Complexity and Cryptography

Thomas Zeugmann

Hokkaido University
Laboratory for Algorithmics

https://www-alg.ist.hokudai.ac.jp/∼thomas/COCRB/

Lecture 7: Complexity Hierachies

Reducing the Number of Tapes I

First, we ask how the number of tapes does influence the complexity of Turing machine computations. As we shall see, the answer depends on both the complexity measure considered and the type of the TM (deterministic or nondeterministic).

Reducing the Number of Tapes I

First, we ask how the number of tapes does influence the complexity of Turing machine computations. As we shall see, the answer depends on both the complexity measure considered and the type of the TM (deterministic or nondeterministic).

Theorem 1 (Hartmanis and Stearns (1965))

Let $f\colon \mathbb{N} \to \mathbb{R}_{\geqslant 0}$ *be any bounding function. Then we have*

$$\mathit{TIME}(f(n)) = \mathit{Time}_2((f(n))^2)\ .$$

Reducing the Number of Tapes II

Proof. Let M be any deterministic k-tape TM, $k > 1$, such that $T_M(n) \leqslant f(n)$ for all $n \in \mathbb{N}$. For showing the theorem, it suffices to construct a deterministic 2-tape TM M' satisfying $L(M') = L(M)$ and $T_{M'}(n) \leqslant c(T_M(n))^2$ for some constant $c > 0$. This is done as follows:

Reducing the Number of Tapes II

Proof. Let M be any deterministic k-tape TM, $k > 1$, such that $T_M(n) \leqslant f(n)$ for all $n \in \mathbb{N}$. For showing the theorem, it suffices to construct a deterministic 2-tape TM M' satisfying $L(M') = L(M)$ and $T_{M'}(n) \leqslant c(T_M(n))^2$ for some constant $c > 0$. This is done as follows:

Let Y be the alphabet used by M on all its $k-1$ work tapes and assume without loss of generality that $\# \notin Y$. Then M' will use the alphabet $(Y \times \{0,1\})^{k-1} \cup \{\#\}$ on its work tape. The ith component of the form $Y \times \{0,1\}$ at position z contains the content of the ith work tape of M at position z, where "1" is used to mark the actual position of the read-write head of M on this tape. The symbol # is used to mark the left and right end of what was written on the work tape of M'.

Reducing the Number of Tapes III

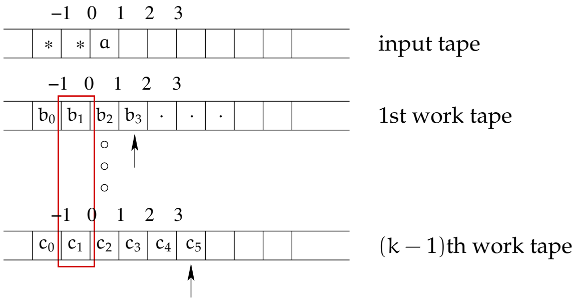


Figure 1: Illustration for the use of the new letters

What is inside the red box is becoming the new letter $((b_1, 0), \ldots, (c_1, 0))$ which is then written in the cell between -1 and 0 of the single work tape of M'.

Reducing the Number of Tapes IV

The simulation of one step of computation performed by M is done by M' as follows: Let w be the input to both M and M'. On the input tape, both machines behave identically. On its work tape M' behaves as follows:

Reducing the Number of Tapes IV

The simulation of one step of computation performed by M is done by M' as follows: Let w be the input to both M and M'. On the input tape, both machines behave identically. On its work tape M' behaves as follows:

(1) The read-write head on the work tape of M' is reading the whole inscription between the two end markers # from left to right. While doing this, M' memorizes in states the $k-1$ tape contents marked with 1, i.e., the actual head positions on the $k-1$ work tapes of M.

