly.

So $\tilde{P}$ has a 50% chance of answering correctly each time and a 25% chance of being correct twice.

Thus Pr [(V $\leftrightarrow \tilde{P}$) accepts $\langle G, H \rangle$] $= \frac{1}{4} < \frac{1}{3}$.

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:
2) V→P Randomly choose G or H and permute to get K, then send K
3) P→V Compare K with G and H. Send "G" or "H" (V's choice in step 2)
4) V *accepts* if P was correct both times. Otherwise V *rejects*.

If G and H are not isomorphic then P can determine which graph V chose randomly.

Thus Pr [(V $\leftrightarrow$ P) accepts $\langle G, H \rangle$] $= 1 \geq \frac{2}{3}$

If G and H are isomorphic then any $\tilde{P}$ has no way to tell which graph V chose randomly.

So $\tilde{P}$ has a 50% chance of answering correctly each time and a 25% chance of being correct twice.

Thus Pr [(V $\leftrightarrow \tilde{P}$) accepts $\langle G, H \rangle$] $= \frac{1}{4} < \frac{1}{3}$.

Check-in 25.1

$\overline{ISO} \in$ IP

Theorem: $\overline{ISO} \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle G, H \rangle$

1) Repeat twice:
2) V→P Randomly choose G or H and permute to get K, then send K
3) P→V Compare K with G and H. Send "G" or "H" (V's choice in step 2)
4) V *accepts* if P was correct both times. Otherwise V *rejects*.

Check-in 25.1

Suppose we change the model to allow the Prover access to the Verifier's random choices. Now consider the same protocol as described above. What language does it describe?

(a) $\{\langle G, H \rangle | \ G \neq H\}$

(b) $\{\langle G, H \rangle |$ G and H are not isomorphic $\}$

(c) $\{\langle G, H \rangle |$ G and H are any two graphs $\}$

(d) $\varnothing$

Facts about IP – Checkin 25.2

Facts about IP – Checkin 25.2

Which of the following is true?
Check all that apply

a) NP $\subseteq$ IP

b) BPP $\subseteq$ IP

c) IP $\subseteq$ PSPACE

Facts about IP – Checkin 25.2

Which of the following is true?
Check all that apply

a) NP $\subseteq$ IP [V is deterministic]

b) BPP $\subseteq$ IP

c) IP $\subseteq$ PSPACE

Facts about IP – Checkin 25.2

Which of the following is true?
Check all that apply

a) NP $\subseteq$ IP [V is deterministic]

b) BPP $\subseteq$ IP [V ignores P]

c) IP $\subseteq$ PSPACE

Facts about IP – Checkin 25.2

Which of the following is true?
Check all that apply

a) NP $\subseteq$ IP [V is deterministic]

b) BPP $\subseteq$ IP [V ignores P]

c) IP $\subseteq$ PSPACE [We won't prove. Idea: explore all possible interactions in poly space.]

Facts about IP – Checkin 25.2

Which of the following is true?
Check all that apply

a) NP $\subseteq$ IP [V is deterministic]

b) BPP $\subseteq$ IP [V ignores P]

c) IP $\subseteq$ PSPACE [We won't prove. Idea: explore all possible interactions in poly space.]

Surprising Theorem: PSPACE $\subseteq$ IP so IP = PSPACE

Facts about IP – Checkin 25.2

Which of the following is true?
Check all that apply

a) NP $\subseteq$ IP [V is deterministic]

b) BPP $\subseteq$ IP [V ignores P]

c) IP $\subseteq$ PSPACE [We won't prove. Idea: explore all possible interactions in poly space.]

Surprising Theorem: PSPACE $\subseteq$ IP so IP = PSPACE

We will prove only a weaker statement: coNP $\subseteq$ IP

$\#SAT$ problem

Defn: $\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

$\#SAT$ problem

Defn: $\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Let $\#\phi =$ the number of satisfying assignments of Boolean formula ϕ.

$\#SAT$ problem

Defn: $\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Let $\#\phi =$ the number of satisfying assignments of Boolean formula ϕ.

So $\#SAT = \{\langle \phi, k \rangle \mid k = \#\phi\}$

$\#SAT$ problem

Defn: $\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Let $\#\phi =$ the number of satisfying assignments of Boolean formula ϕ.

So $\#SAT = \{\langle \phi, k \rangle \mid k = \#\phi\}$

Defn: Language B is NP-hard if $A \leq_P B$ for every $A \in$ NP.

(Note: B is NP-complete if B is NP-hard and $B \in$ NP.)

$\#SAT$ problem

Defn: $\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Let $\#\phi =$ the number of satisfying assignments of Boolean formula ϕ.

So $\#SAT = \{\langle \phi, k \rangle \mid k = \#\phi\}$

Defn: Language B is NP-hard if $A \leq_P B$ for every $A \in$ NP.

(Note: B is NP-complete if B is NP-hard and $B \in$ NP.)

Theorem: $\#SAT$ is coNP-hard

$\#SAT$ problem

Defn: $\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Let $\#\phi =$ the number of satisfying assignments of Boolean formula ϕ.

So $\#SAT = \{\langle \phi, k \rangle \mid k = \#\phi\}$

Defn: Language B is NP-hard if $A \leq_P B$ for every $A \in$ NP.

(Note: B is NP-complete if B is NP-hard and $B \in$ NP.)

Theorem: $\#SAT$ is coNP-hard

Proof: Show $\overline{SAT} \leq_P \#SAT$

$$f(\langle \phi \rangle) = \langle \phi, 0 \rangle$$

$\#SAT$ problem

Defn: $\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Let $\#\phi =$ the number of satisfying assignments of Boolean formula ϕ.

So $\#SAT = \{\langle \phi, k \rangle \mid k = \#\phi\}$

Defn: Language B is NP-hard if $A \leq_P B$ for every $A \in$ NP.

(Note: B is NP-complete if B is NP-hard and $B \in$ NP.)

Theorem: $\#SAT$ is coNP-hard

Proof: Show $\overline{SAT} \leq_P \#SAT$

$$f(\langle \phi \rangle) = \langle \phi, 0 \rangle$$

To show coNP $\subseteq$ IP we will show $\#SAT \in$ IP

$\#SAT$ problem

Defn: $\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Let $\#\phi =$ the number of satisfying assignments of Boolean formula ϕ.

So $\#SAT = \{\langle \phi, k \rangle \mid k = \#\phi\}$

Defn: Language B is NP-hard if $A \leq_P B$ for every $A \in$ NP.

(Note: B is NP-complete if B is NP-hard and $B \in$ NP.)

Theorem: $\#SAT$ is coNP-hard

Proof: Show $\overline{SAT} \leq_P \#SAT$

$$f(\langle \phi \rangle) = \langle \phi, 0 \rangle$$

To show coNP $\subseteq$ IP we will show $\#SAT \in$ IP

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Let $\#\phi$ = the number of satisfying assignments of ϕ.

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

Equivalently:

$$\#\phi(a_1 \ldots a_i) = \sum_{\substack{a_{i+1}, \ldots, a_m \\ \in \{0,1\}}} \phi(a_1 \ldots a_m)$$

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

Equivalently:

$$\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$$

Two useful facts

1. $\#\phi(a_1 \ldots a_i) =$

$$\#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$$

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

Equivalently:

$$\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$$

Two useful facts

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$

2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots a_m)$

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

Equivalently:

$$\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$$

Two useful facts

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$

2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots a_m)$

Check-in 25.3

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variable

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

Equivalently: $\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$

Check-in 25.3

If $\#\phi = 9$ and $\#\phi(0) = 6$ then what do we know?

a) $\#\phi(1) = 3$ c) $\#\phi(00) \leq 5$

b) $\#\phi(1) = 15$ d) none of these

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$

2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots a_m)$

Check-in 25.3

$\#SAT \in$ IP – notation

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(01)$ be ϕ with $x_1 = 0$ and $x_2 = 1$.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variable

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

Equivalently: $\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$

Check-in 25.3

If $\#\phi = 9$ and $\#\phi(0) = 6$ then what do we know?

a) $\#\phi(1) = 3$ c) $\#\phi(00) \leq 5$

b) $\#\phi(1) = 15$ d) none of these

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$

2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots a_m)$

Check-in 25.3

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$$\#\phi(1) = \#\phi(10) + \#\phi(11)$$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

k

=

$\#\phi$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots0}^{m-1}) + \#\phi(\overbrace{0\cdots0}^{m-1}1)$

$\vdots$

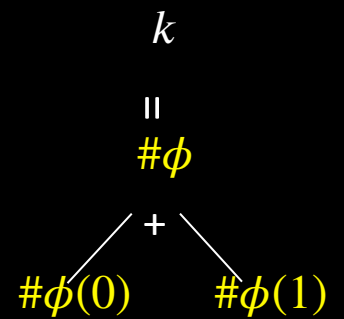
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

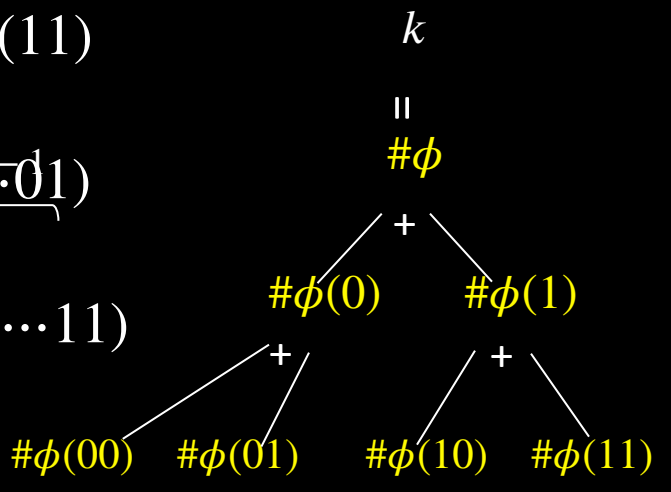
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m}) + \#\phi(\overbrace{0\cdots01}^{m})$

$\vdots$

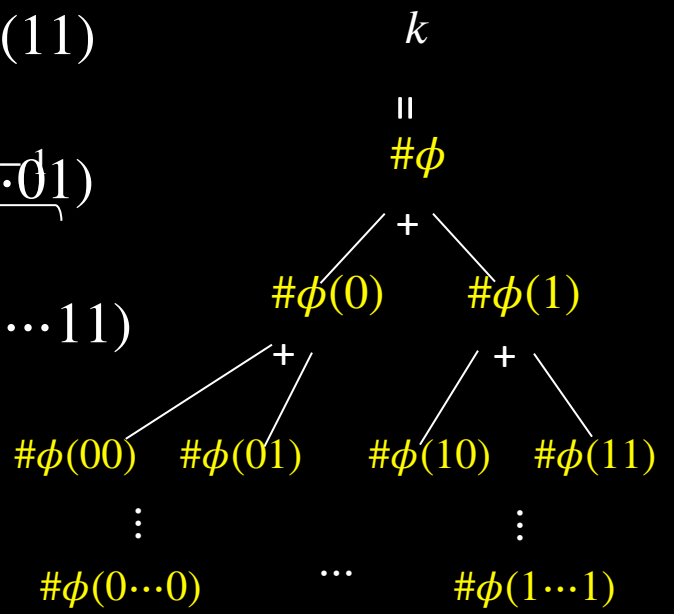
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m}) + \#\phi(\overbrace{0\cdots01}^{m})$

$\vdots$

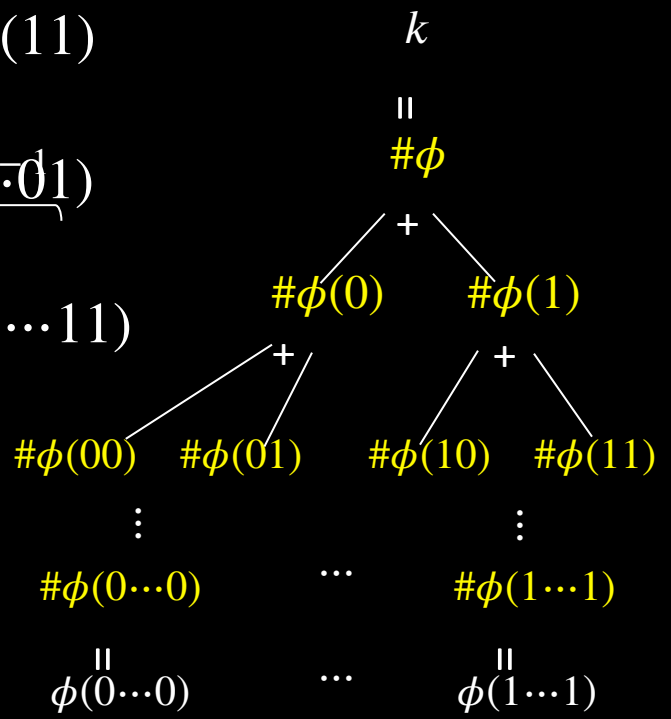
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m}) + \#\phi(\overbrace{0\cdots01}^{m})$

$\vdots$

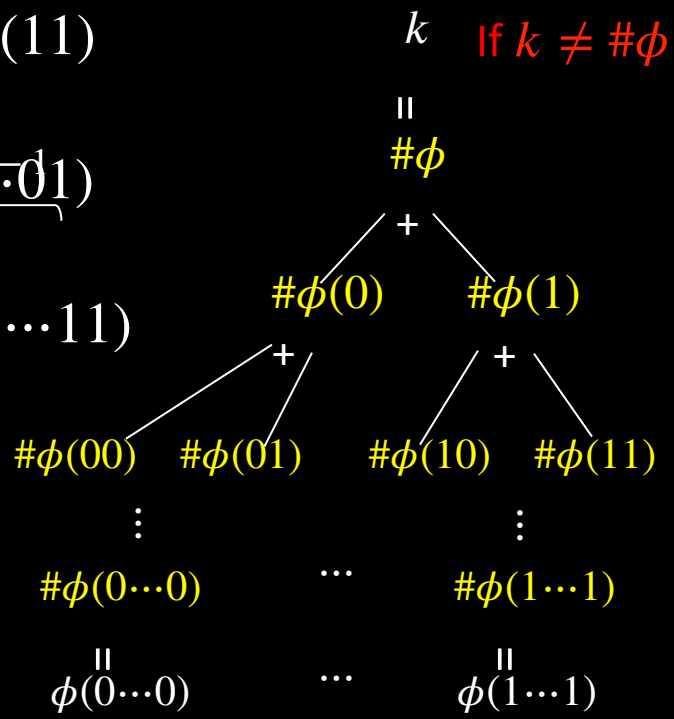
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

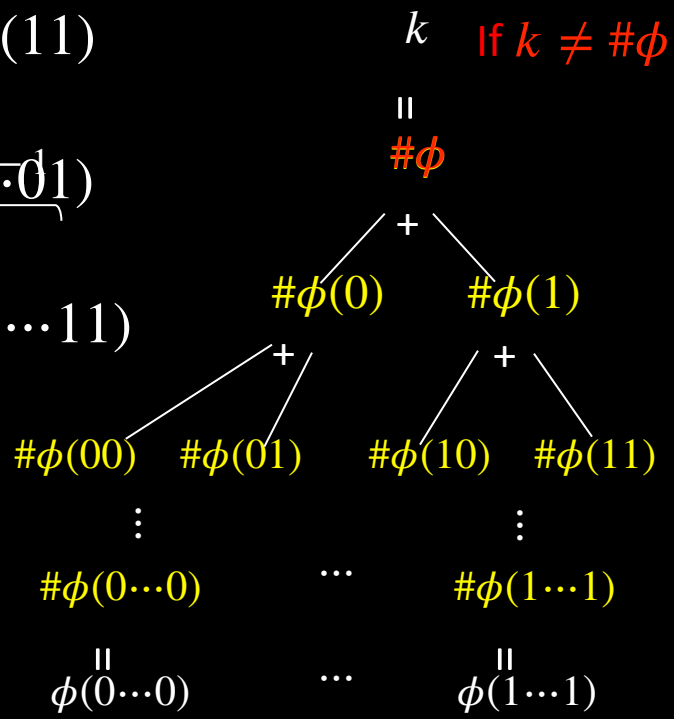
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m}) + \#\phi(\overbrace{0\cdots01}^{m})$

$\vdots$

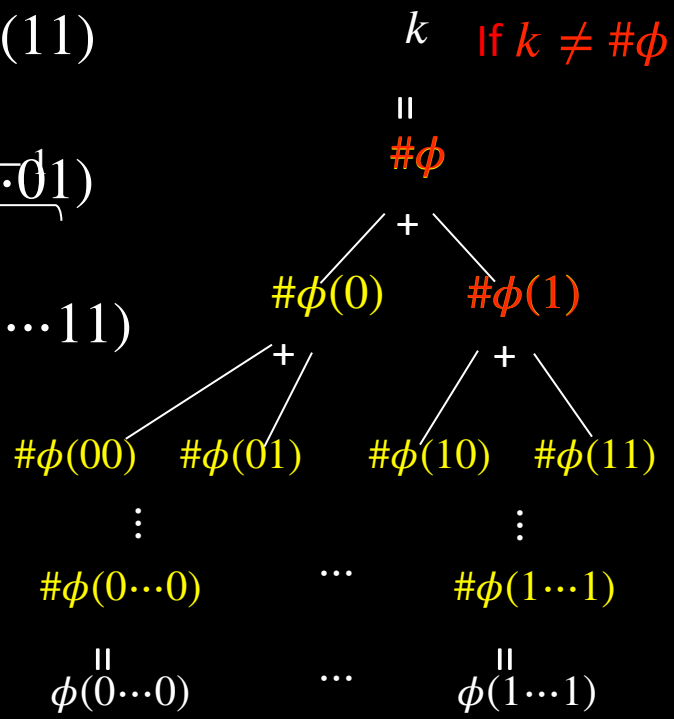
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m}) + \#\phi(\overbrace{0\cdots01}^{m})$

$\vdots$

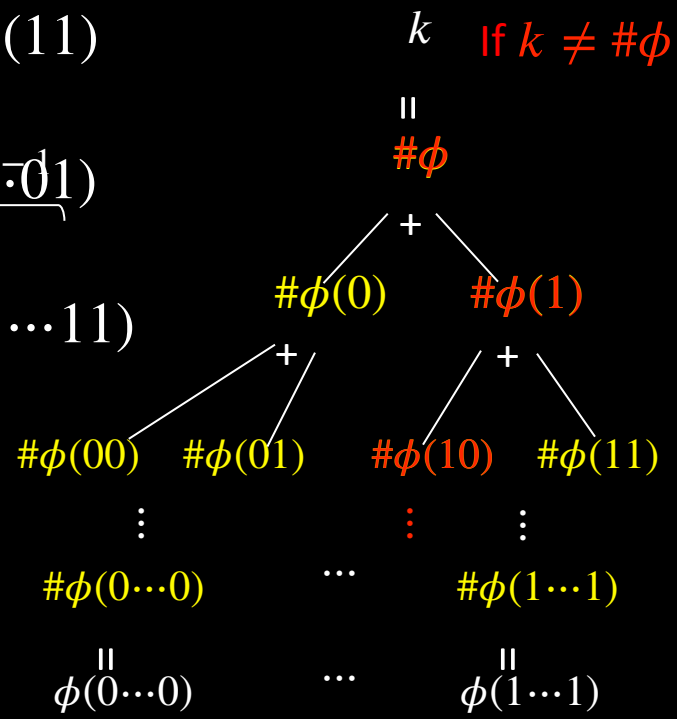
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m}) + \#\phi(\overbrace{0\cdots01}^{m})$

$\vdots$

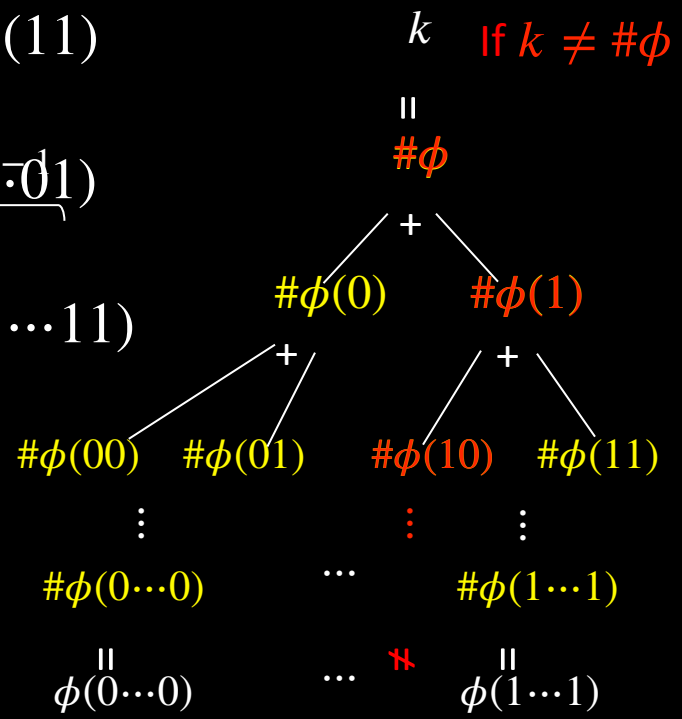
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m}) + \#\phi(\overbrace{0\cdots01}^{m})$

$\vdots$

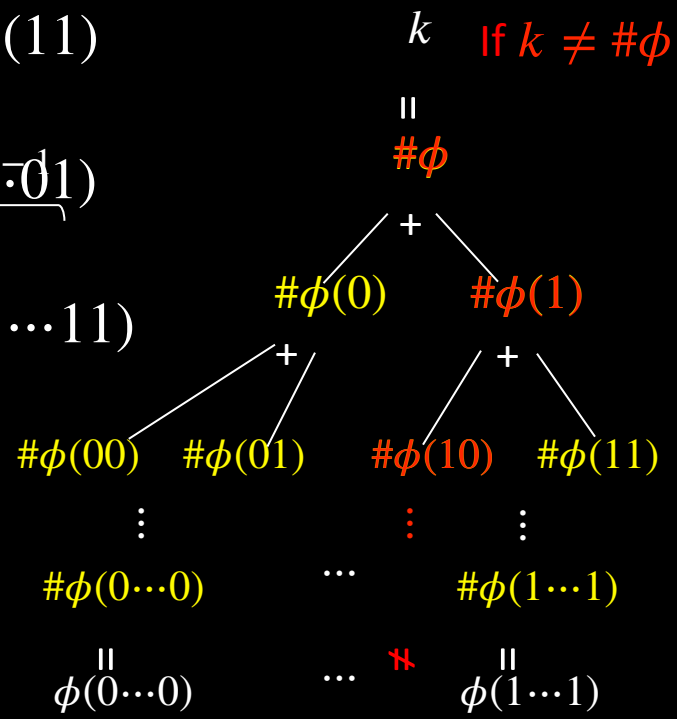
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



Problem: Exponential. How to fix?

