Computational Complexity Theory

Lecture 8: Ladner's theorem

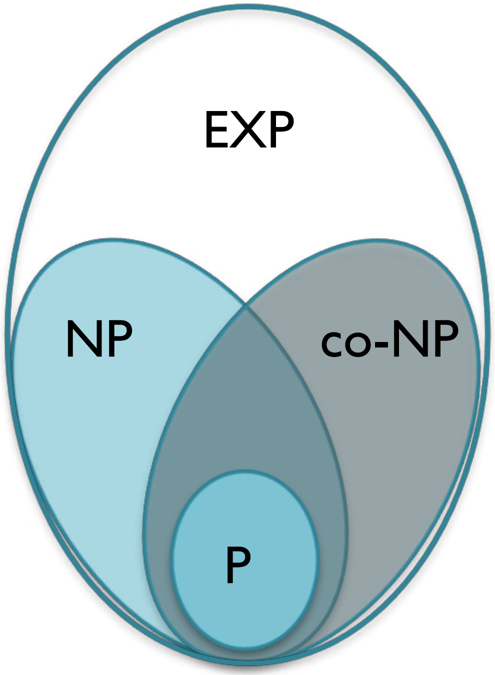
Department of Computer Science,
Indian Institute of Science

Recap: Class EXP

- **Definition.** Class EXP is the exponential time analogue of class P.

$$\mathrm{EXP} = \bigcup_{c \geq 1} \mathrm{DTIME}\left(2^{n^c}\right)$$

- **Observation.** $\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXP}$



Recap: Class EXP

- Definition. Class EXP is the exponential time analogue of class P.

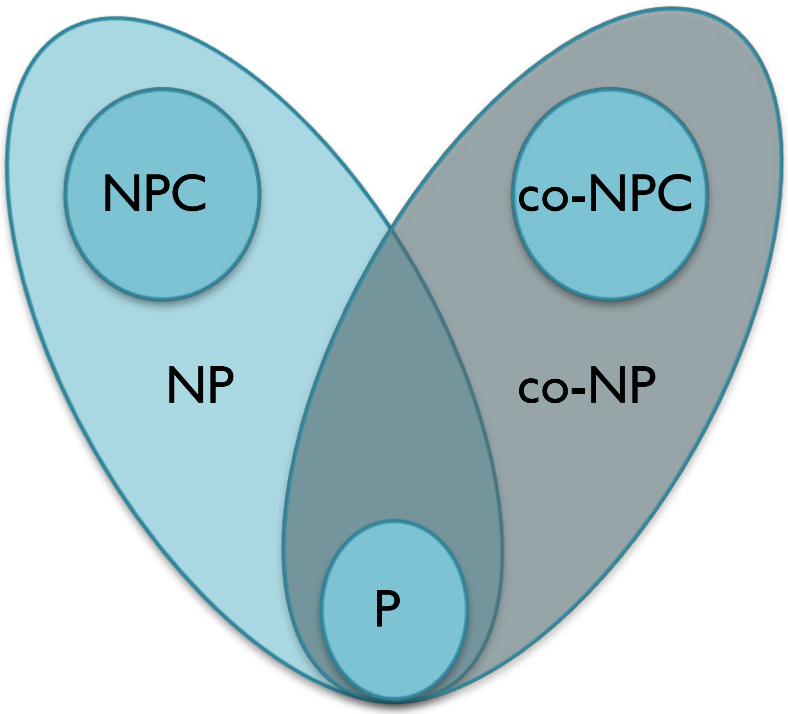
$$EXP = \bigcup_{c \geq 1} DTIME(2^{n^c})$$

- Observation. $P \subseteq NP \subseteq EXP$

- *Exponential Time Hypothesis.* *(Impagliazzo & Paturi 1999)* Any algorithm for 3-SAT takes $\geq 2^{\delta . n}$ time, where $\delta > 0$ is some fixed constant and n is the no. of variables.

In other words, δ cannot be made arbitrarily close to 0.