Reducing the Number of Tapes IV

The simulation of one step of computation performed by M is done by M' as follows: Let w be the input to both M and M'. On the input tape, both machines behave identically. On its work tape M' behaves as follows:

(1) The read-write head on the work tape of M' is reading the whole inscription between the two end markers # from left to right. While doing this, M' memorizes in states the $k-1$ tape contents marked with 1, i.e., the actual head positions on the $k-1$ work tapes of M.

(2) By using the Turing table of M, the machine M' is computing the changes to be made, and

Reducing the Number of Tapes IV

The simulation of one step of computation performed by M is done by M' as follows: Let w be the input to both M and M'. On the input tape, both machines behave identically. On its work tape M' behaves as follows:

(1) The read-write head on the work tape of M' is reading the whole inscription between the two end markers # from left to right. While doing this, M' memorizes in states the $k-1$ tape contents marked with 1, i.e., the actual head positions on the $k-1$ work tapes of M.

(2) By using the Turing table of M, the machine M' is computing the changes to be made, and

(3) then actually performing these changes by moving its read-write head on the work tape from right to left. By doing this, it possibly has also to move its end markers #.

Reducing the Number of Tapes V

Consequently, the length of the tape inscription on the work tape of M' cannot exceed $2 \cdot T_M(w) + 2$. Hence, one step of M's computation can be simulated in $O(T_M(w))$ steps by M'. At all, $T_M(w)$ many steps of M's computation have to be simulated. Thus, M' performs at most $O\left((T_M(w))^2\right)$ many steps. By Theorem 6.3 the assertion of the theorem follows. ∎

Reducing the Number of Tapes V

Consequently, the length of the tape inscription on the work tape of M' cannot exceed $2 \cdot T_M(w) + 2$. Hence, one step of M's computation can be simulated in $O(T_M(w))$ steps by M'. At all, $T_M(w)$ many steps of M's computation have to be simulated. Thus, M' performs at most $O\left((T_M(w))^2\right)$ many steps. By Theorem 6.3 the assertion of the theorem follows. ∎

Note that the same construction also works if we wish to simulate a deterministic k-tape TM by a deterministic one-tape TM. Thus, we even have the following result.

Reducing the Number of Tapes VI

Corollary 1

Let a $f\colon \mathbb{N} \to \mathbb{R}_{\geqslant 0}$ be any bounding function. Then we have $TIME(f(n)) = Time((f(n))^2)$.

Reducing the Number of Tapes VI

Corollary 1

Let a $f\colon \mathbb{N} \to \mathbb{R}_{\geqslant 0}$ *be any bounding function. Then we have* $\mathit{TIME}(f(n)) = \mathit{Time}((f(n))^2)$.

Furthermore, the proof of the latter theorem directly allows for the following corollary:

Corollary 2

Let a $f\colon \mathbb{N} \to \mathbb{R}_{\geqslant 0}$ *be any bounding function. Then we have* $\mathit{SPACE}(f(n)) = \mathit{Space}_2(f(n))$.

Proof. The proof is identical to the proof of Theorem 1, since the simulation given there is not increasing the amount of space needed, i.e., $S_{M'}(w) \leqslant S_M(w)$. ∎

Regular Languages I

Consequently, when studying the amount of space needed to accept non-regular languages, it suffices to deal with 2-tape Turing machines. Recalling a bit automata theory, we directly get the following lemma:

Regular Languages I

Consequently, when studying the amount of space needed to accept non-regular languages, it suffices to deal with 2-tape Turing machines. Recalling a bit automata theory, we directly get the following lemma:

Lemma 1

For every regular language L *there is a deterministic 2-tape TM* M *such that* $L = L(M)$ *and* $S_M(n) = 1$ *for all* $n \in \mathbb{N}$.

Regular Languages I

Consequently, when studying the amount of space needed to accept non-regular languages, it suffices to deal with 2-tape Turing machines. Recalling a bit automata theory, we directly get the following lemma:

Lemma 1

For every regular language L *there is a deterministic 2-tape TM* M *such that* $L = L(M)$ *and* $S_M(n) = 1$ *for all* $n \in \mathbb{N}$*.*

Question

How much space is needed for accepting non-regular languages?