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

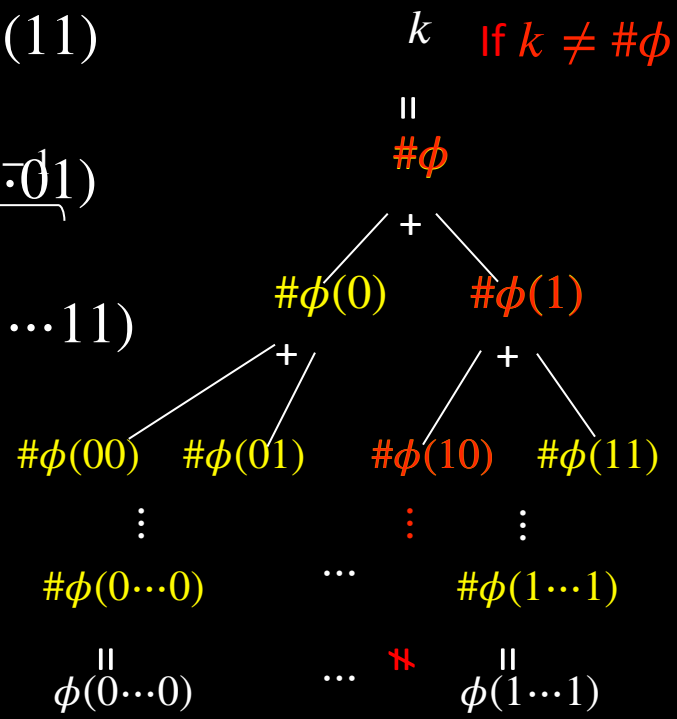
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

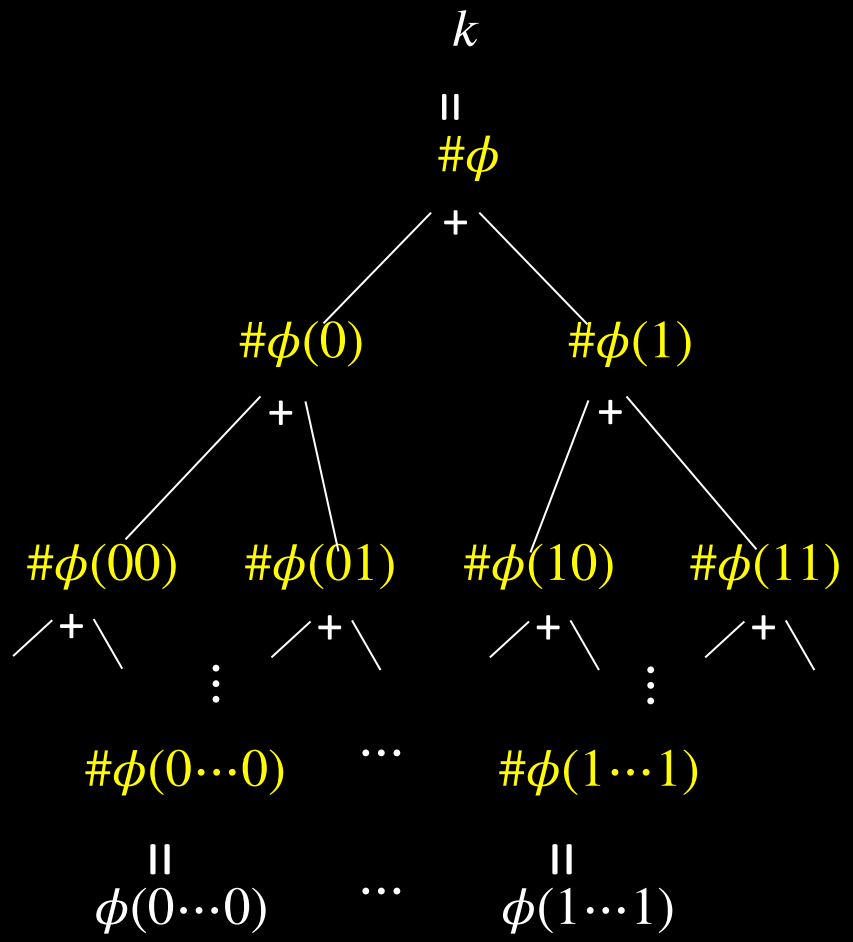
$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.

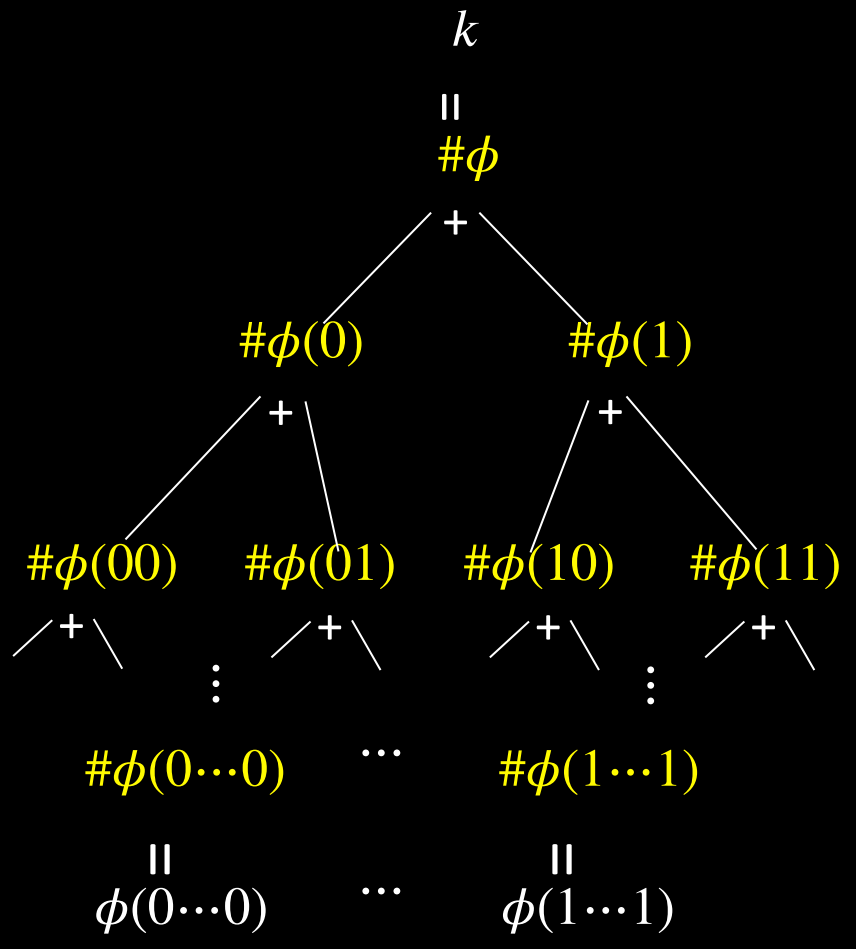


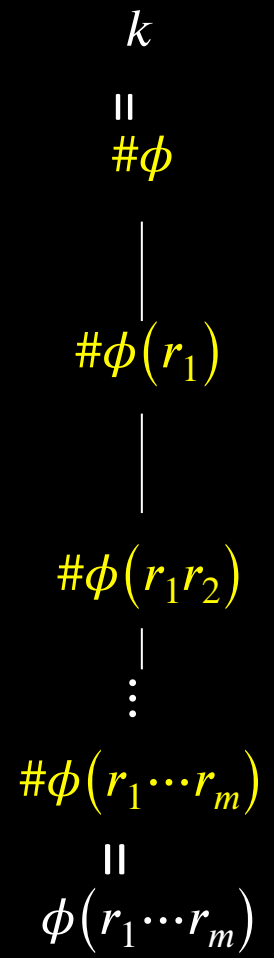
Problem: Exponential. How to fix?

Idea for fixing $\#SAT \in$ IP protocol

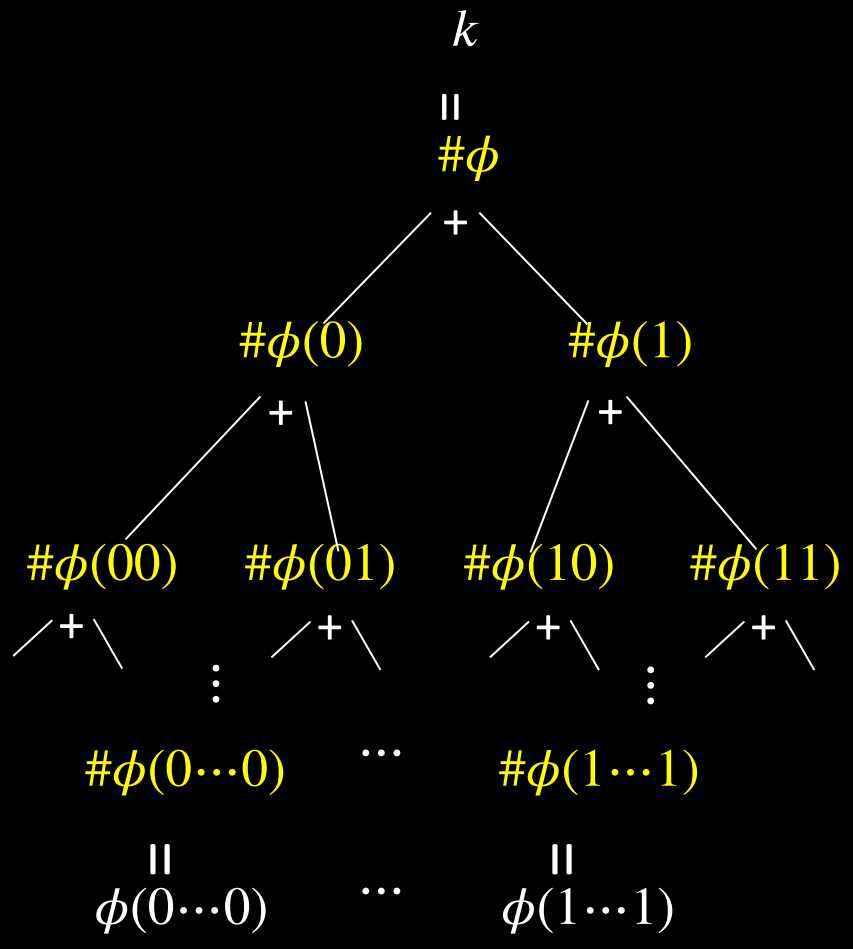


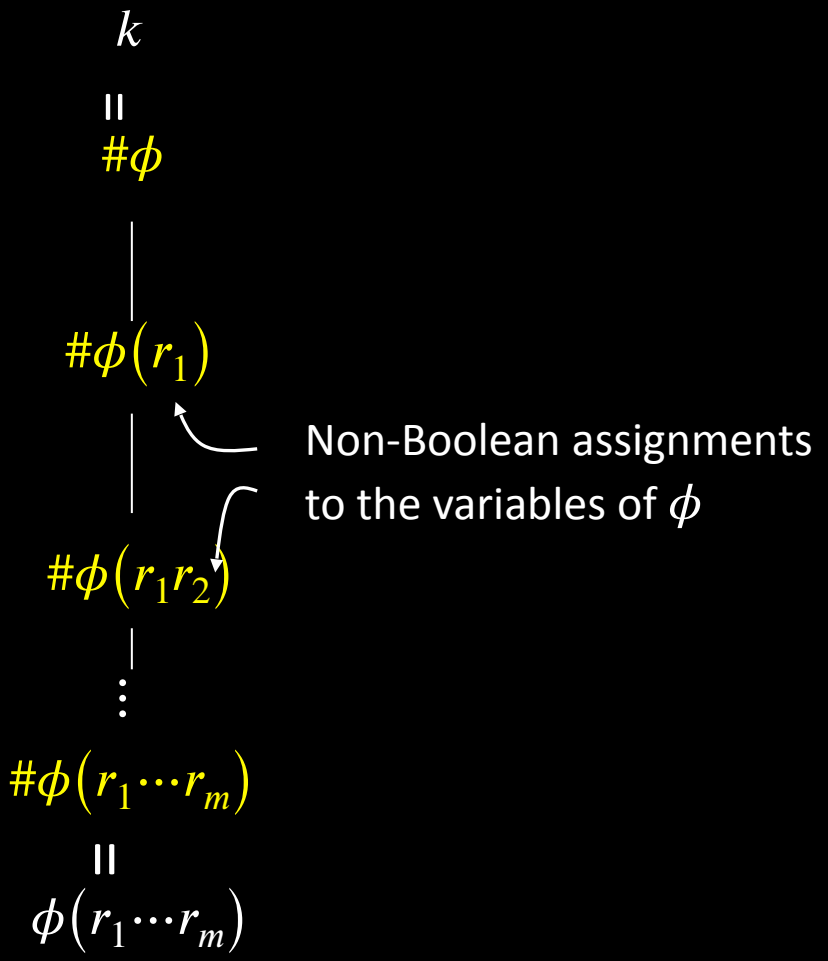
Idea for fixing $\#SAT \in$ IP protocol



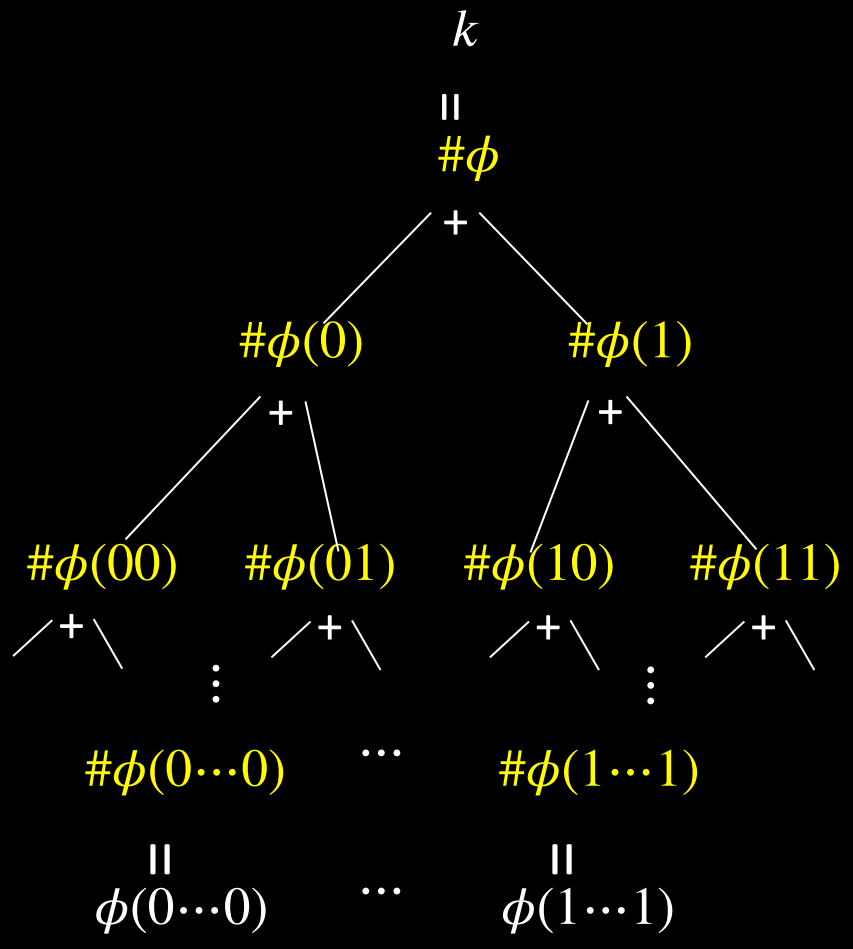


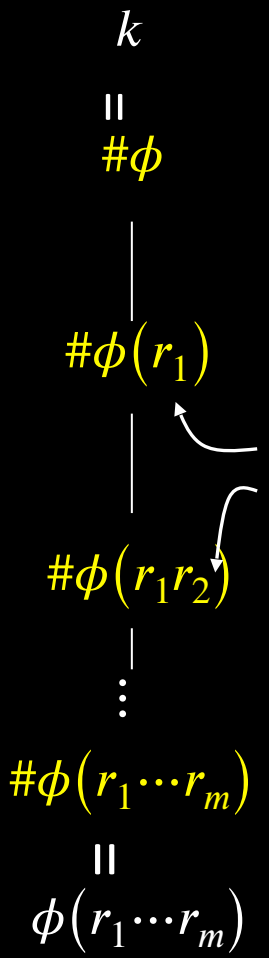
Idea for fixing $\#SAT \in$ IP protocol





Idea for fixing $\#SAT \in$ IP protocol

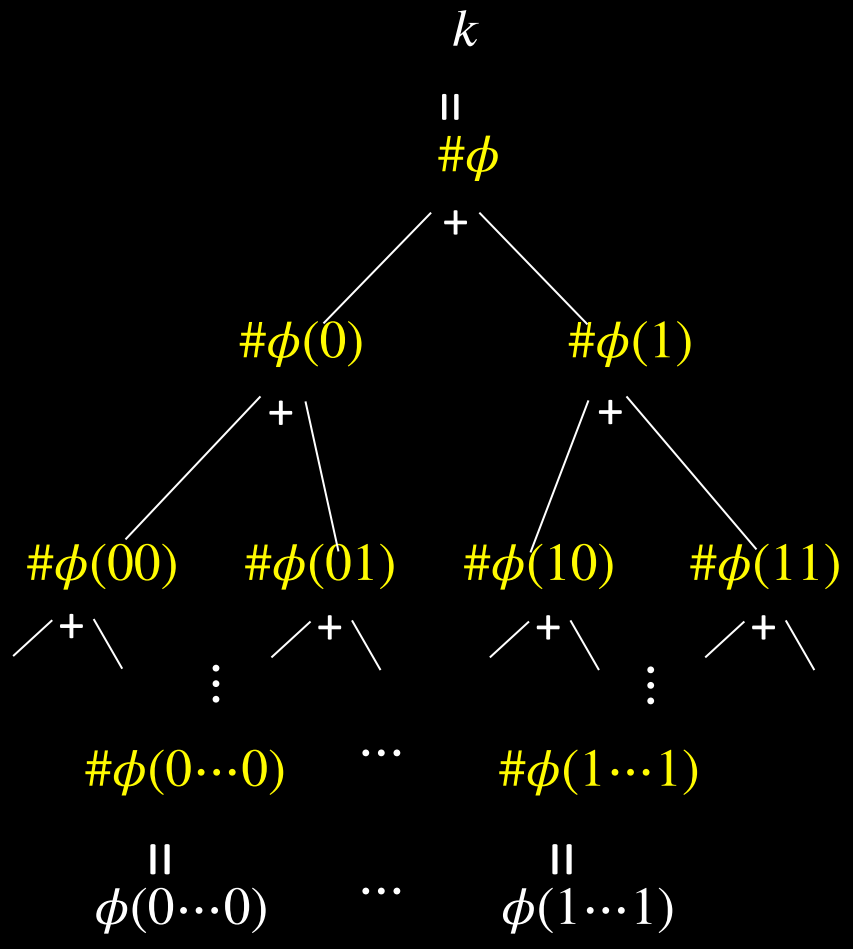


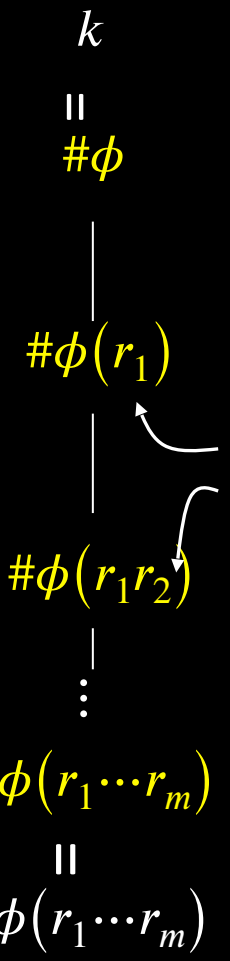


Non-Boolean assignments to the variables of ϕ

To be continued…

Idea for fixing $\#SAT \in$ IP protocol





Non-Boolean assignments to the variables of ϕ

To be continued…

Quick review of today

1. Introduced the interactive proof system model
2. Defined the class IP
3. Showed $\overline{ISO} \in$ IP
4. Started showing $\#SAT \in$ IP to prove that coNP $\subseteq$ IP

18.404/6.840 Lecture 26

Last time:

- Interactive Proof Systems
- The class IP
- Graph isomorphism problem, $\overline{ISO} \in$ IP
- $\#SAT \in$ IP (part 1)

Today: (Sipser §10.4)

- Arithmetization of Boolean formulas
- Finish $\#SAT \in$ IP and conclude that coNP $\subseteq$ IP

Review: Interactive Proofs

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Review: Interactive Proofs

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Review: Interactive Proofs

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{ A \mid$ for some V and P (This P is an "honest" prover)

$$w \in A \rightarrow \Pr[\,(V \leftrightarrow P) \text{ accepts } w\,] \geq \frac{2}{3}$$

$$w \notin A \rightarrow \text{for any prover } \tilde{P} \quad \Pr[\,(V \leftrightarrow \tilde{P}) \text{ accepts } w\,] \leq \frac{1}{3} \}$$

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Review: Interactive Proofs

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{ A \mid$ for some V and P (This P is an "honest" prover)

$$w \in A \rightarrow \Pr[\,(V \leftrightarrow P) \text{ accepts } w\,] \geq \frac{2}{3}$$

$$w \notin A \rightarrow \text{for any prover } \tilde{P} \quad \Pr[\,(V \leftrightarrow \tilde{P}) \text{ accepts } w\,] \leq \frac{1}{3} \}$$

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Think of $\tilde{P}$ as a "crooked" prover trying to make V accept when it shouldn't.

