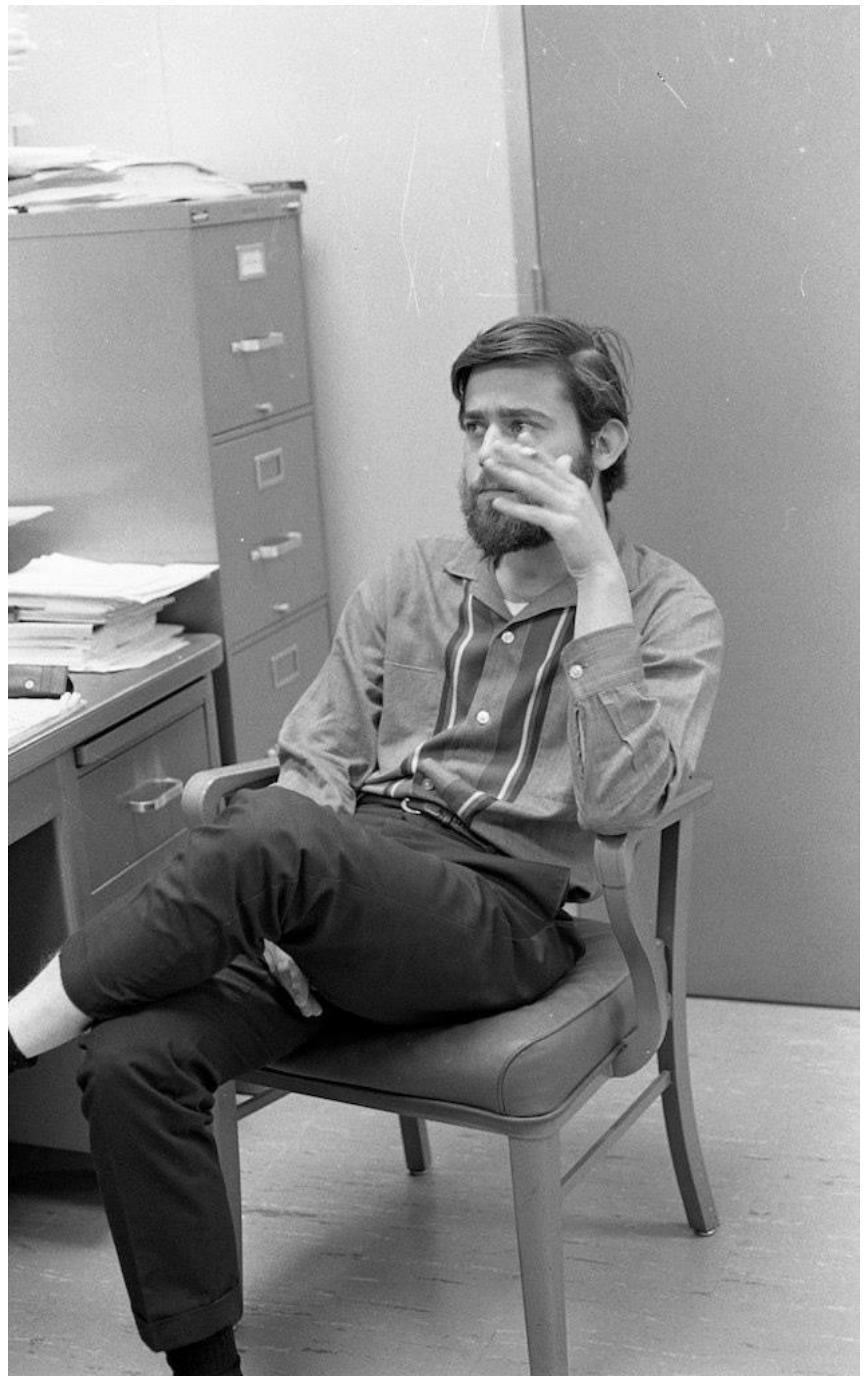
Abstract Complexity Theory

speedup and gap theorem

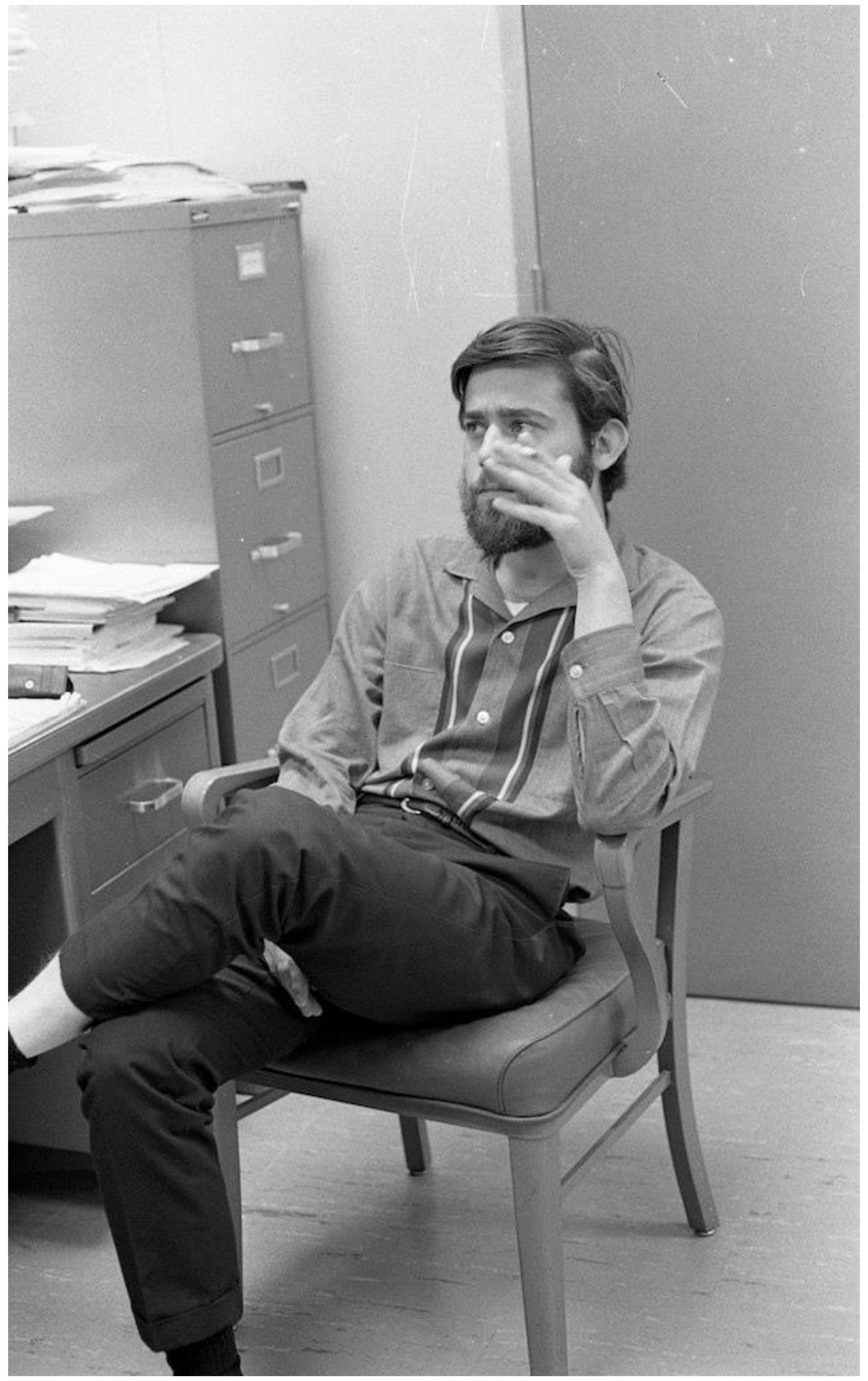
- Manuel Blum 1967: speedup theorem

born 1938 in Caracas, Venezuela



- Manuel Blum 1967: speedup theorem

born 1938 in Caracas, Venezuela



- Allan Borodin 1972: gap theorem

born 1941 in Canada

change of notation:

