Computer Science 511

Nondeterministic Time: Speedup, Emulation, and a Nondeterministic Time Hierarchy

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #9

Goals for Today

- Presentation of results concerning the ***speedup*** and ***emulation*** of nondeterministic computation, along with a ***nondeterministic time hierarchy theorem***.
- The proof of the "nondeterministic time hierarchy theorem" is quite complicated — and it will not be necessary to understand this. Ideally, students should understand the ***results*** presented in these notes (ideally, along with simpler proof) and understand how various results can be applied.

A Nondeterministic Linear Speedup Theorem

Theorem #1 (Nondeterministic Linear Speedup Theorem): Let $L \subseteq \Sigma^\star$ and let $f : \mathbb{N} \to \mathbb{N}$ be a function such that $f(m) \geq m$ for every integer $m \geq 0$. Suppose there exists a k-tape nondeterministic Turing machine M (with one-way infinite tapes) that decides L, such that the depth of the computation tree for ω is at most $f(|\omega|)$, for every input string $\omega \in \Sigma^\star$.

Let $\widehat{k} = \max(k, 2)$ and let ε be any real number such that $\varepsilon > 0$. Then there exists a $\widehat{k}$-tape nondeterministic Turing machine $\widehat{M}$ (with one-way infinite tapes) that decides L, such that the depth of the computation tree for ω is at most $\varepsilon \cdot f(|\omega|) + |\omega| + 3$ for every input string $\omega \in \Sigma^\star$.

A Nondeterministic Linear Speedup Theorem

How This is Proved: Modification of proof of the (deterministic) Linear Speedup Theorem:

- As in the proof of the result for deterministic computation, the nondeterministic Turing machine, $\widehat{M}$, should use a tape alphabet, $\widehat{\Gamma}$ including symbols that can represent the contents of m symbols on M's tape(s).
- The ***initialization phase*** — in which a "compressed" version of the input string — should be deterministic, and exactly the same as the initialization phase for the simulation described in the proof of the original result.

A Nondeterministic Linear Speedup Theorem

- As in the original simulation, m steps in the computation tree should be replaced by at most seven steps — so that m levels in the computation tree, for the input string and the original nondeterministic Turing machine, are replaced by at most seven levels in the computation tree for the same input string, using the new nondeterministic Turing machine.
- The first two steps should be the same as in the simulation used in the original proof (so that these are deterministic).
- When information is read at the beginning of the third step, the nondeterministic Turing machine has all the information that is needed so that — using at its finite control — the decision can be made about how to simulate the next m steps of the original nondeterministic Turing machine.

A Nondeterministic Linear Speedup Theorem

- If the original nondeterministic Turing machine's computation includes an accepting state at the next m levels below the node for the current configuration, then the new nondeterministic Turing machine should end the third step by ***accepting***.
- Otherwise, if the subtree of the computation tree, whose root stores the current configuration, has depth at most m, then the new nondeterministic Turing machine should end the third step by ***rejecting***.

A Nondeterministic Linear Speedup Theorem

- Finally — if the subtree of the computation tree, whose root stores the current configuration, has depth greater than m and does not have any accepting configuration with distance at most m from the root — then the new nondeterministic Turing machine's transition function, for the third move, should be used to nondeterministically choose one of the configurations in the configuration tree, that represents a non-halting configuration, and that is m levels farther down from the node for the current configuration.

  The rest of the third move, along with the fourth, fifth, sixth and (if necessary) seventh move should be used to update the tapes and reposition heads, to obtain a representation of the selected configuration, just like in the original simulation.