Review: Interactive Proofs

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{A \mid$ for some V and P (This P is an "honest" prover)

$$w \in A \rightarrow \Pr[(V \leftrightarrow P) \text{ accepts } w] \geq \frac{2}{3}$$

$$w \notin A \rightarrow \text{for any prover } \tilde{P} \quad \Pr[(V \leftrightarrow \tilde{P}) \text{ accepts } w] \leq \frac{1}{3}\}$$

Equivalently: IP $= \{A \mid$ for some V

$$w \in A \rightarrow \exists P \quad \Pr[(V \leftrightarrow P) \text{ accepts } w] \geq \frac{2}{3}$$

$$w \notin A \rightarrow \nexists P \quad \Pr[(V \leftrightarrow P) \text{ accepts } w] \geq \frac{1}{3}\}$$

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Think of $\tilde{P}$ as a "crooked" prover trying to make V accept when it shouldn't.

Review: Interactive Proofs

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{A \mid$ for some V and P (This P is an "honest" prover)

$$w \in A \rightarrow \Pr[(V \leftrightarrow P) \text{ accepts } w] \geq \frac{2}{3}$$

$$w \notin A \rightarrow \text{for any prover } \tilde{P} \quad \Pr[(V \leftrightarrow \tilde{P}) \text{ accepts } w] \leq \frac{1}{3}\}$$

Equivalently: IP $= \{A \mid$ for some V

$$w \in A \rightarrow \exists P \quad \Pr[(V \leftrightarrow P) \text{ accepts } w] \geq \frac{2}{3}$$

$$w \notin A \rightarrow \nexists P \quad \Pr[(V \leftrightarrow P) \text{ accepts } w] \geq \frac{1}{3}\}$$

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Think of $\tilde{P}$ as a "crooked" prover trying to make V accept when it shouldn't.

Here, we emphasize how P is similar to the certificate for NP-languages.

Review: Interactive Proofs

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{A \mid$ for some V and P (This P is an "honest" prover)

$w \in A \rightarrow$ Pr [(V $\leftrightarrow$ P) accepts w] $\geq \frac{2}{3}$

$w \notin A \rightarrow$ for any prover $\tilde{P}$ Pr [(V $\leftrightarrow \tilde{P}$) accepts w] $\leq \frac{1}{3}$ }

Equivalently: IP $= \{A \mid$ for some V

$w \in A \rightarrow \exists P$ Pr [(V $\leftrightarrow$ P) accepts w] $\geq \frac{2}{3}$

$w \notin A \rightarrow \nexists P$ Pr [(V $\leftrightarrow$ P) accepts w] $\geq \frac{1}{3}$ }

An amplification lemma can improve the error probability from $\frac{1}{3}$ to $\frac{1}{2^{\text{poly}(n)}}$

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Think of $\tilde{P}$ as a "crooked" prover trying to make V accept when it shouldn't.

Here, we emphasize how P is similar to the certificate for NP-languages.

Review: Interactive Proofs

Defn: Pr[(V $\leftrightarrow$ P) accepts w] = probability that V accepts when V interacts with P, given input w.

Defn: IP $= \{A \mid$ for some V and P (This P is an "honest" prover)

$w \in A \rightarrow$ Pr [(V $\leftrightarrow$ P) accepts w] $\geq \frac{2}{3}$

$w \notin A \rightarrow$ for any prover $\tilde{P}$ Pr [(V $\leftrightarrow \tilde{P}$) accepts w] $\leq \frac{1}{3}$ }

Equivalently: IP $= \{A \mid$ for some V

$w \in A \rightarrow \exists P$ Pr [(V $\leftrightarrow$ P) accepts w] $\geq \frac{2}{3}$

$w \notin A \rightarrow \nexists P$ Pr [(V $\leftrightarrow$ P) accepts w] $\geq \frac{1}{3}$ }

An amplification lemma can improve the error probability from $\frac{1}{3}$ to $\frac{1}{2^{\text{poly}(n)}}$

Two interacting parties
Verifier (V): Probabilistic polynomial time TM
Prover (P): Unlimited computational power

Both P and V see input w.
They exchange a polynomial number of polynomial-size messages.
Then V *accepts* or *rejects*.

Think of $\tilde{P}$ as a "crooked" prover trying to make V accept when it shouldn't.

Here, we emphasize how P is similar to the certificate for NP-languages.

coNP $\subseteq$ IP

coNP $\subseteq$ IP

Surprising Theorem: IP = PSPACE

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP: weaker but similar, show $\#SAT \in$ IP ($\#SAT$ is coNP-hard)

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP: weaker but similar, show $\#SAT \in$ IP ($\#SAT$ is coNP-hard)

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP: weaker but similar, show $\#SAT \in$ IP ($\#SAT$ is coNP-hard)

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP: weaker but similar, show $\#SAT \in$ IP ($\#SAT$ is coNP-hard)

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP: weaker but similar, show $\#SAT \in$ IP ($\#SAT$ is coNP-hard)

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP: weaker but similar, show $\#SAT \in$ IP ($\#SAT$ is coNP-hard)

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Two identities

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP: weaker but similar, show $\#SAT \in$ IP ($\#SAT$ is coNP-hard)

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variable

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

Two identities

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$

2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots a_m)$

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP: weaker but similar, show $\#SAT \in$ IP ($\#SAT$ is coNP-hard)

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assignments$\}$

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1$, … , $x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variable

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

Two identities

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$

2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots a_m)$

Check-in 26.1

coNP ⊆ IP

Surprising Theorem: IP = PSPACE

IP ⊆ PSPACE: standard simulation, similar to NP ⊆ PSPACE

PSPACE ⊆ IP: show $TQBF \in$ IP, we won't prove

coNP ⊆ IP: weaker but similar, show $\#SAT \in$ IP ($\#SAT$ is co

$\#SAT = \{\langle \phi, k \rangle \mid$ Boolean formula ϕ has exactly k satisfying assig

Theorem: $\#SAT \in$ IP

Proof: First some notation. Assume ϕ has m variables $x_1, \ldots, x_m$.

Let $\phi(0)$ be ϕ with $x_1 = 0$ (0 substituted for x_1) 0 = FALSE and 1 = TRUE.

Let $\phi(a_1 \ldots a_i)$ be ϕ with $x_1 = a_1, \ldots, x_i = a_i$ for $a_1, \ldots, a_i \in \{0,1\}$.

Call $a_1, \ldots, a_i$ presets. The remaining $x_{i+1}, \ldots, x_m$ stay as unset variable

Let $\#\phi$ = the number of satisfying assignments of ϕ.

Let $\#\phi(0)$ = the number of satisfying assignments of $\phi(0)$.

Let $\#\phi(a_1 \ldots a_i)$ = the number of satisfying assignments of $\phi(a_1 \ldots a_i)$

Check-in 26.1

Let $\phi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

Check all that are true:

a) $\#\phi = 1$ b) $\#\phi = 2$

c) $\#\phi(0) = 1$ d) $\#\phi(0) = 2$

e) $\#\phi(00) = 0$ f) $\#\phi(00) = 1$

Two identities

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$

2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots a_m)$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$$\#\phi(1) = \#\phi(10) + \#\phi(11)$$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

k

$=$

$\#\phi$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

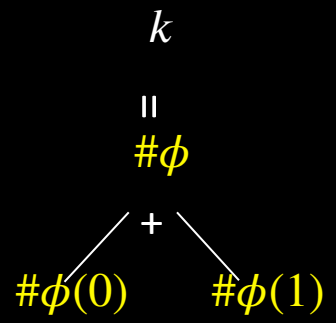
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

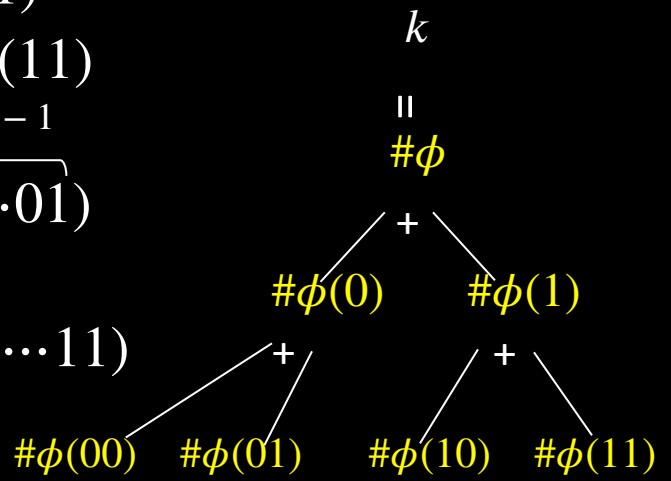
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

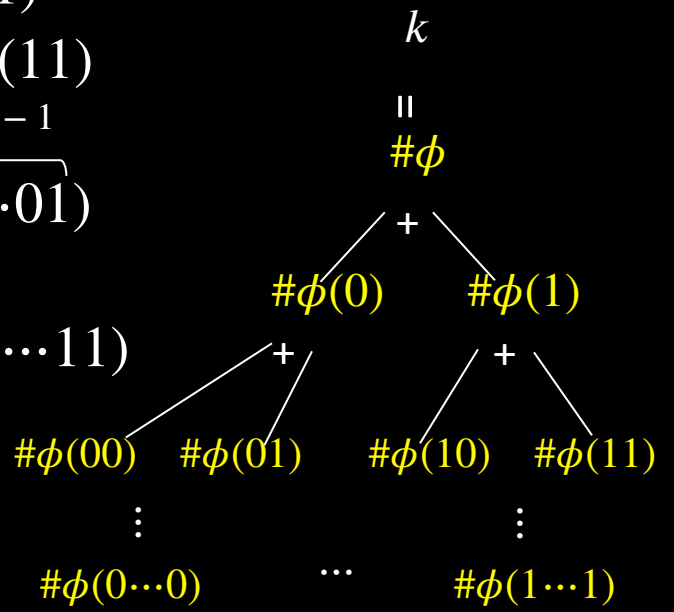
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

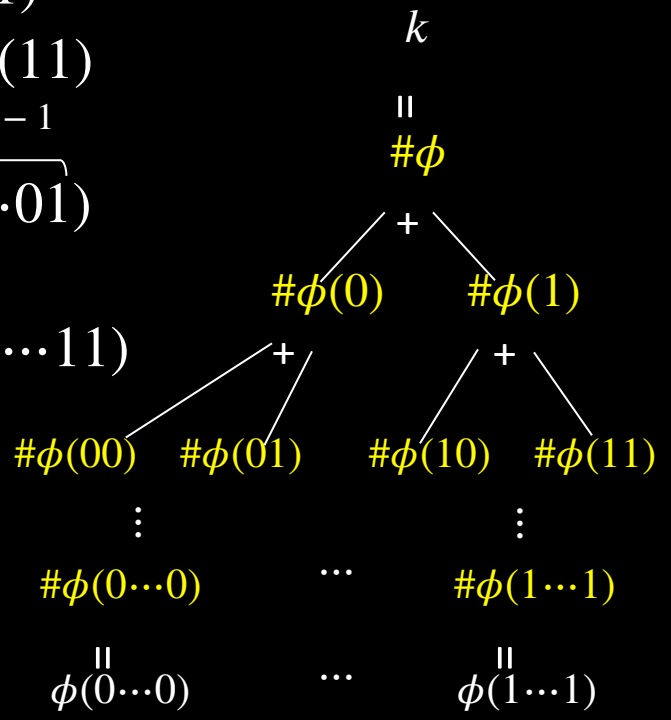
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

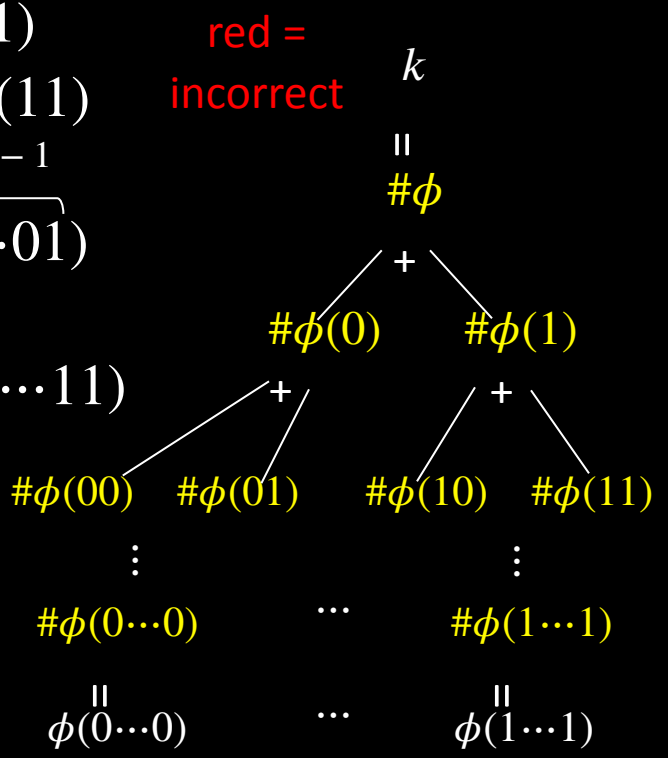
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

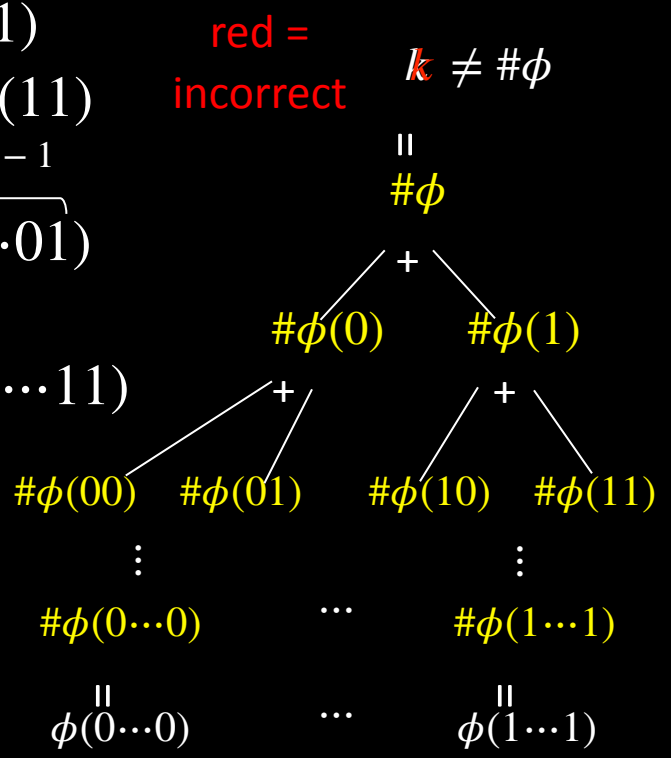
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

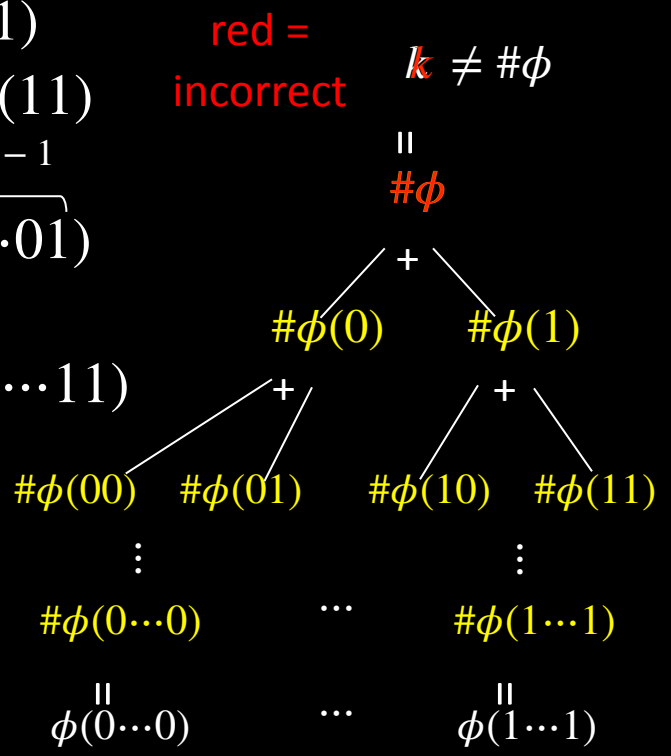
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

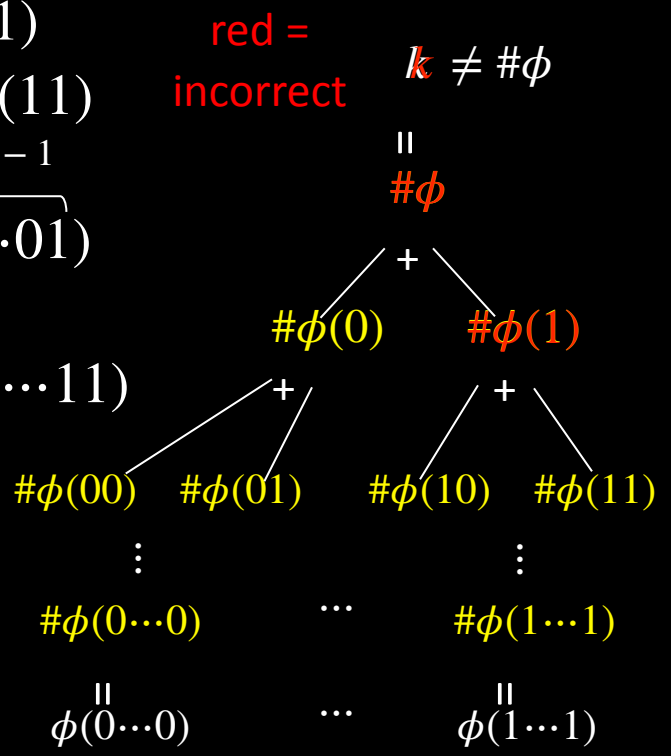
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

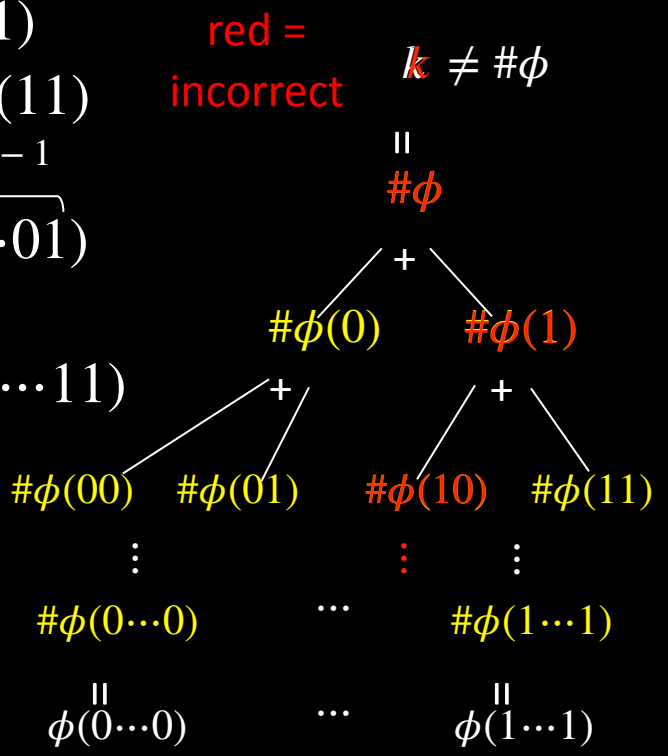
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

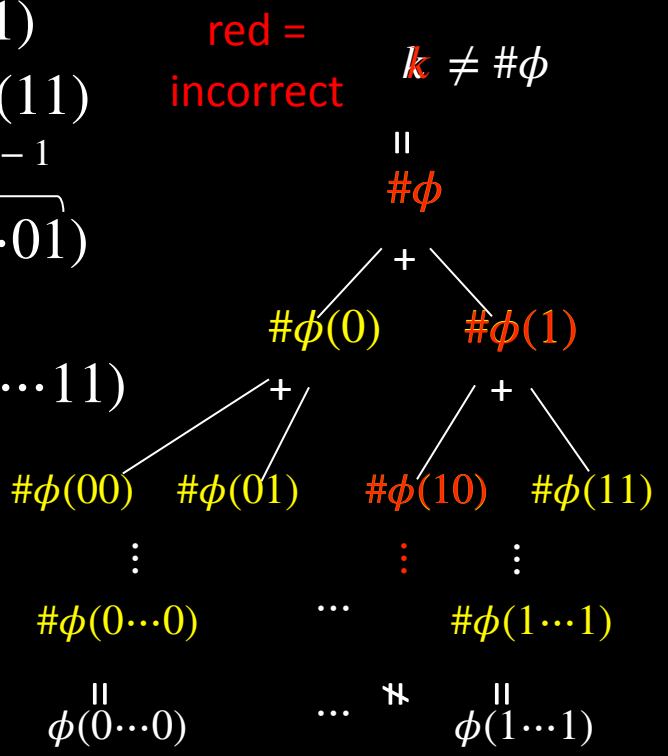
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

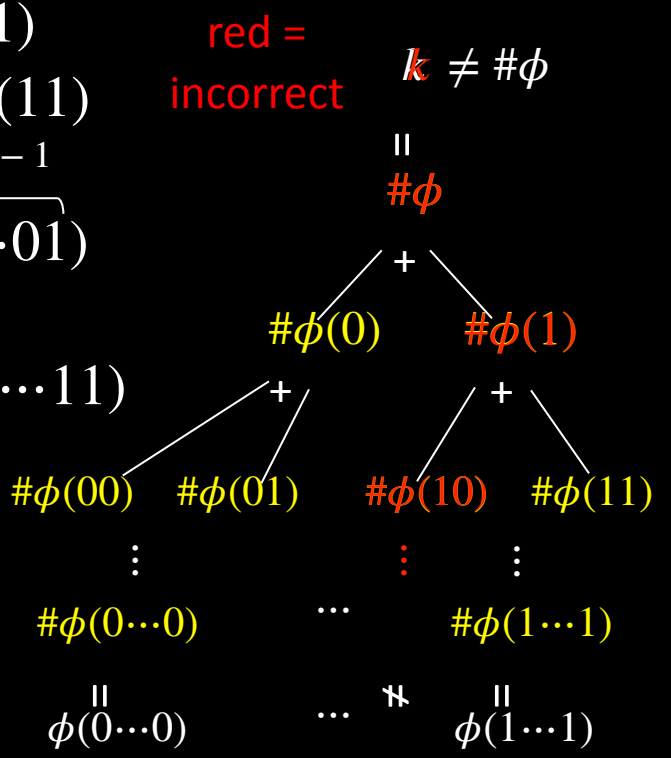
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.