Recap: Class co-NP and NP∩co-NP



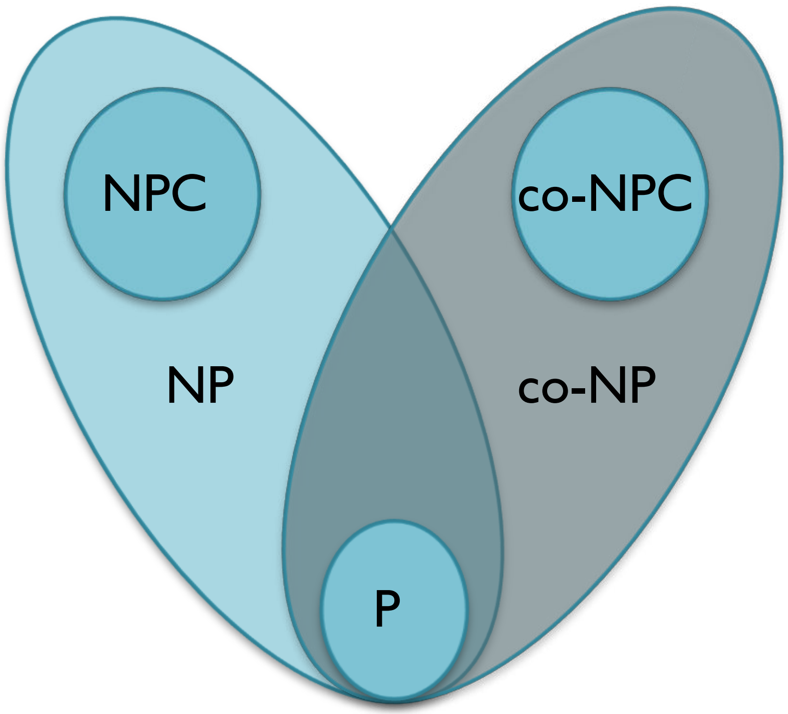
Conjecture: NP ≠ co-NP

⇓

P ≠ NP

General belief: P ≠ NP ∩ co-NP

Recap: Class co-NP and NP∩co-NP



Conjecture: NP ≠ co-NP

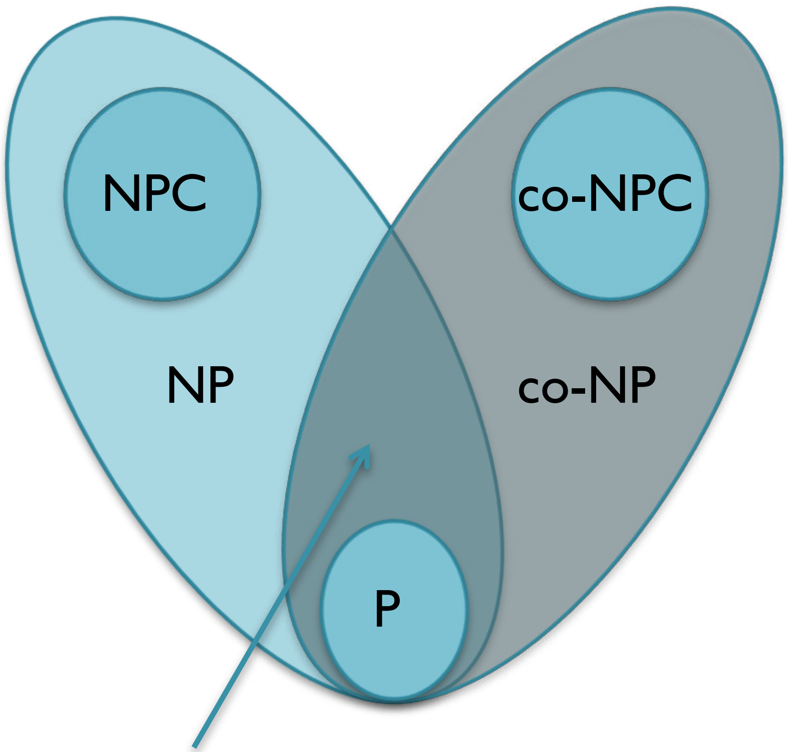
⇓

P ≠ NP

General belief: $P \neq \underbrace{NP \cap co\text{-}NP}_{\text{Edmonds (1966)}}$

