بسم الله الرحمن الرحیم

نظریه علوم کامپیوتر

نظریه علوم کامپیوتر - بهار ۱۴۰۰-۱۴۰۱ - جلسه شانزدهم: سلسله‌مراتب زمان

Theory of computation - 002 - S16 - Time hierarchy

Contents

Last time:

- Log-space reducibility
- L = NL? question
- $PATH$ is NL-complete
- $\overline{2SAT}$ is NL-complete
- NL = coNL

Today: (Sipser §9.1)

- Time and Space Hierarchy Theorems

Review: Major Complexity Classes

$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$

Review: Major Complexity Classes

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$$

$\neq$

Today

Review: Major Complexity Classes

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$$

$$\underbrace{NL \quad \neq \quad PSPACE}_{\text{Today}}$$

The time and space hierarchy theorems show that
if a TM is given more time (or space) then it can do more.*

* certain restrictions apply.

Review: Major Complexity Classes

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$$

$$\underbrace{NL \neq PSPACE}_{\text{Today}}$$

The time and space hierarchy theorems show that
if a TM is given more time (or space) then it can do more.*
* certain restrictions apply.

For example:

$\mathrm{TIME}(n^2) \subsetneq \mathrm{TIME}(n^3)$ [$\subsetneq$ means proper subset]

Review: Major Complexity Classes

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$$

$\neq$

Today

The time and space hierarchy theorems show that
if a TM is given more time (or space) then it can do more.*
* certain restrictions apply.

For example:

$\text{TIME}(n^2) \subsetneq \text{TIME}(n^3)$ [$\subsetneq$ means proper subset]

$\text{SPACE}(n^2) \subsetneq \text{SPACE}(n^3)$

Review: Major Complexity Classes

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$$

$$\underbrace{NL \neq PSPACE}_{\text{Today}}$$

The time and space hierarchy theorems show that
if a TM is given more time (or space) then it can do more.*
* certain restrictions apply.

For example:

$\text{TIME}(n^2) \subsetneq \text{TIME}(n^3)$ [$\subsetneq$ means proper subset]

$\text{SPACE}(n^2) \subsetneq \text{SPACE}(n^3)$

Space Hierarchy Theorem (1/2)

Theorem: For any $f: \mathbb{N} \to \mathbb{N}$ (where f satisfies a technical condition)

there is a language A where A requires $O\big(f(n)\big)$ space, i.e,

Space Hierarchy Theorem (1/2)

Theorem: For any $f:\mathbb{N}\to\mathbb{N}$ (where f satisfies a technical condition)

there is a language A where A requires $O\big(f(n)\big)$ space, i.e,

1) A is decidable in $O\big(f(n)\big)$ space, and

Space Hierarchy Theorem (1/2)

Theorem: For any $f:\mathbb{N} \to \mathbb{N}$ (where f satisfies a technical condition)

there is a language A where A requires $O\big(f(n)\big)$ space, i.e,

1) A is decidable in $O\big(f(n)\big)$ space, and

2) A is not decidable in $o\big(f(n)\big)$ space

Space Hierarchy Theorem (1/2)

Theorem: For any $f : \mathbb{N} \to \mathbb{N}$ (where f satisfies a technical condition)

there is a language A where A <u>requires</u> $O\big(f(n)\big)$ space, i.e,

1) A is decidable in $O\big(f(n)\big)$ space, and

2) A is not decidable in $o\big(f(n)\big)$ space

On other words, $\text{SPACE}\big(o(f(n))\big) \subsetneq \text{SPACE}\big(f(n)\big)$

Notation: $\text{SPACE}\big(o(f(n))\big) = \{B \mid \text{ some TM } M \text{ decides } B \text{ in space } o\big(f(n)\big)\}$

Space Hierarchy Theorem (1/2)

Theorem: For any $f: \mathbb{N} \to \mathbb{N}$ (where f satisfies a technical condition)

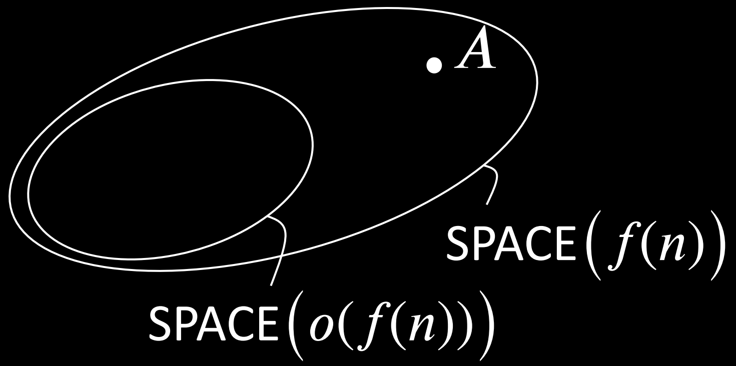
there is a language A where A requires $O\big(f(n)\big)$ space, i.e,

1) A is decidable in $O\big(f(n)\big)$ space, and

2) A is not decidable in $o\big(f(n)\big)$ space

On other words, SPACE$\big(o(f(n))\big) \subsetneq$ SPACE$\big(f(n)\big)$

Notation: SPACE$\big(o(f(n))\big) = \{B \mid$ some TM M decides B in space $o\big(f(n)\big)\}$



Space Hierarchy Theorem (1/2)

Theorem: For any $f : \mathbb{N} \to \mathbb{N}$ (where f satisfies a technical condition)