Problem: Exponential. Will fix.

$\#SAT \in$ IP – 1st attempt

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

2) P sends $\#\phi(00), \#\phi(01), \#\phi(10), \#\phi(11)$; V checks $\#\phi(0) = \#\phi(00) + \#\phi(01)$

$\#\phi(1) = \#\phi(10) + \#\phi(11)$

$\vdots$

m) P sends $\#\phi(\overbrace{0\cdots0}^{m}), \ldots, \#\phi(1\cdots1)$; V checks $\#\phi(0\cdots0) = \#\phi(\overbrace{0\cdots00}^{m-1}) + \#\phi(\overbrace{0\cdots01}^{m-1})$

$\vdots$

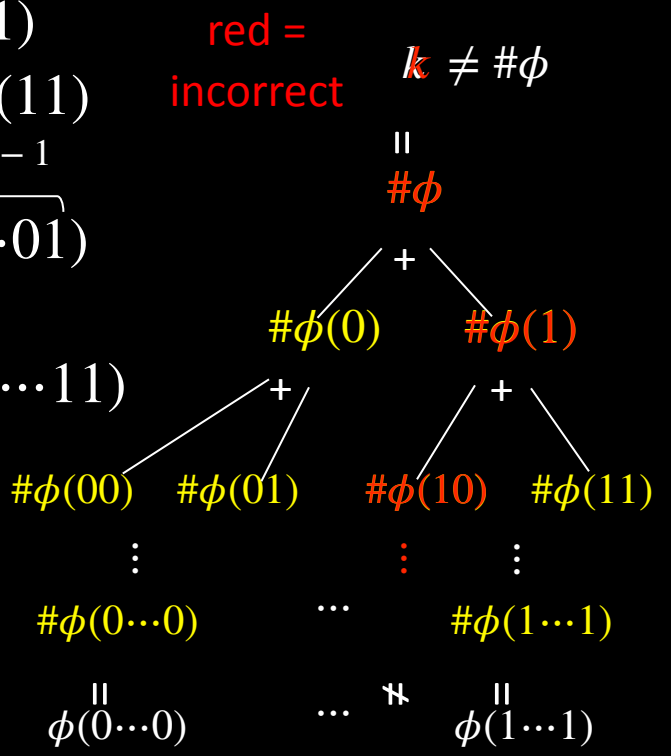
V checks $\#\phi(1\cdots1) = \#\phi(1\cdots10) + \#\phi(1\cdots11)$

$m+1$) V checks $\#\phi(\overbrace{0\cdots0}^{m}) = \phi(0\cdots0)$

$\vdots$

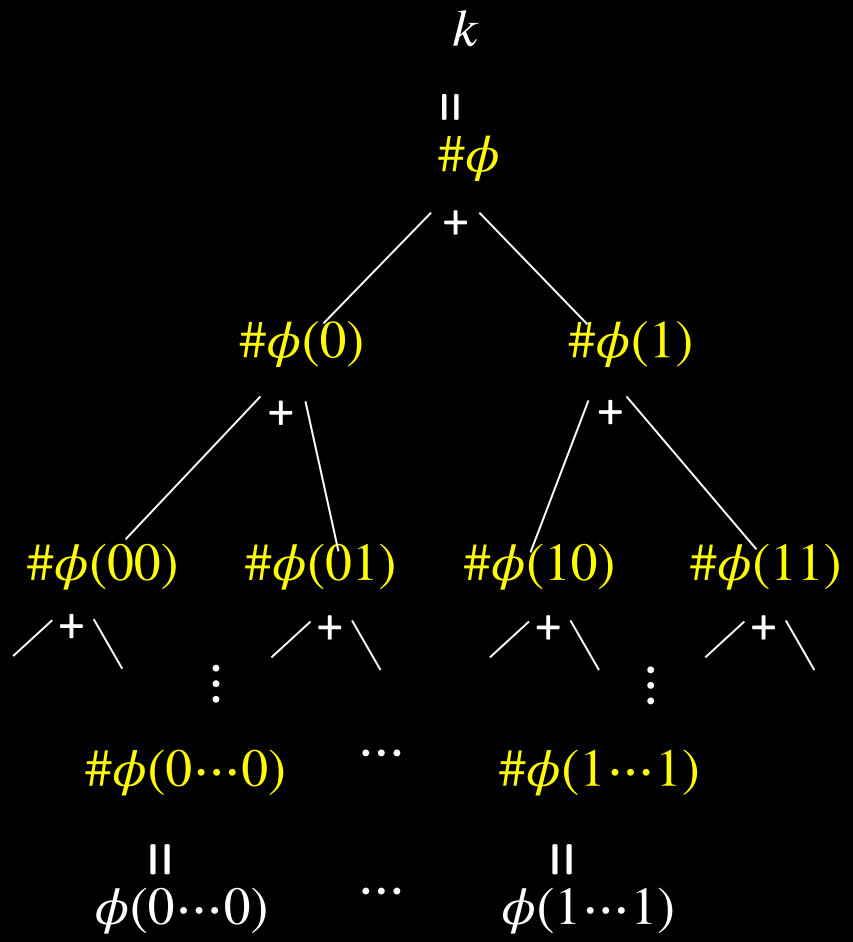
$\#\phi(1\cdots1) = \phi(1\cdots1)$

V *accepts* if all checks are correct. Otherwise V *rejects*.

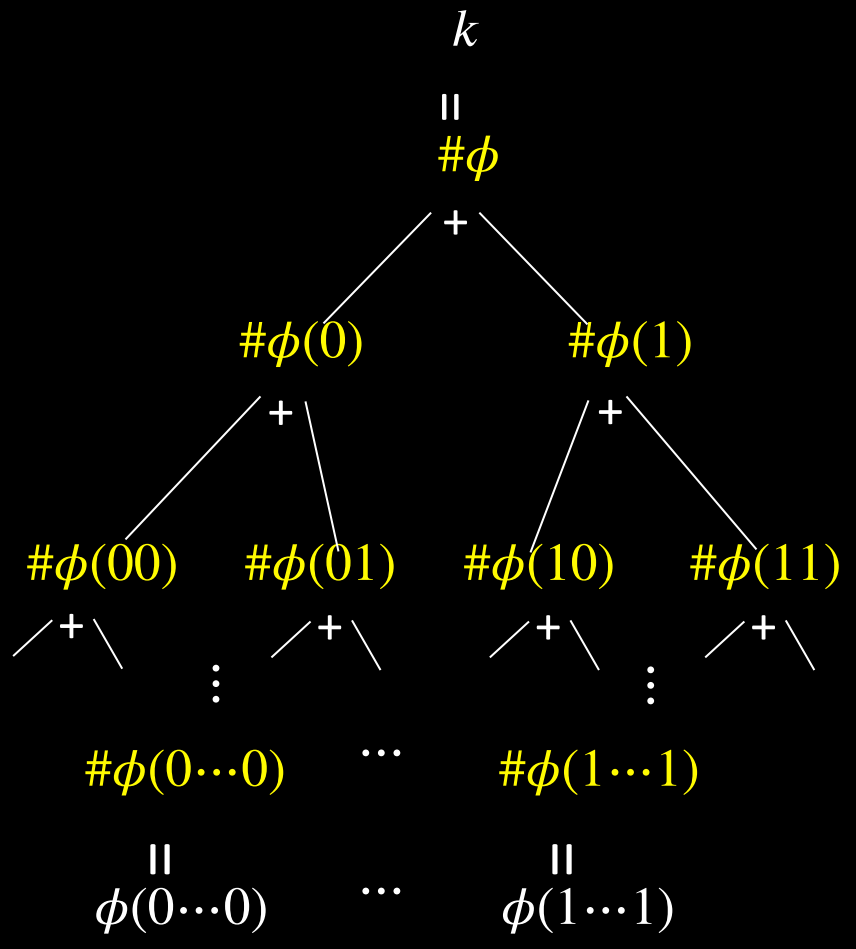


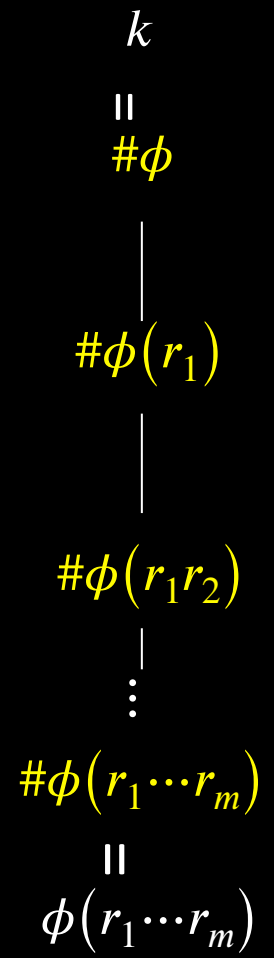
Problem: Exponential. Will fix.

Idea for fixing $\#SAT \in$ IP protocol

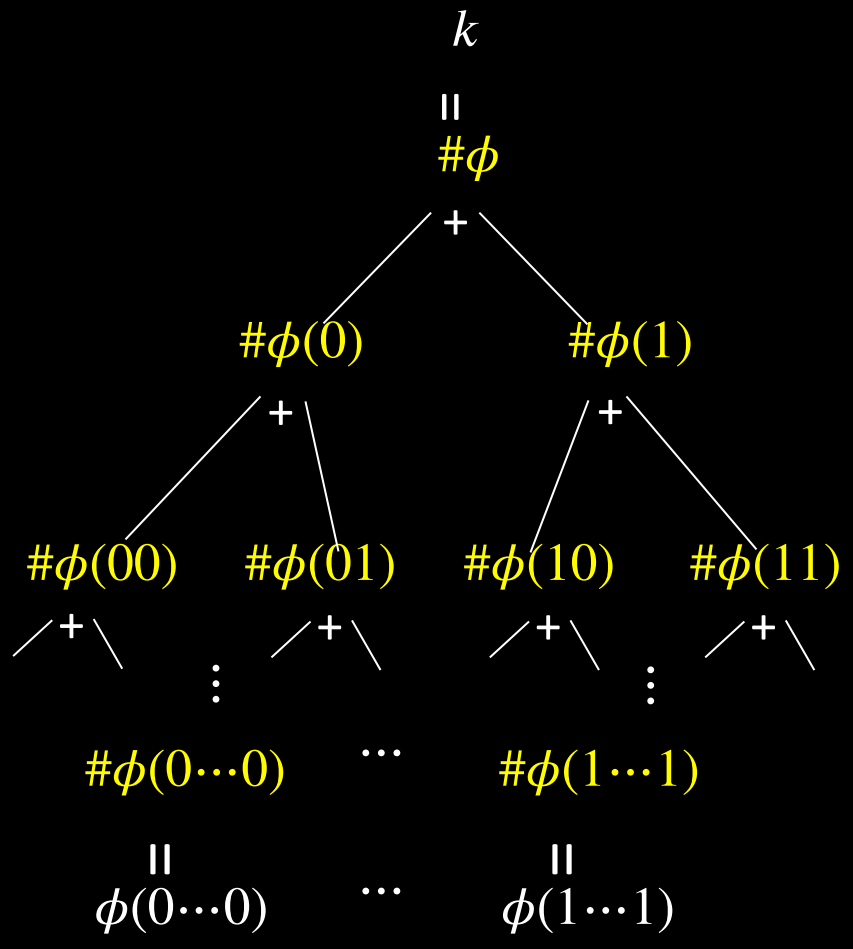


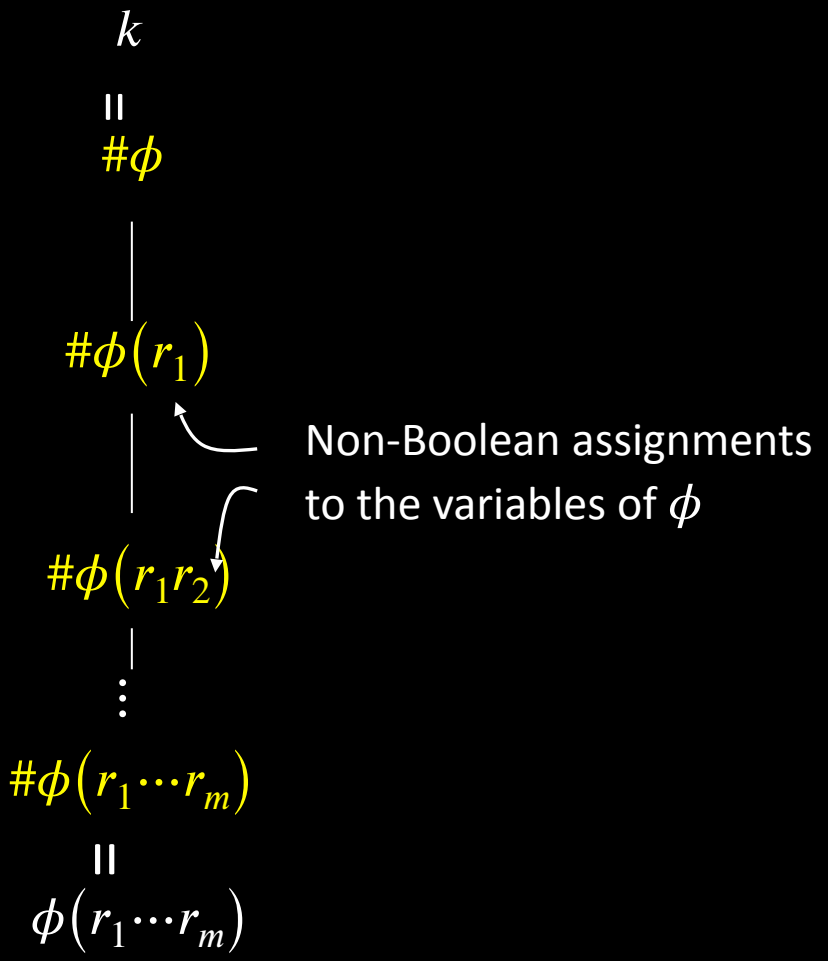
Idea for fixing $\#SAT \in$ IP protocol



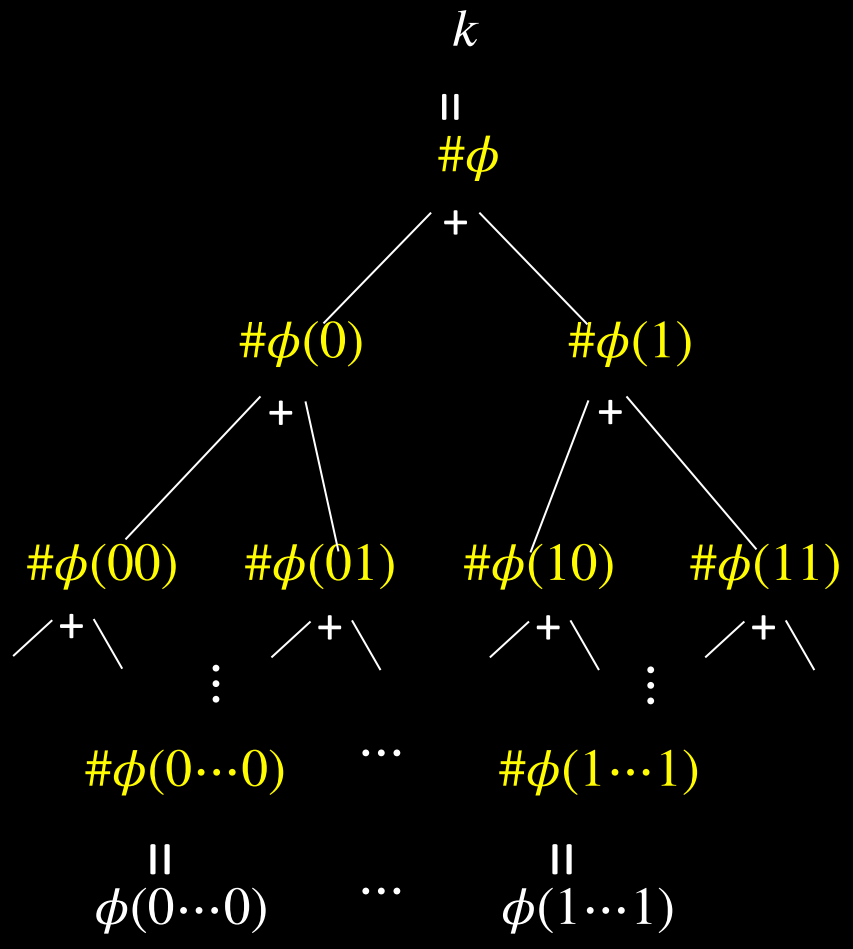


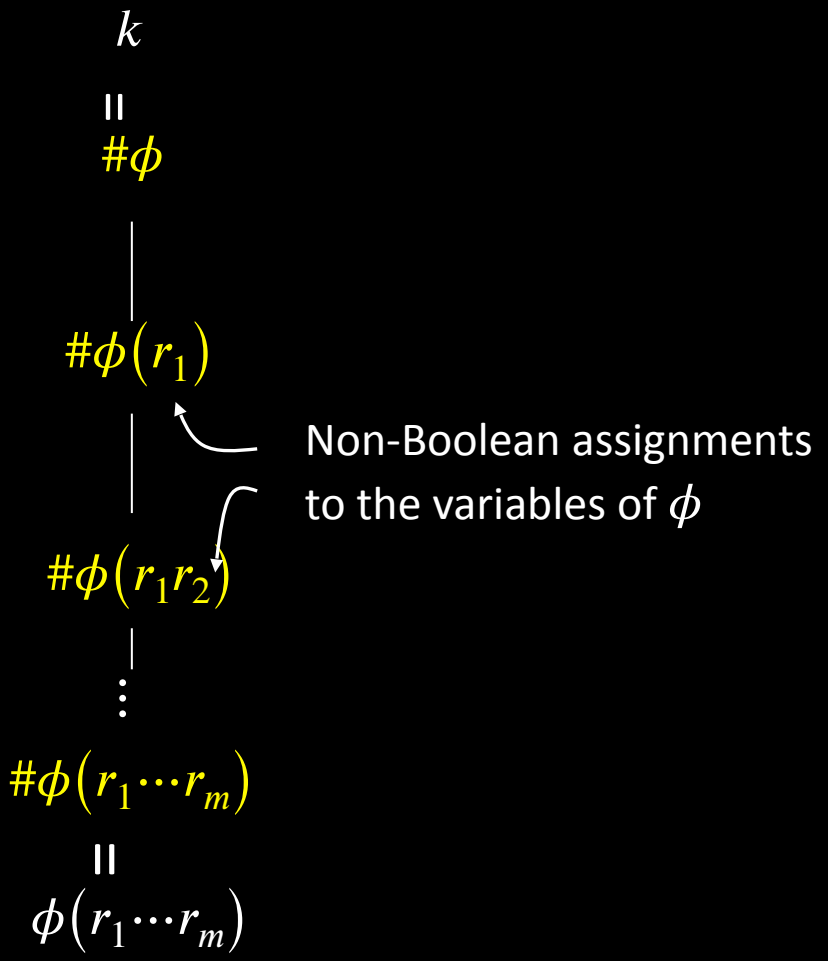
Idea for fixing $\#SAT \in$ IP protocol





Idea for fixing $\#SAT \in$ IP protocol





Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$
\begin{aligned}
a \wedge b &\rightarrow a \times b = ab \\
\overline{a} &\rightarrow (1 - a) \\
a \vee b &\rightarrow a + b - ab
\end{aligned}
$$

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$
\begin{aligned}
a \wedge b &\rightarrow a \times b = ab \\
\overline{a} &\rightarrow (1 - a) \\
a \vee b &\rightarrow a + b - ab \\
\phi &\rightarrow p_\phi
\end{aligned}
$$

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$
\begin{aligned}
a \wedge b &\rightarrow a \times b = ab \\
\overline{a} &\rightarrow (1 - a) \\
a \vee b &\rightarrow a + b - ab \\
\phi &\rightarrow p_\phi \text{ degree}(p_\phi) \leq \left|\phi\right|
\end{aligned}
$$

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$
\begin{aligned}
a \wedge b &\rightarrow a \times b = ab \\
\overline{a} &\rightarrow (1 - a) \\
a \vee b &\rightarrow a + b - ab \\
\phi &\rightarrow p_\phi \text{ degree}(p_\phi) \leq |\phi|
\end{aligned}
$$

Let $\mathbb{F}_q = \{0,1, \ldots, q-1\}$ for prime $q > 2^m$ be a finite field ($+, \times \bmod q$) and let $a_1, \ldots, a_i \in \mathbb{F}_q$

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$
\begin{aligned}
a \wedge b &\rightarrow a \times b = ab \\
\overline{a} &\rightarrow (1 - a) \\
a \vee b &\rightarrow a + b - ab \\
\phi &\rightarrow p_\phi \text{ degree}(p_\phi) \leq |\phi|
\end{aligned}
$$

Let $\mathbb{F}_q = \{0,1, \ldots, q-1\}$ for prime $q > 2^m$ be a finite field ($+, \times$ mod q) and let $a_1, \ldots, a_i \in \mathbb{F}_q$

Let $\phi(a_1 \ldots a_i) = p_\phi$ where $x_1 \cdots x_i = a_1 \cdots a_i$ and remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$
\begin{aligned}
a \wedge b &\rightarrow a \times b = ab \\
\overline{a} &\rightarrow (1 - a) \\
a \vee b &\rightarrow a + b - ab \\
\phi &\rightarrow p_\phi \text{ degree}(p_\phi) \leq |\phi|
\end{aligned}
$$

Let $\mathbb{F}_q = \{0,1, \ldots, q-1\}$ for prime $q > 2^m$ be a finite field ($+, \times$ mod q) and let $a_1, \ldots, a_i \in \mathbb{F}_q$

Let $\phi(a_1 \ldots a_i) = p_\phi$ where $x_1 \cdots x_i = a_1 \cdots a_i$ and remaining $x_{i+1}, \ldots, x_m$ stay as unset variables.