…conjectured P = NP ∩ co-NP

Recap: Class co-NP and NP∩co-NP



Conjecture: NP ≠ co-NP

⬇

P ≠ NP

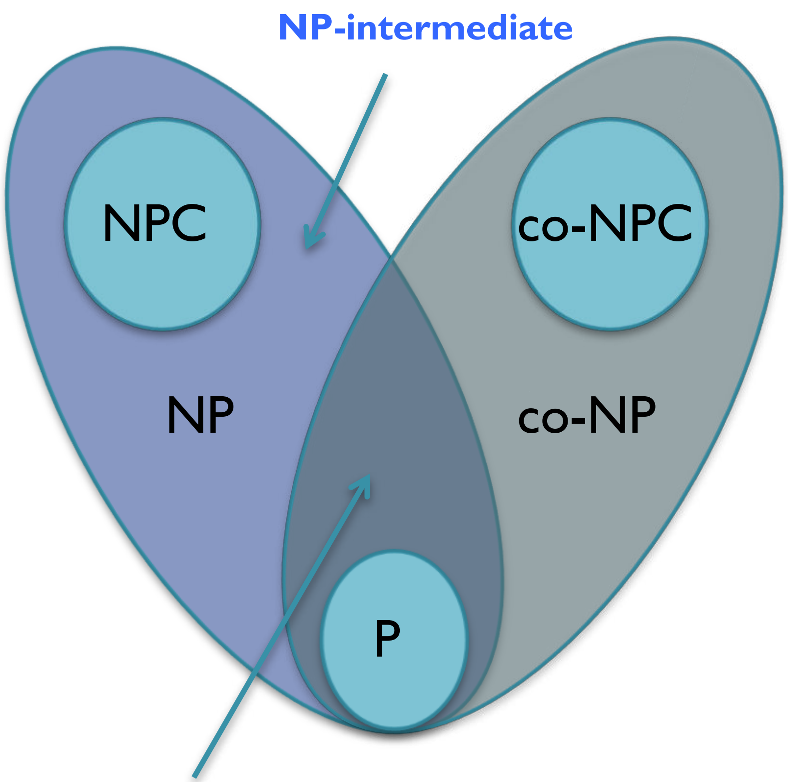
General belief: P ≠ NP ∩ co-NP

Check:
https://cstheory.stackexchange.com/questions/20021/reasons-to-believe-p-ne-np-cap-conp-or-not

- Integer factoring (FACT)
- Approximate shortest vector in a lattice

Ref: "*Lattice problems in NP∩co-NP*" by Aharonov & Regev (2005)

NP-intermediate problems



Conjecture: NP ≠ co-NP

⬇

P ≠ NP

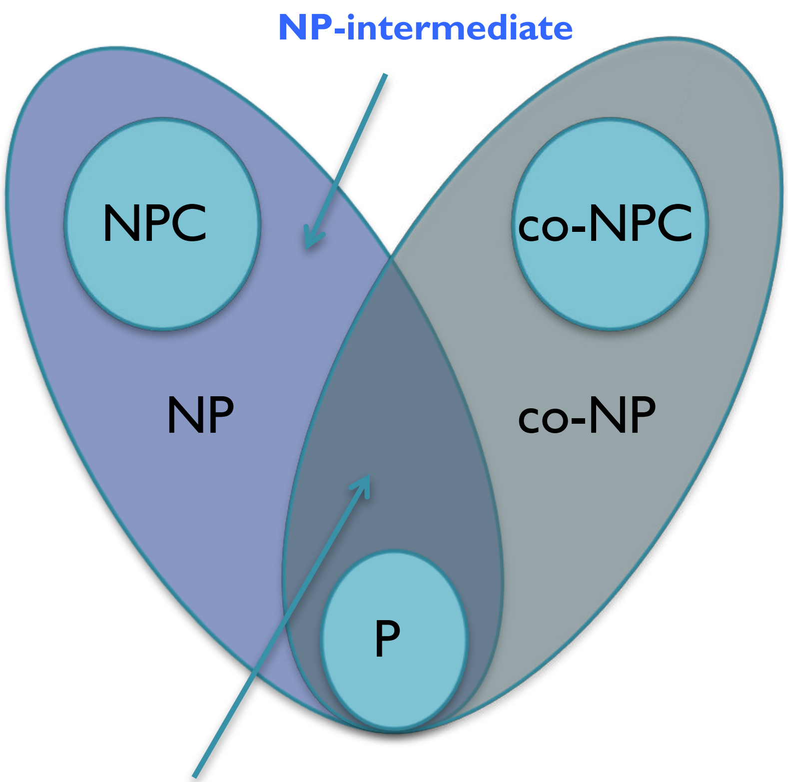
General belief: P ≠ NP ∩ co-NP

Obs: If NP ≠ co-NP and FACT ∉ P then FACT is NP-intermediate.

