Algorithm Theory

Georg Moser Mircea Dan Hernest

Institute of Computer Science @ UIBK

Summer 2007

Complexity Classes

model of computation	multi-string TM
mode	deterministic nondeterministic
resource	time space
bound	$f : \mathbb{N} \to \mathbb{N}$

Definition Complexity class

A complexity class is the set of all languages, decided by a multi-string TM, operating in a mode, so that the TM, on input x uses at most $f(|x|)$ of the resource.

Time Constructible Functions

A function $f: \mathbb{N} \to \mathbb{N}$, where $f(n) \geq n \log n$ is called time constructible if the function that maps

$$1^n \text{ to the binary representation of } f(n)$$

is computable in time $\mathcal{O}(f(n))$.

Example Consider the following functions:

1. $f(n) = n \log n$ is time constructible

   Proof Idea: First represent n in binary; second binary multiplication of n and $\log n$; the latter is (grossly) bounded by $\mathcal{O}(n \cdot \log n)$ steps
2. $f(n) = n\sqrt{n}$ is time constructible
3. $f(n) = c$, $f(n) = n$ are not time constructible

Space Constructible Functions

A function $f: \mathbb{N} \to \mathbb{N}$, where $f(n) \geq \log n$ is called space constructible if the function that maps

$$1^n \text{ to the binary representation of } f(n)$$

is computable in space $\mathcal{O}(f(n))$.

Example Consider the following functions:

1. $f(n) = \log n$ is space constructible

   Proof Idea: First represent n in binary; then count the number of bits in $(n)_2$; needs at most $\mathcal{O}(\log n)$
2. $f(n) = n^2$ is space constructible
3. $f(n) = c$, $f(n) = \log \log n$ are not space constructible

Complements of Complexity Classes

Definition

co$\mathcal{C}$

Let $\mathrm{L} \subseteq \Sigma^*$ be a language
Define $\overline{\mathrm{L}} = \Sigma^* - \mathrm{L}$, its complement
For any class $\mathcal{C}$, **co$\mathcal{C}$** $= \{\overline{\mathrm{L}} \colon \mathrm{L} \in \mathcal{C}\}$

Facts

- deterministic space and time classes are closed under complement
- nondeterministic space is closed under complement
- open problem whether nondeterministic time classes are closed.

Hierarchy Theorems

Theorem

The time hierarchy theorem

If $f(n)$ is a time constructible function, then the class **TIME**$(f(n))$ is strictly contained in **TIME**$(f(2n+1)^3)$.

Corollary: $\mathbf{P} \subsetneq \mathbf{EXP}$

Theorem

The space hierarchy theorem

If $f(n)$ is a space constructible function, then the class **SPACE**$(f(n))$ is strictly contained in **SPACE**$(f(n) \log f(n))$

Corollary: $\mathbf{L} \subsetneq \mathbf{PSPACE}$

Time Hierarchy Theorem

Let $f(n)$ be a time-constructible function, define

$$\mathsf{H}_f := \{\mathsf{M}; x \colon \mathsf{M} \text{ accepts } x \text{ after at most } f(|x|) \text{ steps}\}$$

Lemma $\mathsf{H}_f \in \mathbf{TIME}(f(n)^3)$.

Proof sketch

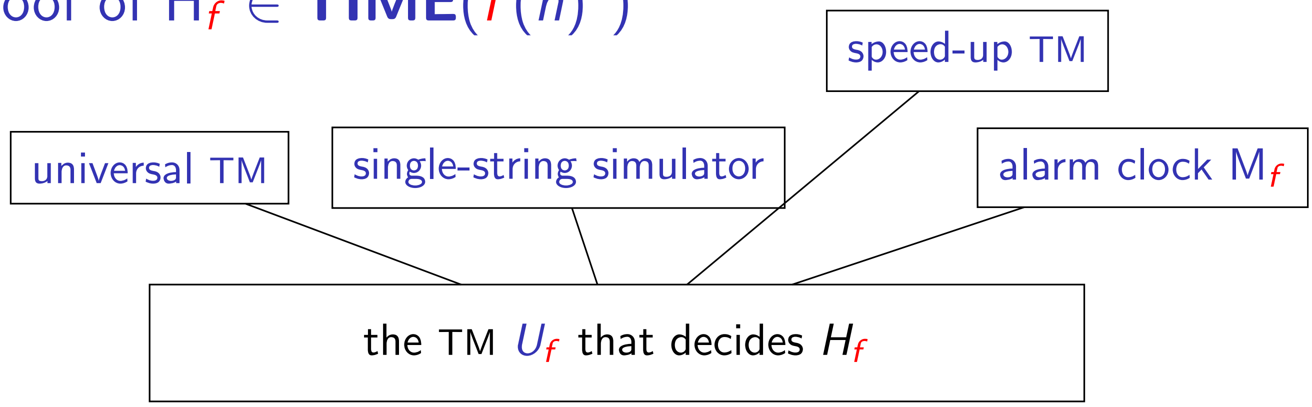
➡ construct a suitable TM

Lemma $\mathsf{H}_f \notin \mathbf{TIME}(f(\lfloor \frac{n}{2} \rfloor))$.