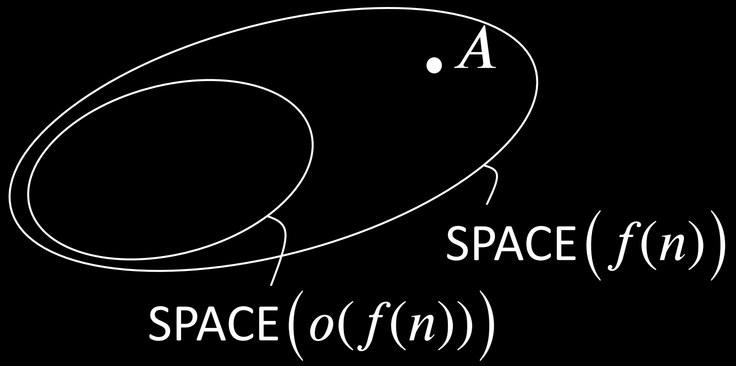
there is a language A where A requires $O\big(f(n)\big)$ space, i.e,

1) A is decidable in $O\big(f(n)\big)$ space, and

2) A is not decidable in $o\big(f(n)\big)$ space

On other words, $\text{SPACE}\big(o(f(n))\big) \subsetneq \text{SPACE}\big(f(n)\big)$

Notation: $\text{SPACE}\big(o(f(n))\big) = \{B \mid \text{ some TM } M \text{ decides } B \text{ in space } o\big(f(n)\big)\}$



Proof outline: (Diagonalization)

Give TM D where

Space Hierarchy Theorem (1/2)

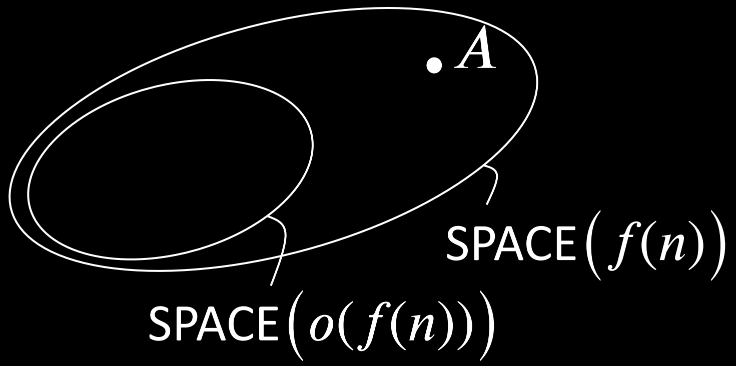
Theorem: For any $f: \mathbb{N} \to \mathbb{N}$ (where f satisfies a technical condition)

there is a language A where A requires $O\big(f(n)\big)$ space, i.e,

1) A is decidable in $O\big(f(n)\big)$ space, and
2) A is not decidable in $o\big(f(n)\big)$ space

On other words, $\text{SPACE}\big(o(f(n))\big) \subsetneq \text{SPACE}\big(f(n)\big)$

Notation: $\text{SPACE}\big(o(f(n))\big) = \{B \mid \text{ some TM } M \text{ decides } B \text{ in space } o\big(f(n)\big)\}$



Proof outline: (Diagonalization)

Give TM D where

1) D runs in $O\big(f(n)\big)$ space

Space Hierarchy Theorem (1/2)

Theorem: For any $f : \mathbb{N} \rightarrow \mathbb{N}$ (where f satisfies a technical condition)

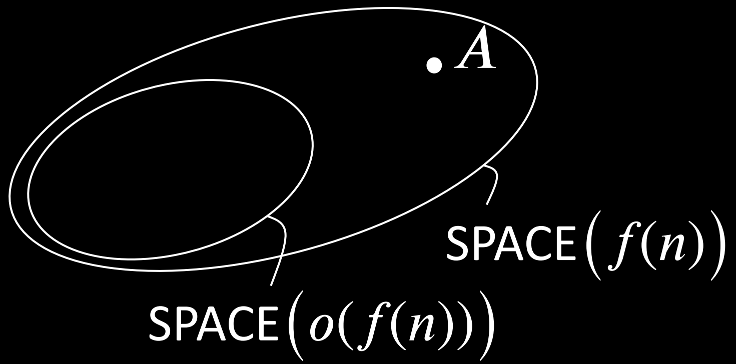
there is a language A where A requires $O(f(n))$ space, i.e,

1) A is decidable in $O(f(n))$ space, and

2) A is not decidable in $o(f(n))$ space

On other words, SPACE$(o(f(n)))$ $\subsetneq$ SPACE$(f(n))$

Notation: SPACE$(o(f(n)))$ $= \{B \mid$ some TM M decides B in space $o(f(n))\}$



Proof outline: (Diagonalization)

Give TM D where

1) D runs in $O(f(n))$ space

2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o(f(n))$ space.

Space Hierarchy Theorem (1/2)

Theorem: For any $f: \mathbb{N} \to \mathbb{N}$ (where f satisfies a technical condition)

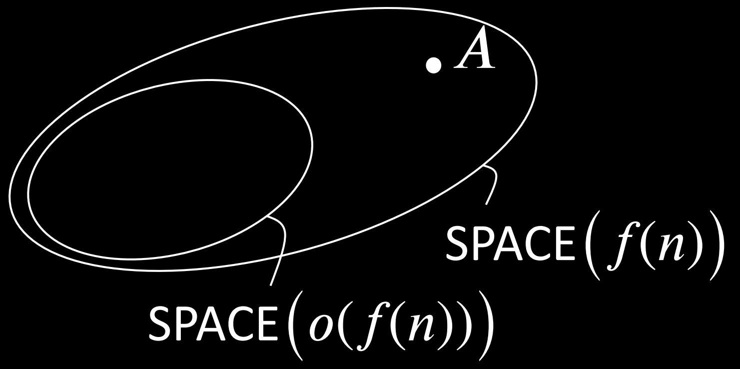
there is a language A where A requires $O\big(f(n)\big)$ space, i.e,

1) A is decidable in $O\big(f(n)\big)$ space, and

2) A is not decidable in $o\big(f(n)\big)$ space

On other words, SPACE$\big(o(f(n))\big) \subsetneq$ SPACE$\big(f(n)\big)$

Notation: SPACE$\big(o(f(n))\big) = \{B \mid$ some TM M decides B in space $o\big(f(n)\big)\}$



Proof outline: (Diagonalization)
Give TM D where

1) D runs in $O\big(f(n)\big)$ space

2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)\big)$ space.