A Nondeterministic Linear Speedup Theorem

- This produces a computation tree that includes the same "root" node (representing the start configuration for the input string) along with the internal nodes, from the original computation tree for this input string, that appeared a levels m, $2m$, $3m$, and so on. An examination of this computation tree should be sufficient to confirm that ω is accepted by $\widehat{M}$ if and only if it was accepted by M.
- One can also see that the computation tree for every input string (and $\widehat{M}$) must be finite, because the computation tree for every string and M was finite. Thus $\widehat{M}$ ***decides*** the same language, L, that M does.
- The proof that the depth of the computation tree is reduced (as claimed) is almost exactly the same as the proof that the number of steps has been reduced, as given in the proof of the original claim. □

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

- As noted in Lecture #3, a k-tape deterministic Turing machine's execution can be simulated using either a one-tape Turing machine or a two-tape Turing machine, with a polynomial increase in the number of steps used.
- A k-tape ***nondeterministic*** Turing machine's execution can be simulated, using a nondeterministic Turing machine with *three* tapes, even more efficiently.
- Consider, in particular, a 3-tape nondeterministic Turing machine

$$\widehat{M} = (\widehat{Q}, \widehat{\Sigma}, \widehat{\Gamma}, \widehat{\delta}, \widehat{q}_0, \widehat{q}_{\text{accept}}, \widehat{q}_{\text{reject}})$$

where $\widehat{\Sigma} = \Sigma \cup \{\#\}$, for a symbol "#" that is not already in M's tape alphabet, Γ.

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

- Suppose that $\widehat{M}$ should accept the set of strings in $\widehat{\Sigma}^\star$ with the form

$$\omega\#^T$$

such that

  - T is a non-negative integer, and
  - $\omega \in \Sigma^\star$ such that M's computation tree for ω includes an accepting configuration with distance at most T from the root (so that — in a sense — M accepts ω using at most T steps).

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

Consider a set S_T of new symbols with the form

$$\xi_{q,\sigma_1,\sigma_2,\ldots,\sigma_k,r,\tau_1,d_1,\tau_2,d_2,\ldots,\tau_k,d_k}$$

such that

- $q, r \in Q$,
- $\sigma_1, \sigma_2, \ldots, \sigma_k, \tau_1, \tau_2, \ldots, \tau_k \in \Gamma$,
- $d_1, d_2, \ldots, d_k \in \{\mathtt{L}, \mathtt{R}, \mathtt{S}\}$, and
- $(r, (\tau_1, d_1), (\tau_2, d_2), \ldots, (\tau_k, d_k)) \in \delta(q, \sigma_1, \sigma_2, \ldots, \sigma_k)$.

These should also be "new" symbols, so that $S_T \cap (\Gamma \cup \{\#\}) = \emptyset$.

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

- Let $\widehat{M}$'s tape alphabet be

$$\widehat{\Gamma} = \Gamma \cup \{\#\} \cup S_T.$$

- $\widehat{M}$ will include three tapes, with the following roles
  - The first tape is the ***input tape***.
  - The second tape is a ***transition tape:*** It, effectively, stores a path down the computation tree of M for ω, with length at most T, that will be checked.
  - The third tape will be used to ***represent M's tapes*** — one after another — as described below.

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

Validation:

- An execution of $\widehat{M}$, on an input string $\zeta \in \widehat{\Sigma}^{\star}$, will begin with a deterministic ***validation*** phase: If the input string consists of a sequence of zero or more symbols in Σ, followed by one or more copies of "#" – with no other symbols after that — then the computation should proceed with the next phase. The input should be ***rejected*** otherwise.
- This can be carried out using a single sweep to the right over the input string, so that (if the input tape head is left at the right end of the input) it can be carried out using at most $|\zeta| + 1$ steps.[1]

[1] This should include "marking" and remembering the leftmost cell of the input tape so that it can be found again, later on.

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

Guessing Transitions:

- Suppose, now, that ζ has not been rejected — so it is a string $\omega\#^T$, for some string $\omega \in \Sigma^\star$ and for some positive integer T.
- In the next step, after copy of "#", a sequence of $t \leq T$ symbols

$$\chi_1, \chi_2, \cdots \chi_t$$

should be nondeterministically guessed, and written onto Tape #2. (The machine can be moving back left on the input tape, over copies of "#" as this happens, in order to determine that at most T symbols from S_T are written out.) This can also be carried out using at most $T \leq |\zeta|$ steps.

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

Validation: Do the states make sense?