Regular Languages II

Answering this question reveals a further surprise.
We also need the following definition:

Definition 1

Let M be a TM. A *macro state* of M is a tuple containing the following:

(1) the head position on the input tape of M,

(2) the actual state of M,

(3) for every work tape of M the inscription of all cells visited so far and the actual position of the read-write head.

If (1) is omitted, then we refer to the resulting tuple as *configuration* of M.

Regular Languages III

Note that the definition of configuration, when applied to one-tape TMs means that we just record the actual state of the TM.

Regular Languages III

Note that the definition of configuration, when applied to one-tape TMs means that we just record the actual state of the TM.
First, we show that enlarging the space available to any constant does not allow to accept non-regular languages.

Theorem 2

Let $c \geqslant 0$ be any constant and let M be a deterministic 2-tape TM such that $S_M(n) \leqslant c$ for all $n \in \mathbb{N}$. Then $L(M) \in \mathcal{REG}$.

Regular Languages III

Note that the definition of configuration, when applied to one-tape TMs means that we just record the actual state of the TM.
First, we show that enlarging the space available to any constant does not allow to accept non-regular languages.

Theorem 2

Let $c \geqslant 0$ be any constant and let M be a deterministic 2-tape TM such that $S_M(n) \leqslant c$ for all $n \in \mathbb{N}$. Then $L(M) \in \mathcal{REG}$.

Proof. We show that there is a constant $\hat{c}$ such that M can work at most $\hat{c}n$ many steps on inputs of length n without reaching a cycle.

Regular Languages IV

Obviously, there are at most

$$\sum_{k=0}^{c} |B|^k = \frac{|B|^{c+1} - 1}{|B| - 1} =: S$$

many pairwise different strings s with $|s| \leqslant c$ which can be written on M's work tape. To write a string of length k on its work tape, M needs k steps. So M needs at most $c \cdot S$ many steps to write all these strings on its work tape. To estimate $\hat{c}$, it suffices to assume that M can write all these strings in every of its states on its work tape. Hence, there are at most $c \cdot |Z| \cdot S$ many pairwise different steps that can be performed by M on every input symbol read on its inputs tape. Setting

$$\hat{c} = c \cdot |Z| \cdot S \tag{1}$$

yields that M can work at most $\hat{c}n$ many steps without reaching a cycle.

Regular Languages V

Next, we argue that there is a deterministic one-tape TM $\widehat{M}$ such that $T_{\widehat{M}}(n) \leqslant \hat{c}n$ for all $n \in \mathbb{N}$ and $L(\widehat{M}) = L(M)$.

Regular Languages V

Next, we argue that there is a deterministic one-tape TM $\widehat{M}$ such that $T_{\widehat{M}}(n) \leqslant \hat{c}n$ for all $n \in \mathbb{N}$ and $L(\widehat{M}) = L(M)$.

The TM $\widehat{M}$ is obtained by encoding all possible inscriptions on M's work tape into states and the changes M can make on these inscriptions into state transitions. Here it is crucial that there are only S many possible inscriptions. Finally, by Theorem 6.2, we directly get $L(M) \in \mathcal{REG}$. ∎

Regular Languages V

Next, we argue that there is a deterministic one-tape TM $\widehat{M}$ such that $T_{\widehat{M}}(n) \leqslant \hat{c}n$ for all $n \in \mathbb{N}$ and $L(\widehat{M}) = L(M)$.

The TM $\widehat{M}$ is obtained by encoding all possible inscriptions on M's work tape into states and the changes M can make on these inscriptions into state transitions. Here it is crucial that there are only S many possible inscriptions. Finally, by Theorem 6.2, we directly get $L(M) \in \mathcal{REG}$. $\blacksquare$

To make further progress, we extend the definition of traces made for one-tape TMs by using configurations.