Let $A = L(D)$.

Space Hierarchy Theorem (1/2)

Theorem: For any $f: \mathbb{N} \to \mathbb{N}$ (where f satisfies a technical condition)

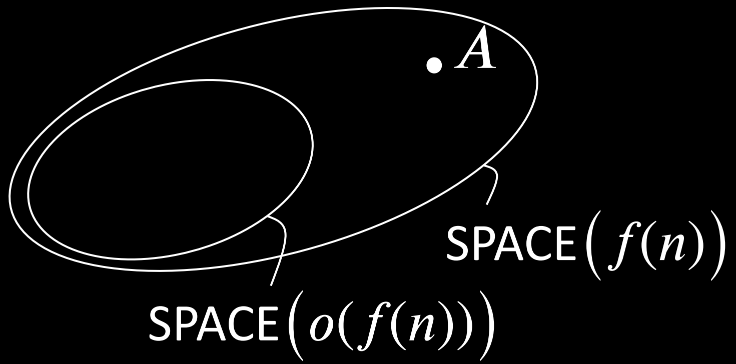
there is a language A where A <u>requires</u> $O\big(f(n)\big)$ space, i.e,

1) A is decidable in $O\big(f(n)\big)$ space, and

2) A is not decidable in $o\big(f(n)\big)$ space

On other words, $\text{SPACE}\big(o(f(n))\big) \subsetneq \text{SPACE}\big(f(n)\big)$

Notation: $\text{SPACE}\big(o(f(n))\big) = \{B \mid \text{ some TM } M \text{ decides } B \text{ in space } o\big(f(n)\big)\}$



Proof outline: (Diagonalization)

Give TM D where

1) D runs in $O\big(f(n)\big)$ space

2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)\big)$ space.

Let $A = L(D)$.

Space Hierarchy Theorem (1/2)

Theorem: For any $f: \mathbb{N} \to \mathbb{N}$ (where f satisfies a technical condition)

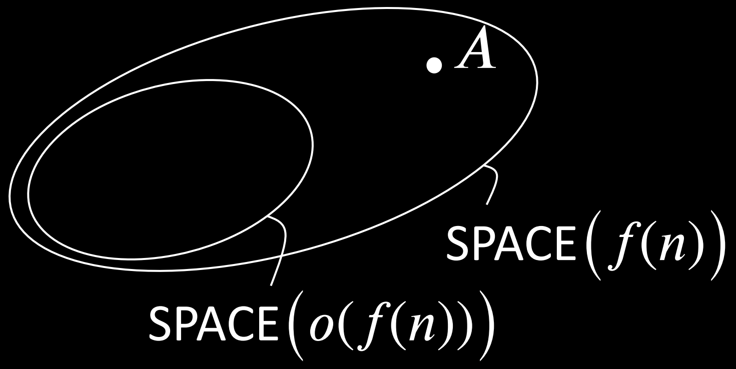
there is a language A where A requires $O\big(f(n)\big)$ space, i.e,

1) A is decidable in $O\big(f(n)\big)$ space, and

2) A is not decidable in $o\big(f(n)\big)$ space

On other words, SPACE$\big(o(f(n))\big) \subsetneq$ SPACE$\big(f(n)\big)$

Notation: SPACE$\big(o(f(n))\big) = \{B \mid$ some TM M decides B in space $o\big(f(n)\big)\}$



Proof outline: (Diagonalization)

Give TM D where

1) D runs in $O\big(f(n)\big)$ space

2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)\big)$ space.

Let $A = L(D)$.

Time Hierarchy Theorem (1/2)

Theorem: For any $f:\mathbb{N} \to \mathbb{N}$ where f is time constructible

there is a language A where A requires $O\big(f(n)\big)$ time, i.e,

Time Hierarchy Theorem (1/2)

Theorem: For any $f:\mathbb{N} \to \mathbb{N}$ where f is time constructible

there is a language A where A requires $O\big(f(n)\big)$ time, i.e,

1) A is decidable in $O\big(f(n)\big)$ time, and
2) A is not decidable in $o\big(f(n)/\log\big(f(n)\big)\big)$ time

Time Hierarchy Theorem (1/2)

Theorem: For any $f:\mathbb{N} \to \mathbb{N}$ where f is time constructible

there is a language A where A requires $O\big(f(n)\big)$ time, i.e,

1) A is decidable in $O\big(f(n)\big)$ time, and
2) A is not decidable in $o\big(f(n)/\log\big(f(n)\big)\big)$ time

On other words, $\text{TIME}\left(o\left(\frac{f(n)}{\log\big(f(n)\big)}\right)\right) \subsetneq \text{TIME}\big(f(n)\big)$

Time Hierarchy Theorem (1/2)

Theorem: For any $f: \mathbb{N} \to \mathbb{N}$ where f is time constructible

there is a language A where A requires $O\big(f(n)\big)$ time, i.e,

1) A is decidable in $O\big(f(n)\big)$ time, and
2) A is not decidable in $o\big(f(n)/\log\big(f(n)\big)\big)$ time

On other words, $\text{TIME}\left(o\left(\dfrac{f(n)}{\log\big(f(n)\big)}\right)\right) \subsetneq \text{TIME}\big(f(n)\big)$

Proof outline: Give TM D where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)/\log\big(f(n)\big)\big)$ time .

Let $A = L(D)$.

Time Hierarchy Theorem (1/2)

Theorem: For any $f:\mathbb{N} \to \mathbb{N}$ where f is time constructible

there is a language A where A requires $O\big(f(n)\big)$ time, i.e,

1) A is decidable in $O\big(f(n)\big)$ time, and
2) A is not decidable in $o\big(f(n)/\log\big(f(n)\big)\big)$ time

On other words, $\text{TIME}\left(o\left(\dfrac{f(n)}{\log\big(f(n)\big)}\right)\right) \subsetneq \text{TIME}\big(f(n)\big)$

Proof outline: Give TM D where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)/\log\big(f(n)\big)\big)$ time .

