Computational Complexity

Lecture 9
More of the Polynomial Hierarchy
Alternation
PH is in terms of verification
PH is in terms of verification

Recall $\Sigma_k^p$
PH is in terms of verification

Recall $\Sigma_k^p$

Languages $L = \{x| \exists w_1 \forall w_2 ... Q w_k \ F(x;w_1,w_2,...,w_k)\}$, where $F$ in $P$
PH is in terms of verification

Recall $\Sigma_k^P$

Languages $L = \{x | \exists w_1 \forall w_2 \ldots Q w_k F(x; w_1, w_2, \ldots, w_k)\}$, where $F$ in $P$

Consider deterministic polynomial time machine $M$ for $F$, with $k$ read-once tapes for the certificates
PH is in terms of verification

Recall $\Sigma_k^P$

Languages $L = \{x | \exists w_1 \forall w_2 \ldots Q w_k F(x;w_1,w_2,\ldots,w_k)\}$, where $F$ in $P$

Consider deterministic polynomial time machine $M$ for $F$, with $k$ read-once tapes for the certificates

Tapes read one after the other
PH is in terms of verification

Recall $\Sigma_k^P$

Languages $L = \{x | \exists w_1 \forall w_2 \ldots Q w_k F(x; w_1, w_2, \ldots, w_k)\}$, where $F$ in $P$

Consider deterministic polynomial time machine $M$ for $F$, with $k$ read-once tapes for the certificates

Tapes read one after the other

$x$ in $L$ if $\exists w_1 \forall w_2 \ldots Q w_k$ such that $M(x; w_1, w_2, \ldots, w_k)$ accepts
PH is in terms of verification

Recall $\Sigma_k^P$

Languages $L = \{x| \exists w_1 \forall w_2 ... Q w_k F(x;w_1,w_2,..,w_k)\}$, where $F$ in $P$

Consider deterministic polynomial time machine $M$ for $F$, with $k$ read-once tapes for the certificates

- Tapes read one after the other

$x$ in $L$ if $\exists w_1 \forall w_2 ... Q w_k$ such that $M(x;w_1,w_2,..,w_k)$ accepts

Plan: Formulate in terms of a non-deterministic TM (with no certificates)
Verification →
Non-determinism
Verification → Non-determinism
Verification → Non-determinism
Verification →
Non-determinism

Read from Tape 1
Verification →
Non-determinism

Read from Tape 1
Read from Tape 1
Verification →
Non-determinism

Read from Tape 1
Read from Tape 1
Read from Tape 1
Verification →
Non-determinism

Read from Tape 1
Read from Tape 1
Read from Tape 2
Verification →
Non-determinism

Read from Tape 1
Read from Tape 1
Read from Tape 2
Verification → Non-determinism

Read from Tape 1

Read from Tape 1

Read from Tape 2

Guess 0
Verification → Non-determinism

Read from Tape 1
Read from Tape 1
Read from Tape 2

 Guess 0
Guess 1
Verification → Non-determinism
Verification → Non-determinism

- Read from Tape 1
- Read from Tape 1
- Read from Tape 2
- Guess 0
- Guess 1
- Guess 0
- Guess 1
- Guess 0
- Guess 1
- Guess 0
- Guess 1
Verification → Non-determinism

Read from Tape 1
Read from Tape 1
Read from Tape 2

Guess 0
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1
Verification → Non-determinism

Read from Tape 1

Read from Tape 1

Read from Tape 2

Guess 0

Guess 1

Guess 0

Guess 1

Guess 0

Guess 1

Guess 0

Guess 1
Verification →
Non-determinism
Verification →

Non-determinism

- Read from Tape 1
- Read from Tape 1
- Read from Tape 2

- Guess 0
- Guess 1
- Guess 0
- Guess 1
- Guess 0
- Guess 1
- Guess 0
Verification → Non-determinism

