

A Different Proof of the Time Hierarchy Theorem

András Z. Salamon and Michael Wehar

FICS 2026-02-24

Time hierarchy

[Hartmanis and Stearns 1965, Hennie and Stearns 1966]

Time Hierarchy Theorem: for well-behaved $t_1(n) \in o(t_2(n))$

$$\text{DTIME}(t_1(n)) \subsetneq \text{DTIME}(t_2(n) \log(t_2(n)))$$

often proved via universal simulation + diagonal argument

...can be viewed as a fixed point argument

Diagonal arguments and fixed points

diagonal arguments have a common categorical structure

[Lawvere 1969, Corollary 1.2] $\Delta(x) = (x, x)$

if $\alpha : Y \rightarrow Y$ has **no fixed point**, then for any $f : A \times A \rightarrow Y$
the diagonal $g = \alpha \circ f \circ \Delta$ is *not representable in f*

Cantor, Russell, Gödel, Tarski, **hierarchy theorems...**

Today: a different construction fitting this pattern

Time hierarchy as diagonal argument

Argument [Yanofsky 2022, Theorem 6.20]:

- category of total computable functions
- $Y = \text{Bool}$, with $\alpha = \text{NOT}$ (no fixed point)
- $f(M, w) = "M \text{ accepts } w \text{ in } t_1(n) \text{ time?}"$
- diagonal: $g(M) = (\alpha \circ f \circ \Delta)(M) = \neg f(M, M)$

hence g not representable in f (so $g \notin \text{DTIME}(t_1(n))$)

also: $g \in \text{DTIME}(t_2(n) \log t_2(n))$

g witnesses $\text{DTIME}(t_1(n)) \subsetneq \text{DTIME}(t_2(n) \log t_2(n))$

Our approach: different witness

The sloth functions

Gödel numbering $(M_i)_{i \in \mathbb{N}}$

$\text{time}(M_i) = \# \text{steps taken by } M_i \text{ on blank tape } (\in \mathbb{N} \cup \{\infty\})$

For well-behaved $t(n)$, define:

$$SF_t : \mathbb{N} \rightarrow \mathbb{N}, SF_t(n) = \min\{i \in \mathbb{N} \mid n \leq \text{time}(M_i) \leq t(n)\}$$

well-behaved: non-decreasing, superadditive, time-constructible, $\geq n$

Roughly: “smallest machine that halts in $[n, t(n)]$ steps”

Why is SF_t interesting?

$$SF_t(n) = \min\{i \in \mathbb{N} \mid n \leq \text{time}(M_i) \leq t(n)\}$$

$SF_t(n)$ needs M_i satisfying **two competing constraints**:

- $\text{time}(M_i) \geq n$ (rules out “halt immediately”)
- $\text{time}(M_i) \leq t(n)$ (rules out “run forever”)

few states \rightarrow small index BUT less control over runtime

Empirical observation



Growth rate of sloth functions

$SF_t(n)$ is $O(\text{poly}(\log n))$ for $t(n) \geq 2n$

Proof idea: given n , find 2^m with $n \leq 2^m \leq 2n$;

there's an $O(\frac{\log m}{\log \log m})$ -state TM running $\geq 2^m$ steps;

its index is $O(\text{poly}(m)) = O(\text{poly}(\log n))$

Sloth \approx inverse of busy beaver cf. [Antunes et al. 2017]

recall: busy beaver eventually $>$ any computable bound

Basic properties

Prop 1: $SF_t(n)$ is $\omega(1)$ (superconstant)

Proof: for any k , let $m = 1 + \max\{\text{time}(M_i) \mid i \leq k, M_i \text{ halts}\}$;
then $SF_t(m) > k$ □

Prop 2: $SF_t(n)$ is computable in $O(t(n) \cdot SF_t(n) \cdot \log SF_t(n))$

Standard idea: in turn simulate M_0, M_1, \dots for $t(n)$ steps,
first making $t(n)$ copies of M_i 's description;
stop when M_i halts in $[n, t(n)]$

The key lemma

Lemma: If $f(n)$ is $o(t(n))$ -time computable and superconstant, then $SF_t(n) < f(n)$ for **infinitely many** n

Idea: build machine $\mathcal{M}(d)$ that searches for k with $2^d < f(2^k)$;
 $\mathcal{M}(d)$ has $O(\frac{\log d}{\log \log d})$ states, runs $\geq 2^k$ steps

therefore $SF_t(2^k) \leq p(d) < 2^d < f(2^k)$ (p is polynomial)