Proof sketch

➡ use diagonalisation

Proof of $\mathsf{H}_f \in \mathbf{TIME}(f(n)^3)$



➡ a universal TM U interprets the first part of its input as the code of another TM and the second as the code of the input; we obtain

$$U(\mathsf{M}; x) = \mathsf{M}(x)$$

➡ states, symbols, special symbols, etc. are encoded as natural numbers in binary

➡ δ is encoded as list of pairs

Initial Phase

U_f has 4 strings.

- U_f uses M_f on the 4th string to compute binary representation of $f(|x|)$

Then

- transform input x to the encoding of $\triangleright x$
- on the second string the code of the initial state s of M is written
- on the third string the description of M is written

Complexity

- $\mathcal{O}(f(|x|)) + \mathcal{O}(n) = \mathcal{O}(f(n))$, where $n = |\mathrm{M}; x|$ (constants independent on M)

Main simulation

U_f simulates one-by-one the steps of M on x

`repeat`

- U_f uses the single-string simulation to represent all strings of M on its first string
- the second string is used to keep track of the current state of M
- gather all information from the first string, update state according to third string, rewrite first string
- U_f advances the alarm-clock, then repeats.

The simulation halts if M halts or the alarm goes off

Complexity

- each step needs $\mathcal{O}(f(n)^2)$ (constants independent on M)
- total time is $\mathcal{O}(f(n)^3)$ use speed-up machine to obtain $f(n)^3$

Reminder: Diagonalisation

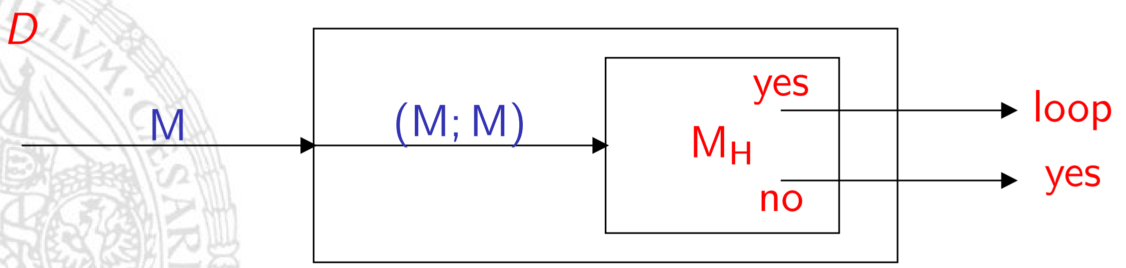
$$\mathsf{H} := \{\mathsf{M}; x\colon \mathsf{M} \text{ halts on } x\}$$

	…	100	101	110	…
⋮	⋮	⋮	⋮	⋮	⋮
100	…	$\circlearrowright$	$\circlearrowright$	$\times$	…
101	…	$\circlearrowright$	$\circlearrowright$	$\circlearrowright$	…
110	…	$\circlearrowright$	$\times$	$\times$	…
⋮	⋮	⋮	⋮	⋮	⋮

$\times$: halting
$\circlearrowright$: looping

Definition $\mathsf{M_H}$

suppose $\mathsf{M_H}$ decides H:

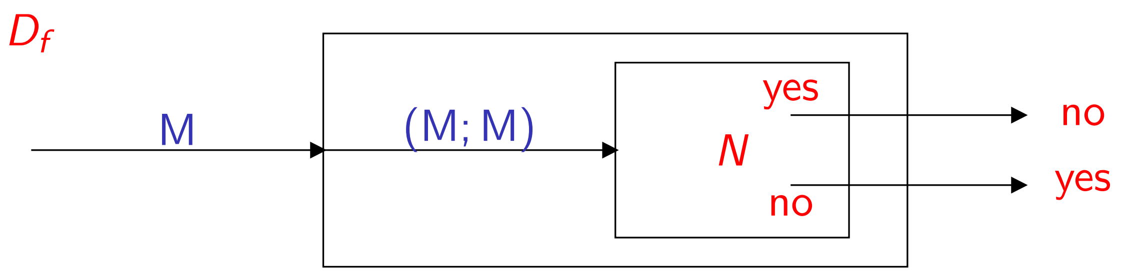


□

Proof of $\mathsf{H}_f \notin \mathbf{TIME}(f(\lfloor \frac{n}{2} \rfloor))$

suppose N decides H_f in time $f(\lfloor \frac{n}{2} \rfloor)$. Define

$$D_f(\mathsf{M})\colon \text{ if } N(\mathsf{M};\mathsf{M}) = \textit{yes} \text{ then } \textit{no}, \text{ else } \textit{yes}$$



Hence, as D_f runs in time $f(n)$:

$$D_f(D_f) = \textit{yes} \Rightarrow N(D_f; D_f) = \textit{no} \Rightarrow D_f; D_f \notin \mathsf{H}_f \Rightarrow D_f(D_f) = \textit{no}$$

$$D_f(D_f) = \textit{no} \Rightarrow N(D_f; D_f) = \textit{yes} \Rightarrow D_f; D_f \in \mathsf{H}_f \Rightarrow D_f(D_f) = \textit{yes}$$

Contradiction □