∃ \in W_1

∀ \in W_2

Read from Tape 1

Guess 0

Guess 1

Read from Tape 1

Guess 0

Guess 1

Read from Tape 2

Guess 0

Guess 1
Verification \rightarrow Non-determinism

∃w_1

∀w_2

Guess 0
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1
Verification → Non-determinism

∃ \forall
Read from Tape 1
Read from Tape 1
Read from Tape 2

Guess 0
Guess 0
Guess 1
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1

Guess 0
Guess 1
ATM
ATM

Alternating Turing Machine
ATM

- Alternating Turing Machine
- At each step, execution can fork into two
ATM

- Alternating Turing Machine
- At each step, execution can fork into two
- Exactly like an NTM or co-NTM
ATM

- Alternating Turing Machine
  - At each step, execution can fork into two
  - Exactly like an NTM or co-NTM
  - Accepting rule is more complex
ATM

- Alternating Turing Machine
  - At each step, execution can fork into two
    - Exactly like an NTM or co-NTM
  - Accepting rule is more complex
    - Like in the game tree for QBF
ATM
ATM

Two kinds of configurations: \( \exists \) and \( \forall \)
ATM

- Two kinds of configurations: \( \exists \) and \( \forall \)
- Depending on the state
ATM

- Two kinds of configurations: $\exists$ and $\forall$
- Depending on the state
- A $\exists$ configuration is accepting if either child is accepting
ATM

- Two kinds of configurations: $\exists$ and $\forall$
  - Depending on the state
  - A $\exists$ configuration is accepting if either child is accepting
  - A $\forall$ configuration is accepting only if both children are accepting
Verification → Non-determinism

∃ \exists w_1

∀ \forall w_2

Read from Tape 1

Read from Tape 1

Read from Tape 2

Guess 0

Guess 1

Guess 0

Guess 1

Guess 0

Guess 1
Verification → Non-determinism

Given a verifier for \( L \) using \( k \) certificate tapes, can build an ATM for \( L \) with at most \( k \) alternations.
Verification → Non-determinism

Given a verifier for L using k certificate tapes, can build an ATM for L with at most k alternations

Non-deterministically guesses tape contents and runs verifier
Verification ← Non-determinism

∃w₁

∀w₂

Read from Tape 1

Read from Tape 2

Guess 0

Guess 1

Guess 0

Guess 1

Guess 0

Guess 1
Verification ← Non-determinism

Given ATM for L with at most k alternations, can build a verifier (using k certificate tapes)
Verification \leftarrow Non-determinism

Given ATM for L with at most k alternations, can build a verifier (using k certificate tapes)

Same time/space requirements (in terms of |x|)
Verification ←
Non-determinism

- Given ATM for L with at most k alternations, can build a verifier (using k certificate tapes)
- Same time/space requirements (in terms of |x|)
- |w_i| = #choices

<table>
<thead>
<tr>
<th>∃</th>
<th>∃</th>
<th>∀</th>
<th>∀</th>
<th>∀</th>
<th>∀</th>
<th>∀</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guess 0</td>
<td>Guess 1</td>
<td>Guess 0</td>
<td>Guess 1</td>
<td>Guess 0</td>
<td>Guess 1</td>
<td>Guess 0</td>
</tr>
</tbody>
</table>

Read from Tape 1
Read from Tape 1
Read from Tape 2
Time, Space, Alternations
Time, Space, Alternations

- Complexity measures
Time, Space, Alternations

- Complexity measures
  - Time: Maximum number of steps in any thread
Time, Space, Alternations

- Complexity measures
  - Time: Maximum number of steps in any thread
  - Space: Maximum space in any configuration reached
Time, Space, Alternations

- Complexity measures
  - Time: Maximum number of steps in any thread
  - Space: Maximum space in any configuration reached
  - Alternations: Maximum number of quantifier switches in any thread
ATIME
ATIME