- M_u with $u \in \mathbb{N}$: TM with Goedel *number u*
- $\varphi_u : \mathbb{N}_0 \to \mathbb{N}_0$
  the function computed by machine M_u. Machine M_u started with $bin(n)$ computes $bin(\varphi_u(n))$ if it halts.
- $\beta_u(n)$: space used M_u started with input $bin(n)$
- $\tau_u(n)$: run time of M_u started with input $bin(n)$
- $A(n)$ holds *faa n*: predicate $A(n)$ for almost all n

$$\exists n_0.\ \forall n \geq n_0.\ A(n)$$

- $A(n)$ hold *io*: predicate a holds infinitely often

$$\forall n_0.\ \exists n > n_0.\ A(n)$$

- Ω: undefined

change of notation:

- M_u with $u \in \mathbb{N}$: TM with Goedel *number u*
- $\varphi_u : \mathbb{N}_0 \to \mathbb{N}_0$
  the function computed by machine M_u. Machine M_u started with $bin(n)$ computes $bin(\varphi_u(n))$ if it halts.
- $\beta_u(n)$: space used M_u started with input $bin(n)$
- $\tau_u(n)$: run time of M_u started with input $bin(n)$
- $A(n)$ holds *faa n*: predicate $A(n)$ for almost all n

$$\exists n_0.\ \forall n \geq n_0.\ A(n)$$

- $A(n)$ hold *io*: predicate a holds infinitely often

$$\forall n_0.\ \exists n > n_0.\ A(n)$$

- Ω: undefined

1 Functions which are hard to compute almost everywhere

Lemma 1. *Let* $T : \mathbb{N} \to \mathbb{N}$ *be total and computable. Then*

$$\exists f.\, \forall i.\, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i *computing* f *uses at least space* $T(n)$ *for almost all inputs* n*.*

Lemma 1. *Let $T : \mathbb{N} \to \mathbb{N}$ be total and computable. Then*

$$\exists f.\, \forall i.\, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i computing f uses at least space $T(n)$ for almost all inputs n.

1	$\beta_1(1)$	$\beta_1(2)$	...	$\beta_1(n)$	...
2	$\beta_2(1)$	$\beta_2(2)$	...	$\beta_2(n)$	...
			...		...
			...		...
n	$\beta_n(1)$	$\beta_n(2)$	...	$\beta_n(n)$	...
			...		...

Table 1: Table of space used by machines M_n. Row n stores in column x the space $\beta_n(x)$ used by machine M_n on input x. Nothe that $\beta_n(x)$ can be Ω/undefined.

$f : \mathbb{N}_0 \to \mathbb{B}$

Consider table 1 and define simultaneously by induction on n

- lists $L(n) \subset \mathbb{N}$ of *cancelled* lines. Machines with indices in $L(n)$ will not compute f
- initially $L(1) = \emptyset$.
- for $n > 1$:

$$s(n) = \begin{cases} \min\{s \leq n \,:\, s \notin L(n-1)\,,\, \beta_s(n) < T(n)\,, \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

Lemma 1. *Let $T : \mathbb{N} \to \mathbb{N}$ be total and computable. Then*

$$\exists f.\, \forall i.\, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i computing f uses at least space $T(n)$ for almost all inputs n.

$f : \mathbb{N}_0 \to \mathbb{B}$

1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$
~~2~~	~~$\beta_2(1)$~~	~~$\beta_2(2)$~~	~~$\dots$~~	~~$\beta_2(n)$~~	~~$\dots$~~
			$\dots$		$\dots$
			~~$\dots$~~		~~$\dots$~~
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$
			$\dots$		$\dots$

Table 1: Table of space used by machines M_n. Row n stores in column x the space $\beta_n(x)$ used by machine M_n on input x. Nothe that $\beta_n(x)$ can be Ω/undefined.

Consider table 1 and define simultaneously by induction on n

- lists $L(n) \subset \mathbb{N}$ of *cancelled* lines. Machines with indices in $L(n)$ will not compute f
- initially $L(1) = \emptyset$.
- for $n > 1$:

$$s(n) = \begin{cases} \min\{s \leq n \;:\; s \notin L(n-1)\,,\, \beta_s(n) < T(n)\,, \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

Lemma 1. *Let $T : \mathbb{N} \to \mathbb{N}$ be total and computable. Then*

$$\exists f.\, \forall i.\, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i computing f uses at least space $T(n)$ for almost all inputs n.

$f : \mathbb{N}_0 \to \mathbb{B}$

1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$
2	$\beta_2(1)$	$\beta_2(2)$	$\dots$	$\beta_2(n)$	$\dots$
			$\dots$		$\dots$
			$\dots$		$\dots$
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$
			$\dots$		$\dots$

Table 1: Table of space used by machines M_n. Row n stores in column x the space $\beta_n(x)$ used by machine M_n on input x. Nothe that $\beta_n(x)$ can be Ω/undefined.

Consider table 1 and define simultaneously by induction on n

- lists $L(n) \subset \mathbb{N}$ of *cancelled* lines. Machines with indices in $L(n)$ will not compute f
- initially $L(1) = \emptyset$.
- for $n > 1$: columns

$$s(n) = \begin{cases} \min\{s \leq n \,:\, s \notin L(n-1)\,,\ \beta_s(n) < T(n)\,, & \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

Lemma 1. *Let $T : \mathbb{N} \to \mathbb{N}$ be total and computable. Then*

$$\exists f. \forall i. \, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i computing f uses at least space $T(n)$ for almost all inputs n.

1	$\beta_1(1)$	$\beta_1(2)$	...	$\beta_1(n)$	...
~~2~~	~~$\beta_2(1)$~~	~~$\beta_2(2)$~~	~~...~~	~~$\beta_2(n)$~~	~~...~~
			...		...
			~~...~~		~~...~~
n	$\beta_n(1)$	$\beta_n(2)$	...	$\beta_n(n)$	...
			...		...

Table 1: Table of space used by machines M_n. Row n stores in column x the space $\beta_n(x)$ used by machine M_n on input x. Nothe that $\beta_n(x)$ can be Ω/undefined.

$f : \mathbb{N}_0 \to \mathbb{B}$

Consider table 1 and define simultaneously by induction on n

- lists $L(n) \subset \mathbb{N}$ of *cancelled* lines. Machines with indices in $L(n)$ will not compute f
- initially $L(1) = \emptyset$.
- for $n > 1$: columns

$$s(n) = \begin{cases} \min\{s \leq n \; : \; s \notin L(n-1) \, , \, \beta_s(n) < T(n) \, , \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

look for too easy computation

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

diagonalize

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

and forget index s

Lemma 1. *Let $T : \mathbb{N} \to \mathbb{N}$ be total and computable. Then*

$$\exists f.\, \forall i.\, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i computing f uses at least space $T(n)$ for almost all inputs n.

$f : \mathbb{N}_0 \to \mathbb{B}$

1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$
~~2~~	~~$\beta_2(1)$~~	~~$\beta_2(2)$~~	~~$\dots$~~	~~$\beta_2(n)$~~	~~$\dots$~~
			$\dots$		$\dots$
			~~$\dots$~~		~~$\dots$~~
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$
			$\dots$		$\dots$

Table 1: Table of space used by machines M_n. Row n stores in column x the space $\beta_n(x)$ used by machine M_n on input x. Nothe that $\beta_n(x)$ can be Ω/undefined.

Consider table 1 and define simultaneously by induction on n

- lists $L(n) \subset \mathbb{N}$ of *cancelled* lines. Machines with indices in $L(n)$ will not compute f
- initially $L(1) = \emptyset$.
- for $n > 1$: columns

$$s(n) = \begin{cases} \min\{s \leq n \,:\, s \notin L(n-1)\,,\, \beta_s(n) < T(n)\,, \\ \quad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

- f is computable and total because machine M_s with alphabet Σ and set of states Z can make on space $T(n)$ at most

$$|\Sigma|^{T(n)} \cdot |Z| \cdot T(n)$$

steps without repeating a configuration.