Regular Languages VI

Definition 2

Let $k \in \mathbb{N}$, $k \geqslant 1$, let M be a k-tape TM and $w \in \Sigma^*$. We define the *trace* of M on input w at position j to be $TR_M(w, j)$, where

$$TR_M(w, j) =_{df} \text{the string formed of configurations of } M \text{ such that the } i\text{th position of } TR_M(w, j) \text{ is } M\text{'s configuration when its head on the input tape crosses the border } j \text{ for the } i\text{th time}.$$

Furthermore, $TR_M(w)$ and $TR_M(n)$ are defined as before; i.e., for every input string $w \in \Sigma^*$ and all $n \in \mathbb{N}$ we set

$$\begin{aligned} TR_M(w) &=_{df} \max\{|TR_M(w, j)| \mid j \in \mathbb{Z}\} \\ TR_M(n) &=_{df} \max\{TR_M(w) \mid |w| = n\} . \end{aligned}$$

Regular Languages VII

Note that for $k = 1$ we just obtain the definition of traces previously made for deterministic one-tape TMs.

Exercise 1. *Prove the following generalization of Lemma 6.1:*
Let M be a deterministic TM. If $TR_M(uvw, |uv|) = TR_M(uvw, |u|)$ then, for all $i, j \in \mathbb{N}$, $TR_M(uv^iv^jw, |uv^i|) = TR_M(uw, |u|)$ and $uv^iv^jw \in L(M)$ iff $uw \in L(M)$.

Regular Languages VII

Note that for $k = 1$ we just obtain the definition of traces previously made for deterministic one-tape TMs.

Exercise 1. *Prove the following generalization of Lemma 6.1:* *Let* M *be a deterministic TM. If* $TR_M(uvw, |uv|) = TR_M(uvw, |u|)$ *then, for all* $i, j \in \mathbb{N}$, $TR_M(uv^i v^j w, |uv^i|) = TR_M(uw, |u|)$ *and* $uv^i v^j w \in L(M)$ *iff* $uw \in L(M)$.

Next, we show a complexity gap for the complexity measure "space complexity" with respect to the acceptability of non-regular languages.

Regular Languages VIII

Theorem 3 (Stearns, Hartmanis, and Lewis II (1965))

Let M be any deterministic 2-tape TM such that $S_M(n) = o(\log\log n)$. Then $L(M)$ is regular.

Regular Languages VIII

Theorem 3 (Stearns, Hartmanis, and Lewis II (1965))

Let M be any deterministic 2-tape TM such that $S_M(n) = o(\log \log n)$. Then $L(M)$ is regular.

By Theorem 2 it suffices to show that there is a constant $c > 0$ such that $S_M(n) < c$. We continue indirectly.
Suppose there is an infinite sequence $(v_i)_{i \in \mathbb{N}}$ of strings such that

$$S_M(v_{i+1}) \; > \; S_M(v_i) \quad \text{for all } i \in \mathbb{N} \, . \tag{2}$$

We can choose the strings v_i in a way such that for all $i \in \mathbb{N}$

$$S_M(u) < S_M(v_i) \quad \text{for all strings } u \text{ with } |u| < |v_i| \, .$$

Regular Languages IX

Among all traces $TR_M(v_i, j)$, where $0 < j < |v_i|$, there are no two equal traces.

Suppose there are positions j_1 and j_2 with $0 < j_1 < j_2 < |v_i|$ and $TR_M(v_i, j_1) = TR_M(v_i, j_2)$. Then all symbols between j_1 and j_2 in v_i can be erased without reducing the space needed, a contradiction to (2).

By Exercise 1, without loss of generality, we can assume that each single trace does not contain two identical configurations, since the part of M's work done between such identical configurations cannot influence the acceptance of the input string given.