Let $A = L(D)$.

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\big(f(n)\big)$ but $A \notin$ TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\big(f(n)\big)$ but $A \notin$ TIME$\Big(o\big(f(n)/\log(f(n))\big)\Big)$

$A = L(D)$ where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M

   that runs in $o\big(f(n)/\log(f(n))\big)$ time.

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\big(f(n)\big)$ but $A \notin$ TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$

$A = L(D)$ where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\Big(f(n)/\log\big(f(n)\big)\Big)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\big(f(n)\big)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\big(f(n)\big)$ but $A \notin$ TIME$\Big(o\Big(f(n)/\log\big(f(n)\big)\Big)\Big)$

$A = L(D)$ where

For no M in TIME$\Big(o\Big(f(n)/\log\big(f(n)\big)\Big)\Big)$, L(M) != L(D)

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\Big(f(n)/\log\big(f(n)\big)\Big)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\big(f(n)\big)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in \text{TIME}\big(f(n)\big)$ but $A \notin \text{TIME}\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$

$A = L(D)$ where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)/\log\big(f(n)\big)\big)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\big(f(n)\big)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

For no M in TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$, L(M) != L(D)

Contradiction, M(<M>10^k) runs in o(f/log f)

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\big(f(n)\big)$ but $A \notin$ TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$

$A = L(D)$ where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)/\log\big(f(n)\big)\big)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\big(f(n)\big)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

For no M in TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$, L(M) != L(D)

Contradiction, M(<M>10^k) runs in o(f/log f)
D(<M>1) rejects iff M(<M>10^k) accepts

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\big(f(n)\big)$ but $A \notin$ TIME$\Big(o\Big(f(n)/\log\big(f(n)\big)\Big)\Big)$

$A = L(D)$ where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\Big(f(n)/\log\big(f(n)\big)\Big)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\big(f(n)\big)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

For no M in TIME$\Big(o\Big(f(n)/\log\big(f(n)\big)\Big)\Big)$, L(M) != L(D)

Contradiction, M(<M>10^k) runs in o(f/log f)
D(<M>1) rejects iff M(<M>10^k) accepts

Why do we lose a factor of $\log\big(f(n)\big)$?

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\big(f(n)\big)$ but $A \notin$ TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$

$A = L(D)$ where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)/\log\big(f(n)\big)\big)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\big(f(n)\big)$ steps.
   Accept if M rejects,
   Reject if M accepts or hasn't halted."

For no M in TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$, L(M) != L(D)

Contradiction, M(<M>10^k) runs in o(f/log f)
D(<M>1) rejects iff M(<M>10^k) accepts

Why do we lose a factor of $\log\big(f(n)\big)$?

D must halt within $O\big(f(n)\big)$ time.

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\left(f(n)\right)$ but $A \notin$ TIME$\left(o\left(f(n)/\log\left(f(n)\right)\right)\right)$

$A = L(D)$ where

1) D runs in $O\left(f(n)\right)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\left(f(n)/\log\left(f(n)\right)\right)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\left(f(n)\right)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

For no M in TIME$\left(o\left(f(n)/\log\left(f(n)\right)\right)\right)$, L(M) != L(D)

Contradiction, M(<M>10^k) runs in o(f/log f)
D(<M>1) rejects iff M(<M>10^k) accepts

Why do we lose a factor of $\log\left(f(n)\right)$?

D must halt within $O\left(f(n)\right)$ time.

To do so, D counts the number of steps it uses and stops if the limit is exceeded. The counter has size $\log\left(f(n)\right)$ and is stored on the tape.

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\left(f(n)\right)$ but $A \notin$ TIME$\left(o\left(f(n)/\log\left(f(n)\right)\right)\right)$

$A = L(D)$ where

1) D runs in $O\left(f(n)\right)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\left(f(n)/\log\left(f(n)\right)\right)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\left(f(n)\right)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

For no M in TIME$\left(o\left(f(n)/\log\left(f(n)\right)\right)\right)$, L(M) != L(D)

Contradiction, M(<M>10^k) runs in o(f/log f)
D(<M>1) rejects iff M(<M>10^k) accepts

Why do we lose a factor of $\log\left(f(n)\right)$?

D must halt within $O\left(f(n)\right)$ time.

To do so, D counts the number of steps it uses and stops if the limit is exceeded. The counter has size $\log\left(f(n)\right)$ and is stored on the tape.

It must be kept near the current head location.

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in$ TIME$\big(f(n)\big)$ but $A \notin$ TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$

$A = L(D)$ where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)/\log\big(f(n)\big)\big)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\big(f(n)\big)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

For no M in TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$, L(M) != L(D)

Contradiction, M(<M>10^k) runs in o(f/log f)
D(<M>1) rejects iff M(<M>10^k) accepts

Why do we lose a factor of $\log\big(f(n)\big)$?

D must halt within $O\big(f(n)\big)$ time.

To do so, D counts the number of steps it uses and stops if the limit is exceeded. The counter has size $\log\big(f(n)\big)$ and is stored on the tape.

It must be kept near the current head location.

Cost of moving it adds a $O\Big(\log\big(f(n)\big)\Big)$ overhead factor. So to halt within $O\big(f(n)\big)$ time, D stops when the counter reaches $f(n)/\log\big(f(n)\big)$.

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in \text{TIME}\big(f(n)\big)$ but $A \notin \text{TIME}\Big(o\Big(f(n)/\log\big(f(n)\big)\Big)\Big)$

$A = L(D)$ where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\Big(f(n)/\log\big(f(n)\big)\Big)$ time.

$D =$ "On input w

1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\big(f(n)\big)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

*Note: D can simulate M with a log factor time overhead due to the step counter.