$\Sigma_k\text{TIME}, \Pi_k\text{TIME}$
ATIME

$\Sigma_k \text{TIME}, \Pi_k \text{TIME}$

$\Sigma_k \text{TIME}(T)$: languages decided by ATMs with at most $k$ alternations starting with $\exists$, in time $T(n)$
ATIME

\[ \Sigma_k \text{TIME}, \Pi_k \text{TIME} \]

\[ \Sigma_k \text{TIME}(T) : \text{languages decided by ATMs with at most } k \text{ alternations starting with } \exists, \text{ in time } T(n) \]

\[ \Sigma_k \text{TIME}(\text{poly}) = \Sigma_k^p \]
ATIME

\( \Sigma_k \text{TIME, } \Pi_k \text{TIME} \)

\( \Sigma_k \text{TIME}(T) \): languages decided by ATMs with at most \( k \) alternations starting with \( \exists \), in time \( T(n) \)

\( \Sigma_k \text{TIME}(\text{poly}) = \Sigma_k^P \)

\( \text{Latter being exactly the certificate version} \)
ATIME

\[ \Sigma_k \text{TIME}, \Pi_k \text{TIME} \]

\[ \Sigma_k \text{TIME}(T): \text{languages decided by ATMs with at most } k \text{ alternations starting with } \exists, \text{ in time } T(n) \]

\[ \Sigma_k \text{TIME(poly)} = \Sigma_k^p \]

\[ \text{Latter being exactly the certificate version} \]

ATIME
ATIME

\( \Sigma_k^{\text{TIME}}, \Pi_k^{\text{TIME}} \)

\( \Sigma_k^{\text{TIME}}(T) \): languages decided by ATMs with at most \( k \) alternations starting with \( \exists \), in time \( T(n) \)

\( \Sigma_k^{\text{TIME}}(\text{poly}) = \Sigma_k^{\text{P}} \)

\( \text{Latter being exactly the certificate version} \)

ATIME

\( \text{ATIME}(T) \): languages decided by ATMs in time \( T(n) \)
ATIME vs. DSPACE
ATIME vs. DSPACE

\[ \text{ATIME}(T) \subseteq \text{DSPACE}(T^2) \]
ATIME vs. DSPACE

\[ \text{ATIME}(T) \subseteq \text{DSPACE}(T^2) \]

\[ \text{c.f. NTIME}(T) \subseteq \text{DSPACE}(T) \]
ATIME vs. DSPACE

- ATIME(T) ⊆ DSPACE(T^2)
- c.f. NTIME(T) ⊆ DSPACE(T)
- AP ⊆ PSPACE
ATIME vs. DSPACE

- $\text{ATIME}(T) \subseteq \text{DSPACE}(T^2)$
- c.f. $\text{NTIME}(T) \subseteq \text{DSPACE}(T)$
- $\text{AP} \subseteq \text{PSPACE}$
- But $\text{PSPACE} \subseteq \text{AP}$
ATIME vs. DSPACE

- ATIME(T) ⊆ DSPACE(T^2)
- c.f. NTIME(T) ⊆ DSPACE(T)
- AP ⊆ PSPACE
- But PSPACE ⊆ AP
- TQBF in AP (why?)
ATIME vs. DSPACE

- ATIME(T) ⊆ DSPACE(T^2)
- c.f. NTIME(T) ⊆ DSPACE(T)
- AP ⊆ PSPACE
- But PSPACE ⊆ AP
- TQBF in AP (why?)
- AP = PSPACE
ATIME(T) ⊆ DSPACE(T^2)
ATIME(T) ⊆ DSPACE(T^2)

Evaluate if the start configuration is accepting, recursively
ATIME(T) ⊆ DSPACE(T^2)

- Evaluate if the start configuration is accepting, recursively
- A ∃ configuration is accepting if any child is, and
  a ∀ configuration is accepting if all children are
\textbf{ATIME}(T) \subseteq \textbf{DSPACE}(T^2)