Regular Languages X

Provided $M = [B, Z, A]$ uses ℓ cells on its work tape, there are at most

$$c_\ell = |Z| \cdot \frac{|B|^{\ell+1} - 1}{|B| - 1} \cdot \ell \tag{3}$$

many possible configurations of length at most ℓ. By Exercise 1, we conclude that there are at most $c_\ell^{c_\ell}$ many different traces that can be built from the possible configurations. Since there are no two identical traces among all traces $TR_M(v_i, j)$, where $0 < j < |v_i|$, we therefore have

$$|v_i| \leqslant c_\ell^{c_\ell}\,, \quad \text{where} \quad \ell = S_M(v_i)\,. \tag{4}$$

Using (3) and (4), we see that there is an r such that $|v_i| \leqslant r^{r^\ell}$. Hence, $\log\log|v_i| \leqslant c \cdot \ell = c \cdot S_M(v_i)$ for some constant c. Therefore, $1 \leqslant c \cdot S_M(v_i)/\log\log|v_i|$, a contradiction to $S_M(n) = o(\log\log n)$.

Regular Languages XI

This contradiction is due to (2), and so (2) cannot hold. Hence, $S_M(n) \leqslant c$ for a constant $c > 0$. By Theorem 2, we therefore have $L(M) \in \mathcal{REG}$. ∎

Regular Languages XI

This contradiction is due to (2), and so (2) cannot hold. Hence, $S_M(n) \leqslant c$ for a constant $c > 0$. By Theorem 2, we therefore have $L(M) \in \mathcal{REG}$. ∎

The gap established in Theorem 3 cannot be improved. Let

$$L_{bin} = \{1 * 10 * 11 * \cdots * bin(k) \mid k \in \mathbb{N}^{+}\},$$

where $bin(k)$ denotes the binary representation of the number k. Clearly, $L_{bin} \notin \mathcal{REG}$ (Nerode's theorem).

Regular Languages XI

This contradiction is due to (2), and so (2) cannot hold. Hence, $S_M(n) \leqslant c$ for a constant $c > 0$. By Theorem 2, we therefore have $L(M) \in \mathcal{REG}$. $\blacksquare$

The gap established in Theorem 3 cannot be improved. Let

$$L_{bin} = \{1 * 10 * 11 * \cdots * bin(k) \mid k \in \mathbb{N}^+\} ,$$

where $bin(k)$ denotes the binary representation of the number k.
Clearly, $L_{bin} \notin \mathcal{REG}$ (Nerode's theorem).
But we have the following lemma:

Lemma 2

There exists a deterministic TM M such that $S_M(n) = \log\log n$ and $L(M) = L_{bin}$.

Regular Languages XII

Applying the techniques developed so far, we can prove the following:

Theorem 4

Let M be any deterministic TM accepting the language $L(M) = \{0^n 1^n \mid n \in \mathbb{N}\}$. Then we have $S_M(n) \neq o(\log n)$.

Proof. The proof is left as an exercise. ∎

Regular Languages XII

Applying the techniques developed so far, we can prove the following:

Theorem 4

Let M be any deterministic TM accepting the language $L(M) = \{0^n1^n \mid n \in \mathbb{N}\}$. *Then we have* $S_M(n) \neq o(\log n)$.

Proof. The proof is left as an exercise. ∎

Next, we turn our attention to complexity hierarchies for deterministic Turing machines.