- Integer factoring (FACT)
- Approximate shortest vector in a lattice

Ref: "*Lattice problems in NP∩co-NP*" by Aharonov & Regev (2005)

NP-intermediate problems



Conjecture: NP ≠ co-NP

⇓

P ≠ NP

General belief: P ≠ NP ∩ co-NP

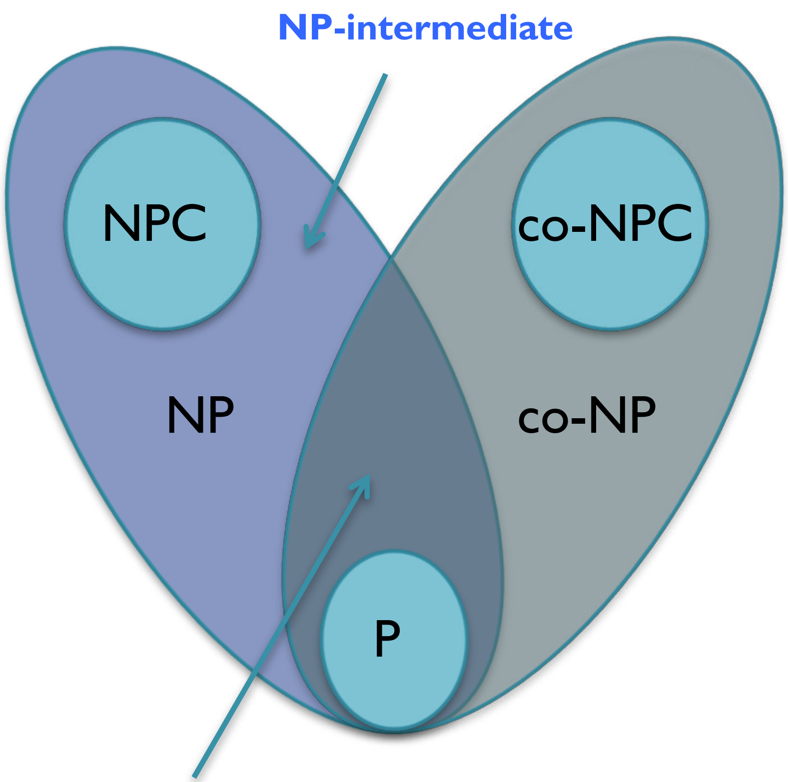
Obs: If NP ≠ co-NP and FACT ∉ P then FACT is NP-intermediate.

Ladner's theorem: P ≠ NP implies existence of a NP-intermediate language.

- Integer factoring (FACT)
- Approximate shortest vector in a lattice

Ref: "*Lattice problems in NP∩co-NP*" by Aharonov & Regev (2005)

NP-intermediate problems



- Integer factoring (FACT)
- Approximate shortest vector in a lattice

Ref: "*Lattice problems in NP∩co-NP*" by Aharonov & Regev (2005)

Conjecture: NP ≠ co-NP

⇓

P ≠ NP

General belief: P ≠ NP ∩ co-NP

Obs: If NP ≠ co-NP and FACT ∉ P then FACT is NP-intermediate.

Ladner's theorem: P ≠ NP implies existence of a NP-intermediate language.

(proved using ***diagonalization***)

Recap: Diagonalization

- *Diagonalization* refers to a class of techniques used in complexity theory to separate complexity classes.
- These techniques are characterized by two main features:
    1. There's a universal TM U that when given strings α and x, simulates M_α on x with only a *small* overhead.