- If line s is ever cancelled, then $f \neq \varphi_s$

$$s \in L(n) \to f \neq \varphi_s$$

Lemma 1. *Let $T : \mathbb{N} \to \mathbb{N}$ be total and computable. Then*

$$\exists f.\, \forall i.\, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i computing f uses at least space $T(n)$ for almost all inputs n.

$f : \mathbb{N}_0 \to \mathbb{B}$

Consider table 1 and define simultaneously by induction on n

- lists $L(n) \subset \mathbb{N}$ of *cancelled* lines. Machines with indices in $L(n)$ will not compute f
- initially $L(1) = \emptyset$.
- for $n > 1$: columns

$$s(n) = \begin{cases} \min\{s \leq n \;:\; s \notin L(n-1)\;,\; \beta_s(n) < T(n)\;, \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$
~~2~~	~~$\beta_2(1)$~~	~~$\beta_2(2)$~~	~~$\dots$~~	~~$\beta_2(n)$~~	~~$\dots$~~
			$\dots$		$\dots$
			~~$\dots$~~		~~$\dots$~~
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$
			$\dots$		$\dots$

Table 1: Table of space used by machines M_n. Row n stores in column x the space $\beta_n(x)$ used by machine M_n on input x. Nothe that $\beta_n(x)$ can be Ω/undefined.

Lemma 2. *if i is an index, such that $\beta_i(n) < T(n)$ infinitely often, then $f \neq \varphi_i$*

- sufficient: $i = s(n)$ for some n

Lemma 1. *Let $T : \mathbb{N} \to \mathbb{N}$ be total and computable. Then*

$$\exists f.\, \forall i.\, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i computing f uses at least space $T(n)$ for almost all inputs n.

$f : \mathbb{N}_0 \to \mathbb{B}$

Consider table 1 and define simultaneously by induction on n

- lists $L(n) \subset \mathbb{N}$ of *cancelled* lines. Machines with indices in $L(n)$ will not compute f
- initially $L(1) = \emptyset$.
- for $n > 1$: columns

$$s(n) = \begin{cases} \min\{s \leq n \;:\; s \notin L(n-1)\,,\, \beta_s(n) < T(n)\,, & \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

1	$\beta_1(1)$	$\beta_1(2)$	...	$\beta_1(n)$	...
2	$\beta_2(1)$	$\beta_2(2)$	...	$\beta_2(n)$	...
			...		...
			...		...
n	$\beta_n(1)$	$\beta_n(2)$	...	$\beta_n(n)$	...
			...		...

Table 1: Table of space used by machines M_n. Row n stores in column x the space $\beta_n(x)$ used by machine M_n on input x. Nothe that $\beta_n(x)$ can be Ω/undefined.

Lemma 2. *if i is an index, such that $\beta_i(n) < T(n)$ infinitely often, then $f \neq \varphi_i$*

- sufficient: $i = s(n)$ for some n
- Let

  $$i \leq n_1 < n_2 < \ldots$$

  be infinite sequence of arguments with

  $$\beta_i(n_j) < T(n_j) \quad \text{for all } j$$

- for all $n \geq i$: index i is candidate for $s(n)$.
  cases

Lemma 1. *Let $T : \mathbb{N} \to \mathbb{N}$ be total and computable. Then*

$$\exists f.\, \forall i.\, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i computing f uses at least space $T(n)$ for almost all inputs n.

$f : \mathbb{N}_0 \to \mathbb{B}$

Consider table 1 and define simultaneously by induction on n

- lists $L(n) \subset \mathbb{N}$ of *cancelled* lines. Machines with indices in $L(n)$ will not compute f
- initially $L(1) = \emptyset$.
- for $n > 1$: columns

$$s(n) = \begin{cases} \min\{s \leq n \,:\, s \notin L(n-1)\,,\, \beta_s(n) < T(n)\,, \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$
2	$\beta_2(1)$	$\beta_2(2)$	$\dots$	$\beta_2(n)$	$\dots$
			$\dots$		$\dots$
			$\dots$		$\dots$
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$
			$\dots$		$\dots$

Table 1: Table of space used by machines M_n. Row n stores in column x the space $\beta_n(x)$ used by machine M_n on input x. Nothe that $\beta_n(x)$ can be Ω/undefined.

Lemma 2. *if i is an index, such that $\beta_i(n) < T(n)$ infinitely often, then $f \neq \varphi_i$*

- sufficient: $i = s(n)$ for some n
- Let

  $$i \leq n_1 < n_2 < \dots$$

  be infinite sequence of arguments with

  $$\beta_i(n_j) < T(n_j) \quad \text{for all } j$$

- for all $n \geq i$: index i is candidate for $s(n)$.
  cases

1. i is smallest such candidate:

   $$i \in L(n)\,,\; f \neq \varphi_i$$

   done

Lemma 1. *Let $T : \mathbb{N} \to \mathbb{N}$ be total and computable. Then*

$$\exists f.\, \forall i.\, (\varphi_i = f \to \beta_i(n) \geq T(n) \text{ faa } n$$

Every machine M_i computing f uses at least space $T(n)$ for almost all inputs n.

$f : \mathbb{N}_0 \to \mathbb{B}$

Consider table 1 and define simultaneously by induction on n

- lists $L(n) \subset \mathbb{N}$ of *cancelled* lines. Machines with indices in $L(n)$ will not compute f
- initially $L(1) = \emptyset$.
- for $n > 1$: columns

$$s(n) = \begin{cases} \min\{s \leq n \;:\; s \notin L(n-1)\,,\; \beta_s(n) < T(n)\,, \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

1	$\beta_1(1)$	$\beta_1(2)$	...	$\beta_1(n)$	...
~~2~~	~~$\beta_2(1)$~~	~~$\beta_2(2)$~~	~~...~~	~~$\beta_2(n)$~~	~~...~~
			...		...
			~~...~~		~~...~~
n	$\beta_n(1)$	$\beta_n(2)$	...	$\beta_n(n)$	...
			...		...

Table 1: Table of space used by machines M_n. Row n stores in column x the space $\beta_n(x)$ used by machine M_n on input x. Nothe that $\beta_n(x)$ can be Ω/undefined.

Lemma 2. *if i is an index, such that $\beta_i(n) < T(n)$ infinitely often, then $f \neq \varphi_i$*

- sufficient: $i = s(n)$ for some n
- Let

  $$i \leq n_1 < n_2 < \dots$$

  be infinite sequence of arguments with

  $$\beta_i(n_j) < T(n_j) \quad \text{for all } j$$

- for all $n \geq i$: index i is candidate for $s(n)$.
  cases

  2. otherwise a smaller index $i' < i$ is included in $L(n)$

     $$L(n) = L(n-1) \cup \{i\}$$

     This can happen at most $i-1$ times.

     $$i \in L(n_i)$$

2 Speedup theorem for space

2.1 statement

- for long time the most famous theorem in computer science
- today almost forgotten. Lecturers tend to say, it's too difficult for students. But maybe its too difficult for the lecturers??
- until today my favourite theorem in computer science
- **stating that some functions have no fastest or most space efficient program**
- proof is of course completely understandable...

Lemma 3. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\beta_j(n)) \leq \beta_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f\ \to \exists j.\ (\varphi_j = f\ \wedge\ r(\beta_j(n)) \leq \beta_i(n)\ \mathit{faa}\ n)$$

2 Speedup theorem for space

2.1 statement

- for long time the most famous theorem in computer science
- today almost forgotten. Lecturers tend to say, it's too difficult for students. But maybe its too difficult for the lecturers??
- until today my favourite theorem in computer science
- **stating that some functions have no fastest or most space efficient program**
- proof is of course completely understandable...

Lemma 3. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\beta_j(n)) \leq \beta_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f \ \to \exists j.\ (\varphi_j = f \ \wedge \ r(\beta_j(n)) \leq \beta_i(n)\ faa\ n)$$

example:

$$r(n) = 2^n \quad , \quad 2^{\beta_j(n)} \leq \beta_i(n)$$

now repeat for j...