Deterministic Complexity Hierarchies I

Before dealing with complexity hierarchies, it is meaningful to ask whether or not we can improve Theorem 1. This is indeed the case as the following theorem shows:

Deterministic Complexity Hierarchies I

Before dealing with complexity hierarchies, it is meaningful to ask whether or not we can improve Theorem 1. This is indeed the case as the following theorem shows:

Theorem 5 (Hennie and Stearns (1966))

Let $f\colon \mathbb{N} \to \mathbb{N}$ *be any bounding function. Then we have*

$$\mathit{TIME}(f(n)) = \mathit{Time}_3\left(f(n) \cdot \log f(n)\right) .$$

Deterministic Complexity Hierarchies I

Before dealing with complexity hierarchies, it is meaningful to ask whether or not we can improve Theorem 1. This is indeed the case as the following theorem shows:

Theorem 5 (Hennie and Stearns (1966))

Let $f\colon \mathbb{N} \to \mathbb{N}$ *be any bounding function. Then we have*

$$TIME(f(n)) = Time_3\left(f(n) \cdot \log f(n)\right) \ .$$

We omit the proof of Theorem 5 here, since there are several more important theorems we want to deal with in this course.

Deterministic Complexity Hierarchies II

Theorem 6

Assuming an appropriate enumeration of all deterministic TMs the following holds:

(1) *There is a universal deterministic TM* U *for all deterministic TMs* $(M_i)_{i \in \mathbb{N}}$ *such that*

$$L(U) = \{bin(i) * w \mid w \in L(M_i),\ i \in \mathbb{N}\},$$

and for every $i \in \mathbb{N}$ *there is a constant* c_i *such that for all* $w \in \Sigma^*$ *the condition* $T_U(|bin(i) * w|) < c_i \cdot T_{M_i}(|w|) \cdot \log T_{M_i}(|w|)$ *is fulfilled.*

Deterministic Complexity Hierarchies II

Theorem 6

Assuming an appropriate enumeration of all deterministic TMs the following holds:

(1) *There is a universal deterministic TM U for all deterministic TMs $(M_i)_{i\in\mathbb{N}}$ such that*

$$L(U) = \{bin(i) * w \mid w \in L(M_i),\ i \in \mathbb{N}\}\,,$$

and for every $i \in \mathbb{N}$ there is a constant c_i such that for all $w \in \Sigma^*$ the condition $T_U(|bin(i) * w|) < c_i \cdot T_{M_i}(|w|) \cdot \log T_{M_i}(|w|)$ is fulfilled.

(2) *There is a universal deterministic TM U for all deterministic TMs $(M_i)_{i\in\mathbb{N}}$ such that*

$$L(U) = \{bin(i) * w \mid w \in L(M_i),\ i \in \mathbb{N}\}\,,$$

and for every $i \in \mathbb{N}$ there is a constant c_i such that for all $w \in \Sigma^*$ the condition $S_U(|bin(i) * w|) < c_i \cdot S_{M_i}(|w|)$ is fulfilled.

Deterministic Complexity Hierarchies III

Proof. We showed the existence of universal TMs in our course *Theory of Computation*. For showing the theorem, the enumeration is chosen in a way such that the code of M_i at the universal machine is just $\mathit{bin}(i)$. Moreover, the universal deterministic TM U possesses an input tape and at least two work tapes. Any fixed number $k \geqslant 2$ of work tapes will do.

Deterministic Complexity Hierarchies III

Proof. We showed the existence of universal TMs in our course *Theory of Computation*. For showing the theorem, the enumeration is chosen in a way such that the code of M_i at the universal machine is just $bin(i)$. Moreover, the universal deterministic TM U possesses an input tape and at least two work tapes. Any fixed number $k \geqslant 2$ of work tapes will do.

On input $bin(i) * w$, the universal TM U simulates the deterministic TM M_i on input w step by step. Assuming the coding is appropriately chosen, for doing this, it suffices that U reads in each simulation step the “programming code” $bin(i)$, memorizes the actions to be performed and performs the step to be simulated.

Deterministic Complexity Hierarchies IV

Hence, the constant c_i is obtained by counting the number of steps the universal machine needs for simulating one step of M_i. Furthermore, the simulation has to incorporate the reduction of work tapes to the previously fixed number k. By Theorem 5, this reduction requires that in the deterministic case $T_{M_i}(w) \cdot \log\left(T_{M_i}(w)\right)$ steps have to be simulated. Thus, Part (1) of the theorem follows.