    2. Every string represents some TM, and every TM can be represented by infinitely many strings.

Recap: Time Hierarchy Theorem

- Let $f(n)$ and $g(n)$ be time-constructible functions s.t.,

  $$f(n) \,.\, \log f(n) = o(g(n)).$$

- Theorem. *(Hartmanis & Stearns 1965)*

  $$\mathrm{DTIME}(f(n)) \subsetneq \mathrm{DTIME}(g(n))$$

- Theorem. $\mathrm{P} \subsetneq \mathrm{EXP}$

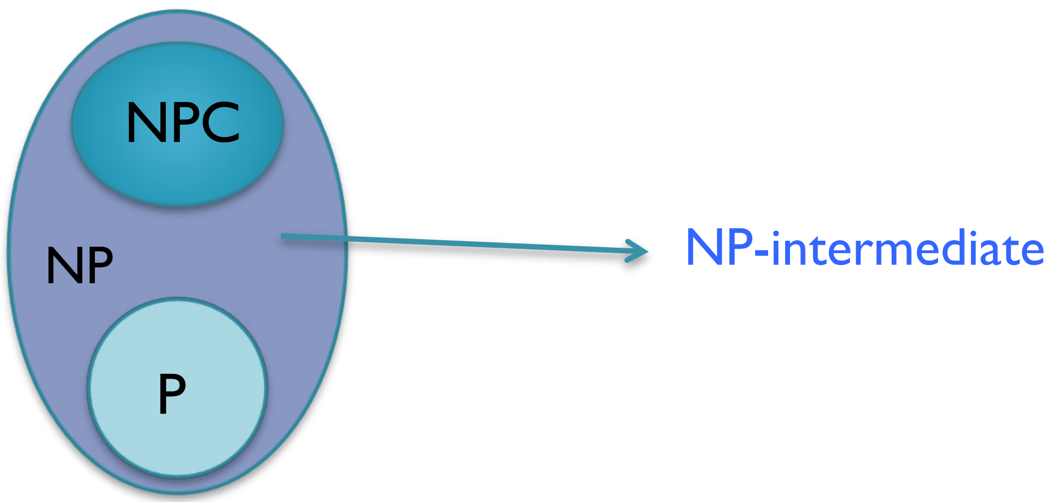
- This type of results are called **<u>lower bounds</u>**.

Ladner’s Theorem

- Another application of Diagonalization

NP-intermediate problems

- Definition. A language L in NP is *NP-intermediate* if L is neither in P nor NP-complete.



NP-intermediate problems

- Definition. A language L in NP is *NP-intermediate* if L is neither in P nor NP-complete.

- Theorem. *(Ladner 1975)* If $P \neq NP$ then there is a NP-intermediate language.

NP-intermediate problems

- Definition. A language L in NP is *NP-intermediate* if L is neither in P nor NP-complete.

- Theorem. *(Ladner 1975)* If P $\neq$ NP then there is a NP-intermediate language.

  Proof. A delicate argument using diagonalization.

NP-intermediate problems

- Definition. A language L in NP is *NP-intermediate* if L is neither in P nor NP-complete.

- Theorem. *(Ladner 1975)* If P $\neq$ NP then there is a NP-intermediate language.

  Proof. Let H: $\mathbb{N} \rightarrow \mathbb{N}$ be a function.

NP-intermediate problems

- **Definition.** A language L in NP is *NP-intermediate* if L is neither in P nor NP-complete.

- **Theorem.** *(Ladner 1975)* If $P \neq NP$ then there is a NP-intermediate language.

  Proof. Let $H: \mathbb{N} \rightarrow \mathbb{N}$ be a function.

  Let $\quad SAT_H = \{\Psi 0 1^{m^{H(m)}} : \Psi \in SAT \text{ and } |\Psi| = m\}$

NP-intermediate problems

- Definition. A language L in NP is *NP-intermediate* if L is neither in P nor NP-complete.

- Theorem. *(Ladner 1975)* If P $\neq$ NP then there is a NP-intermediate language.

  Proof. Let H: $\mathbb{N} \rightarrow \mathbb{N}$ be a function.

Let $SAT_H = \{\Psi 0 1^{m^{H(m)}} : \Psi \in SAT \text{ and } |\Psi| = m\}$

H would be defined in such a way that SAT_H is NP-intermediate (assuming P $\neq$ NP)

Ladner's theorem: Constructing H

- Theorem. There's a function $H: \mathbb{N} \rightarrow \mathbb{N}$ such that

  1. $H(m)$ is computable from m in $O(m^3)$ time.

Ladner's theorem: Constructing H

- Theorem. There's a function $H: \mathbb{N} \to \mathbb{N}$ such that

  1. $H(m)$ is computable from m in $O(m^3)$ time.