For no M in TIME$\Big(o\Big(f(n)/\log\big(f(n)\big)\Big)\Big)$, L(M) != L(D)

Contradiction, M(<M>10^k) runs in o(f/log f)
D(<M>1) rejects iff M(<M>10^k) accepts

Why do we lose a factor of $\log\big(f(n)\big)$?

D must halt within $O\big(f(n)\big)$ time.

To do so, D counts the number of steps it uses and stops if the limit is exceeded. The counter has size $\log\big(f(n)\big)$ and is stored on the tape.
It must be kept near the current head location.
Cost of moving it adds a $O\Big(\log\big(f(n)\big)\Big)$ overhead factor. So to halt within $O\big(f(n)\big)$ time, D stops when the counter reaches $f(n)/\log\big(f(n)\big)$.

Time Hierarchy Theorem (2/2)

Goal: Exhibit $A \in \text{TIME}\big(f(n)\big)$ but $A \notin \text{TIME}\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$

$A = L(D)$ where

1) D runs in $O\big(f(n)\big)$ time
2) D ensures that $L(D) \neq L(M)$ for every TM M that runs in $o\big(f(n)/\log\big(f(n)\big)\big)$ time.

$D =$ "On input w
1. Compute $f(n)$.
2. If $w \neq \langle M \rangle 10^*$ for some TM M, *reject*.
3. Simulate* M on w for $f(n)/\log\big(f(n)\big)$ steps.

   Accept if M rejects,

   Reject if M accepts or hasn't halted."

*Note: D can simulate M with a log factor time overhead due to the step counter.

For no M in TIME$\Big(o\big(f(n)/\log\big(f(n)\big)\big)\Big)$, L(M) != L(D)

Contradiction, M(<M>10^k) runs in o(f/log f)
D(<M>1) rejects iff M(<M>10^k) accepts

Why do we lose a factor of $\log\big(f(n)\big)$?

D must halt within $O\big(f(n)\big)$ time.

To do so, D counts the number of steps it uses and stops if the limit is exceeded. The counter has size $\log\big(f(n)\big)$ and is stored on the tape.
It must be kept near the current head location.
Cost of moving it adds a $O\Big(\log\big(f(n)\big)\Big)$ overhead factor. So to halt within $O\big(f(n)\big)$ time, D stops when the counter reaches $f(n)/\log\big(f(n)\big)$.

Recap: Separating Complexity Classes

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE$$

$$\underbrace{NL \neq PSPACE}_{\text{Space Hierarchy Theorem}}$$

$NL \subseteq SPACE\left(\log^2 n\right) \subsetneq SPACE(n) \subseteq$

PSPACE

Recap: Separating Complexity Classes

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE}$$

$\mathrm{NL} \neq \mathrm{PSPACE}$

Space Hierarchy Theorem

$\mathrm{NL} \subseteq \mathrm{SPACE}\left(\log^2 n\right) \subsetneq \mathrm{SPACE}(n) \subseteq$

PSPACE

Check-in 21.3

Recap: Separating Complexity Classes

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE}$$

$\mathrm{NL} \neq \mathrm{PSPACE}$

Space Hierarchy Theorem

$\mathrm{NL} \subseteq \mathrm{SPACE}\left(\log^2 n\right) \subsetneq \mathrm{SPACE}(n) \subseteq$
PSPACE

Check-in 21.3

Consider these two famous unsolved questions:

1. Does L = P?
2. Does P = PSPACE?

What do the hierarchy theorems tell us about these questions?

a) Nothing
b) At least one of these has answer "NO"
c) At least one of these has answer "YES"

Exponential Complexity Classes

Defn: EXPTIME = $\bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

EXPSPACE = $\bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Exponential Complexity Classes

Defn: $\text{EXPTIME} = \bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

$\text{EXPSPACE} = \bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

$$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}$$

Exponential Complexity Classes

Defn: $\text{EXPTIME} = \bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

$\text{EXPSPACE} = \bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}$

Space Hierarchy Theorem

Exponential Complexity Classes

Defn: EXPTIME = $\bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

EXPSPACE = $\bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

$\neq$

L $\subseteq$ NL $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ EXPSPACE

$\neq$ $\neq$

Space Hierarchy Theorem

Defn: B is <u>EXPTIME-complete</u> if

1) $B \in$ EXPTIME
2) For all $A \in$ EXPTIME, $A \leq_{\text{P}} B$

Exponential Complexity Classes

Defn: EXPTIME = $\bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

EXPSPACE = $\bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

$\neq$

L $\subseteq$ NL $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ EXPSPACE

$\neq$ $\neq$

Space Hierarchy Theorem

Defn: B is <u>EXPTIME-complete</u> if

1) $B \in$ EXPTIME

2) For all $A \in$ EXPTIME, $A \leq_{\mathrm{P}} B$

Same for <u>EXPSPACE-complete</u>

Exponential Complexity Classes

Defn: $\text{EXPTIME} = \bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

$\text{EXPSPACE} = \bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

$\text{P} \neq \text{EXPTIME}$

$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}$

$\text{NL} \neq \text{PSPACE}$ $\quad$ $\text{PSPACE} \neq \text{EXPSPACE}$

Space Hierarchy Theorem

Defn: B is <u>EXPTIME-complete</u> if

1) $B \in$ EXPTIME
2) For all $A \in$ EXPTIME, $A \leq_{\text{P}} B$

Same for <u>EXPSPACE-complete</u>

Theorem: If B is EXPTIME-complete then $B \notin$ P

Exponential Complexity Classes

Defn: EXPTIME = $\bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

EXPSPACE = $\bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

$$\text{L} \subseteq \text{NL} \subseteq \underbrace{\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME}}_{\text{P} \neq \text{EXPTIME}} \subseteq \text{EXPSPACE}$$