Let $$\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$$

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$\begin{aligned} a \wedge b &\rightarrow a \times b = ab \\ \overline{a} &\rightarrow (1-a) \\ a \vee b &\rightarrow a + b - ab \\ \phi &\rightarrow p_\phi \text{ degree}(p_\phi) \leq |\phi| \end{aligned}$$

Let $\mathbb{F}_q = \{0,1,\ldots,\ q-1\}$ for prime $q > 2^m$ be a finite field ($+, \times$ mod q) and let $a_1, \ldots,\ a_i \in \mathbb{F}_q$

Let $\phi(a_1 \ldots a_i) = p_\phi$ where $x_1 \cdots x_i = a_1 \cdots a_i$ and remaining $x_{i+1},\ \ldots,\ x_m$ stay as unset variables.

Let $$\#\phi(a_1 \ldots\ a_i) = \sum_{a_{i+1},\ldots,\ a_m \in \{0,1\}} \phi(a_1 \ldots\ a_m)$$

Important: For Boolean $a_1 \ldots\ a_i$ the values of $\phi(a_1 \ldots a_i)$ and $\#\phi(a_1 \ldots\ a_i)$ are unchanged from the previous definition.

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$\begin{aligned} a \wedge b &\rightarrow a \times b = ab \\ \overline{a} &\rightarrow (1-a) \\ a \vee b &\rightarrow a + b - ab \\ \phi &\rightarrow p_\phi \text{ degree}(p_\phi) \leq |\phi| \end{aligned}$$

Let $\mathbb{F}_q = \{0,1,\ldots,\ q-1\}$ for prime $q > 2^m$ be a finite field ($+, \times$ mod q) and let $a_1, \ldots,\ a_i \in \mathbb{F}_q$

Let $\phi(a_1 \ldots a_i) = p_\phi$ where $x_1 \cdots x_i = a_1 \cdots a_i$ and remaining $x_{i+1},\ \ldots,\ x_m$ stay as unset variables.

Let $$\#\phi(a_1 \ldots\ a_i) = \sum_{a_{i+1},\ldots,\ a_m \in \{0,1\}} \phi(a_1 \ldots\ a_m)$$

Important: For Boolean $a_1 \ldots\ a_i$

the values of $\phi(a_1 \ldots a_i)$ and $\#\phi(a_1 \ldots\ a_i)$ are unchanged from the previous definition.

We have <u>extended</u> these functions to non-Boolean values

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$\begin{aligned} a \wedge b &\rightarrow a \times b = ab \\ \overline{a} &\rightarrow (1-a) \\ a \vee b &\rightarrow a + b - ab \\ \phi &\rightarrow p_\phi \text{ degree}(p_\phi) \leq |\phi| \end{aligned}$$

Let $\mathbb{F}_q = \{0,1,\ldots,\ q-1\}$ for prime $q > 2^m$ be a finite field ($+, \times$ mod q) and let $a_1, \ldots,\ a_i \in \mathbb{F}_q$

Let $\phi(a_1 \ldots a_i) = p_\phi$ where $x_1 \cdots x_i = a_1 \cdots a_i$ and remaining $x_{i+1},\ \ldots,\ x_m$ stay as unset variables.

Let $$\#\phi(a_1 \ldots\ a_i) = \sum_{a_{i+1},\ldots,\ a_m \in \{0,1\}} \phi(a_1 \ldots\ a_m)$$

identities still true

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$
2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots\ a_m)$

Important: For Boolean $a_1 \ldots\ a_i$

the values of $\phi(a_1 \ldots a_i)$ and $\#\phi(a_1 \ldots\ a_i)$ are unchanged from the previous definition.

We have extended these functions to non-Boolean values

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$\begin{aligned} a \wedge b &\rightarrow a \times b = ab \\ \overline{a} &\rightarrow (1-a) \\ a \vee b &\rightarrow a + b - ab \\ \phi &\rightarrow p_\phi \text{ degree}(p_\phi) \leq |\phi| \end{aligned}$$

Let $\mathbb{F}_q = \{0,1,\ldots,\ q-1\}$ for prime $q > 2^m$ be a finite field ($+, \times$ mod q) and let $a_1, \ldots,\ a_i \in \mathbb{F}_q$

Let $\phi(a_1 \ldots a_i) = p_\phi$ where $x_1 \cdots x_i = a_1 \cdots a_i$ and remaining $x_{i+1},\ \ldots,\ x_m$ stay as unset variables.

Let $$\#\phi(a_1 \ldots\ a_i) = \sum_{a_{i+1},\ldots,\ a_m \in \{0,1\}} \phi(a_1 \ldots\ a_m)$$

identities still true

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$
2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots\ a_m)$

Important: For Boolean $a_1 \ldots\ a_i$

the values of $\phi(a_1 \ldots a_i)$ and $\#\phi(a_1 \ldots\ a_i)$ are unchanged from the previous definition.

We have extended these functions to non-Boolean values

Check-in 26.2

Arithmetizing Boolean formulas

Simulate $\wedge$ and $\vee$ with $+$ and $\times$

$$\begin{aligned} a \wedge b &\rightarrow a \times b = ab \\ \overline{a} &\rightarrow (1-a) \\ a \vee b &\rightarrow a + b - ab \\ \phi &\rightarrow p_\phi \text{ degree}(p_\phi) \leq |\phi| \end{aligned}$$

Let $\mathbb{F}_q = \{0,1,\ldots,\ q-1\}$ for prime $q > 2^m$ be a finite field ($+, \times$ mod q) and let $a_1, \ldots,\ a_i \in \mathbb{F}_q$

Let $\phi(a_1 \ldots a_i) = p_\phi$ where $x_1 \cdots x_i = a_1 \cdots a_i$ and remaining $x_{i+1},\ \ldots,\ x_m$ stay as unset variables.

Let $$\#\phi(a_1 \ldots\ a_i) = \sum_{a_{i+1},\ldots,\ a_m \in \{0,1\}} \phi(a_1 \ldots\ a_m)$$

identities still true

1. $\#\phi(a_1 \ldots a_i) = \#\phi(a_1 \ldots a_i 0) + \#\phi(a_1 \ldots a_i 1)$
2. $\#\phi(a_1 \ldots a_m) = \phi(a_1 \ldots\ a_m)$

Check-in 26.2

Let $\phi = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$. Check all that are true:

a) $p_\phi = (x_1 + x_2 - x_1 x_2)\Big((1-x_1) + (1-x_2) - (1-x_1)(1-x_2) \Big)$

b) $p_\phi = (x_1 + x_2)\Big((1-x_1) + (1-x_2) \Big)$

c) $p_\phi = (x_1 + x_2 - 2x_1 x_2)$

$\#SAT \in$ IP – version 1

$\#SAT \in$ IP – version 1

Theorem: $\#SAT \in$ IP

$\#SAT \in$ IP – version 1

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

$\#SAT \in$ IP – version 1

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

$\#SAT \in$ IP – version 1

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(0), \#\phi(1)$; V checks $\#\phi = \#\phi(0) + \#\phi(1)$

$\#SAT \in$ IP – version 1

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

$\#SAT \in$ IP – version 1

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

$\#SAT \in$ IP – version 1

Theorem: $\#SAT \in$ IP

Recall $\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

[P needs to show $\#\phi(z)$ is correct]

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \ldots a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} \phi(a_1 \ldots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \dots a_i) = \sum_{a_{i+1}, \dots, a_m \in \{0,1\}} \phi(a_1 \dots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

V checks $\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$ [by evaluating polynomial for $\#\phi(r_1 z)$]

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \dots a_i) = \sum_{a_{i+1}, \dots, a_m \in \{0,1\}} \phi(a_1 \dots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

V checks $\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$ [by evaluating polynomial for $\#\phi(r_1 z)$]

V sends random $r_2 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1 r_2)$ is correct]

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \dots a_i) = \sum_{a_{i+1}, \dots, a_m \in \{0,1\}} \phi(a_1 \dots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

V checks $\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$ [by evaluating polynomial for $\#\phi(r_1 z)$]

V sends random $r_2 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1 r_2)$ is correct]

⋮

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \dots a_i) = \sum_{a_{i+1}, \dots, a_m \in \{0,1\}} \phi(a_1 \dots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

V checks $\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$ [by evaluating polynomial for $\#\phi(r_1 z)$]

V sends random $r_2 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1 r_2)$ is correct]

⋮

m) P sends $\#\phi(r_1 \cdots r_{m-1} z)$ as a polynomial in z

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \dots a_i) = \sum_{a_{i+1}, \dots, a_m \in \{0,1\}} \phi(a_1 \dots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

V checks $\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$ [by evaluating polynomial for $\#\phi(r_1 z)$]

V sends random $r_2 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1 r_2)$ is correct]

$\vdots$

m) P sends $\#\phi(r_1 \cdots r_{m-1} z)$ as a polynomial in z

V checks $\#\phi(r_1 \cdots r_{m-1}) = \#\phi(r_1 \cdots r_{m-1} 0) + \#\phi(r_1 \cdots r_{m-1} 1)$

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \dots a_i) = \sum_{a_{i+1}, \dots, a_m \in \{0,1\}} \phi(a_1 \dots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

V checks $\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$ [by evaluating polynomial for $\#\phi(r_1 z)$]

V sends random $r_2 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1 r_2)$ is correct]

⋮

m) P sends $\#\phi(r_1 \cdots r_{m-1} z)$ as a polynomial in z

V checks $\#\phi(r_1 \cdots r_{m-1}) = \#\phi(r_1 \cdots r_{m-1} 0) + \#\phi(r_1 \cdots r_{m-1} 1)$

V sends random $r_m \in \mathbb{F}_q$

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \dots a_i) = \sum_{a_{i+1}, \dots, a_m \in \{0,1\}} \phi(a_1 \dots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

V checks $\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$ [by evaluating polynomial for $\#\phi(r_1 z)$]

V sends random $r_2 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1 r_2)$ is correct]

$\vdots$

m) P sends $\#\phi(r_1 \cdots r_{m-1} z)$ as a polynomial in z

V checks $\#\phi(r_1 \cdots r_{m-1}) = \#\phi(r_1 \cdots r_{m-1} 0) + \#\phi(r_1 \cdots r_{m-1} 1)$

V sends random $r_m \in \mathbb{F}_q$

$m+1$) V checks $\#\phi(r_1 \cdots r_m) = \phi(r_1 \cdots r_m)$

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \dots a_i) = \sum_{a_{i+1}, \dots, a_m \in \{0,1\}} \phi(a_1 \dots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

V checks $\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$ [by evaluating polynomial for $\#\phi(r_1 z)$]

V sends random $r_2 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1 r_2)$ is correct]

$\vdots$

m) P sends $\#\phi(r_1 \cdots r_{m-1} z)$ as a polynomial in z

V checks $\#\phi(r_1 \cdots r_{m-1}) = \#\phi(r_1 \cdots r_{m-1} 0) + \#\phi(r_1 \cdots r_{m-1} 1)$

V sends random $r_m \in \mathbb{F}_q$

$m+1$) V checks $\#\phi(r_1 \cdots r_m) = \phi(r_1 \cdots r_m)$

V *accepts* if all checks are correct. Otherwise V *rejects*.

$\#SAT \in$ IP – version 1

Recall $\#\phi(a_1 \dots a_i) = \sum_{a_{i+1}, \dots, a_m \in \{0,1\}} \phi(a_1 \dots a_m)$

Theorem: $\#SAT \in$ IP

Proof: Protocol for V and (the honest) P on input $\langle \phi, k \rangle$

0) P sends $\#\phi$; V checks $k = \#\phi$

1) P sends $\#\phi(z)$ as a polynomial in z [sends coefficients – recall deg $p_\phi \leq |\phi|$]

V checks $\#\phi = \#\phi(0) + \#\phi(1)$ [by evaluating polynomial for $\#\phi(z)$]

V sends random $r_1 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1)$ is correct]

2) P sends $\#\phi(r_1 z)$ as a polynomial in z

V checks $\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$ [by evaluating polynomial for $\#\phi(r_1 z)$]

V sends random $r_2 \in \mathbb{F}_q$ [P needs to show $\#\phi(r_1 r_2)$ is correct]

$\vdots$

m) P sends $\#\phi(r_1 \cdots r_{m-1} z)$ as a polynomial in z

V checks $\#\phi(r_1 \cdots r_{m-1}) = \#\phi(r_1 \cdots r_{m-1} 0) + \#\phi(r_1 \cdots r_{m-1} 1)$

V sends random $r_m \in \mathbb{F}_q$

$m+1$) V checks $\#\phi(r_1 \cdots r_m) = \phi(r_1 \cdots r_m)$

V *accepts* if all checks are correct. Otherwise V *rejects*.

$\#SAT \in$ IP – version 2

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends **Verifier sends**

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends **Verifier sends**

$\#\phi$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

Verifier sends

Verifier checks

$\#\phi$

$\#\phi = k$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

Verifier sends

Verifier checks

$\#\phi$

$\#\phi(z)$

$= 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi = k$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z)$

$= 3z^d - 5z^{d-1} + \cdots + 7$

Verifier sends

Verifier checks

$\#\phi = k$

$\#\phi(0)$ $\#\phi(1)$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z)$

$= 3z^d - 5z^{d-1} + \cdots + 7$

Verifier sends

Verifier checks

$\#\phi = k$

$+$

$\#\phi(0)$ $\#\phi(1)$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z)$

$= 3z^d - 5z^{d-1} + \cdots + 7$

Verifier sends

r_1

Verifier checks

$\#\phi = k$

$+$

$\#\phi(0)$ $\#\phi(1)$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z)$

$= 3z^d - 5z^{d-1} + \cdots + 7$

Verifier sends

r_1

Verifier checks

$\#\phi = k$

$+$

$\#\phi(0)$ $\#\phi(1)$

$\#\phi(r_1)$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

Verifier sends

r_1

Verifier checks

$\#\phi = k$

$+$

$\#\phi(0)$ $\#\phi(1)$

$\#\phi(r_1)$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

Verifier sends

r_1

Verifier checks

$\#\phi = k$

$+$

$\#\phi(0)$ $\#\phi(1)$

$\#\phi(r_1)$

$\#\phi(r_1 0)$ $\#\phi(r_1 1)$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

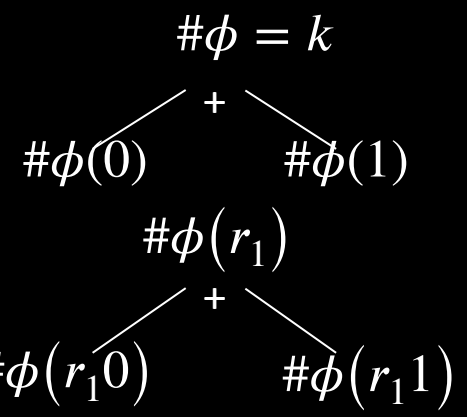
$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

Verifier sends

r_1

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

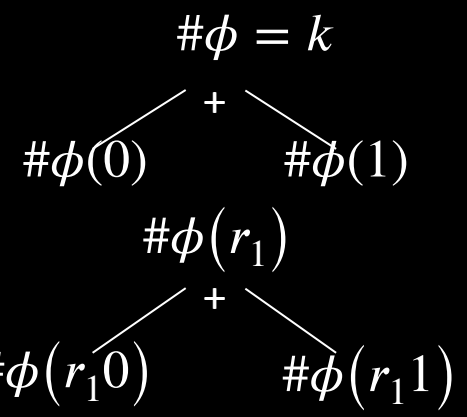
$\#\phi(r_1 z) = \cdots$

Verifier sends

r_1

r_2

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

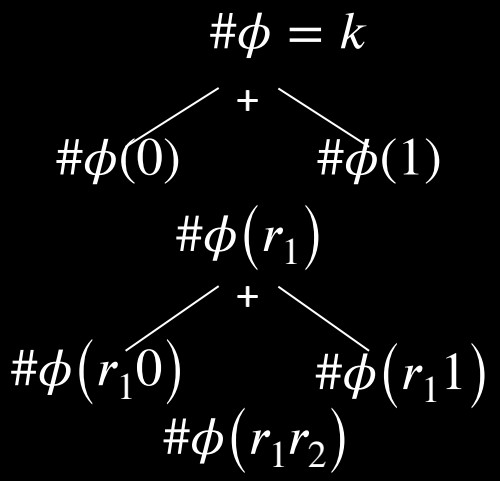
$\#\phi(r_1 z) = \cdots$

Verifier sends

r_1

r_2

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

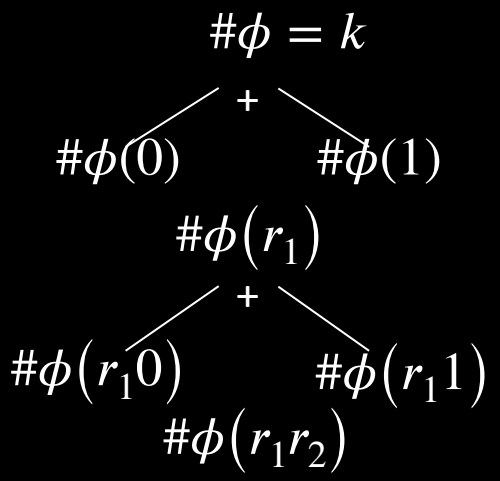
$\#\phi(r_1 r_2 z) = \cdots$

Verifier sends

r_1

r_2

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

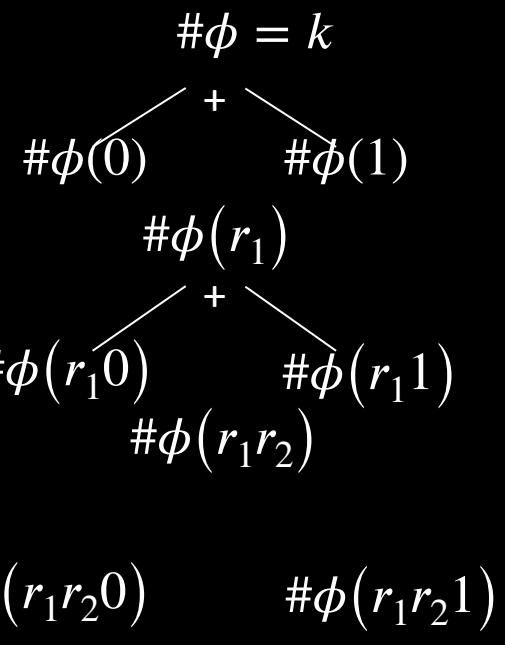
$\#\phi(r_1 r_2 z) = \cdots$

Verifier sends

r_1

r_2

Verifier checks



$\#\phi(r_1 r_2 0)$ $\#\phi(r_1 r_2 1)$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