Lemma 3. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\beta_j(n)) \leq \beta_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f\ \to \exists j.\ (\varphi_j = f\ \wedge\ r(\beta_j(n)) \leq \beta_i(n)\ faa\ n)$$

example:

$$r(n) = 2^n \quad , \quad 2^{\beta_j(n)} \leq \beta_i(n)$$

now repeat for j...

2.2 constructing f and lower bound

ideas:

- space consumption of M_i computing f, i.e. with $f = \varphi_i$ must be large enough such that

  $$r^{-1}(r^{-1}(\ldots r^{-1}(\beta_i(n))\ldots))$$

  is defined

- with larger programs computing f becomes easier.

- w.l.o.g r strictly monotonically increasing and

  $$r(n) \geq 2n \quad \text{for all } n$$

- define numbers

  $$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\ldots r(1))) \\ &\geq 2^i \end{aligned}$$

- replace in the above proof $T(n)$ by $r_{n-s}(1)$ for candidates s of canelled rows. For large s this bound becomes smaller.

Lemma 3. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\beta_j(n)) \leq \beta_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f\ \to \exists j.\ (\varphi_j = f\ \wedge\ r(\beta_j(n)) \leq \beta_i(n)\ \mathit{faa}\ n)$$

- replace in the above proof $T(n)$ by $r_{n-s}(1)$ for candidates s of canelled rows. For large s this bound becomes smaller.

$r(n) \geq 2n$ for all n

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\ldots r(1))) \\ &\geq 2^i \end{aligned}$$

1	$\beta_1(1)$	$\beta_1(2)$	$\ldots$	$\beta_1(n)$	$\ldots$
~~2~~	$\cancel{\beta_2(1)}$	$\cancel{\beta_2(2)}$	~~$\ldots$~~	$\cancel{\beta_2(n)}$	~~$\ldots$~~
			$\ldots$		$\ldots$
			~~$\ldots$~~		~~$\ldots$~~
n	$\beta_n(1)$	$\beta_n(2)$	$\ldots$	$\beta_n(n)$	$\ldots$
			$\ldots$		$\ldots$

Lemma 3. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\beta_j(n)) \le \beta_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f\ \to \exists j.\ (\varphi_j = f\ \wedge\ r(\beta_j(n)) \le \beta_i(n)\ \mathit{faa}\ n)$$

- replace in the above proof $T(n)$ by $r_{n-s}(1)$ for candidates s of canelled rows. For large s this bound becomes smaller.

$r(n) \ge 2n$ for all n

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\ldots r(1))) \\ &\ge 2^i \end{aligned}$$

1	$\beta_1(1)$	$\beta_1(2)$	$\ldots$	$\beta_1(n)$	$\ldots$
~~2~~	$\cancel{\beta_2(1)}$	$\cancel{\beta_2(2)}$	$\cancel{\ldots}$	$\cancel{\beta_2(n)}$	$\cancel{\ldots}$
			$\ldots$		$\ldots$
			$\cancel{\ldots}$		$\cancel{\ldots}$
n	$\beta_n(1)$	$\beta_n(2)$	$\ldots$	$\beta_n(n)$	$\ldots$
			$\ldots$		$\ldots$

defining f: Initially $L(1) = \emptyset$ and for $n > 1$:

$$s(n) = \begin{cases} \min\{s \le n\ :\ s \notin L(n-1)\ ,\ \underline{\beta_s(n) < r_{n-s}}\ , \\ \quad \varphi_s(n) \ne \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \ne \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \ne \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

Lemma 3. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\beta_j(n)) \leq \beta_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f\ \to \exists j.\ (\varphi_j = f\ \wedge\ r(\beta_j(n)) \leq \beta_i(n)\ faa\ n)$$

- replace in the above proof $T(n)$ by $r_{n-s}(1)$ for candidates s of canelled rows. For large s this bound becomes smaller.

$r(n) \geq 2n$ for all n

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\ldots r(1))) \\ &\geq 2^i \end{aligned}$$

1	$\beta_1(1)$	$\beta_1(2)$	...	$\beta_1(n)$	...
~~2~~	~~$\beta_2(1)$~~	~~$\beta_2(2)$~~	~~...~~	~~$\beta_2(n)$~~	~~...~~
			...		...
			~~...~~		~~...~~
n	$\beta_n(1)$	$\beta_n(2)$	...	$\beta_n(n)$	...
			...		...

defining f: Initially $L(1) = \emptyset$ and for $n > 1$:

$$s(n) = \begin{cases} \min\{s \leq n\ :\ s \notin L(n-1)\ ,\ \underline{\beta_s(n) < r_{n-s}}\ , & \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

as above show:

Lemma 4. *f is total and computable and*

$$\varphi_i = f\ \to\ \beta_i(n) \geq r_{n-i}\ faa\ n$$

Lemma 3. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\beta_j(n)) \le \beta_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f \ \to \exists j.\ (\varphi_j = f \ \wedge \ r(\beta_j(n)) \le \beta_i(n)\ \mathit{faa}\ n)$$

- replace in the above proof $T(n)$ by $r_{n-s}(1)$ for candidates s of canelled rows. For large s this bound becomes smaller.

$r(n) \ge 2n \quad$ for all n

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\ldots r(1))) \\ &\ge 2^i \end{aligned}$$

1	$\beta_1(1)$	$\beta_1(2)$	...	$\beta_1(n)$	...
~~2~~	~~$\beta_2(1)$~~	~~$\beta_2(2)$~~	~~...~~	~~$\beta_2(n)$~~	~~...~~
			...		...
			...		...
n	$\beta_n(1)$	$\beta_n(2)$	...	$\beta_n(n)$	...
			...		...

defining f: Initially $L(1) = \emptyset$ and for $n > 1$:

$$s(n) = \begin{cases} \min\{s \le n \ :\ s \notin L(n-1)\ ,\ \underline{\beta_s(n) < r_{n-s}}\ , & \\ \qquad \varphi_s(n) \ne \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \ne \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \ne \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

as above show:

Lemma 4. *f is total and computable and*

$$\varphi_i = f \ \to \ \beta_i(n) \ge r_{n-i}\ \mathit{faa}\ n$$

2.3 stating upper bound

Lemma 5. *For all $k \in \mathbb{N}$: if machine M_k computes f, then there is a machine $M_{j(k)}$ computing f with space consumption $\beta_{j(k)}(n) \le r_{n-k}$ for almost all n*

$$\varphi_k = f \to \exists j(k).\ (\varphi_{j(k)} = f \ \wedge \ \beta_{j(k)}(n) \le r_{n-k} \quad \mathit{faa}\ n)$$

Lemma 3. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\beta_j(n)) \le \beta_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f\ \to \exists j.\ (\varphi_j = f\ \wedge\ r(\beta_j(n)) \le \beta_i(n)\ faa\ n)$$

- replace in the above proof $T(n)$ by $r_{n-s}(1)$ for candidates s of canelled rows. For large s this bound becomes smaller.