The lower bound

Theorem: $SF_t(n)$ is not $o(t(n))$ -time computable

Proof: suppose SF_t is $o(t(n))$ -time computable;

by Prop 1, SF_t is superconstant;

now apply the key lemma with $f = SF_t$:

$$SF_t(n) < SF_t(n) \text{ for infinitely many } n \quad \perp$$

The diagonal structure

Define $G : \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}$ as $G(f) = \min\{n \mid SF_t(n) < f(n)\}$

key lemma: $G(f)$ defined for any $o(t(n))$ -time superconstant f

If SF_t were such an f :

$$SF_t(G(SF_t)) < SF_t(G(SF_t)) \quad \perp$$

$\{\langle n, SF_t(n) \rangle\}$ is witness

The hierarchy theorem (with sloths)

Corollary: For $t_1(n) \cdot SF_{t_2}(n) = o(t_2(n))$,

$$\text{DTIME}(t_1(n)) \subsetneq \text{DTIME}(t_2(n) \cdot SF_{t_2}(n) \cdot \log SF_{t_2}(n)).$$

Proof:

- Upper bound: SF_{t_2} computable in $O(t_2 \cdot SF_{t_2} \cdot \log SF_{t_2})$
- Lower bound: SF_{t_2} not $o(t_1(n))$ -time computable

now encode SF_{t_2} as a language to separate the classes □

Summary

- hierarchy theorems are diagonal arguments
- we give a **different witness** based on sloth function SF_t
- $SF_t(n) =$ “smallest machine” halting in $[n, t(n)]$ steps
- key property: $SF_t < f$ i.o. for $f \in \text{DTIME}(o(t(n)) \cap \omega(1))$
- lower bound via self-reference: $SF_t < SF_t$ is absurd
- this matches Lawvere's $\alpha \circ f \circ \Delta$ pattern

Open questions

1. Exact growth rate of $SF_t(n)$?
2. Relationship between SF_t for varying t ?
3. Nondeterministic hierarchy via sloth functions?
4. Tighter bounds via this approach?

an extended version is in preparation

Andras.Salamon@st-andrews.ac.uk

Anti-fixed points of compose

$\text{FDTIME}(t(n)) = \{f : \mathbb{N} \rightarrow \mathbb{N} \mid f \text{ is } O(t(n))\text{-time computable}\}$

Define $\text{compose} : \mathcal{P}(\mathbb{N}^{\mathbb{N}}) \rightarrow \mathcal{P}(\mathbb{N}^{\mathbb{N}})$ as

$$\text{compose}(X) = \{f \circ g \mid f \in \text{FDTIME}(n^2), g \in X\}$$

Lemma: $\text{compose}(\text{FDTIME}(t(n))) = \text{FDTIME}((t(n))^2)$

Theorem: $\text{FDTIME}(t(n)) \neq \text{compose}(\text{FDTIME}(t(n)))$

Every natural $\text{FDTIME}(t(n))$ class is a non-fixed-point of compose

Encoding and indices

Encode TM with q states via self-delimiting binary: each transition $(q_i, \sigma, q_j, \sigma', d)$ needs at most $5 \log q + C$ bits

Encoding length: $O(q \log q)$ bits \Rightarrow index $\leq 2^{O(q \log q)}$

For $\mathcal{M}(d)$ with $q = O\left(\frac{\log d}{\log \log d}\right)$ states:

$$q \log q = O\left(\frac{\log d}{\log \log d} \cdot \log \log d\right) = O(\log d)$$

So index $= 2^{O(\log d)} = d^{O(1)}$ **polynomial in d , as claimed**

$(\log \log d)^{-1}$ factor in state count absorbs the encoding blowup

Why upper bounds on SF_t are difficult

For generous $t(n)$ ($= 2^{2^n}$) need smallest M_i with $\text{time}(M_i) \geq n$

Attempt 1: walk through n states in a line. $\text{time} = n$

Index $\approx n^n$ (terrible)

Attempt 2: binary counter, $O(\log n)$ states. $\text{time} \approx n$

Index $= 2^{O(\log n \cdot \log \log n)} = n^{O(\log \log n)}$ (superpolynomial in n)

Attempt 3: encode $m = \lceil \log n \rceil$ in $O(\frac{\log m}{\log \log m})$ states, use 2^m

Index $= 2^{O(\log m)} = \text{poly}(\log n)$

encode $\log n$ (not n), then exponentiate: each level adds complexity [Chaitin 1966, Chistikov 2014, Wehar 2013]

Sloth vs. “Busy Beaver logical depth”

Antunes, Bauwens, Souto & Teixeira [2017] define:

$$\text{bb}(n) = \min\{|p| : U(p) \text{ halts and } \text{time}(p) \geq n\}$$

“Inverse Busy Beaver”: converts runtime to *description length*

measures	$\text{bb}(n)$	$SF_t(n)$
runtime	description length	machine index
purpose	lower bound only ($\geq n$)	window $[n, t(n)]$
setting	depth of strings	separating DTIME classes
	Kolmogorov complexity	constructive

SF_t is constructive, time-bounded, index-based analogue of bb