- Evaluate if the start configuration is accepting, recursively
  - A \exists configuration is accepting if any child is, and
  - a \forall configuration is accepting if all children are

- Space needed: depth \times size of configuration
$\text{ATIME}(T) \subseteq \text{DSPACE}(T^2)$

- Evaluate if the start configuration is accepting, recursively
  
  - $A \exists$ configuration is accepting if any child is, and
  - $A \forall$ configuration is accepting if all children are

- Space needed: depth $\times$ size of configuration

- Depth = $\#$ alternations = $O(T)$. Also, size of configuration = $O(T)$ as any thread runs for time $O(T)$
ATIME(T) ⊆ DSPACE(T^2)

Evaluate if the start configuration is accepting, recursively

- A ∃ configuration is accepting if any child is, and
- a ∀ configuration is accepting if all children are

Space needed: depth x size of configuration

- Depth = # alternations = O(T). Also, size of configuration = O(T) as any thread runs for time O(T)

O(T^2)
ASPACE vs. DTIME
ASPACE vs. DTIME

ASPACE(S) = DTIME(2^{O(S)})
ASPACE vs. DTIME

\[ \text{ASPACE}(S) = \text{DTIME}(2^{O(S)}) \]

Recall, already seen \( \text{NSPACE}(S) \subseteq \text{DTIME}(2^{O(S)}) \)
ASPACE vs. DTIME

\[ \text{ASPACE}(S) = \text{DTIME}(2^{O(S)}) \]

Recall, already seen \( \text{NSPACE}(S) \subseteq \text{DTIME}(2^{O(S)}) \)

Run poly-time connectivity check in a configuration graph of size at most \( 2^{O(S)} \)
ASPACE vs. DTIME

\[ \text{ASPACE}(S) = \text{DTIME}(2^{O(S)}) \]

- Recall, already seen \( \text{NSPACE}(S) \subseteq \text{DTIME}(2^{O(S)}) \)

- Run poly-time connectivity check in a configuration graph of size at most \( 2^{O(S)} \)

- Instead of connectivity, can recursively label all accepting nodes (2 lookups per node: in \( \text{poly}(S) \) time). So \( \text{ASPACE}(S) \subseteq \text{DTIME}(2^{O(S)}) \)
ASPACE vs. DTIME

- $\text{ASPACE}(S) = \text{DTIME}(2^{O(S)})$

  - Recall, already seen $\text{NSPACE}(S) \subseteq \text{DTIME}(2^{O(S)})$

  - Run poly-time connectivity check in a configuration graph of size at most $2^{O(S)}$

  - Instead of connectivity, can recursively label all accepting nodes (2 lookups per node: in $\text{poly}(S)$ time). So $\text{ASPACE}(S) \subseteq \text{DTIME}(2^{O(S)})$

- To show $\text{DTIME}(2^{O(S)}) \subseteq \text{ASPACE}(S)$
\text{DTIME}(2^{O(S)}) \subseteq \text{ASPACE}(S)
DTIME($2^{O(S)}$) ⊆ ASPACE(S)

To decide, is configuration after t steps accepting
DTIME($2^{O(S)}$) ⊆ ASPACE(S)

To decide, is configuration after $t$ steps accepting

Assume configuration starts with $\alpha$ (first tape cell blank, head on it, unique accept state) iff an accept configuration
DTIME($2^{O(S)}$) ⊆ ASPACE(S)

To decide, is configuration after $t$ steps accepting

Assume configuration starts with $\alpha$ (first tape cell blank, head on it, unique accept state) iff an accept configuration

Once there, stays there
DTIME\left(2^{O(S)}\right) \subseteq \text{ASPACE}(S)

- To decide, is configuration after \( t \) steps accepting
  - Assume configuration starts with \( \alpha \) (first tape cell blank, head on it, unique accept state) iff an accept configuration
    - Once there, stays there
  - Is first cell of config after \( t \) steps \( \alpha \)
DTIME\(\left(2^{O(S)}\right) \subseteq \text{ASPACE}(S)\)