For $1 \leq i \leq t$, suppose that χ_i represents a move

$$(s_i, (\tau_{i,1}, d_{i,1}), (\tau_{i,2}, d_{i,2}), \ldots, (\tau_{i,k}, d_{i,k})) \in \delta(r_i, \sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,k}).$$

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

Using another sweep over the symbols on Tape #2, one should confirm that each of the following conditions are satisfied — ***rejecting*** if this is not the case:

- $r_1 = q_0$ (so that the sequence of guessed transitions begins with the start state).
- $r_{j+1} = s_j$ for every integer j such that $1 \leq j \leq t - 1$ (so these moves can be followed, in order).
- $s_t = q_{\text{accept}}$ (so this sequence would result in ω being accepted by M).

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

Validation: Would the tapes make sense?

For $1 \leq j \leq k$, the simulation should deterministically check whether the contents of M's i^{th} tape would allow each one of the guessed transitions to be applied:

- When $j = 1$ the third tape should be initialized to store

$$\#\omega \sqcup \#$$

with the tape head pointing to the leftmost symbol in ω (or the copy of "$\sqcup$" between copies of "#", if $\omega = \lambda$).

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

- When $2 \leq j \leq k$, this should start by initializing the third tape to store

  $$\# \sqcup \#$$

  with the tape head pointing to the copy of "$\sqcup$" between the copies of "#", instead.

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

For $1 \leq i \leq t$, the transition represented by the symbol χ_j should be applied: If χ_i represents the move

$$(s_i, (\tau_{i,1}, d_{i,1}), (\tau_{i,2}, d_{i,2}), \ldots, (\tau_{i,k}, d_{i,k})) \in \delta(r_i, \sigma_{i,1}, \sigma_{i,2}, \ldots, \sigma_{i,k})$$

then

- the simulation should end with ζ being ***rejected*** if symbol $\sigma_{i,j}$ is not visible on Tape #3.
- Otherwise the copy of $\sigma_{i,j}$ on Tape #3 should be replaced by a copy of $\tau_{i,j}$, and the third tape head should be moved in direction $d_{i,j}$.

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

- ***One Special Case:*** If $d_{i,j}$ was "L" and "#" is now visible then the tape head for Tape #3 should be moved back to the right again (because this was an attempt to move left on the leftmost cell of the j^{th} tape).
- ***Another Special Case:*** If $d_{i,j}$ was "R" and "#" is now visible then this should be changed to "␣", the symbol to the right should be changed to "#", and the tape head should be moved back to the "␣" that was just written — extending the part of the tape in use.

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

If all t moves are checked, without ζ being rejected, then...

- If $1 \leq j \leq k-1$ then Tape #3 should be erased (by moving to the right copy of "#" and erasing symbols, while moving left, until the left "#" is found — and this tape should initialized so that M's $j+1^{\text{st}}$ tape can be checked.
- If $j = k$ then — since all tapes have been checked — ζ should be ***accepted***.

Nondeterministic Simulation of a k-Tape Nondeterministic Turing Machine

One can show the following.

- $\widehat{M}$ accepts a string $\zeta \in \widehat{\Sigma}^\star$ if and only if $\zeta = \omega\#^T$ for some string $\omega \in \Sigma^\star$, and for some positive integer T, such that M accepts ω using at most T steps.
- The number of steps used by $\widehat{M}$, when executed on an input string $\zeta \in \widehat{\Sigma}^\star$ is
  - In $O(|\omega| + k \times T)$ if $\zeta = \omega\#^T$, for a string $\omega \in \Sigma^\star$ and for a positive integer T, and is
  - at most $|\zeta| + 1$, otherwise.

Encoding a Nondeterministic Turing Machine

Recall, from Lecture #4, that a deterministic multi-tape Turing machine can be encoded as a string of symbols over an alphabet

$$\Sigma_{\text{UTM}} = \{\texttt{s}, \texttt{q}, \texttt{0}, \texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}, \texttt{5}, \texttt{6}, \texttt{7}, \texttt{8}, \texttt{9}, \texttt{L}, \texttt{R}, \texttt{S}, \texttt{Y}, \texttt{N}, \texttt{(}, \texttt{)}, \texttt{,}, \texttt{;}, \texttt{\#}\}$$

A k-tape ***nondeterministic*** Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

can be encoded, using a string of symbols over this alphabet, in almost exactly the same way.