$\text{NL} \neq \text{PSPACE}$, $\text{PSPACE} \neq \text{EXPSPACE}$

Space Hierarchy Theorem

Defn: B is <u>EXPTIME-complete</u> if

1) $B \in$ EXPTIME
2) For all $A \in$ EXPTIME, $A \leq_{\text{P}} B$

Same for <u>EXPSPACE-complete</u>

Theorem: If B is EXPTIME-complete then $B \notin$ P

Theorem: If B is EXPSPACE-complete then $B \notin$ PSPACE (and $B \notin$ P)

Exponential Complexity Classes

Defn: EXPTIME = $\bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

EXPSPACE = $\bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

P $\neq$ EXPTIME

L $\subseteq$ NL $\subseteq$ P $\subseteq$ NP $\subseteq$ PSPACE $\subseteq$ EXPTIME $\subseteq$ EXPSPACE

NL $\neq$ PSPACE, PSPACE $\neq$ EXPSPACE

Space Hierarchy Theorem

Defn: B is <u>EXPTIME-complete</u> if

1) $B \in$ EXPTIME

2) For all $A \in$ EXPTIME, $A \leq_\text{P} B$

Same for <u>EXPSPACE-complete</u>

Theorem: If B is EXPTIME-complete then $B \notin$ P

Theorem: If B is EXPSPACE-complete then $B \notin$ PSPACE (and $B \notin$ P)

intractable

Exponential Complexity Classes

Defn: EXPTIME = $\bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

EXPSPACE = $\bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

$$\overbrace{\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}}$$

$\text{NL} \neq \text{PSPACE}$ (Space Hierarchy Theorem), $\text{P} \neq \text{EXPTIME}$ (Time Hierarchy Theorem), $\text{PSPACE} \neq \text{EXPSPACE}$ (Space Hierarchy Theorem)

Space Hierarchy Theorem

Defn: B is <u>EXPTIME-complete</u> if

1) $B \in$ EXPTIME
2) For all $A \in$ EXPTIME, $A \leq_{\text{P}} B$

Same for <u>EXPSPACE-complete</u>

Theorem: If B is EXPTIME-complete then $B \notin$ P
Theorem: If B is EXPSPACE-complete then $B \notin$ PSPACE (and $B \notin$ P)

$\Big\}$ intractable

Defn: $EQ_{\text{REX}\uparrow} = \{\langle R_1, R_2 \rangle \mid R_1$ and R_2 are equivalent regular expressions with exponentiation$\}$

Exponential Complexity Classes

Defn: EXPTIME = $\bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

EXPSPACE = $\bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

$\neq$

$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}$

$\neq$ $\neq$

Space Hierarchy Theorem

Defn: B is <u>EXPTIME-complete</u> if

1) $B \in$ EXPTIME
2) For all $A \in$ EXPTIME, $A \leq_{\text{P}} B$

Same for <u>EXPSPACE-complete</u>

Theorem: If B is EXPTIME-complete then $B \notin$ P

Theorem: If B is EXPSPACE-complete then $B \notin$ PSPACE (and $B \notin$ P)

$\left.\begin{matrix} \\ \\ \end{matrix}\right\}$ intractable

Defn: $EQ_{\text{REX}\uparrow} = \{\langle R_1, R_2\rangle \mid R_1$ and R_2 are equivalent regular expressions with exponentiation$\}$

Notation: If R is a regular expression write R^k to mean $\overbrace{RR\cdots R}^{k}$ (exponent is written in binary).

Exponential Complexity Classes

Defn: EXPTIME = $\bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

EXPSPACE = $\bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

$$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}$$

Space Hierarchy Theorem

Defn: B is <u>EXPTIME-complete</u> if

1) $B \in$ EXPTIME
2) For all $A \in$ EXPTIME, $A \leq_{\text{P}} B$

Same for <u>EXPSPACE-complete</u>

Theorem: If B is EXPTIME-complete then $B \notin$ P

Theorem: If B is EXPSPACE-complete then $B \notin$ PSPACE (and $B \notin$ P)

intractable

Defn: $EQ_{\text{REX}\uparrow} = \{\langle R_1, R_2 \rangle \mid R_1$ and R_2 are equivalent regular expressions with exponentiation$\}$

Notation: If R is a regular expression write R^k to mean $\overbrace{RR\cdots R}^{k}$ (exponent is written in binary).

Exponential Complexity Classes

Defn: EXPTIME = $\bigcup_k \text{TIME}\left(2^{(n^k)}\right)$

EXPSPACE = $\bigcup_k \text{SPACE}\left(2^{(n^k)}\right)$

Time Hierarchy Theorem

$$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}$$

Space Hierarchy Theorem

Defn: B is <u>EXPTIME-complete</u> if

1) $B \in$ EXPTIME
2) For all $A \in$ EXPTIME, $A \leq_{\text{P}} B$

Same for <u>EXPSPACE-complete</u>

Theorem: If B is EXPTIME-complete then $B \notin$ P

Theorem: If B is EXPSPACE-complete then $B \notin$ PSPACE (and $B \notin$ P)

$\Bigg\}$ intractable

Defn: $EQ_{\text{REX}\uparrow} = \{\langle R_1, R_2\rangle \mid R_1$ and R_2 are equivalent regular expressions with exponentiation$\}$

Notation: If R is a regular expression write R^k to mean $\overbrace{RR\cdots R}^{k}$ (exponent is written in binary).