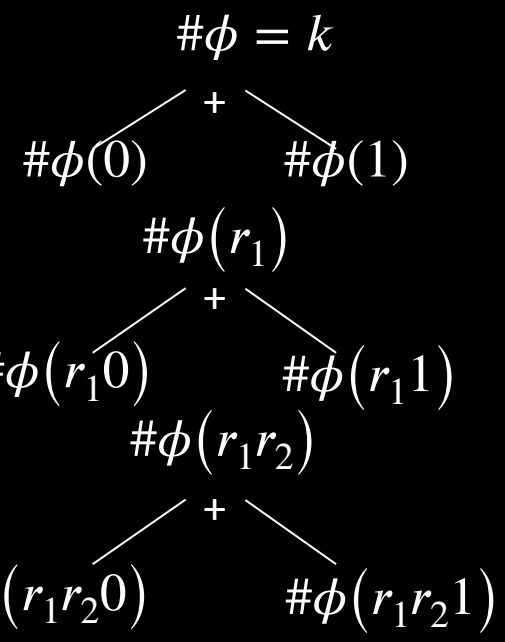
$\#\phi(r_1 r_2 z) = \cdots$

Verifier sends

r_1

r_2

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

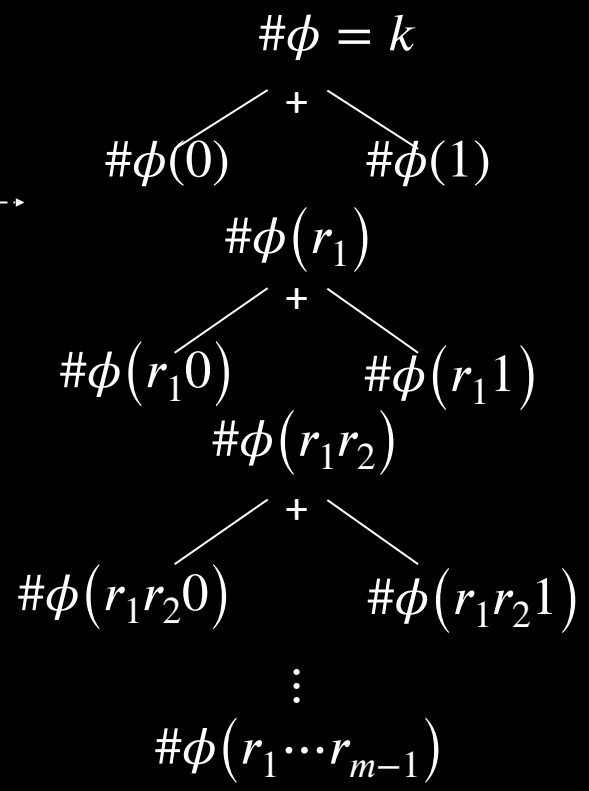
$\#\phi(r_1 r_2 z) = \cdots$

Verifier sends

r_1

r_2

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

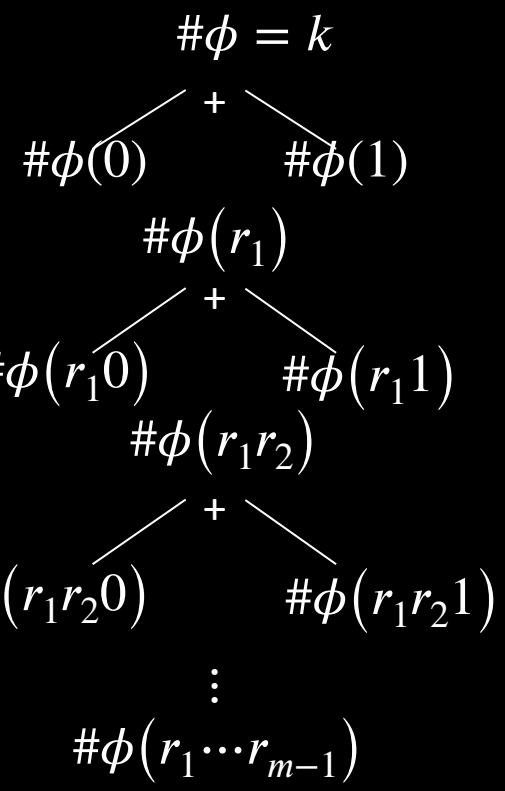
$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

r_1

r_2

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

Verifier sends

Verifier checks

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

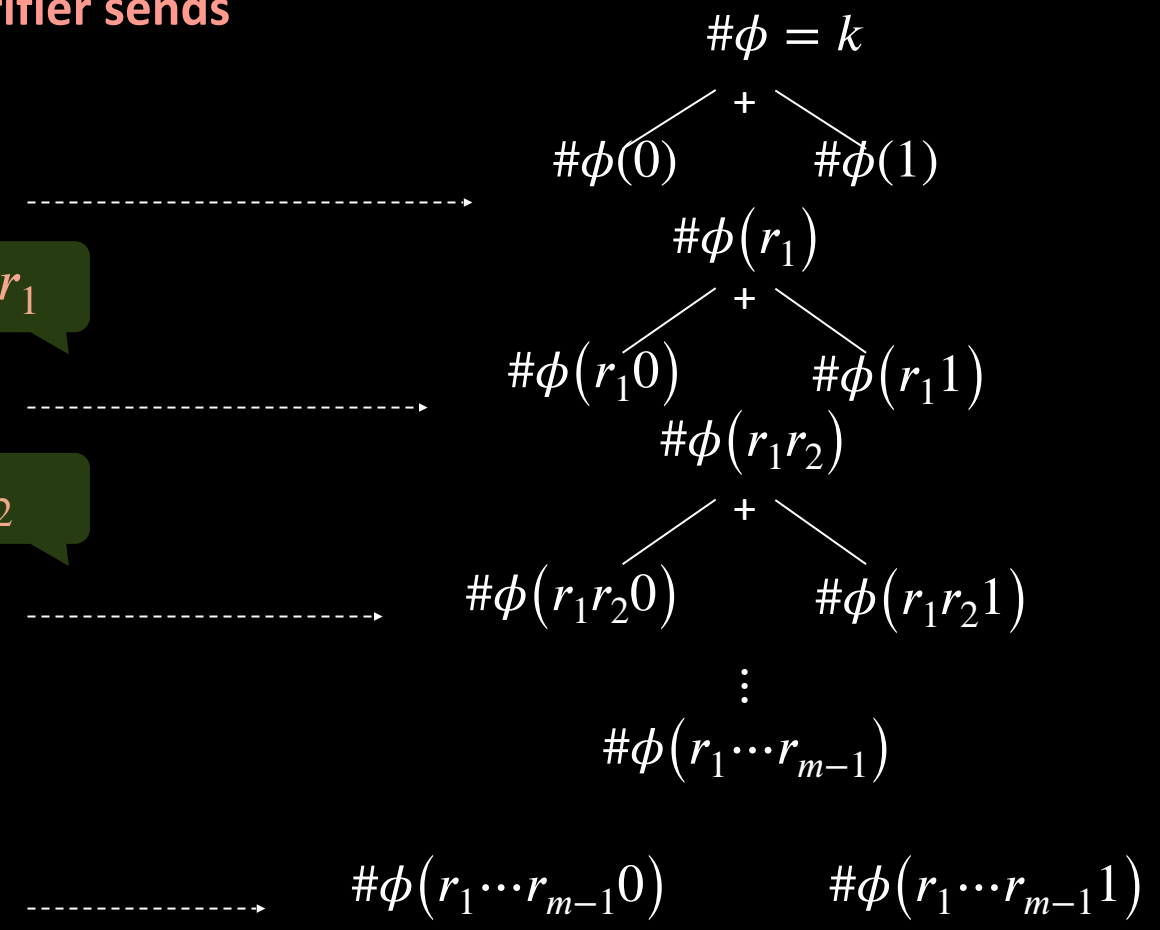
r_1

$\#\phi(r_1 z) = \cdots$

r_2

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

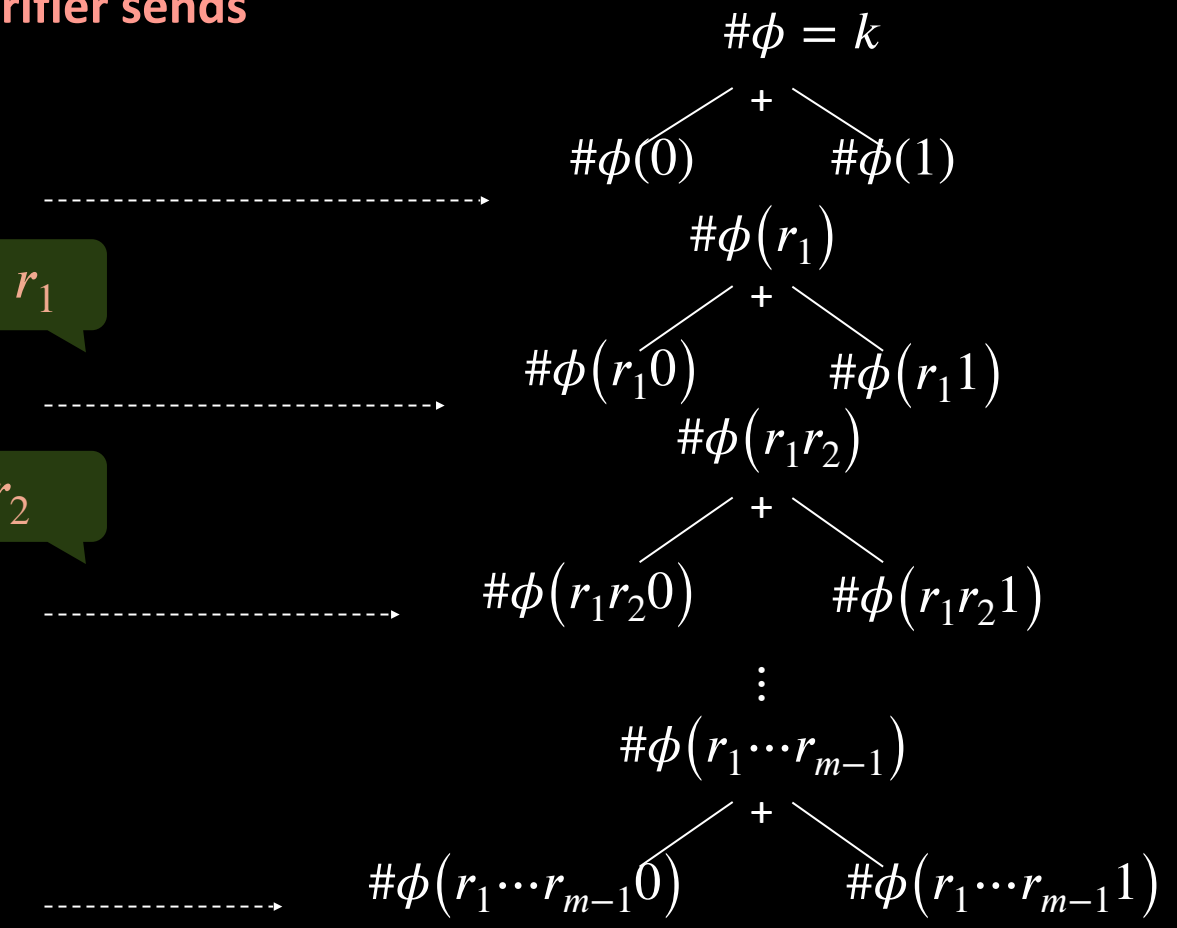
$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

r_1

r_2

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

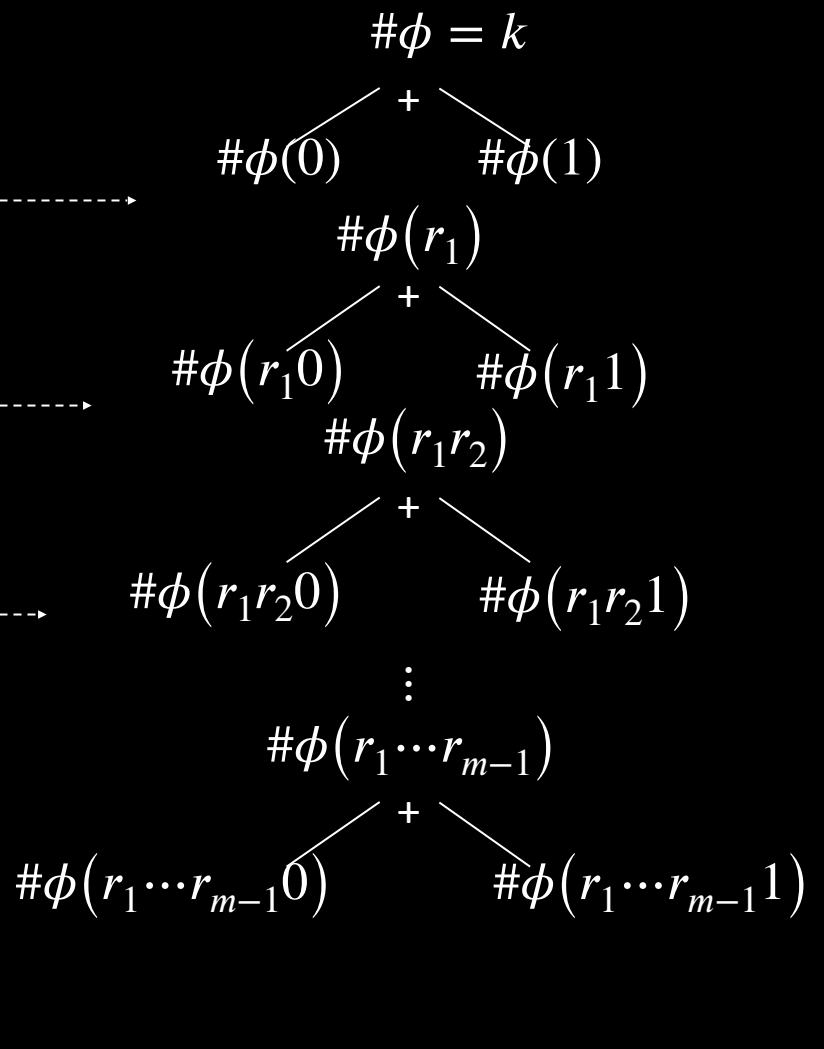
Verifier sends

r_1

r_2

r_m

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

r_1

r_2

r_m

Verifier checks

$\#\phi = k$

$+$

$\#\phi(0)$ $\#\phi(1)$

$\#\phi(r_1)$

$+$

$\#\phi(r_1 0)$ $\#\phi(r_1 1)$

$\#\phi(r_1 r_2)$

$+$

$\#\phi(r_1 r_2 0)$ $\#\phi(r_1 r_2 1)$

$\vdots$

$\#\phi(r_1 \cdots r_{m-1})$

$+$

$\#\phi(r_1 \cdots r_{m-1} 0)$ $\#\phi(r_1 \cdots r_{m-1} 1)$

$\#\phi(r_1 \cdots r_m)$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

r_1

r_2

r_m

Verifier checks

$\#\phi = k$

$+$

$\#\phi(0)$ $\#\phi(1)$

$\#\phi(r_1)$

$+$

$\#\phi(r_1 0)$ $\#\phi(r_1 1)$

$\#\phi(r_1 r_2)$

$+$

$\#\phi(r_1 r_2 0)$ $\#\phi(r_1 r_2 1)$

$\vdots$

$\#\phi(r_1 \cdots r_{m-1})$

$+$

$\#\phi(r_1 \cdots r_{m-1} 0)$ $\#\phi(r_1 \cdots r_{m-1} 1)$

$\#\phi(r_1 \cdots r_m)$

$\phi(r_1 \cdots r_m)$

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

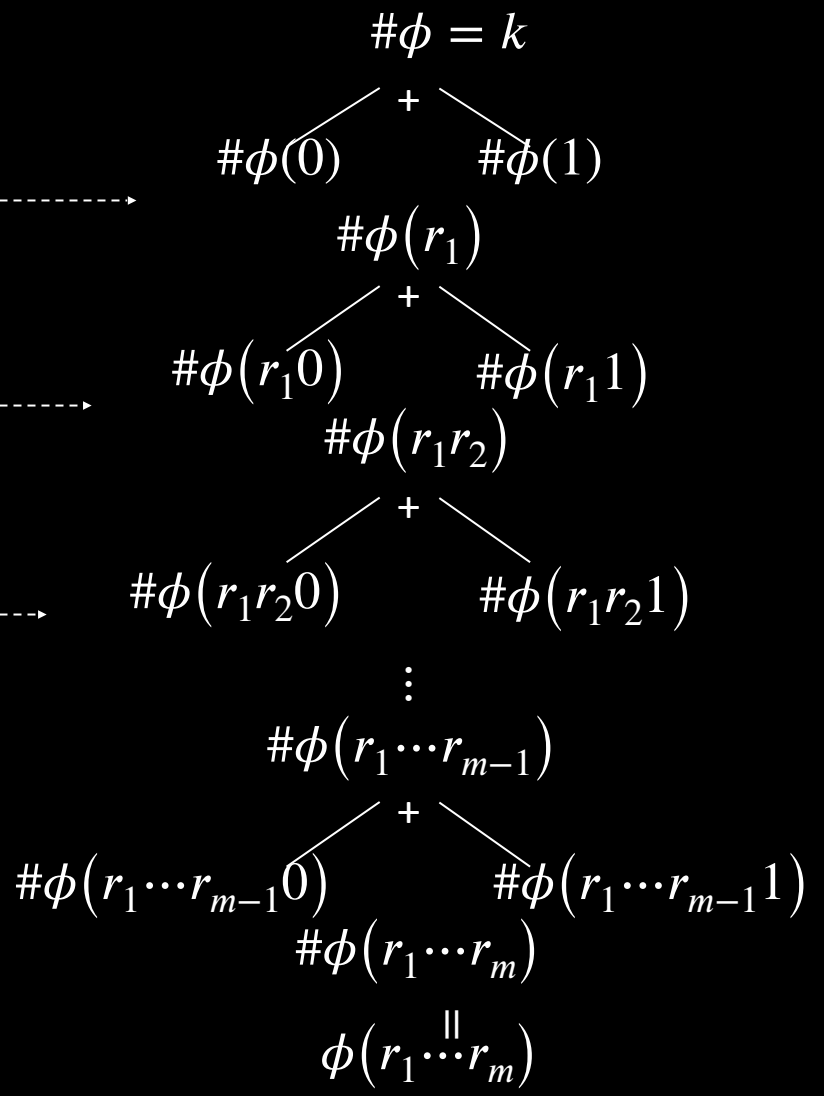
r_1

r_2

r_m

accept

Verifier checks



$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

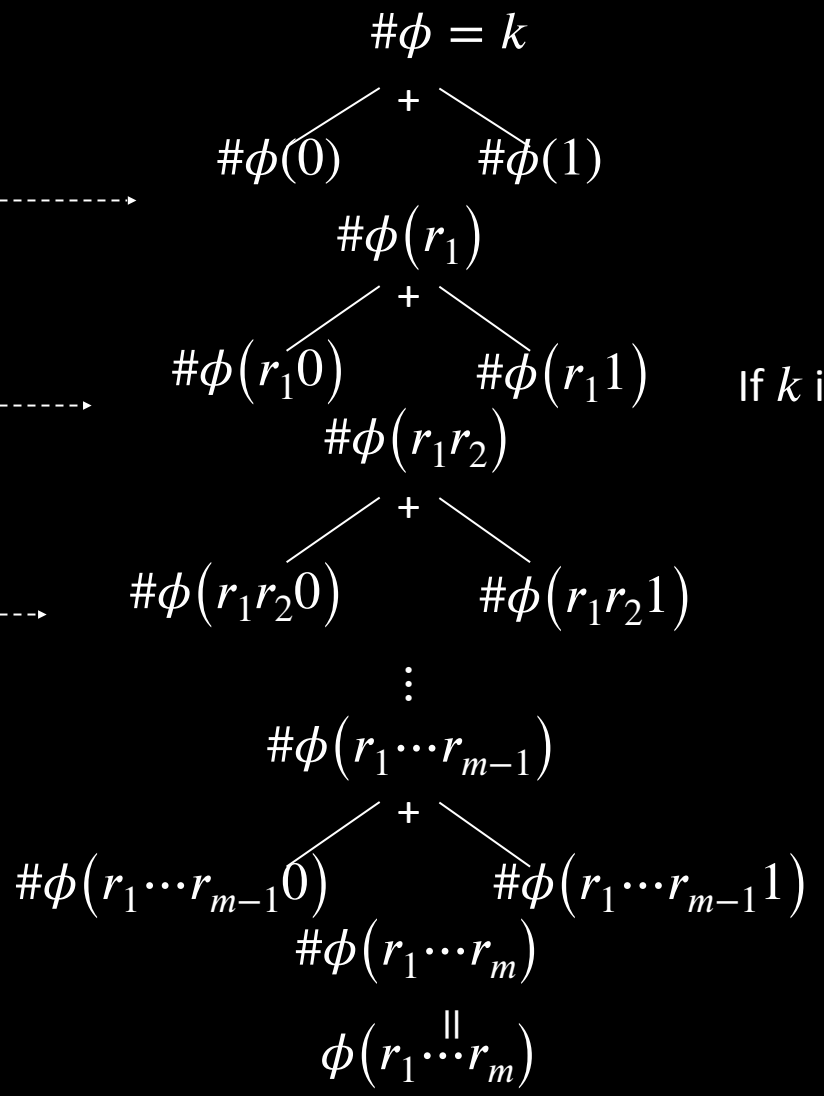
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

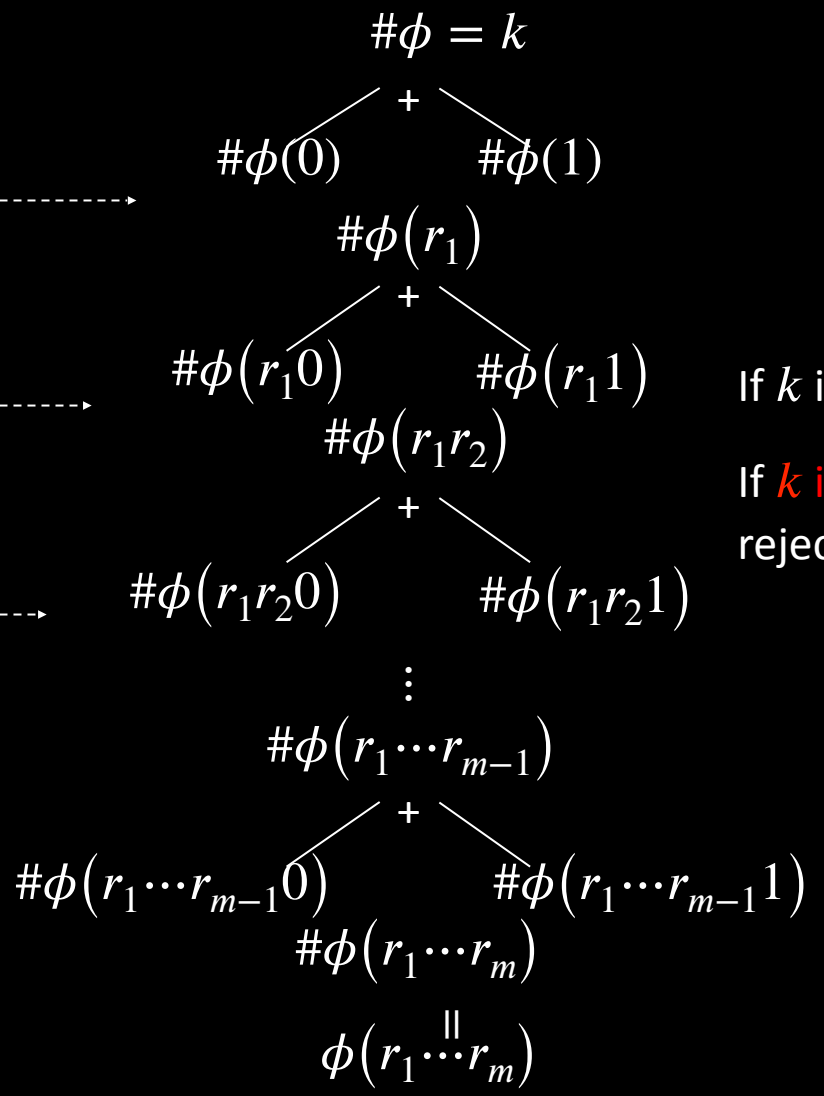
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