Encoding a Nondeterministic Turing Machine

- States, symbols in Γ, and directions of motion ("`L`", "`R`" or "`S`") can be encoded as strings in $\Sigma^\star_{\text{UTM}}$ in exactly the same way as for deterministic Turing machines.
- An element of $Q \times \Gamma^k$ (that is, a possible input for the transition function) can be encoded as a comma-separated list of component: For a state $q \in Q$ and tape symbols $\sigma_1, \sigma_2, \ldots, \sigma_k$, the input $(q, \sigma_1, \sigma_2, \ldots, \sigma_k)$ can be encoded as

$$\mathrm{e}(q), \mathrm{e}(\sigma_1), \mathrm{e}(\sigma_2), \ldots, \mathrm{e}(\sigma_k)$$

  ... a string with length in $O(\log |Q| + k \log |\Gamma|)$.

Encoding a Nondeterministic Turing Machine

- A move, in $Q \times (\Gamma \times \{\mathtt{L}, \mathtt{R}, \mathtt{S}\})^k$, which might be an element of one of the outputs of a transition function, can be encoded in a similar way — with brackets added: If $r \in Q$, $\tau_1, \tau_2, \ldots, \tau_k \in \Gamma$, and $d_1, d_2, \ldots, d_k \in \{\mathtt{L}, \mathtt{R}, \mathtt{S}\}$, then

$$(r, (\tau_1, d_1), (\tau_2, d_2), \ldots, (\tau_k, d_k))$$

  can be encoded as

$$(\mathrm{e}(r), ((\mathrm{e}(\tau_1), \mathrm{e}(d_1)), (\mathrm{e}(\tau_2), \mathrm{e}(d_2)), \ldots, (\mathrm{e}(\tau_k), \mathrm{e}(d_k))))$$

  ... also a string with length in $O(\log |Q| + k \log |\Gamma|)$.

Encoding a Nondeterministic Turing Machine

- Suppose that $q \in Q$, $\sigma_1, \sigma_2, \ldots, \sigma_k \in \Gamma$, and

$$\delta(q, \sigma_1, \sigma_2, \ldots, \sigma_k) = \{\varphi_1, \varphi_2, \ldots, \varphi_m\}$$

for a non-negative integer m and for

$$\varphi_1, \varphi_2, \ldots, \varphi_m \in Q \times (\Gamma \times \{\mathtt{L}, \mathtt{R}, \mathtt{S}\})^k.$$

Suppose furthermore, that $\varphi_1, \varphi_2, \ldots, \varphi_m$ are ordered in lexicographic order (that is, by nondecreasing state; for moves with the same state, by first tape symbol; and so on).

Encoding a Nondeterministic Turing Machine

- Then the transitions $\delta(q, \sigma_1, \sigma_2, \ldots, \sigma_k)$ can be encoded using the string

$$(\mathsf{e}(q), \mathsf{e}(\sigma_1), \mathsf{e}(\sigma_2), \ldots, \mathsf{e}(\sigma_k), (\mathsf{e}(\varphi_1), \mathsf{e}(\varphi_2), \ldots, \mathsf{e}(\varphi_m)))$$

— a string with length in $O((m+1)(\log|Q| + k\log|\Gamma|)$, where

$$m = |\delta(q, \sigma_1, \sigma_2, \ldots, \sigma_k) \leq 2^k \times |Q| \times |\Gamma|^k.$$

Encoding a Nondeterministic Turing Machine

- The transition function, δ, can now be encoded as a string $\mathrm{e}(\delta) \in \Sigma^{\star}_{\mathrm{UTM}}$ — a comma-separated list of encodings of transitions $\delta(q, \sigma_1, \sigma_2, \ldots, \sigma_k$, for inputs $(q, \sigma_1, \sigma_2, \ldots, \sigma_k)$, such that $q \in Q \setminus \{q_{\mathrm{accept}}, q_{\mathrm{reject}}\}$ and $\sigma_1, \sigma_2, \ldots, \sigma_k \in \Gamma$, sorted in lexicographic order — enclosed by brackets.
- The string $\mathrm{e}(\delta)$ has length in $\Omega(|Q| \times |\Gamma|^k \log(|Q| \times |\Gamma|^k))$, and in $O(2^k \times |Q|^2 \times |\Gamma|^{2k} \times \log(|Q| \times |\Gamma|^k))$.