Deterministic Complexity Hierarchies IV

Hence, the constant c_i is obtained by counting the number of steps the universal machine needs for simulating one step of M_i. Furthermore, the simulation has to incorporate the reduction of work tapes to the previously fixed number k. By Theorem 5, this reduction requires that in the deterministic case $T_{M_i}(w) \cdot \log\left(T_{M_i}(w)\right)$ steps have to be simulated. Thus, Part (1) of the theorem follows.

For showing Part (2), the only difference is the application of Corollary 2 for the tape reduction in the deterministic case. Note that this corollary also causes the improvement in Part (2) concerning the space complexity needed by the universal deterministic TM $\mathcal{U}$. ∎

Deterministic Complexity Hierarchies V

Theorem 7 (Hennie and Stearns (1966))

Let $f, g\colon \mathbb{N} \to \mathbb{N}$ *be any bounding functions such that* g *is* T*-constructible and* $f(n) \log f(n) = o(g(n))$. *Then we have*

$$\mathit{TIME}(f(n)) \quad \subset \quad \mathit{TIME}(g(n)) \, .$$

Deterministic Complexity Hierarchies V

Theorem 7 (Hennie and Stearns (1966))

Let $f, g \colon \mathbb{N} \to \mathbb{N}$ *be any bounding functions such that* g *is* T*-constructible and* $f(n)\log f(n) = o(g(n))$. *Then we have*

$$\mathit{TIME}(f(n)) \quad \subset \quad \mathit{TIME}(g(n))\,.$$

Proof. Since the assumption implies $f(n) < f(n)\log f(n) < g(n)$ for all but finitely many n, the inclusion $\mathit{TIME}(f(n)) \subseteq \mathit{TIME}(g(n))$ is obvious.

Deterministic Complexity Hierarchies V

Theorem 7 (Hennie and Stearns (1966))

Let $f, g\colon \mathbb{N} \to \mathbb{N}$ *be any bounding functions such that* g *is* T*-constructible and* $f(n)\log f(n) = o(g(n))$. *Then we have*

$$\mathit{TIME}(f(n)) \quad\subset\quad \mathit{TIME}(g(n)) \ .$$

Proof. Since the assumption implies $f(n) < f(n)\log f(n) < g(n)$ for all but finitely many n, the inclusion $\mathit{TIME}(f(n)) \subseteq \mathit{TIME}(g(n))$ is obvious.

We have to show that there is a language $L_d \in \mathit{TIME}(g(n)) \setminus \mathit{TIME}(f(n))$. This is done by *diagonalization*.

Deterministic Complexity Hierarchies VI

We construct the wanted language L_d by using the universal TM U from Theorem 6 and by defining a deterministic TM M as follows: Then, L_d contains all strings w, $w = 0^k bin(i)$ that are accepted by M and nothing else.

Deterministic Complexity Hierarchies VI

We construct the wanted language L_d by using the universal TM U from Theorem 6 and by defining a deterministic TM M as follows: Then, L_d contains all strings w, $w = 0^k bin(i)$ that are accepted by M and nothing else.

On input w, $w = 0^k bin(i)$, the machine M works as U on input $bin(i) * w$. Moreover, M uses a clock for $g(n)$ and rejects its input if U does not stop within $g(|w|)$ many steps. If U does stop within $g(|w|)$ many steps, then M rejects the input if U accepts and accepts if U rejects.

Deterministic Complexity Hierarchies VI

We construct the wanted language L_d by using the universal TM U from Theorem 6 and by defining a deterministic TM M as follows: Then, L_d contains all strings w, $w = 0^k bin(i)$ that are accepted by M and nothing else.

On input w, $w = 0^k bin(i)$, the machine M works as U on input $bin(i) * w$. Moreover, M uses a clock for $g(n)$ and rejects its input if U does not stop within $g(|w|)$ many steps. If U does stop within $g(|w|)$ many steps, then M rejects the input if U accepts and accepts if U rejects.