  2. If $SAT_H \in P$ then $H(m) \leq C$ (a constant).

for every m

Ladner's theorem: Constructing H

- Theorem. There's a function $H: \mathbb{N} \rightarrow \mathbb{N}$ such that

    1. $H(m)$ is computable from m in $O(m^3)$ time.

    2. If $SAT_H \in P$ then $H(m) \leq C$ (a constant).

    3. If $SAT_H \notin P$ then $H(m) \rightarrow \infty$ with m.

Ladner's theorem: Constructing H

- **Theorem.** There's a function $H: \mathbb{N} \rightarrow \mathbb{N}$ such that

    1. $H(m)$ is computable from m in $O(m^3)$ time.

    2. If $SAT_H \in P$ then $H(m) \leq C$ (a constant).

    3. If $SAT_H \notin P$ then $H(m) \rightarrow \infty$ with m.

Proof: Later (uses diagonalization).

Let's see the proof of Ladner's theorem assuming the existence of such a "special" H.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose $SAT_H \in P$. Then $H(m) \leq C$.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose $SAT_H \in P$. Then $H(m) \leq C$.
- This implies a poly-time algorithm for SAT as follows:

Ladner's theorem: Proof

$P \neq NP$

- Suppose $SAT_H \in P$. Then $H(m) \leq C$.
- This implies a poly-time algorithm for SAT as follows:
  - On input ϕ, find $m = |\phi|$.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose $SAT_H \in P$. Then $H(m) \leq C$.
- This implies a poly-time algorithm for SAT as follows:
  - On input ϕ, find $m = |\phi|$.
  - Compute $H(m)$, and construct the string $\phi\, 0\, 1^{m^{H(m)}}$

Ladner's theorem: Proof

$P \neq NP$

- Suppose $SAT_H \in P$. Then $H(m) \leq C$.
- This implies a poly-time algorithm for SAT as follows:
  - On input ϕ, find $m = |\phi|$.
  - Compute $H(m)$, and construct the string $\phi\, 0\, 1^{m^{H(m)}}$
  - Check if $\phi\, 0\, 1^{m^{H(m)}}$ belongs to SAT_H.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose $SAT_H \in P$. Then $H(m) \leq C$.
- This implies a poly-time algorithm for SAT as follows:
  - On input ϕ, find $m = |\phi|$.
  - Compute $H(m)$, and construct the string $\phi\, 0\, 1^{m^{H(m)}}$
  - Check if $\underbrace{\phi\, 0\, 1^{m^{H(m)}}}_{\text{length at most } m + 1 + m^C}$ belongs to SAT_H.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose $SAT_H \in P$. Then $H(m) \leq C$.
- This implies a poly-time algorithm for SAT as follows:
  - On input ϕ, find $m = |\phi|$.
  - Compute $H(m)$, and construct the string $\phi\ 0\ 1^{m^{H(m)}}$
  - Check if $\phi\ 0\ 1^{m^{H(m)}}$ belongs to SAT_H.
- As $P \neq NP$, it must be that $SAT_H \notin P$.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \to \infty$ with m.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \to \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\ 0\ 1^k$$

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \rightarrow \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H$$

$$\underbrace{\phi}_{|\phi| = n} \xmapsto{f} \underbrace{\Psi\, 0\, 1^k}_{|\Psi\, 0\, 1^k| = n^c}$$

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \to \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H$$

$$\underbrace{\phi}_{|\phi| = n} \xmapsto{f} \underbrace{\Psi\, 0\, 1^k}_{|\Psi\, 0\, 1^k| = n^c}$$

Let m_0 be the largest s.t. $H(m_0) \leq 2c$.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \to \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad \phi \xmapsto{f} \Psi\, 0\, 1^k$$

Let m_0 be the largest s.t. $H(m_0) \leq 2c$.

➢ On input ϕ, compute $f(\phi) = \Psi\, 0\, 1^k$. Let $m = |\Psi|$.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \to \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\ 0\ 1^k$$

Let m_0 be the largest s.t. $H(m_0) \leq 2c$.

> On input ϕ, compute $f(\phi) = \Psi\ 0\ 1^k$. Let $m = |\Psi|$.
> Compute $H(m)$ and check if $k = m^{H(m)}$.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \to \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\, 0\, 1^k$$

Let m_0 be the largest s.t. $H(m_0) \leq 2c$.