To decide, is configuration after \(t\) steps accepting

Assume configuration starts with \(\alpha\) (first tape cell blank, head on it, unique accept state) iff an accept configuration

Once there, stays there

Is first cell of config after \(t\) steps \(\alpha\)

\(C(i,j,x)\) : if after \(i\) steps, \(j^{\text{th}}\) cell of config is \(x\)
\textbf{DTIME}(2^{O(S)}) \subseteq \text{ASPACE}(S)

To decide, is configuration after \( t \) steps accepting

Assume configuration starts with \( \alpha \) (first tape cell blank, head on it, unique accept state) iff an accept configuration

Once there, stays there

Is first cell of config after \( t \) steps \( \alpha \)

\( C(i,j,x) : \text{if after } i \text{ steps, } j^{\text{th}} \text{ cell of config is } x \)

Need to check \( C(t,1,\alpha) \)
ATM for TM simulation
ATM for TM simulation

\[ C(i, j, x) : \text{if after } i \text{ steps, } j^{th} \text{ cell of config is } x \]
ATM for TM simulation

$C(i,j,x)$: if after $i$ steps, $j^{th}$ cell of config is $x$

Recall reduction in Cook’s theorem
ATM for TM simulation

\[ C(i,j,x) : \text{if after } i \text{ steps, } j^{\text{th}} \text{ cell of config is } x \]

- Recall reduction in Cook’s theorem

- If \( C(i-1,j-1,a), C(i-1,j,b), C(i-1,j+1,c) \) then \( C(i,j,x) \) iff \( x=F(a,b,c) \)
ATM for TM simulation

- \( C(i,j,x) \): if after \( i \) steps, \( j^{th} \) cell of config is \( x \)

- Recall reduction in Cook’s theorem

  - If \( C(i-1,j-1,a), C(i-1,j,b), C(i-1,j+1,c) \) then \( C(i,j,x) \) iff \( x=F(a,b,c) \)

- \( C(i,j,x) \): \( \exists a,b,c \) st \( x=F(a,b,c) \) and \( C(i-1,j-1,a), C(i-1,j,b), C(i-1,j+1,c) \)
ATM for TM simulation

- $C(i,j,x)$: if after $i$ steps, $j^{th}$ cell of config is $x$

- Recall reduction in Cook’s theorem
  - If $C(i-1,j-1,a)$, $C(i-1,j,b)$, $C(i-1,j+1,c)$ then $C(i,j,x)$ iff $x=F(a,b,c)$
  - $C(i,j,x)$: $\exists a,b,c \text{ st } x=F(a,b,c)$ and $C(i-1,j-1,a)$, $C(i-1,j,b)$, $C(i-1,j+1,c)$

- Base case: $C(0,j,x)$ easy to check from input
ATM for TM simulation

- $C(i,j,x)$: if after $i$ steps, $j^{th}$ cell of config is $x$

- Recall reduction in Cook’s theorem

- If $C(i-1,j-1,a)$, $C(i-1,j,b)$, $C(i-1,j+1,c)$ then $C(i,j,x)$ iff $x=F(a,b,c)$

- $C(i,j,x)$: $\exists a,b,c$ st $x=F(a,b,c)$ and $C(i-1,j-1,a)$, $C(i-1,j,b)$, $C(i-1,j+1,c)$

- Base case: $C(0,j,x)$ easy to check from input

- Naive recursion: Extra $O(S)$ space to store $i,j$ at each level for $2^{O(S)}$ levels!
ATM for TM simulation
ATM for TM simulation

- ATM to check if $C(i,j,x)$
ATM for TM simulation

- ATM to check if $C(i,j,x)$

- $C(i,j,x): \exists a, b, c \text{ st } x = F(a, b, c) \text{ and } C(i-1,j-1,a), C(i-1,j,b), C(i-1,j+1,c)$
ATM for TM simulation