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: $\mathbf{P}^A = \{B \mid B$ is decidable in polynomial time with an oracle for $A\}$

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: $\mathrm{P}^A = \{B \mid B$ is decidable in polynomial time with an oracle for $A\}$

Thus NP $\subseteq \mathrm{P}^{SAT}$

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a “black box” that can answer queries “is $x \in A$?” for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: $\mathrm{P}^A = \{B \mid B$ is decidable in polynomial time with an oracle for $A\}$

Thus NP $\subseteq \mathrm{P}^{SAT}$

NP $= \mathrm{P}^{SAT}$? Probably No because coNP $\subseteq \mathrm{P}^{SAT}$

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: $\mathrm{P}^A = \{B \mid B$ is decidable in polynomial time with an oracle for $A\}$

Thus NP $\subseteq \mathrm{P}^{SAT}$

NP $= \mathrm{P}^{SAT}$? Probably No because coNP $\subseteq \mathrm{P}^{SAT}$

Defn: $\mathbf{NP}^A = \{B \mid B$ is decidable in nondeterministic polynomial time with an oracle for $A\}$

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: $\mathrm{P}^A = \{B \mid B$ is decidable in polynomial time with an oracle for $A\}$

Thus NP $\subseteq \mathrm{P}^{SAT}$

NP $= \mathrm{P}^{SAT}$? Probably No because coNP $\subseteq \mathrm{P}^{SAT}$

Defn: $\mathbf{NP}^A = \{B \mid B$ is decidable in nondeterministic polynomial time with an oracle for $A\}$

Recall $\text{MIN-FORMULA} = \{\langle\phi\rangle \mid \phi$ is a minimal Boolean formula $\}$

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: $\mathrm{P}^A = \{B \mid B$ is decidable in polynomial time with an oracle for $A\}$

Thus NP $\subseteq \mathrm{P}^{SAT}$

NP $= \mathrm{P}^{SAT}$? Probably No because coNP $\subseteq \mathrm{P}^{SAT}$

Defn: $\mathrm{NP}^A = \{B \mid B$ is decidable in nondeterministic polynomial time with an oracle for $A\}$

Recall $\mathrm{MIN\text{-}FORMULA} = \{\langle\phi\rangle \mid \phi$ is a minimal Boolean formula $\}$

Example: $\overline{\text{MIN–FORMULA}} \in \mathrm{NP}^{SAT}$

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: $\mathrm{P}^A = \{B \mid B$ is decidable in polynomial time with an oracle for $A\}$

Thus NP $\subseteq \mathrm{P}^{SAT}$

NP $= \mathrm{P}^{SAT}$? Probably No because coNP $\subseteq \mathrm{P}^{SAT}$

Defn: $\mathrm{NP}^A = \{B \mid B$ is decidable in nondeterministic polynomial time with an oracle for $A\}$

Recall MIN-FORMULA $= \{\langle\phi\rangle \mid \phi$ is a minimal Boolean formula $\}$

Example: $\overline{\textbf{MIN–FORMULA}} \in \mathbf{NP}^{SAT}$

"On input $\langle\phi\rangle$

1. Guess shorter formula ψ

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: **P**$^A = \{B \mid B$ is decidable in polynomial time with an oracle for $A\}$

Thus NP $\subseteq$ P^{SAT}

NP $=$ P^{SAT}? Probably No because coNP $\subseteq$ P^{SAT}

Defn: **NP**$^A = \{B \mid B$ is decidable in nondeterministic polynomial time with an oracle for $A\}$

Recall $\text{MIN-FORMULA} = \{\langle\phi\rangle \mid \phi$ is a minimal Boolean formula $\}$

Example: $\overline{\textbf{MIN-FORMULA}} \in$ **NP**^{SAT}

"On input $\langle\phi\rangle$

1. Guess shorter formula ψ
2. Use SAT oracle to solve the coNP problem: ϕ and ψ are equivalent

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: **P**$^A = \{B \,|\, B$ is decidable in polynomial time with an oracle for $A\}$

Thus NP $\subseteq$ P^{SAT}

NP $=$ P^{SAT}? Probably No because coNP $\subseteq$ P^{SAT}

Defn: **NP**$^A = \{B \,|\, B$ is decidable in nondeterministic polynomial time with an oracle for $A\}$

Recall MIN-FORMULA $= \{\langle\phi\rangle \,|\, \phi$ is a minimal Boolean formula $\}$

Example: $\overline{\textbf{MIN–FORMULA}} \in$ **NP**^{SAT}

"On input $\langle\phi\rangle$

1. Guess shorter formula ψ
2. Use SAT oracle to solve the coNP problem: ϕ and ψ are equivalent
3. *Accept* if ϕ and ψ are equivalent. *Reject* if not."

Computation with Oracles

Let A be any language.

Defn: A TM M with oracle for A, written M^A, is a TM equipped with a "black box" that can answer queries "is $x \in A$?" for free.

Example: A TM with an oracle for SAT can decide all $B \in$ NP in polynomial time.

Defn: $\mathbf{P}^A = \{B \mid B$ is decidable in polynomial time with an oracle for $A\}$

Thus NP $\subseteq$ P^{SAT}

NP $=$ P^{SAT}? Probably No because coNP $\subseteq$ P^{SAT}

Defn: $\mathbf{NP}^A = \{B \mid B$ is decidable in nondeterministic polynomial time with an oracle for $A\}$

Recall $\text{MIN-FORMULA} = \{\langle\phi\rangle \mid \phi$ is a minimal Boolean formula $\}$

Example: $\overline{\textbf{MIN–FORMULA}} \in \mathbf{NP}^{SAT}$

"On input $\langle\phi\rangle$

1. Guess shorter formula ψ
2. Use SAT oracle to solve the coNP problem: ϕ and ψ are equivalent
3. *Accept* if ϕ and ψ are equivalent. *Reject* if not."

Oracles and P versus NP

Theorem: There is an oracle A where $\mathrm{P}^A = \mathrm{NP}^A$

Oracles and P versus NP

Theorem: There is an oracle A where $\mathrm{P}^A = \mathrm{NP}^A$

Proof: Let $A = TQBF$

Oracles and P versus NP

Theorem: There is an oracle A where $\text{P}^A = \text{NP}^A$

Proof: Let $A = TQBF$

$\text{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq \text{P}^{TQBF}$

Oracles and P versus NP

Theorem: There is an oracle A where $\mathrm{P}^A = \mathrm{NP}^A$

Proof: Let $A = TQBF$

$\mathrm{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq \mathrm{P}^{TQBF}$

Relevance to the P versus NP question

Oracles and P versus NP

Theorem: There is an oracle A where $\text{P}^A = \text{NP}^A$

Proof: Let $A = TQBF$

$\text{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq$ P^{TQBF}

Relevance to the P versus NP question

Recall: We showed $EQ_{\text{REX}\uparrow} \notin$ PSPACE.