$r(n) \ge 2n$ for all n

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\ldots r(1))) \\ &\ge 2^i \end{aligned}$$

1	$\beta_1(1)$	$\beta_1(2)$	$\ldots$	$\beta_1(n)$	$\ldots$
2	$\beta_2(1)$	$\beta_2(2)$	$\ldots$	$\beta_2(n)$	$\ldots$
			$\ldots$		$\ldots$
			$\ldots$		$\ldots$
n	$\beta_n(1)$	$\beta_n(2)$	$\ldots$	$\beta_n(n)$	$\ldots$
			$\ldots$		$\ldots$

defining f: Initially $L(1) = \emptyset$ and for $n > 1$:

$$s(n) = \begin{cases} \min\{s \le n\ :\ s \notin L(n-1)\ ,\ \beta_s(n) < r_{n-s}\ , & \\ \quad \varphi_s(n) \ne \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \ne \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1)\ \cup\ \{s(n)\} & s(n) \ne \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

as above show:

Lemma 4. *f is total and computable and*

$$\varphi_i = f\ \to\ \beta_i(n) \ge r_{n-i}\ faa\ n$$

2.3 stating upper bound

Lemma 5. *For all $k \in \mathbb{N}$: if machine M_k computes f, then there is a machine $M_{j(k)}$ computing f with space consumption $\beta_{j(k)}(n) \le r_{n-k}$ for almost all n*

$$\varphi_k = f \to \exists j(k).\ (\varphi_{j(k)} = f\ \wedge\ \beta_{j(k)}(n) \le r_{n-k}\quad faa\ n)$$

implies speedup theorem (lemma 3): Let $\varphi_i = f$ and $k \ge i+1$ with $\varphi_k = f$. Then $\varphi_{j(k)} = f$ and

$$\begin{aligned} r(\beta_{j(k)}(n)) &\le r(r_{n-k}) \quad \text{(lemma 5} \\ &\le r(r_{n-(i+1)}) \\ &= r_{n-i} \\ &\le \beta_i(n) \quad \text{(lemma 4)} \end{aligned}$$

$$r(n) \geq 2n \quad \text{for all } n$$

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\ldots r(1))) \\ &\geq 2^i \end{aligned}$$

defining f: Initially $L(1) = \emptyset$ and for $n > 1$:

$$s(n) = \begin{cases} \min\{s \leq n \;:\; s \notin L(n-1)\;,\; \beta_s(n) < r_{n-s}\;, & \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

Lemma 5. *For all $k \in \mathbb{N}$: if machine M_k computes f, then there is a machine $M_{j(k)}$ computing f with space consumption $\beta_{j(k)}(n) \leq r_{n-k}$ for almost all n*

$$\varphi_k = f \rightarrow \exists j(k).\ (\varphi_{j(k)} = f \;\wedge\; \beta_{j(k)}(n) \leq r_{n-k} \quad \mathit{faa}\ n)$$

$$r(n) \geq 2n \quad \text{for all } n$$

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\ldots r(1))) \\ &\geq 2^i \end{aligned}$$

defining f: Initially $L(1) = \emptyset$ and for $n > 1$:

$$s(n) = \begin{cases} \min\{s \leq n \;:\; s \notin L(n-1)\;,\; \beta_s(n) < r_{n-s}\;, \\ \quad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

Lemma 5. *For all $k \in \mathbb{N}$: if machine M_k computes f, then there is a machine $M_{j(k)}$ computing f with space consumption $\beta_{j(k)}(n) \leq r_{n-k}$ for almost all n*

$$\varphi_k = f \rightarrow \exists j(k).\ (\varphi_{j(k)} = f \ \wedge \beta_{j(k)}(n) \leq r_{n-k} \quad \textit{faa } n)$$

- by showing (*non constructively*) the existence of an efficient program
- avoiding simulation of machines M_s for $s < 2k$

- **crucial observation:** For all k there is $v \in \mathbb{N}$ (non constructive) such that every index
$$s \in [1:2k]$$
which will ever be included in a list $L(n)$ is already in $L(v)$

2.4 proving the upper bound

- machine $M_{j(k)}$ stores in finite memory
$$f(1)\ldots,f(v),L(v)$$
- in table 2 it does not need to simulate the lines with the large space bounds above line $2k$

s			$\ldots$		$\ldots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\ldots$	$\beta_1(n)$	$\ldots$	hard
$*$			$\ldots$		$\ldots$	
			$\ldots$		$\ldots$	
$*$			$\ldots$		$\ldots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\ldots$	$\beta_{2k}(n)$	$\ldots$	
			$\ldots$		$\ldots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\ldots$	$\beta_n(n)$	$\ldots$	
			$\ldots$		$\ldots$	mild

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$ Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

$$r(n) \geq 2n \quad \text{for all } n$$

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\dots r(1))) \\ &\geq 2^i \end{aligned}$$

defining f: Initially $L(1) = \emptyset$ and for $n > 1$:

$$s(n) = \begin{cases} \min\{s \leq n \ : \ s \notin L(n-1) \ , \ \beta_s(n) < r_{n-s} \ , \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

Lemma 5. *For all $k \in \mathbb{N}$: if machine M_k computes f, then there is a machine $M_{j(k)}$ computing f with space consumption $\beta_{j(k)}(n) \leq r_{n-k}$ for almost all n*

$$\varphi_k = f \rightarrow \exists j(k). \ (\varphi_{j(k)} = f \ \wedge \ \beta_{j(k)}(n) \leq r_{n-k} \quad \textit{faa } n)$$

- machine $M_{j(k)}$ stores in finite memory

$$f(1) \dots, f(v), L(v)$$

- in table 2 it does not need to simulate the lines with the large space bounds above line $2k$

s			$\dots$		$\dots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$	hard
$*$			$\dots$		$\dots$	
			$\dots$		$\dots$	
$*$			$\dots$		$\dots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\dots$	$\beta_{2k}(n)$	$\dots$	
			$\dots$		$\dots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$	
			$\dots$		$\dots$	mild

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$. Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

$r(n) \geq 2n \quad$ for all n

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\dots r(1))) \\ &\geq 2^i \end{aligned}$$

defining f: Initially $L(1) = \emptyset$ and for $n > 1$:

$$s(n) = \begin{cases} \min\{s \leq n \ : \ s \notin L(n-1)\ ,\ \beta_s(n) < r_{n-s}\ , \\ \qquad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

Lemma 5. *For all $k \in \mathbb{N}$: if machine M_k computes f, then there is a machine $M_{j(k)}$ computing f with space consumption $\beta_{j(k)}(n) \leq r_{n-k}$ for almost all n*

$$\varphi_k = f \rightarrow \exists j(k).\ (\varphi_{j(k)} = f \ \wedge\ \beta_{j(k)}(n) \leq r_{n-k} \quad \textit{faa } n)$$

- machine $M_{j(k)}$ stores in finite memory

$$f(1) \dots, f(v), L(v)$$

- in table 2 it does not need to simulate the lines with the large space bounds above line $2k$

s			$\dots$		$\dots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$	hard
$*$			$\dots$		$\dots$	
			$\dots$		$\dots$	
$*$			$\dots$		$\dots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\dots$	$\beta_{2k}(n)$	$\dots$	
			$\dots$		$\dots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$	
			$\dots$		$\dots$	mild

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$. Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

Turing machine $M_{j(k)}$: with input $bin(n)$

- for $n \leq v$: print $bin(f(v))$. Done.
- for $n > v$ print $L(v)$; then proceed in stages

$$m = 2k+1, \dots, n$$

where stage m computes $L(m)$ and $bin(f(n))$ if $m = n$.

$r(n) \geq 2n$ for all n

$$\begin{aligned} r_0 &= 1 \\ r_i &= r(r_{i-1}) \\ &= r(r(\dots r(1))) \\ &\geq 2^i \end{aligned}$$

defining f: Initially $L(1) = \emptyset$ and for $n > 1$:

$$s(n) = \begin{cases} \min\{s \leq n \ : \ s \notin L(n-1) \ , \ \beta_s(n) < r_{n-s} \ , \\ \quad \varphi_s(n) \neq \Omega\} & \text{if this exists} \\ \Omega & \text{otherwise} \end{cases}$$

$$f(n) = \begin{cases} 0 & s(n) \neq \Omega \wedge \varphi_s(n) = 1 \\ 1 & \text{otherwise} \end{cases}$$

$$L(n) = \begin{cases} L(n-1) \cup \{s(n)\} & s(n) \neq \Omega \\ L(n-1) & \text{otherwise} \end{cases}$$

Lemma 5. *For all $k \in \mathbb{N}$: if machine M_k computes f, then there is a machine $M_{j(k)}$ computing f with space consumption $\beta_{j(k)}(n) \leq r_{n-k}$ for almost all n*

$$\varphi_k = f \rightarrow \exists j(k). \ (\varphi_{j(k)} = f \ \wedge \ \beta_{j(k)}(n) \leq r_{n-k} \quad \mathit{faa}\ n)$$

- machine $M_{j(k)}$ stores in finite memory

$$f(1) \dots, f(v), L(v)$$

- in table 2 it does not need to simulate the lines with the large space bounds above line $2k$

s			$\dots$		$\dots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$	hard
$*$			$\dots$		$\dots$	
			$\dots$		$\dots$	
$*$			$\dots$		$\dots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\dots$	$\beta_{2k}(n)$	$\dots$	
			$\dots$		$\dots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$	
			$\dots$		$\dots$	mild

column m

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$. Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

Turing machine $M_{j(k)}$: with input $bin(n)$

- for $n \leq v$: print $bin(f(v))$. Done.
- for $n > v$ print $L(v)$; then proceed in stages

$$m = 2k+1, \dots, n$$

  where stage m computes $L(m)$ and $bin(f(n))$ if $m = n$.