- ATM to check if $C(i,j,x)$

- $C(i,j,x): \exists a,b,c \text{ s.t. } x=F(a,b,c) \text{ and } C(i-1,j-1,a), C(i-1,j,b), C(i-1,j+1,c)$

- Tail-recursion in parallel forks
ATM for TM simulation

- ATM to check if $C(i,j,x)$

- $C(i,j,x): \exists a,b,c \text{ s.t. } x=F(a,b,c) \text{ and } C(i-1,j-1,a), C(i-1,j,b), C(i-1,j+1,c)$

- **Tail-recursion in parallel forks**

  - Check $x=F(a,b,c)$; then enter universal state, and non-deterministically choose one of the three conditions to check
ATM for TM simulation

- ATM to check if $C(i,j,x)$

- $C(i,j,x): \exists a, b, c \text{ st } x=F(a,b,c) \text{ and } C(i-1,j-1,a), C(i-1,j,b), C(i-1,j+1,c)$

- Tail-recursion in parallel forks

  - Check $x=F(a,b,c)$; then enter universal state, and non-deterministically choose one of the three conditions to check

  - Overwrite $C(i,j,x)$ with $C(i-1,...)$ and reuse space
ATM for TM simulation

- ATM to check if \( C(i,j,x) \)

- \( C(i,j,x): \exists a,b,c \text{ st } x=F(a,b,c) \text{ and } C(i-1,j-1,a), C(i-1,j,b), C(i-1,j+1,c) \)

- **Tail-recursion in parallel forks**
  - Check \( x=F(a,b,c) \); then enter universal state, and non-deterministically choose one of the three conditions to check
  - Overwrite \( C(i,j,x) \) with \( C(i-1,...) \) and reuse space
  - Stay within the same \( O(S) \) space at each level!
ATM for TM simulation

- ATM to check if $C(i,j,x)$

- $C(i,j,x): \exists a,b,c \text{ st } x = F(a,b,c)$ and $C(i-1,j-1,a)$, $C(i-1,j,b)$, $C(i-1,j+1,c)$

- **Tail-recursion in parallel forks**
  - Check $x = F(a,b,c)$; then enter universal state, and non-deterministically choose one of the three conditions to check
  - Overwrite $C(i,j,x)$ with $C(i-1,...)$ and reuse space
  - Stay within the same $O(S)$ space at each level!

*Gets the AND check for free. No need to use a stack.*
ASPACE vs. DTIME
ASPACE vs. DTIME

\[ \text{ASPACE}(S) = \text{DTIME}(2^{O(S)}) \]
ASPACE vs. DTIME

- $\text{ASPACE}(S) = \text{DTIME}(2^{O(S)})$
- $\text{APSPACE} = \text{EXP}$
ASPACE vs. DTIME

- $\text{ASPACE}(S) = \text{DTIME}(2^{O(S)})$
- $\text{APSPACE} = \text{EXP}$
- $\text{AL} = \text{P}$
Zoo
Zoo
DTISP(T,S)
DTISP(T,S)

Theorem: NTIME(n) ∉ DTISP(n^{1+\varepsilon},n^{\delta}) for some \varepsilon, \delta > 0
**DTISP(T,S)**

- **Theorem**: $\text{NTIME}(n) \not\subseteq \text{DTISP}(n^{1+\varepsilon}, n^{\delta})$ for some $\varepsilon, \delta > 0$

- In particular, cannot solve SAT in linear time **and** poly-logarithmic space
Theorem: NTIME(n) ∉ DTISP(n^{1+\epsilon},n^\delta) for some \( \epsilon, \delta > 0 \)

In particular, cannot solve SAT in linear time and poly-logarithmic space.
Theorem: $\text{NTIME}(n) \not\subset \text{DTISP}(n^{1+\varepsilon}, n^\delta)$ for some $\varepsilon, \delta > 0$