- On input ϕ, compute $f(\phi) = \Psi\, 0\, 1^k$. Let $m = |\Psi|$.
- Compute $H(m)$ and check if $k = m^{H(m)}$. (Homework: Verify that this can be done in poly(n) time.)

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \to \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\, 0\, 1^k$$

Let m_0 be the largest s.t. $H(m_0) \leq 2c$.

- On input ϕ, compute $f(\phi) = \Psi\, 0\, 1^k$. Let $m = |\Psi|$.
- Compute $H(m)$ and check if $k = m^{H(m)}$.

Either $m \leq m_0$ (in which case the task reduces to checking if a constant-size Ψ is satisfiable),

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \rightarrow \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$SAT \leq_p SAT_H$ $\qquad\qquad$ $\phi \xmapsto{f} \Psi\, 0\, 1^k$

Let m_0 be the largest s.t. $H(m_0) \leq 2c$.

- On input ϕ, compute $f(\phi) = \Psi\, 0\, 1^k$. Let $m = |\Psi|$.
- Compute $H(m)$ and check if $k = m^{H(m)}$.

or $H(m) > 2c$ (as $H(m)$ tends to infinity with m).

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \rightarrow \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\ 0\ 1^k$$

- On input ϕ, compute $f(\phi) = \Psi\ 0\ 1^k$. Let $m = |\Psi|$.
- Compute $H(m)$ and check if $k = m^{H(m)}$.
- Hence, w.l.o.g. $\qquad |f(\phi)| \geq k > m^{2c}$

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \rightarrow \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\, 0\, 1^k$$

- On input ϕ, compute $f(\phi) = \Psi\, 0\, 1^k$. Let $m = |\Psi|$.
- Compute $H(m)$ and check if $k = m^{H(m)}$.
- Hence, w.l.o.g. $n^c = |f(\phi)| \geq k > m^{2c}$

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \rightarrow \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\, 0\, 1^k$$

- On input ϕ, compute $f(\phi) = \Psi\, 0\, 1^k$. Let $m = |\Psi|$.
- Compute $H(m)$ and check if $k = m^{H(m)}$.
- Hence, $\sqrt{n} \geq m$.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \rightarrow \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$SAT \leq_p SAT_H$ $\qquad\qquad$ $\phi \xmapsto{f} \Psi\, 0\, 1^k$

- On input ϕ, compute $f(\phi) = \Psi\, 0\, 1^k$. Let $m = |\Psi|$.
- Compute $H(m)$ and check if $k = m^{H(m)}$.
- Hence, $\sqrt{n} \geq m$. Also $\phi \in SAT$ iff $\Psi \in SAT$

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \rightarrow \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\, 0\, 1^k$$

- On input ϕ, compute $f(\phi) = \Psi\, 0\, 1^k$. Let $m = |\Psi|$.
- Compute $H(m)$ and check if $k = m^{H(m)}$.
- Hence, $\sqrt{n} \geq m$. Also $\phi \in SAT$ iff $\Psi \in SAT$

Thus, checking if an n-size formula ϕ is satisfiable reduces to checking if a $\sqrt{n}$-size formula Ψ is satisfiable.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \rightarrow \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\, 0\, 1^k$$

- On input ϕ, compute $f(\phi) = \Psi\, 0\, 1^k$. Let $m = |\Psi|$.
- Compute $H(m)$ and check if $k = m^{H(m)}$.
- Hence, $\sqrt{n} \geq m$. Also $\phi \in SAT$ iff $\Psi \in SAT$

Do this recursively! Only $O(\log \log n)$ recursive steps required.

Ladner's theorem: Proof

$$P \neq NP$$

- Suppose SAT_H is NP-complete. Then $H(m) \rightarrow \infty$ with m.
- This also implies a poly-time algorithm for SAT:

$$SAT \leq_p SAT_H \qquad\qquad \phi \xmapsto{f} \Psi\ 0\ 1^k$$

  - On input ϕ, compute $f(\phi) = \Psi\ 0\ 1^k$. Let $m = |\Psi|$.
  - Compute $H(m)$ and check if $k = m^{H(m)}$.
  - Hence, $\sqrt{n} \geq m$. Also $\phi \in SAT$ iff $\Psi \in SAT$

- Hence SAT_H is not NP-complete, as $P \neq NP$.