Oracles and P versus NP

Theorem: There is an oracle A where $\mathrm{P}^A = \mathrm{NP}^A$

Proof: Let $A = TQBF$

$\mathrm{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq$ P^{TQBF}

Relevance to the P versus NP question

Recall: We showed $EQ_{\mathrm{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Oracles and P versus NP

Theorem: There is an oracle A where $\mathrm{P}^A = \mathrm{NP}^A$

Proof: Let $A = TQBF$

$\mathrm{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq \mathrm{P}^{TQBF}$

Relevance to the P versus NP question

Recall: We showed $EQ_{\mathrm{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Oracles and P versus NP

Theorem: There is an oracle A where $\mathrm{P}^A = \mathrm{NP}^A$

Proof: Let $A = TQBF$

$\mathrm{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq$ P^{TQBF}

Relevance to the P versus NP question

Recall: We showed $EQ_{\mathrm{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Reason: Suppose YES.

Oracles and P versus NP

Theorem: There is an oracle A where $\text{P}^A = \text{NP}^A$

Proof: Let $A = TQBF$

$\text{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq$ P^{TQBF}

Relevance to the P versus NP question

Recall: We showed $EQ_{\text{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Reason: Suppose YES.
The Hierarchy Theorems are proved by a diagonalization.

Oracles and P versus NP

Theorem: There is an oracle A where $\text{P}^A = \text{NP}^A$

Proof: Let $A = TQBF$

$\text{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq \text{P}^{TQBF}$

Relevance to the P versus NP question

Recall: We showed $EQ_{\text{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Reason: Suppose YES.
The Hierarchy Theorems are proved by a diagonalization.
In this diagonalization, the TM D simulates some TM M.

Oracles and P versus NP

Theorem: There is an oracle A where $\text{P}^A = \text{NP}^A$

Proof: Let $A = TQBF$

$\text{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq \text{P}^{TQBF}$

Relevance to the P versus NP question

Recall: We showed $EQ_{\text{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Reason: Suppose YES.
The Hierarchy Theorems are proved by a diagonalization.
In this diagonalization, the TM D simulates some TM M.

If both TMs were oracle TMs D^A and M^A with the same oracle A,
the simulation and the diagonalization would still work.

Oracles and P versus NP

Theorem: There is an oracle A where $\mathrm{P}^A = \mathrm{NP}^A$

Proof: Let $A = TQBF$

$\mathrm{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq \mathrm{P}^{TQBF}$

Relevance to the P versus NP question

Recall: We showed $EQ_{\mathrm{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Reason: Suppose YES.
The Hierarchy Theorems are proved by a diagonalization.
In this diagonalization, the TM D simulates some TM M.
If both TMs were oracle TMs D^A and M^A with the same oracle A,
the simulation and the diagonalization would still work.
Therefore, if we could prove P $\neq$ NP by a diagonalization,
we would also prove that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every oracle A.

Oracles and P versus NP

Theorem: There is an oracle A where $\mathrm{P}^A = \mathrm{NP}^A$

Proof: Let $A = TQBF$

$\mathrm{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq \mathrm{P}^{TQBF}$

Relevance to the P versus NP question

Recall: We showed $EQ_{\mathrm{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Reason: Suppose YES.
The Hierarchy Theorems are proved by a diagonalization.
In this diagonalization, the TM D simulates some TM M.
If both TMs were oracle TMs D^A and M^A with the same oracle A,
the simulation and the diagonalization would still work.
Therefore, if we could prove P $\neq$ NP by a diagonalization,
we would also prove that $\mathrm{P}^A \neq \mathrm{NP}^A$ for every oracle A.

But that is false!

Oracles and P versus NP

Theorem: There is an oracle A where $\text{P}^A = \text{NP}^A$

Proof: Let $A = TQBF$

$\text{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq \text{P}^{TQBF}$

Relevance to the P versus NP question

Recall: We showed $EQ_{\text{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Reason: Suppose YES.
The Hierarchy Theorems are proved by a diagonalization.
In this diagonalization, the TM D simulates some TM M.
If both TMs were oracle TMs D^A and M^A with the same oracle A,
the simulation and the diagonalization would still work.
Therefore, if we could prove P $\neq$ NP by a diagonalization,
we would also prove that $\text{P}^A \neq \text{NP}^A$ for every oracle A.
But that is false!

Check-in 22.3

Oracles and P versus NP

Theorem: There is an oracle A where $\text{P}^A = \text{NP}^A$

Proof: Let $A = TQBF$

$\text{NP}^{TQBF} \subseteq$ NPSPACE $=$ PSPACE $\subseteq \text{P}^{TQBF}$

Relevance to the P versus NP question

Recall: We showed $EQ_{\text{REX}\uparrow} \notin$ PSPACE.

Could we show $SAT \notin$ P using a similar method? NO.

Reason: Suppose YES.
The Hierarchy Theorems are proved by a diagonalization.
In this diagonalization, the TM D simulates some TM M.

If both TMs were oracle TMs D^A and M^A with the same oracle A, the simulation and the diagonalization would still work.
Therefore, if we could prove P $\neq$ NP by a diagonalization, we would also prove that $\text{P}^A \neq \text{NP}^A$ for every oracle A.

But that is false!

Check-in 22.3

Which of these are known to be true?
Check all that apply.

(a) $\text{P}^{SAT} = \text{P}^{\overline{SAT}}$

(b) $\text{NP}^{SAT} = \text{coNP}^{SAT}$

(c) MIN-FORMULA $\in \text{P}^{TQBF}$

(d) $\text{NP}^{TQBF} = \text{coNP}^{TQBF}$

Check-in 22.3