Encoding a Nondeterministic Turing Machine

- A k-tape nondeterministic Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

  can now be encoded exactly as a k-tape deterministic Turing machine can be encoded — except that the encoding $\text{e}(\delta)$ of the transition function, δ, is now as described above.
- Like the length of the encoding of the transition function, the length of the encoding of M is in

$$O(2^k \times |Q|^2 \times |\Gamma|^{2k} \times \log(|Q| \times |\Gamma|^k)).$$

Encoding a Nondeterministic Turing Machine

- Recall that the input alphabet Σ_{UTM} also includes a symbol, "#", which has not been used to define encodings of nondeterministic Turing machines — yet.
- This will be used to define arbitrarily long encodings of nondeterministic Turing machines: If $\zeta \in \Sigma^{\star}_{\mathrm{UTM}}$ is an encoding of a nondeterministic Turing machine M then $\zeta\#$ is a an encoding of the same nondeterministic Turing machine M.
- It follows, by a repeated application of this rule, that $\zeta\#^{h}$ is also an encoding of M, for every non-negative integer h.

Encoding a Nondeterministic Turing Machine

- In order to simplify some constructions it will be assumed that any string $\zeta \in \Sigma^{\star}_{\mathrm{UTM}}$, which *does not* encode a nondeterministic Turing machine (as now described) represents a three-state nondeterministic Turing machine which always rejects its input string using one step — so that its language is empty.

Encoding a Nondeterministic Turing Machine

- Since $|\Sigma_{\text{UTM}}| = 22$, there is a bijective function

$$\varphi : \Sigma_{\text{UTM}} \to \{i \in \mathbb{N} \mid 0 \leq i \leq 21\}.$$

  Furthermore, we may choose φ so that $\varphi(\,;\,) = 0$.
- This can be used (extending φ so that it also maps strings in $\Sigma^{\star}_{\text{UTM}}$ to natural numbers) to obtain an ***enumeration***

$$M_0, M_1, M_2, \ldots$$

  of nondeterministic Turing machine, in which every nondeterministic Turing machine is listed ***infinitely often***.
- See the discussion of enumerations of Turing machines, in Lecture #4, for details.

Nondeterministic Universal Turing Machines — Problems

Each of the following problems can now be considered.

- *Validation* of an encoding of a nondeterministic Turing machine M.
- *Simulation* of a computation: The input also includes an encoding of an input string ω for M, with the execution of M on input ω to be simulated.
- *Timed Simulation* of a computation: The input also includes an encoding (in unary) of a positive integer T, so that nondeterministic Turing machine M, input string ω for M and integer T are encoded in the input string; the input should be accepted if and only if M accepts ω using at most T steps.

Nondeterministic Universal Turing Machines — Validation

Let $L_{\text{NTM}} \subseteq \Sigma^{\star}_{\text{UTM}}$ be the language of valid encodings of nondeterministic multi-tape Turing machines, as described above.

- There exists a *deterministic* multi-tape Turing machine that decides the language L_{NTM}, using at number of steps that is at most linear in the length of the input string.
- This can be shown by modifying the proof that the language $L_{\text{TM}} \subseteq \Sigma^{\star}_{\text{UTM}}$, of encodings of *deterministic* Turing machines, is decidable in linear time.

Nondeterministic Universal Turing Machines — Simulation

Next consider the following languages — which are both subsets of $\Sigma^{\star}_{\mathrm{UTM}}$:

- $L_{\mathrm{NTM+I}}$: The language of encodings of nondeterministic Turing machines M and input strings ω for M.
- $A_{\mathrm{NTM+I}}$: The language of encodings of nondeterministic Turing machines M and input strings ω for M such that M ***accepts*** ω.

Once again, an argument for encodings of *deterministic* Turing machines, and their inputs, can be used to show that there is a deterministic multi-tape Turing machine that decides the language $L_{\mathrm{NTM+I}}$, using a number of steps that is at most linear in the length of the input string.