- stage m: for

$$s \in [2k+1 : m] \setminus L(m-1)$$

1. simulate M_s with input $bin(m)$
2. space used $> r_{m-s}$ or $\leq r_{m-s}$ and $\varphi_s = \Omega$: abort, next s
3. space used $\leq r_{m-s}$ and $\varphi_s(m) \neq \Omega$: set $s(m) = s$, $L(m) = L(m-1) \cup \{s(m)\}$

s			$\dots$		$\dots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$	hard
$*$			$\dots$		$\dots$	
			$\dots$		$\dots$	
$*$			$\dots$		$\dots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\dots$	$\beta_{2k}(n)$	$\dots$	
			$\dots$		$\dots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$	
			$\dots$		$\dots$	mild

column m

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$. Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

Turing machine $M_{j(k)}$: with input $bin(n)$

- for $n \leq v$: print $bin(f(v))$. Done.
- for $n > v$ print $L(v)$; then proceed in stages

$$m = 2k+1, \dots, n$$

where stage m computes $L(m)$ and $bin(f(n))$ if $m = n$.

- stage m: for

$$s \in [2k+1 : m] \setminus L(m-1)$$

1. simulate M_s with input $bin(m)$
2. space used $> r_{m-s}$ or $\leq r_{m-s}$ and $\varphi_s = \Omega$: abort, next s
3. space used $\leq r_{m-s}$ and $\varphi_s(m) \neq \Omega$: set $s(m) = s$, $L(m) = L(m-1) \cup \{s(m)\}$

s			$\dots$		$\dots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$	hard
$*$			$\dots$		$\dots$	
			$\dots$		$\dots$	
$*$			$\dots$		$\dots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\dots$	$\beta_{2k}(n)$	$\dots$	
			$\dots$		$\dots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$	
			$\dots$		$\dots$	mild

column m

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$. Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

Turing machine $M_{j(k)}$: with input $bin(n)$

- for $n \leq v$: print $bin(f(v))$. Done.
- for $n > v$ print $L(v)$; then proceed in stages

$$m = 2k+1, \dots, n$$

where stage m computes $L(m)$ and $bin(f(n))$ if $m = n$.

- stage m: for

$$s \in [2k+1 : m] \setminus L(m-1)$$

1. simulate M_s with input $bin(m)$
2. space used $> r_{m-s}$ or $\leq r_{m-s}$ and $\varphi_s = \Omega$: abort, next s
3. space used $\leq r_{m-s}$ and $\varphi_s(m) \neq \Omega$: set $s(m) = s$, $L(m) = L(m-1) \cup \{s(m)\}$

space used for searching column m: for $m \geq v$

- universal TM U has to simulate M_s with input $bin(m)$ on space r_{m-s}.
- $bin(s) = code(u)$ for $u \in \{0,1,\#\}^*$ with

$$|u| = O(|bin(s)|) = O(\log s)$$

s			$\dots$		$\dots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$	hard
$*$			$\dots$		$\dots$	
			$\dots$		$\dots$	
$*$			$\dots$		$\dots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\dots$	$\beta_{2k}(n)$	$\dots$	
			$\dots$		$\dots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$	
			$\dots$		$\dots$	mild

column m

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$. Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

Turing machine $M_{j(k)}$: with input $bin(n)$

- for $n \leq v$: print $bin(f(v))$. Done.
- for $n > v$ print $L(v)$; then proceed in stages

$$m = 2k+1, \dots, n$$

where stage m computes $L(m)$ and $bin(f(n))$ if $m = n$.

- stage m: for

$$s \in [2k+1 : m] \setminus L(m-1)$$

1. simulate M_s with input $bin(m)$
2. space used $> r_{m-s}$ or $\leq r_{m-s}$ and $\varphi_s = \Omega$: abort, next s
3. space used $\leq r_{m-s}$ and $\varphi_s(m) \neq \Omega$: set $s(m) = s$, $L(m) = L(m-1) \cup \{s(m)\}$

space used for searching column m: for $m \geq v$

- universal TM U has to simulate M_s with input $bin(m)$ on space r_{m-s}.
- $bin(s) = code(u)$ for $u \in \{0,1,\#\}^*$ with

$$|u| = O(|bin(s)|) = O(\log s)$$

- U needs space

$$O(|u|) \cdot r_{m-s} \leq C \cdot (\log s) \cdot r_{m-s} \quad \text{for some } C \text{ almost all } n$$

s			$\dots$		$\dots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$	hard
$*$			$\dots$		$\dots$	
			$\dots$		$\dots$	
$*$			$\dots$		$\dots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\dots$	$\beta_{2k}(n)$	$\dots$	
			$\dots$		$\dots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$	
			$\dots$		$\dots$	mild

column m

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$. Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

Turing machine $M_{j(k)}$: with input $bin(n)$

- for $n \leq v$: print $bin(f(v))$. Done.
- for $n > v$ print $L(v)$; then proceed in stages

$$m = 2k+1, \dots, n$$

  where stage m computes $L(m)$ and $bin(f(n))$ if $m = n$.

- stage m: for

$$s \in [2k+1 : m] \setminus L(m-1)$$

  1. simulate M_s with input $bin(m)$
  2. space used $> r_{m-s}$ or $\leq r_{m-s}$ and $\varphi_s = \Omega$: abort, next s
  3. space used $\leq r_{m-s}$ and $\varphi_s(m) \neq \Omega$: set $s(m) = s$, $L(m) = L(m-1) \cup \{s(m)\}$

space used for searching column m: for $m \geq v$

- universal TM U has to simulate M_s with input $bin(m)$ on space r_{m-s}.
- $bin(s) = code(u)$ for $u \in \{0,1,\#\}^*$ with

$$|u| = O(|bin(s)|) = O(\log s)$$

- U needs space

$$O(|u|) \cdot r_{m-s} \leq C \cdot (\log s) \cdot r_{m-s} \quad \text{for some } C \text{ almost all } n$$

- w.l.o.g. choose k large enough such that

$$s \geq 2k \;\rightarrow C \cdot \log s \leq s$$

s			$\dots$		$\dots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$	hard
$*$			$\dots$		$\dots$	
			$\dots$		$\dots$	
$*$			$\dots$		$\dots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\dots$	$\beta_{2k}(n)$	$\dots$	
			$\dots$		$\dots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$	
			$\dots$		$\dots$	mild

column m

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$. Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

Turing machine $M_{j(k)}$: with input $bin(n)$

- for $n \leq v$: print $bin(f(v))$. Done.
- for $n > v$ print $L(v)$; then proceed in stages

$$m = 2k+1,\dots,n$$

where stage m computes $L(m)$ and $bin(f(n))$ if $m = n$.