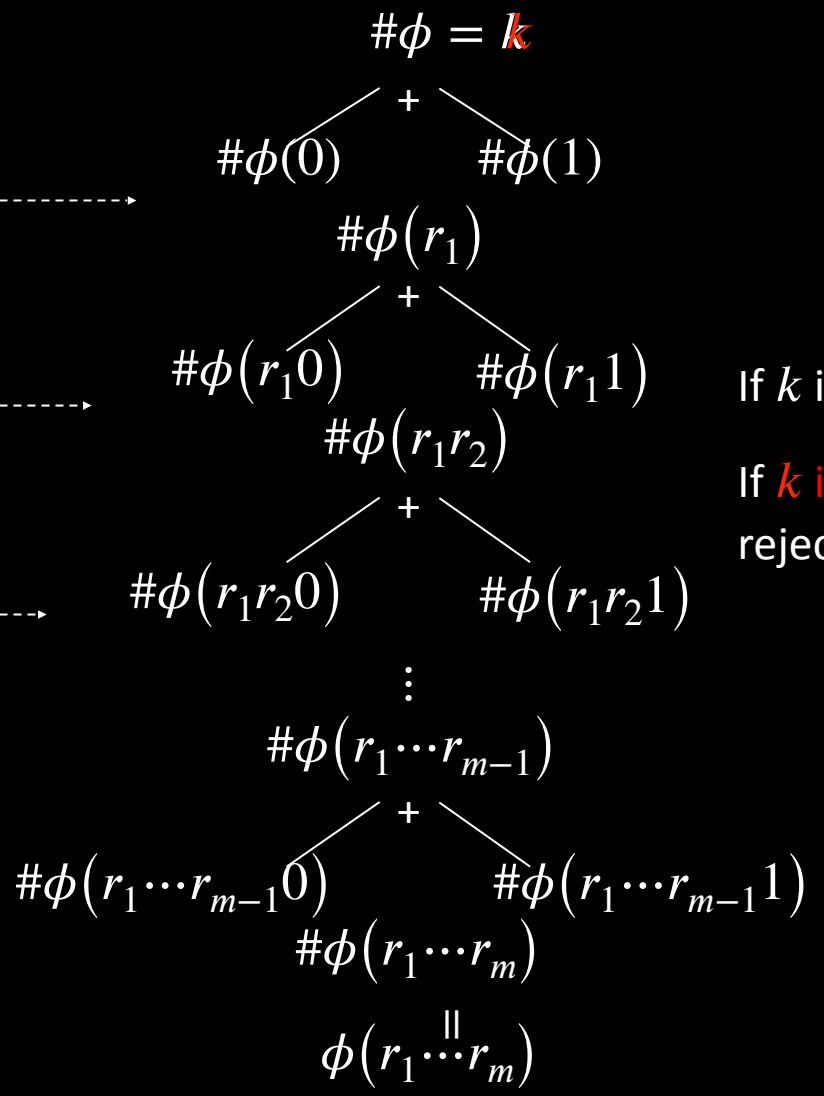
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

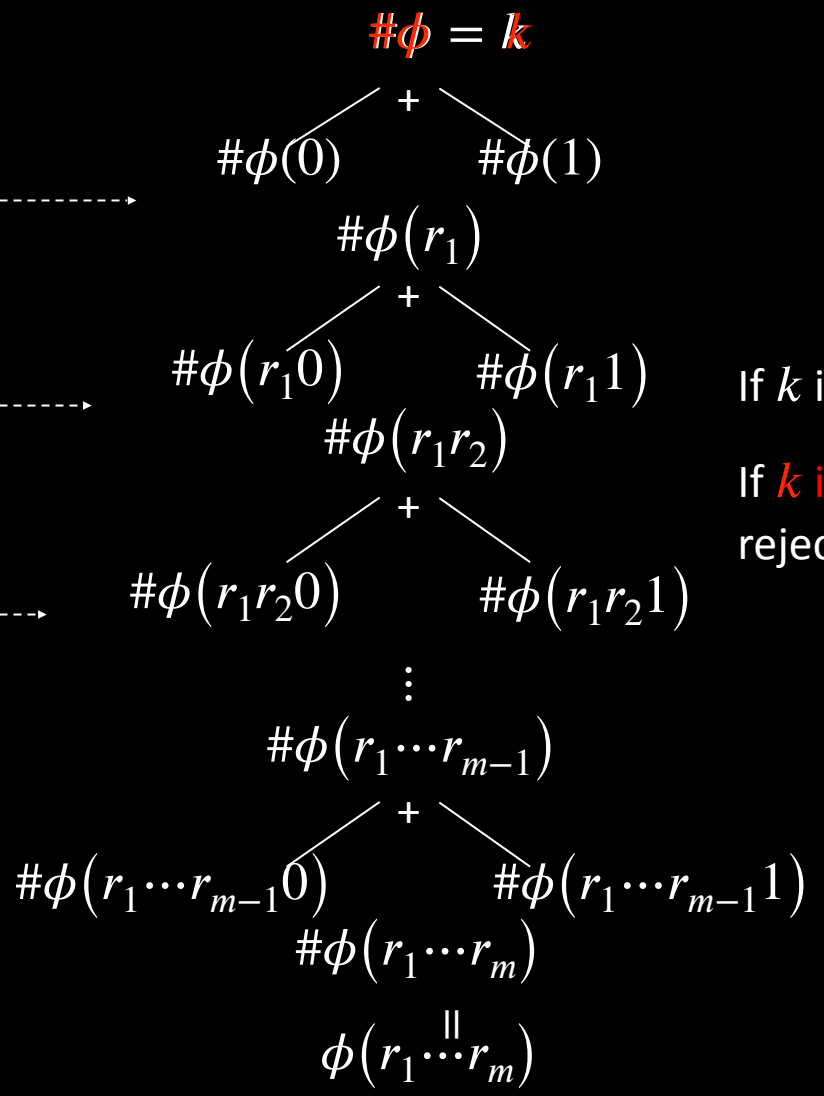
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

r_1

r_2

r_m

accept

Verifier checks

$\#\phi = k$

$\#\phi(0) + \#\phi(1)$

$\#\phi(r_1) = \#\phi(r_1 0) + \#\phi(r_1 1)$

$\#\phi(r_1 r_2) = \#\phi(r_1 r_2 0) + \#\phi(r_1 r_2 1)$

$\vdots$

$\#\phi(r_1 \cdots r_{m-1}) = \#\phi(r_1 \cdots r_{m-1} 0) + \#\phi(r_1 \cdots r_{m-1} 1)$

$\#\phi(r_1 \cdots r_m) \stackrel{?}{=} \phi(r_1 \cdots r_m)$

If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

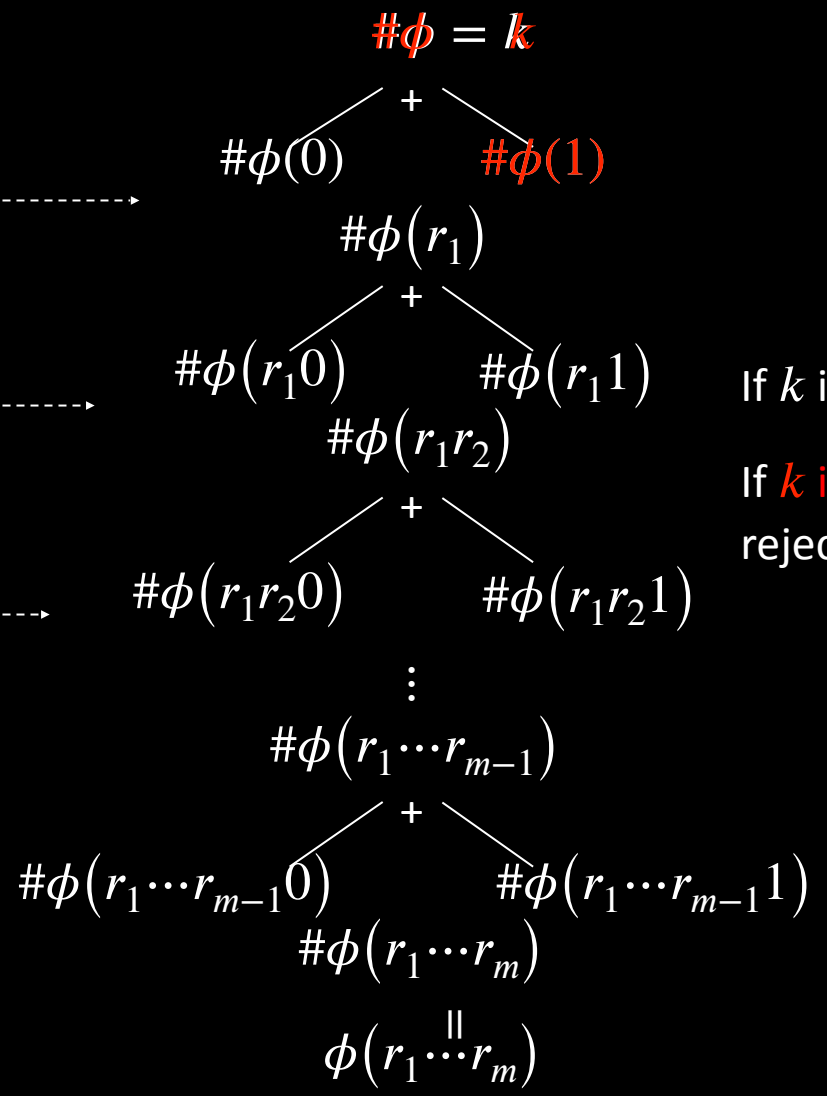
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

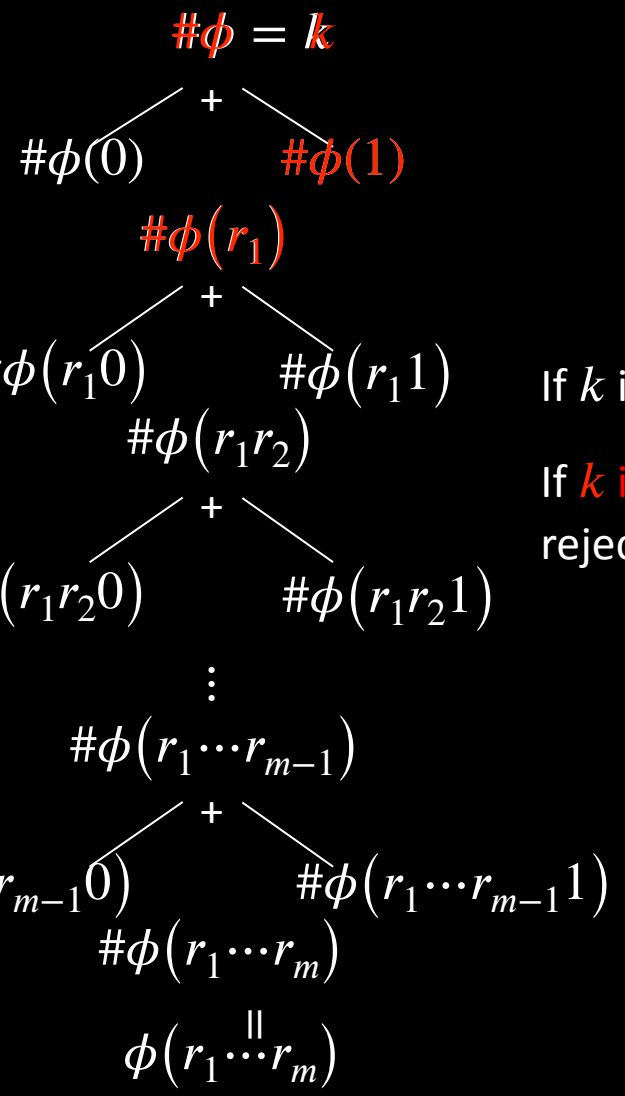
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

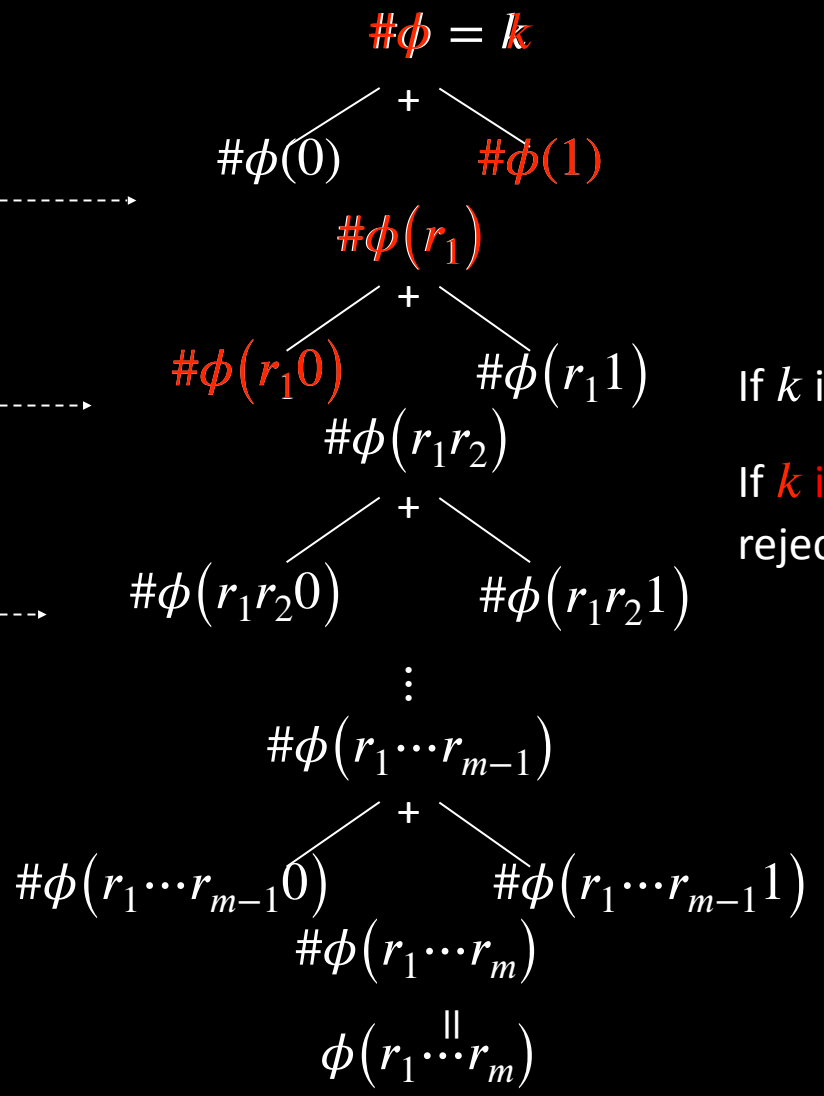
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

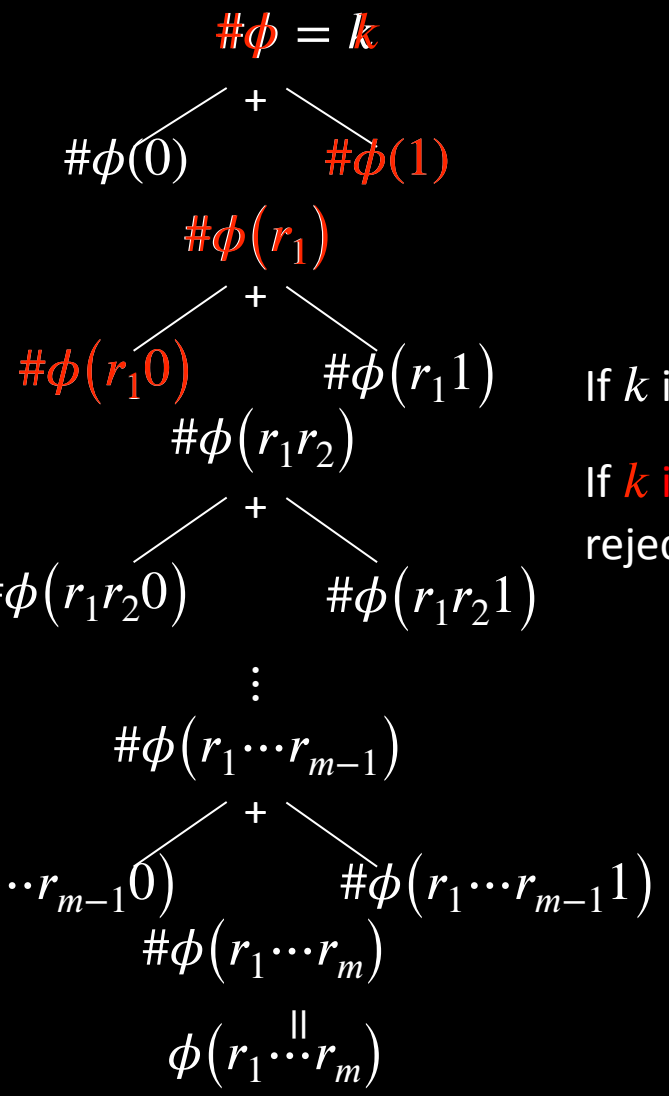
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

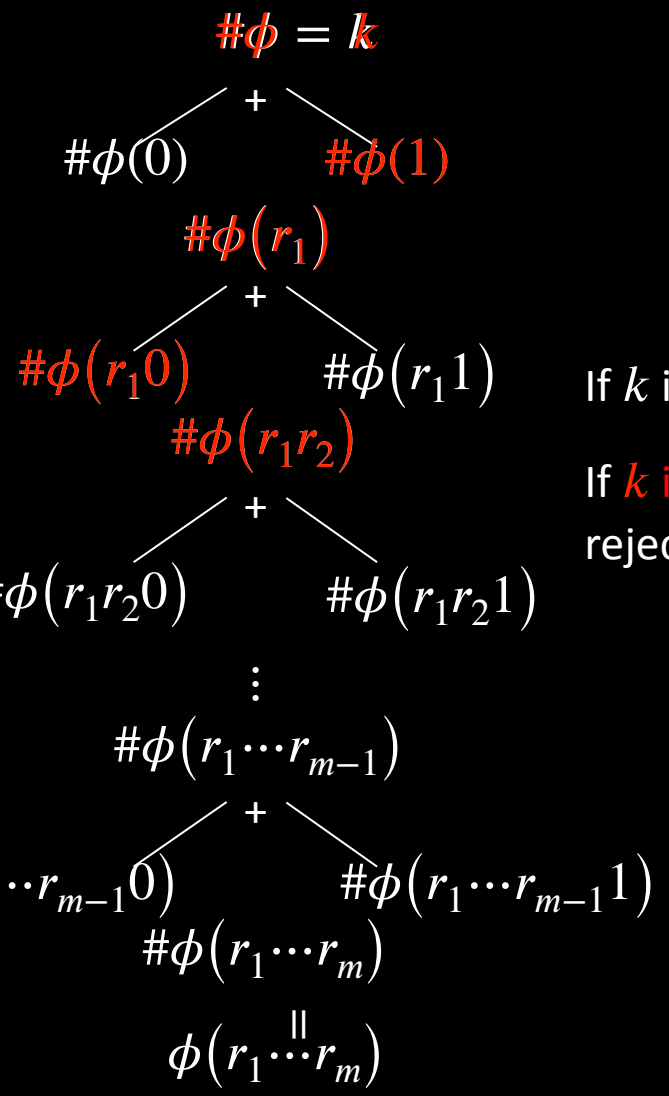
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

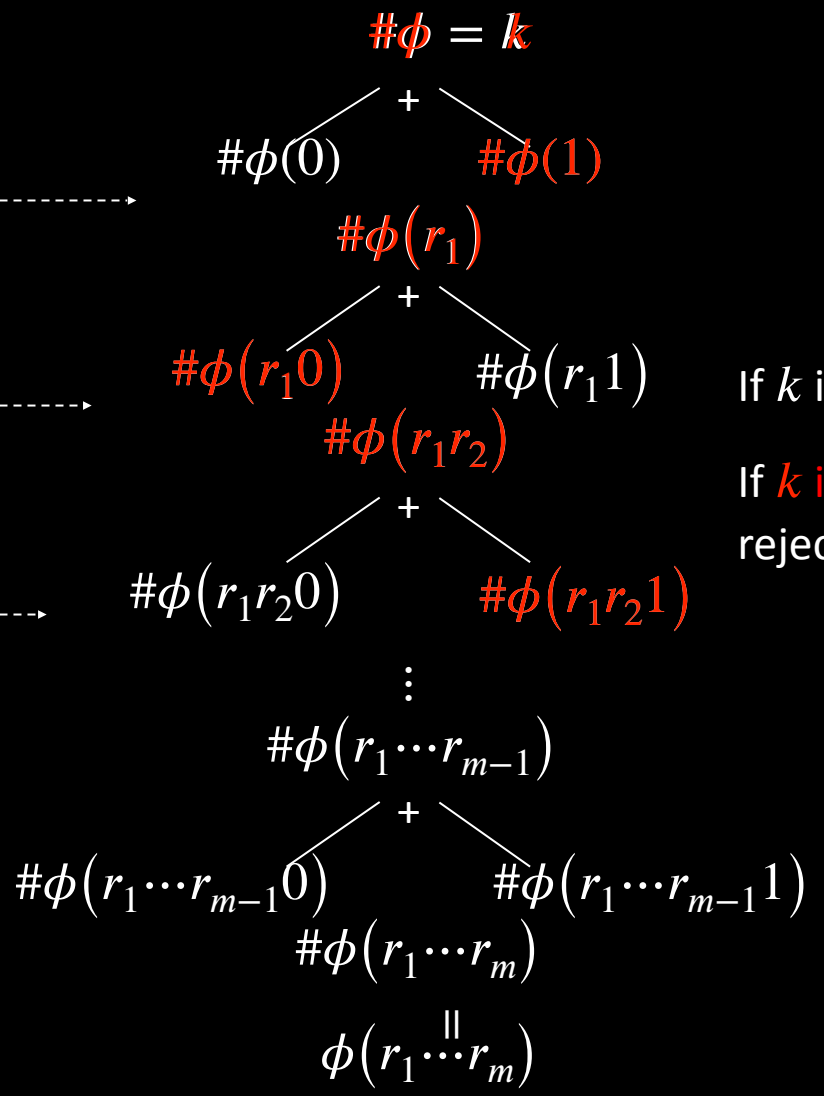
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