In particular, cannot solve SAT in linear time and poly-logarithmic space

Commonly Believed: can’t solve in less than exponential time or with less than linear space
Theorem: \( \text{NTIME}(n) \not\subset \text{DTISP}(n^{1+\varepsilon},n^{\delta}) \) for some \( \varepsilon, \delta > 0 \)

In particular, cannot solve SAT in linear time and poly-logarithmic space

Commonly Believed: can’t solve in less than exponential time or with less than linear space

Follows (after careful choice of parameters) from A tighter variant of Cook-Levin reduction, can reduce NTIME(n) problems to SAT in \((n \log^c(n), \log^c(n))\) time-space
Theorem: $\text{NTIME}(n) \not\subset \text{DTISP}(n^{1+\varepsilon}, n^{\delta})$ for some $\varepsilon, \delta > 0$

In particular, cannot solve SAT in linear time and poly-logarithmic space.

Commonly Believed: can’t solve in less than exponential time or with less than linear space.

Follows (after careful choice of parameters) from

$\text{DTISP}(T,S) \subseteq \Sigma_2 \text{TIME}(T^{1/2} \cdot S)$
DTISP(T,S)

Theorem: \( \text{NTIME}(n) \not\subset \text{DTISP}(n^{1+\varepsilon}, n^\delta) \) for some \( \varepsilon, \delta > 0 \)

In particular, cannot solve SAT in linear time and poly-logarithmic space.

Commonly Believed: can’t solve in less than exponential time or with less than linear space.

Follows (after careful choice of parameters) from

\[ \text{DTISP}(T,S) \subseteq \Sigma_2 \text{TIME}(T^{1/2} S) \]
Theorem: $\text{NTIME}(n) \not\subset \text{DTISP}(n^{1+\varepsilon}, n^\delta)$ for some $\varepsilon, \delta > 0$

In particular, cannot solve SAT in linear time and poly-logarithmic space.

Commonly Believed: can’t solve in less than exponential time or with less than linear space.

Follows (after careful choice of parameters) from:

- $\text{DTISP}(T,S) \subseteq \Sigma_2 \text{TIME}(T^{1/2} S)$

- $\text{NTIME}(n) \subseteq \text{DTIME}(n^{1+\varepsilon}) \Rightarrow \Sigma_2 \text{TIME}(T) \subseteq \text{NTIME}(T^{1+\varepsilon})$ (Why?)

A tighter variant of Cook-Levin reduction, can reduce NTIME(n) problems to SAT in $(n \log^c(n), \log^c(n))$ time-space.
Theorem: $\text{NTIME}(n) \not\subset \text{DTISP}(n^{1+\varepsilon}, n^\delta)$ for some $\varepsilon, \delta > 0$

In particular, cannot solve SAT in linear time and poly-logarithmic space.

Commonly Believed: can’t solve in less than exponential time or with less than linear space.

Follows (after careful choice of parameters) from

1. $\text{DTISP}(T,S) \subseteq \Sigma_2\text{TIME}(T^{1/2} S)$
2. $\text{NTIME}(n) \subseteq \text{DTIME}(n^{1+\varepsilon}) \Rightarrow \Sigma_2\text{TIME}(T) \subseteq \text{NTIME}(T^{1+\varepsilon})$ (Why?)
3. $\text{NTIME}(n) \subseteq \text{DTISP}(n^{1+\varepsilon}, n^\delta) \Rightarrow \text{NTIME}(n^t) \subseteq \text{NTIME}(n^{t(1/2+\varepsilon')})$!
Today
Today

- ATM to define levels of PH
Today

- ATM to define levels of PH
- ATIME and ASPACE
Today

- ATM to define levels of PH
- ATIME and ASPACE
- AP = PSPACE and APSPACE = EXP
Today

- ATM to define levels of PH
- ATIME and ASPACE
  - AP = PSPACE and APSPACE = EXP
- Using $\Sigma_2$TIME for a DTISP lower-bound