- stage m: for

$$s \in [2k+1 : m] \setminus L(m-1)$$

1. simulate M_s with input $bin(m)$
2. space used $> r_{m-s}$ or $\leq r_{m-s}$ and $\varphi_s = \Omega$: abort, next s
3. space used $\leq r_{m-s}$ and $\varphi_s(m) \neq \Omega$: set $s(m) = s$, $L(m) = L(m-1) \cup \{s(m)\}$

space used for searching column m: for $m \geq v$

- universal TM U has to simulate M_s with input $bin(m)$ on space r_{m-s}.
- $bin(s) = code(u)$ for $u \in \{0,1,\#\}^*$ with

$$|u| = O(|bin(s)|) = O(\log s)$$

- U needs space

$$O(|u|) \cdot r_{m-s} \leq C \cdot (\log s) \cdot r_{m-s} \quad \text{for some } C \text{ almost all } n$$

- w.l.o.g. choose k large enough such that

$$s \geq 2k \;\rightarrow C \cdot \log s \leq s$$

- space for simulation

$$\begin{aligned} C \cdot (\log s) \cdot r_{m-s} &\leq s \cdot r_{m-s} \\ &= 2^{\log s} r_{m-s} \\ &\leq r_{m-s+\log s} \quad (r(n) \geq 2n) \\ &\leq r_{m-s/2} \quad (s/2 \geq \log s) \\ &\leq r_{m-k} \end{aligned}$$

s			$\dots$		$\dots$	bound r_{n-s}
1	$\beta_1(1)$	$\beta_1(2)$	$\dots$	$\beta_1(n)$	$\dots$	hard
$*$			$\dots$		$\dots$	
			$\dots$		$\dots$	
$*$			$\dots$		$\dots$	
$2k$	$\beta_{2k}(1)$	$\beta_{2k}(2)$	$\dots$	$\beta_{2k}(n)$	$\dots$	
			$\dots$		$\dots$	
n	$\beta_n(1)$	$\beta_n(2)$	$\dots$	$\beta_n(n)$	$\dots$	
			$\dots$		$\dots$	mild

column m

Table 2: all lines $*$ above line $2k$ that will ever be cancelled are in list $L(v)$. Machine $M_{j(k)}$ does not need to simulate machine M_s above line $2k$.

Turing machine $M_{j(k)}$: with input $bin(n)$

- for $n \leq v$: print $bin(f(v))$. Done.
- for $n > v$ print $L(v)$; then proceed in stages

$$m = 2k+1, \dots, n$$

where stage m computes $L(m)$ and $bin(f(n))$ if $m = n$.

- stage m: for

$$s \in [2k+1 : m] \setminus L(m-1)$$

1. simulate M_s with input $bin(m)$
2. space used $> r_{m-s}$ or $\leq r_{m-s}$ and $\varphi_s = \Omega$: abort, next s
3. space used $\leq r_{m-s}$ and $\varphi_s(m) \neq \Omega$: set $s(m) = s$, $L(m) = L(m-1) \cup \{s(m)\}$

space used for searching column m: for $m \geq v$

- universal TM U has to simulate M_s with input $bin(m)$ on space r_{m-s}.
- $bin(s) = code(u)$ for $u \in \{0,1,\#\}^*$ with

$$|u| = O(|bin(s)|) = O(\log s)$$

- U needs space

$$O(|u|) \cdot r_{m-s} \leq C \cdot (\log s) \cdot r_{m-s} \quad \text{for some } C \text{ almost all } n$$

- w.l.o.g. choose k large enough such that

$$s \geq 2k \rightarrow C \cdot \log s \leq s$$

- space for simulation

$$\begin{aligned} C \cdot (\log s) \cdot r_{m-s} &\leq s \cdot r_{m-s} \\ &= 2^{\log s} r_{m-s} \\ &\leq r_{m-s+\log s} \quad (r(n) \geq 2n) \\ &\leq r_{m-s/2} \quad (s/2 \geq \log s) \\ &\leq r_{m-k} \end{aligned}$$

space for storing lists $L(n)$: for $m \leq n$; on extra track. Space

$$\begin{aligned} O(n \log n) &\leq 2^{n-k} \quad \text{ffa } n \\ &\leq r_{n-k} \end{aligned}$$

3 Speedup theorem for time

Lemma 6. *Let* $r : \mathbb{N} \to \mathbb{N}$ *be total and computable. Then there is a total computable function* f *such that for every machine* M_i *computing* f*, there is a machine* M_j *computing* f *such that* $r(\beta_j(n)) \leq \beta_i(n))$ *for almost all* n

$$\forall i.\ \varphi_i = f\ \to \exists j.\ (\varphi_j = f\ \wedge\ r(\tau_j(n)) \leq \tau_i(n)\ faa\ n)$$

3 Speedup theorem for time

Lemma 6. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\tau_j(n)) \leq \tau_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f \ \to \exists j.\ (\varphi_j = f \ \wedge\ r(\tau_j(n)) \leq \tau_i(n)\ \mathit{faa}\ n)$$

- with input length $\log n$:

$$\begin{aligned}
\beta_i(n)/2 &\leq \beta_i(n) - \log n \\
&\leq \tau_i(n) \\
&\leq 2^{O(\beta_i(n))} \quad \text{if } \varphi_i(n) \neq \Omega \\
&\leq 2^{\beta_i^2(n)} \quad \text{faa } n \text{ (extremely coarse)}
\end{aligned}$$

3 Speedup theorem for time

Lemma 6. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\tau_j(n)) \leq \tau_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f\ \to \exists j.\ (\varphi_j = f\ \wedge\ r(\tau_j(n)) \leq \tau_i(n)\ \mathit{faa}\ n)$$

- with input length $\log n$:

$$\begin{aligned}\beta_i(n)/2 &\leq \beta_i(n) - \log n \\ &\leq \tau_i(n) \\ &\leq 2^{O(\beta_i(n))} \quad \text{if } \varphi_i(n) \neq \Omega \\ &\leq 2^{\beta_i^2(n)} \quad \text{faa } n \text{ (extremely coarse)}\end{aligned}$$

- set

$$r'(x) = 2r(2^{x^2})$$

  by speedup theorem for space (lemma 3) there is f such that for all i with $\varphi_i = f$ there is j with $\varphi_j = f$ such that

$$r'(\beta_j(n)) \leq \beta_i(n)$$

3 Speedup theorem for time

Lemma 6. *Let $r : \mathbb{N} \to \mathbb{N}$ be total and computable. Then there is a total computable function f such that for every machine M_i computing f, there is a machine M_j computing f such that $r(\tau_j(n)) \leq \tau_i(n))$ for almost all n*

$$\forall i.\ \varphi_i = f\ \to \exists j.\ (\varphi_j = f\ \wedge\ r(\tau_j(n)) \leq \tau_i(n)\ faa\ n)$$

- with input length $\log n$:

$$\begin{aligned}
\beta_i(n)/2 &\leq \beta_i(n) - \log n \\
&\leq \tau_i(n) \\
&\leq 2^{O(\beta_i(n))} \quad \text{if } \varphi_i(n) \neq \Omega \\
&\leq 2^{\beta_i^2(n)} \quad \text{faa } n \text{ (extremely coarse)}
\end{aligned}$$

- set

$$r'(x) = 2r(2^{x^2})$$

by speedup theorem for space (lemma 3) there is f such that for all i with $\varphi_i = f$ there is j with $\varphi_j = f$ such that

$$r'(\beta_j(n)) \leq \beta_i(n)$$

Then

$$\begin{aligned}
2r(\tau_j(n)) &\leq 2r(2^{\beta_j^2})(n) \\
&= r'(\beta_j(n)) \\
&\leq \beta_i(n) \\
&\leq 2\tau_i(n)
\end{aligned}$$

4 Gap theorem

- in hierarchy: resource bounds are time or space constructible.
- is this hypothesis necessary?
- answer of gap theorem: yes.
- here shown for space.

def: space complexity classes: here

$$C_{t(n)} = \{\varphi_i \ : \ \beta_i(n) \leq t(n) \text{ faa } n\}$$

4 Gap theorem

- in hierarchy: resource bounds are time or space constructible.
- is this hypothesis necessary?
- answer of gap theorem: yes.
- here shown for space.

def: space complexity classes: here

$$C_{t(n)} = \{\varphi_i \;:\; \beta_i(n) \leq t(n) \text{ faa } n\}$$

4.1 Gap theorem for space

Lemma 7. *For every total recursive function $r : \mathbb{N} \to \mathbb{N}$ there is a total recursive function $t : \mathbb{N} \to \mathbb{N}$ such that*

$$C_{r(t(n))} = C_{t(n)}$$

this means: increasing resource bound from $t(n)$ to $r(t(n))$ does not allow to compute more functions.