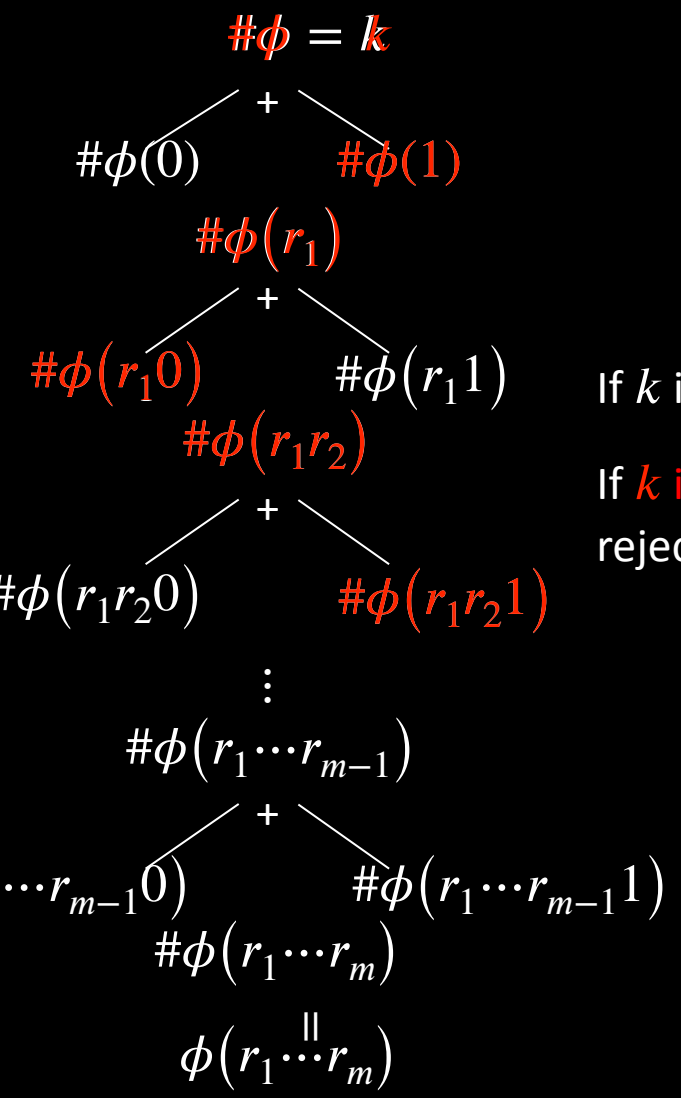
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

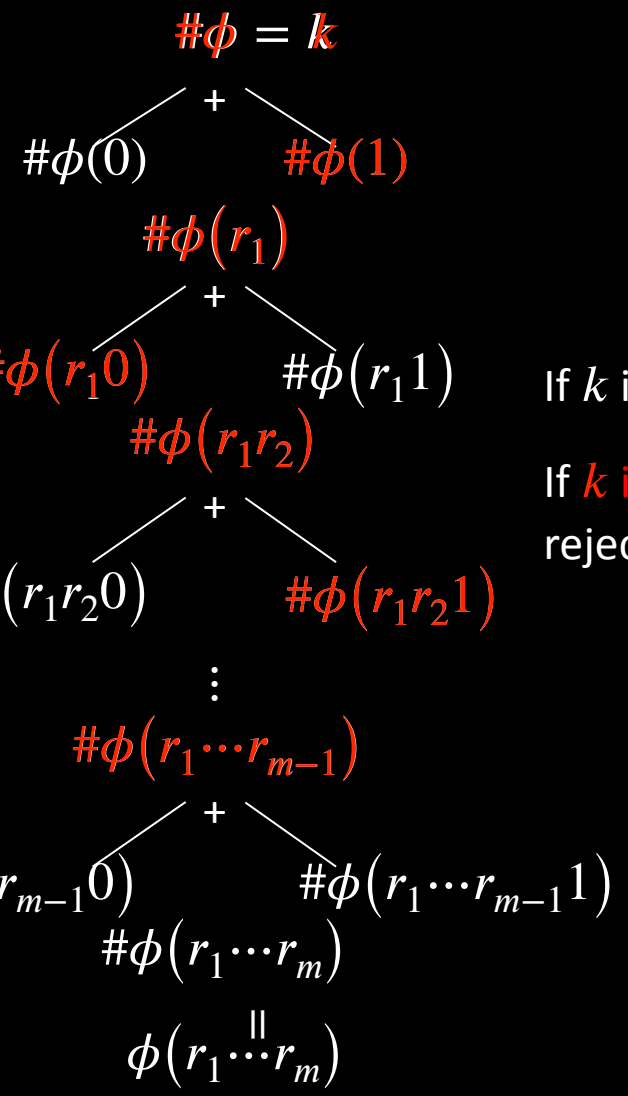
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

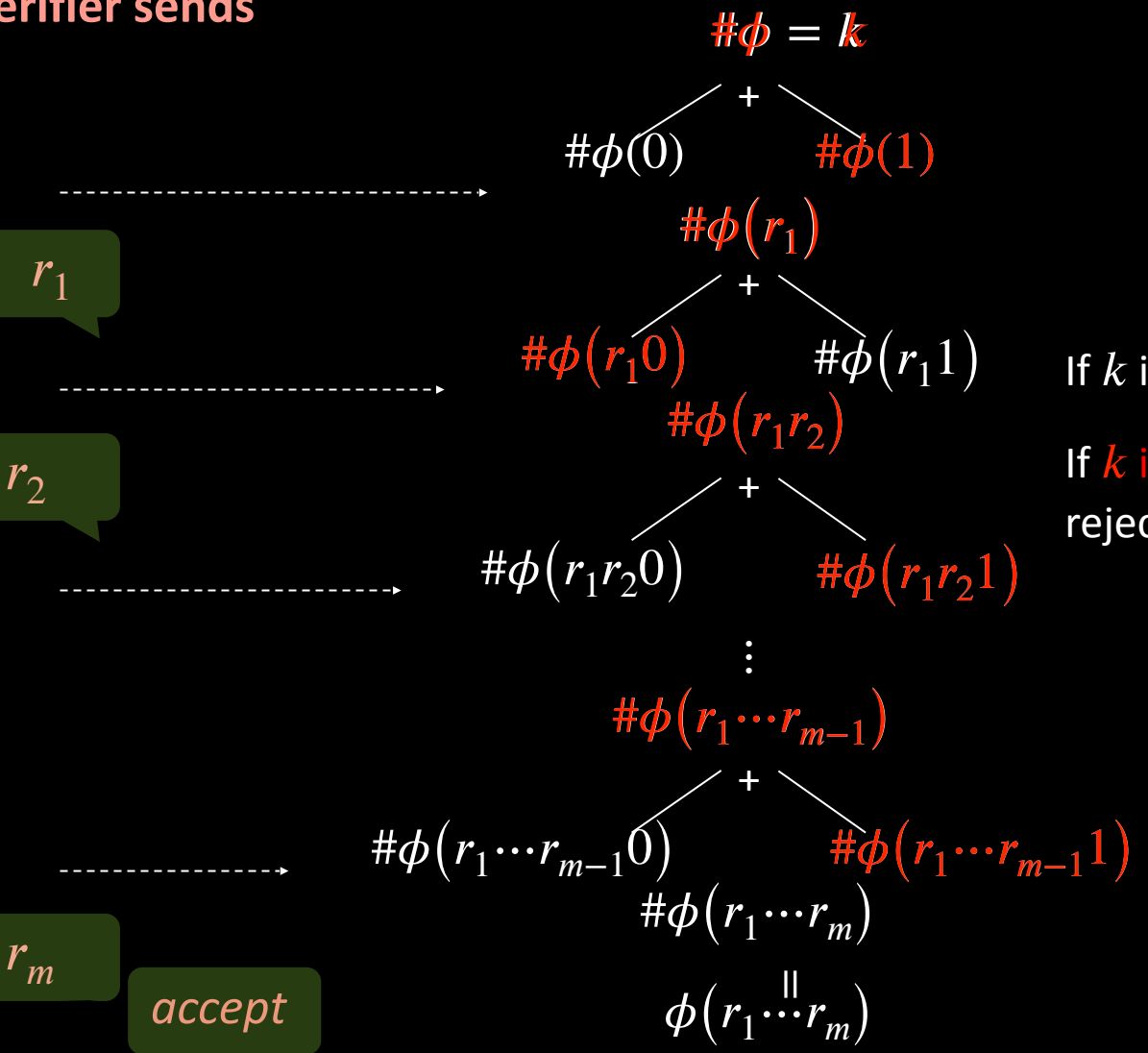
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

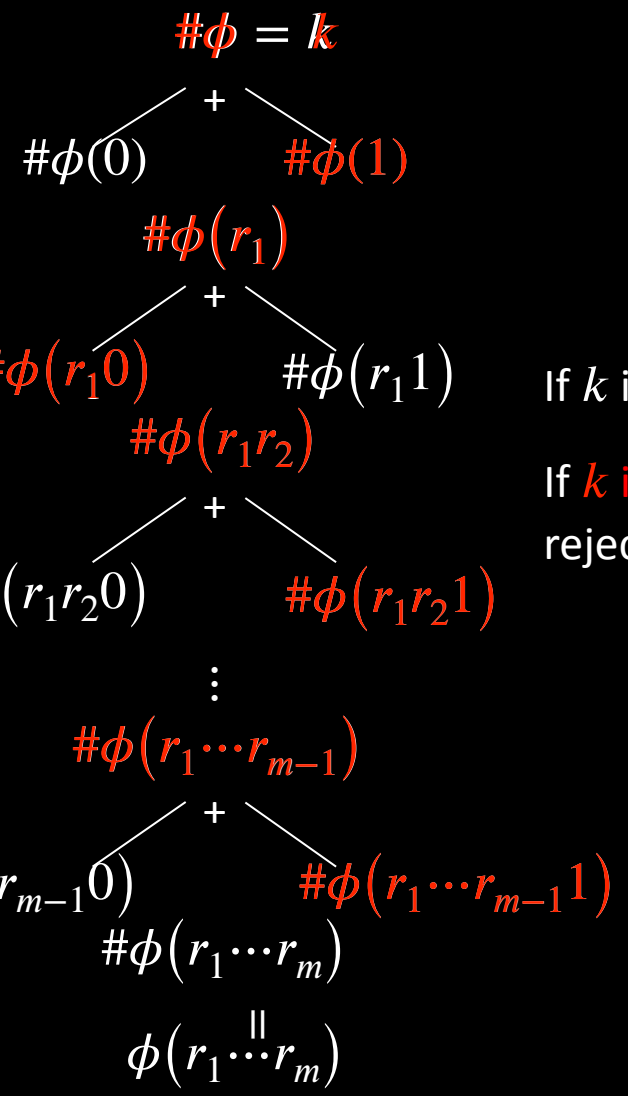
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

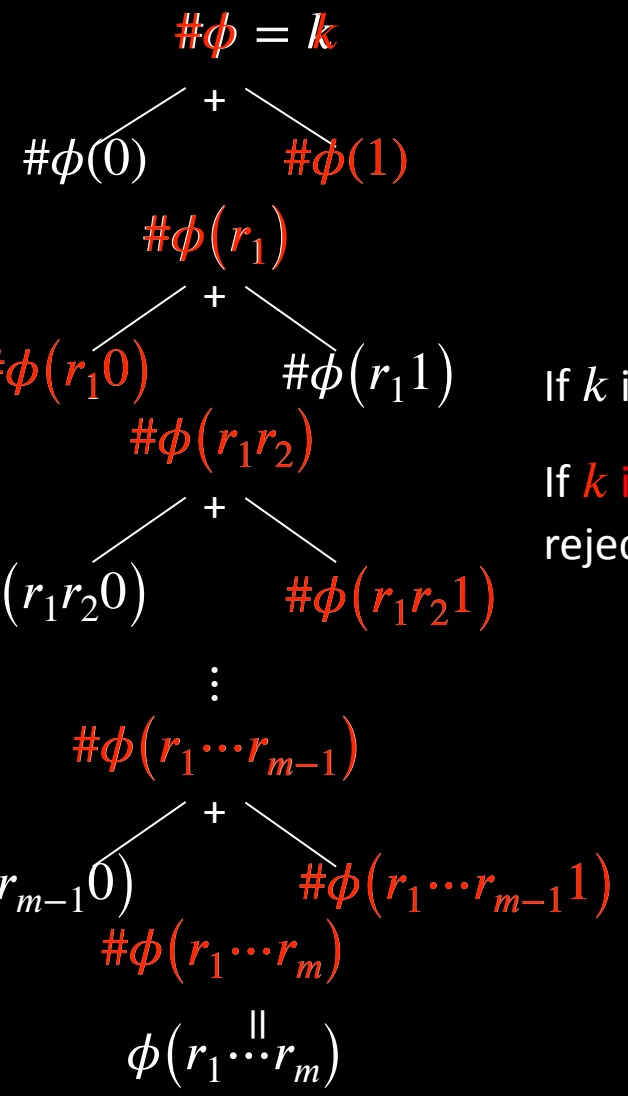
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

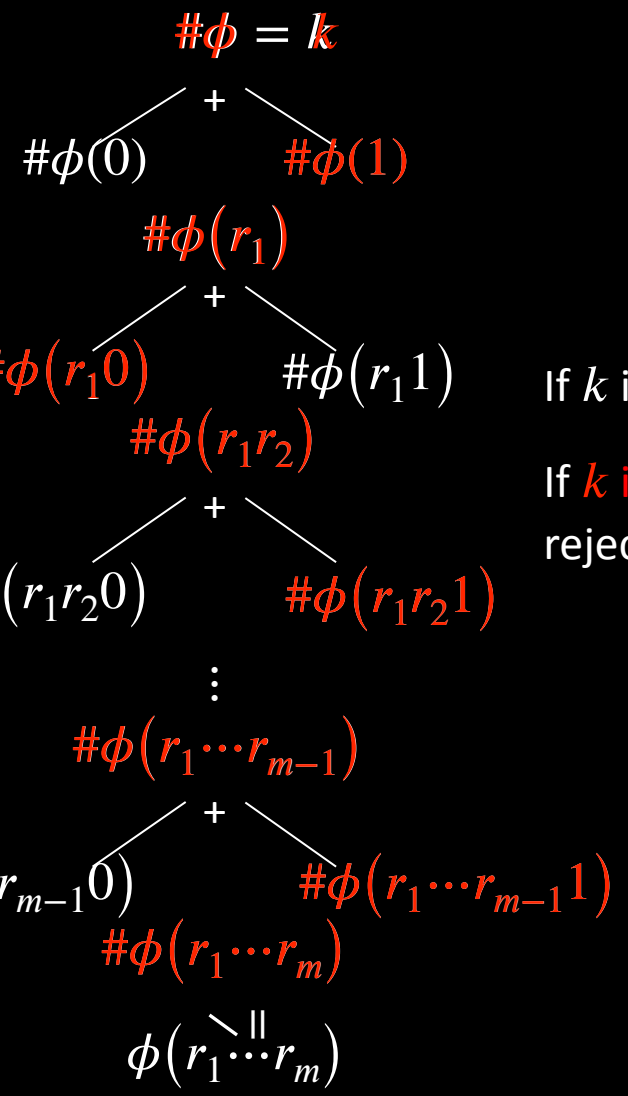
r_1

r_2

r_m

accept

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

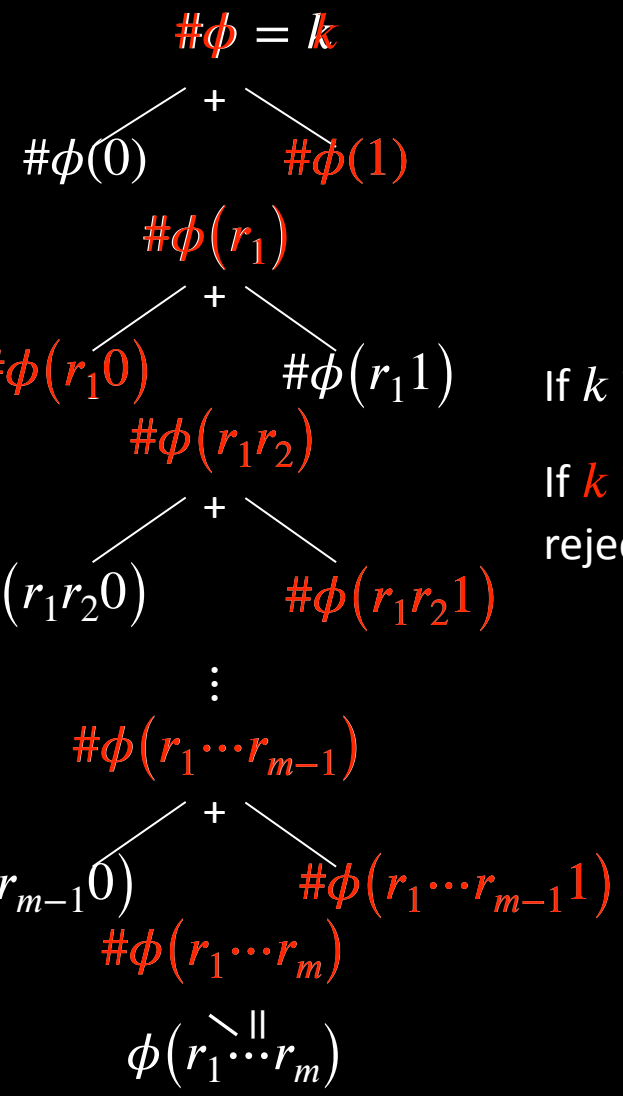
r_1

r_2

r_m

reject

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

$\#SAT \in$ IP – version 2

Input $\langle \phi, k \rangle$

Prover sends

$\#\phi$

$\#\phi(z) = 3z^d - 5z^{d-1} + \cdots + 7$

$\#\phi(r_1 z) = \cdots$

$\#\phi(r_1 r_2 z) = \cdots$

$\#\phi(r_1 \cdots r_{m-1} z) = \cdots$

Verifier sends

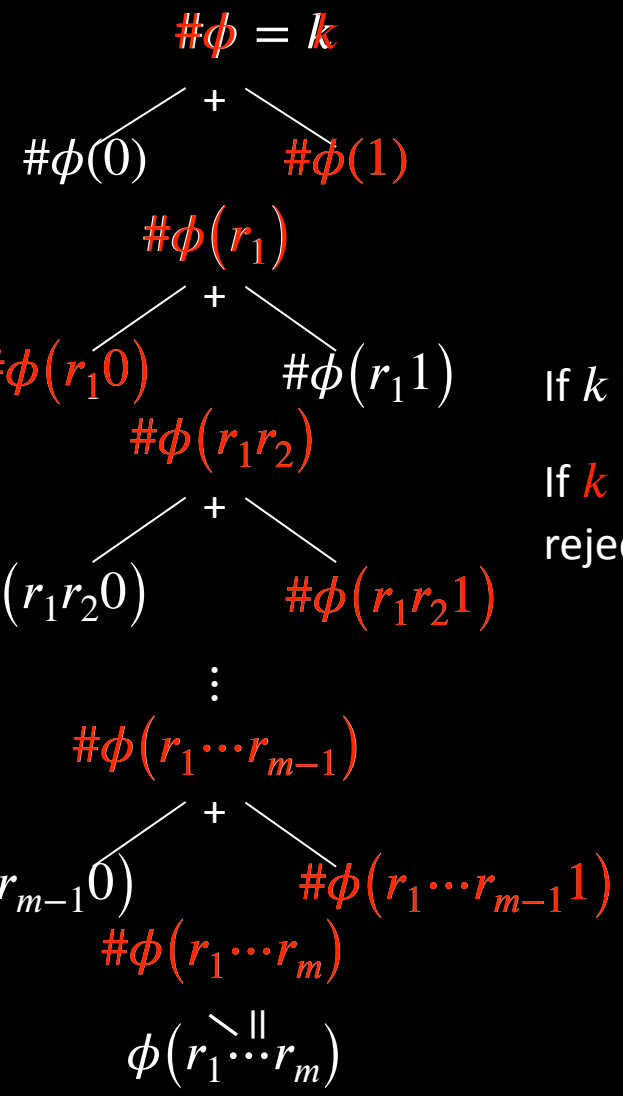
r_1

r_2

r_m

reject

Verifier checks



If k is correct, V will accept.

If k is wrong, V probably will reject, whatever P does.

Quick review of today

Quick review of today

Finished $\#SAT \in$ IP and coNP $\subseteq$ IP

Quick review of today

Finished $\#SAT \in$ IP and coNP $\subseteq$ IP

Additional subjects:

18.405/6.841 Advanced complexity F2021

18.425/6.875 Cryptography F2021

6.842 Randomness and Computation ?

Quick review of today

Finished $\#SAT \in$ IP and coNP $\subseteq$ IP

Additional subjects:

18.405/6.841 Advanced complexity F2021

18.425/6.875 Cryptography F2021

6.842 Randomness and Computation ?

Good luck on the final!

Quick review of today

Finished $\#SAT \in$ IP and coNP $\subseteq$ IP

Additional subjects:
18.405/6.841 Advanced complexity F2021
18.425/6.875 Cryptography F2021
6.842 Randomness and Computation ?

Good luck on the final!

Best wishes for the holidays and the New Year!