Ladner's theorem: Properties of H

- Theorem. There's a function $H: N \rightarrow N$ such that

  1. $H(m)$ is computable from m in $O(m^3)$ time.

  2. If $SAT_H \in P$ then $H(m) \leq C$ (a constant).

  3. If $SAT_H \notin P$ then $H(m) \rightarrow \infty$ with m.

- $SAT_H = \{\Psi 0 1^{m^{H(m)}} : \Psi \in SAT$ and $|\Psi| = m\}$

Construction of H

- Observation. The value of $H(m)$ determines membership in SAT_H of strings whose length is $\geq m$.

- Therefore, it is OK to define $H(m)$ based on strings in SAT_H whose lengths are $< m$ (say, $\log m$).

Construction of H

- Observation. The value of $H(m)$ determines membership in SAT_H of strings whose length is $\geq m$.
- Therefore, it is OK to define $H(m)$ based on strings in SAT_H whose lengths are $< m$ (say, $\log m$).
- Think of computing $H(m)$ sequentially: Compute $H(1)$, $H(2),\ldots,H(m-1)$. Just before computing $H(m)$, find $SAT_H \cap \{0,1\}^{\log m}$.

Construction of H

- Observation. The value of $H(m)$ determines membership in SAT_H of strings whose length is $\geq m$.

- Therefore, it is OK to define $H(m)$ based on strings in SAT_H whose lengths are $< m$ (say, $\log m$).

- Construction. $H(m)$ is the smallest $k < \log \log m$ s.t.
  1. M_k decides membership of **all** length up to $\log m$ strings x in SAT_H within $k.|x|^k$ time.
  2. If no such k exists then $H(m) = \log \log m$.

Construction of H

- Observation. The value of $H(m)$ determines membership in SAT_H of strings whose length is $\geq m$.

- Therefore, it is OK to define $H(m)$ based on strings in SAT_H whose lengths are $< m$ (say, $\log m$).

- Homework. Prove that $H(m)$ is computable from m in $O(m^3)$ time.

Construction of H

- Claim. If $SAT_H \in P$ then $H(m) \leq C$ (a constant).
- Proof. There is a poly-time M that decides membership of every x in SAT_H within $c.|x|^c$ time.

Construction of H

- Claim. If $SAT_H \in P$ then $H(m) \leq C$ (a constant).
- Proof. There is a poly-time M that decides membership of every x in SAT_H within $c.|x|^c$ time.

- As M can be represented by infinitely many strings, there's an $\alpha \geq c$ s.t. $M = M_\alpha$ decides membership of every x in SAT_H within $\alpha.|x|^\alpha$ time.

- So, for every m satisfying $\alpha < \log \log m$, $H(m) \leq \alpha$.

Construction of H

- Claim. If $H(m) \leq C$ (a constant) for infinitely many m, then $SAT_H \in P$.
- Proof. There's a $k \leq C$ s.t. $H(m) = k$ for infinitely many m.

Construction of H

- Claim. If $H(m) \leq C$ (a constant) for infinitely many m, then $SAT_H \in P$.
- Proof. There's a $k \leq C$ s.t. $H(m) = k$ for infinitely many m.

- Pick any $x \in \{0,1\}^*$. Think of a large enough m s.t. $|x| \leq \log m$ and $H(m) = k$.

Construction of H

- Claim. If $H(m) \leq C$ (a constant) for infinitely many m, then $SAT_H \in P$.
- Proof. There's a $k \leq C$ s.t. $H(m) = k$ for infinitely many m.

- Pick any $x \in \{0,1\}^*$. Think of a large enough m s.t. $|x| \leq \log m$ and $H(m) = k$.

- This means x is correctly decided by M_k in $k.|x|^k$ time. So, M_k is a poly-time machine deciding SAT_H.

Natural NP-intermediate problems ??

- Integer factoring
- Approximate shortest vector in a lattice
- Minimum Circuit Size Problem

  ("*Multi-output MCSP is NP-hard*", Ilango, Loff & Oliveira 2020)

- Graph isomorphism

  ("*GI in QuasiP time*", Babai 2015)