Nondeterministic Universal Turing Machines — Simulation

- It is also possible to describe a multi-tape deterministic Turing machine whose language is $A_{\text{NTM+I}}$ — so this language is ***recognizable***.
- However, one can also show that

$$A_{\text{TM+I}} \preceq_{\text{M}} A_{\text{NTM+I}}$$

so that $A_{\text{NTM+I}}$ is ***undecidable***, because $A_{\text{TM+I}}$ is.

Nondeterministic Timed Simulation

Now consider the following languages — which are both subsets of $\Sigma^{\star}_{\text{UTM}}$:

- $L_{\text{NTM+I+Time}}$: The language of encodings of nondeterministic Turing machines M, input strings ω for M, and positive integers T (encoded in unary).
- $A_{\text{NTM+I+Time}}$: The language of encodings of nondeterministic Turing machines M input strings ω for M, and positive integers T (encoded in unary) such that M ***accepts*** ω using at most T steps.

An argument for encodings of *deterministic* Turing machines, their inputs, and time bounds can be modified to show that there is a deterministic multi-tape Turing machine that decides the language $L_{\text{NTM+I+Time}}$ using a number of steps that is at most linear in the length of the input string.

Nondeterministic Timed Simulation

- A "nondeterministic universal Turing machine", which decides the language $A_{\text{NTM+I+Time}}$, can be be based on the simulation of a nondeterministic k-tape Turing machine, by a nondeterministic three time, given at the beginning of these lecture notes.

Consider an execution of such a "nondeterministic universal Turing machine" on an input string $\zeta \in \Sigma^{\star}_{\text{UTM}}$ with length N.

- It can be (deterministically) decided whether $\zeta \in L_{\text{NTM+I+Time}}$ using a number of steps at most linear in $|\zeta|$ — and the input string can be ***rejected*** if this is not the case.

Nondeterministic Timed Simulation

Suppose, instead, that $\zeta \in L_{\text{NT+I+Time}}$ and that ζ encodes a nondeterministic Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}),$$

an input string $\omega \in \Sigma^\star$ with length n, and a positive integer T. Let ℓ be the length of the substring of ζ that encodes M's transition function, δ — so that

- $\ell \leq |\zeta|$,
- $\ell \in O(2^k \times |Q|^2 \times |\Gamma|^{2k} \times \log(|Q| \times |\Gamma|^k))$, and
- $\ell \in \Omega(|Q| \times |\Gamma|^k \times \log(|Q| \times |\Gamma|^k))$.

Nondeterministic Timed Simulation

- A sequence of at most T ***transitions*** to be applied, during a simulation of an execution of M on input ω, can be nondeterministically guessed, with their encodings written onto a tape, using a number of steps at most linear in $\ell \times T$.
- The encoding of the sequence of transitions, to be used, will be a string with length in

$$O(T \times \log(|Q| \times |\Gamma|^k)) = O(T \times (\log |Q| + k \times \log |\Gamma|)).$$

- Processing on each of M's tapes can be considered using a number of steps in $O(T \times (\log |Q| + k \times \log |\Gamma|))$ steps as well.
- Indeed, the nondeterministic universal Turing machine's computation on input ζ can be completed (after confirming that $\zeta \in L_{\text{NTM+I+Time}}$) using a number of steps at most linear in $O(T \times (\ell + k \log |Q| + k^2 \log |\Gamma|))$.

A Nondeterministic-Time Hierarchy Theorem

Nondeterministic Time Hierarchy Theorem: Let $f, g : \mathbb{N} \to \mathbb{N}$ be time constructible functions such that f and g are nondecreasing functions, $f(n) \geq n$ for all $n \in \mathbb{N}$, and $g(n+1) \in o(f(n))$. Then

$$\text{NTIME}(g(n)) \subsetneq \text{NTIME}(f(n)).$$

- This result is used using some of the results given above.
- This *cannot* be established using a direct application of a diagonalization argument — because exchanging accepting and rejecting states does not guarantee that a string would be nondeterministically rejected after this, if it was nondeterministically accepted before!
- A supplemental document gives more details about the proof of its result. Its *application* is similar to an application of the Deterministic Time Hierarchy Theorem.