4 Gap theorem

- in hierarchy: resource bounds are time or space constructible.
- is this hypothesis necessary?
- answer of gap theorem: yes.
- here shown for space.

def: space complexity classes: here

$$C_{t(n)} = \{\varphi_i \;:\; \beta_i(n) \leq t(n) \text{ faa } n\}$$

4.1 Gap theorem for space

Lemma 7. *For every total recursive function $r : \mathbb{N} \to \mathbb{N}$ there is a total recursive function $t : \mathbb{N} \to \mathbb{N}$ such that*

$$C_{r(t(n))} = C_{t(n)}$$

this means: increasing resource bound from $t(n)$ to $r(t(n))$ does not allow to compute more functions.

- aiming at bound $t(n)$ such that for all i

$$\beta_i(n) \leq r(t(n)) \;\to\; \beta_i(n) \leq t(n)$$

- resp.

$$\beta_i(n) \notin [t(n)+1 : r(t(n))]$$

- define

$$t(n) = \min\{m \;:\; \forall i \leq n.\; \beta_i(n) \notin [m+1 : r(m)]\}$$

4 Gap theorem

- in hierarchy: resource bounds are time or space constructible.
- is this hypothesis necessary?
- answer of gap theorem: yes.
- here shown for space.

def: space complexity classes: here

$$C_{t(n)} = \{\varphi_i \ : \ \beta_i(n) \leq t(n) \text{ faa } n\}$$

4.1 Gap theorem for space

Lemma 7. *For every total recursive function $r : \mathbb{N} \to \mathbb{N}$ there is a total recursive function $t : \mathbb{N} \to \mathbb{N}$ such that*

$$C_{r(t(n))} = C_{t(n)}$$

this means: increasing resource bound from $t(n)$ to $r(t(n))$ does not allow to compute more functions.

- aiming at bound $t(n)$ such that for all i

$$\beta_i(n) \leq r(t(n)) \ \to \ \beta_i(n) \leq t(n)$$

- resp.

$$\beta_i(n) \notin [t(n)+1 : r(t(n))]$$

- define

$$t(n) = \min\{m \ : \ \forall i \leq n.\ \beta_i(n) \notin [m+1 : r(m)]\}$$

- $t(n)$ computable but possibly partial

4 Gap theorem

- in hierarchy: resource bounds are time or space constructible.
- is this hypothesis necessary?
- answer of gap theorem: yes.
- here shown for space.

def: space complexity classes: here

$$C_{t(n)} = \{\varphi_i \ : \ \beta_i(n) \leq t(n) \text{ faa } n\}$$

4.1 Gap theorem for space

Lemma 7. *For every total recursive function $r : \mathbb{N} \to \mathbb{N}$ there is a total recursive function $t : \mathbb{N} \to \mathbb{N}$ such that*

$$C_{r(t(n))} = C_{t(n)}$$

this means: increasing resource bound from $t(n)$ to $r(t(n))$ does not allow to compute more functions.

- aiming at bound $t(n)$ such that for all i

$$\beta_i(n) \leq r(t(n)) \ \rightarrow \ \beta_i(n) \leq t(n)$$

- resp.

$$\beta_i(n) \notin [t(n)+1 : r(t(n))]$$

- define

$$t(n) = \min\{m \ : \ \forall i \leq n.\ \beta_i(n) \notin [m+1 : r(m)]\}$$

- $t(n)$ computable but possibly partial

- define *in a non constructive way*

$$m_n = \max\{\beta_i(n) \ : \ i \leq n \ , \ \beta_i(n) \neq \Omega\}$$

4 Gap theorem

- in hierarchy: resource bounds are time or space constructible.
- is this hypothesis necessary?
- answer of gap theorem: yes.
- here shown for space.

def: space complexity classes: here

$$C_{t(n)} = \{\varphi_i \; : \; \beta_i(n) \leq t(n) \text{ faa } n\}$$

4.1 Gap theorem for space

Lemma 7. *For every total recursive function $r : \mathbb{N} \to \mathbb{N}$ there is a total recursive function $t : \mathbb{N} \to \mathbb{N}$ such that*

$$C_{r(t(n))} = C_{t(n)}$$

this means: increasing resource bound from $t(n)$ to $r(t(n))$ does not allow to compute more functions.

- aiming at bound $t(n)$ such that for all i

$$\beta_i(n) \leq r(t(n)) \; \to \; \beta_i(n) \leq t(n)$$

- resp.

$$\beta_i(n) \notin [t(n)+1 : r(t(n))]$$

- define

$$t(n) = \min\{m \; : \; \forall i \leq n. \; \beta_i(n) \notin [m+1 : r(m)]\}$$

- $t(n)$ computable but possibly partial

- define *in a non constructive way*

$$m_n = \max\{\beta_i(n) \; : \; i \leq n \, , \; \beta_i(n) \neq \Omega\}$$

Then

$$\forall i \leq n. \; \beta_i(n) \notin [m_n + 1 : r(m_n)]$$

4 Gap theorem

- in hierarchy: resource bounds are time or space constructible.
- is this hypothesis necessary?
- answer of gap theorem: yes.
- here shown for space.

def: space complexity classes: here

$$C_{t(n)} = \{\varphi_i \ : \ \beta_i(n) \leq t(n) \text{ faa } n\}$$

4.1 Gap theorem for space

Lemma 7. *For every total recursive function $r : \mathbb{N} \to \mathbb{N}$ there is a total recursive function $t : \mathbb{N} \to \mathbb{N}$ such that*

$$C_{r(t(n))} = C_{t(n)}$$

this means: increasing resource bound from $t(n)$ to $r(t(n))$ does not allow to compute more functions.

- aiming at bound $t(n)$ such that for all i

$$\beta_i(n) \leq r(t(n)) \ \to \ \beta_i(n) \leq t(n)$$

- resp.

$$\beta_i(n) \notin [t(n)+1 : r(t(n))]$$

- define

$$t(n) = \min\{m \ : \ \forall i \leq n.\ \beta_i(n) \notin [m+1 : r(m)]\}$$

- $t(n)$ computable but possibly partial

- define *in a non constructive way*

$$m_n = \max\{\beta_i(n) \ : \ i \leq n \ , \ \beta_i(n) \neq \Omega\}$$

Then

$$\forall i \leq n.\ \beta_i(n) \notin [m_n + 1 : r(m_n)]$$

- $t(n) \leq m_n$ is total.

- Let $\varphi_i \in C_{r(t(n))}$, i.e.

$$\beta_i(n) \leq r(t(n)) \text{ faa } n$$

and let $n \geq i$. Then

$$\beta_i(n) \notin [t(n)+1 : r(t(n))]$$

hence

$$\beta_i(n) \leq t(n)$$

4.2 Making the resource bound $t(n)$ monotonic.

Lemma 8. *For every total recursive function $r : \mathbb{N} \to \mathbb{N}$ there is a monotonic total recursive function $t : \mathbb{N} \to \mathbb{N}$ such that*

$$C_{r(t(n))} = C_{t(n)}$$

4.2 Making the resource bound $t(n)$ monotonic.

Lemma 8. *For every total recursive function $r : \mathbb{N} \to \mathbb{N}$ there is a monotonic total recursive function $t : \mathbb{N} \to \mathbb{N}$ such that*

$$C_{r(t(n))} = C_{t(n)}$$

- define

$$\begin{aligned} t(1) &= 1 \\ m(n) &= \min\{m \ : \ \forall i \leq n.\ \beta_i(n) \notin [t(n)+m+1 : r(t(n)+m)]\} \\ t(n+1) &= t(n)+m(n) \end{aligned}$$

- completing the proof: exercise