Thus, by construction we already know $L_d \in TIME(g(n))$.

Deterministic Complexity Hierarchies VII

Suppose there is a deterministic TM M_i such that

$$L_d = L(M_i) \quad \text{and} \quad T_{M_i}(n) \leqslant f(n)\,. \tag{5}$$

By Theorem 6, we may conclude that for input $0^k bin(i)$ the following holds:

$$\begin{aligned} T_U(|bin(i) * 0^k bin(i)|) &\leqslant c_i \cdot T_{M_i}(|0^k bin(i)|) \log\left(T_{M_i}(|0^k bin(i)|)\right) \\ &\leqslant c_i \cdot f(|0^k bin(i)|) \log\left(f(|0^k bin(i)|)\right)\,. \end{aligned}$$

Moreover, if k is sufficiently large, we also have

$$T_U(|bin(i) * 0^k bin(i)|) \leqslant g(|0^k bin(i)|)\,.$$

Deterministic Complexity Hierarchies VIII

Hence, Theorem 6 and the construction of M imply that

$0^k bin(i) \in L(M_i)$

$$\begin{aligned}
&\text{iff} \quad 0^k bin(i) \in L(M) \quad (* \text{ since } L(M_i) = L_d = L(M) *) \\
&\text{iff} \quad bin(i) * 0^k bin(i) \notin L(U) \quad (* \text{ by construction of } M *) \\
&\text{iff} \quad 0^k bin(i) \notin L(M_i) \quad (* \text{ by definition of } L(U) *) \,.
\end{aligned}$$

Deterministic Complexity Hierarchies VIII

Hence, Theorem 6 and the construction of M imply that

$0^k bin(i) \in L(M_i)$

$$\begin{array}{lll} \text{iff} & 0^k bin(i) \in L(M) & (* \text{ since } L(M_i) = L_d = L(M) *) \\ \text{iff} & bin(i) * 0^k bin(i) \notin L(U) & (* \text{ by construction of } M *) \\ \text{iff} & 0^k bin(i) \notin L(M_i) & (* \text{ by definition of } L(U) *)\,. \end{array}$$

Thus, we have obtained the contradiction $0^k bin(i) \in L(M_i)$ iff $0^k bin(i) \notin L(M_i)$. This contradiction is caused by our Supposition (5). Consequently, our supposition is false. Since there is obviously a deterministic TM M_i with $L_d = L(M_i)$, we must conclude that every TM M_i with $L_d = L(M_i)$ has to satisfy $T_{M_i}(n) > f(n)$. So we arrive at $L_d \notin TIME(f(n))$. ∎

Deterministic Complexity Hierarchies IX

Applying the same ideas as in the proof of Theorem 7 we directly get the following hierarchy theorem for deterministic space complexity:

Theorem 8

Let g be any S-constructible function and let $f(n)$ be any space bound such that $f(n) = o(g(n))$. Then we have

$$\mathit{SPACE}(f(n)) \subset \mathit{SPACE}(g(n)) .$$

Deterministic Complexity Hierarchies IX

Applying the same ideas as in the proof of Theorem 7 we directly get the following hierarchy theorem for deterministic space complexity:

Theorem 8

Let g be any S-constructible function and let $f(n)$ be any space bound such that $f(n) = o(g(n))$. Then we have

$$\mathit{SPACE}(f(n)) \subset \mathit{SPACE}(g(n)) .$$

The hierarchy theorems already proved do yield only infinite deterministic complexity time and space hierarchies provided there are arbitrarily complex T-constructible and S-constructible functions. Therefore, we should prove the following exercise:

Deterministic Complexity Hierarchies X

Exercise 2. *For every general recursive function* $f\colon \mathbb{N} \to \mathbb{N}$ *there exist general recursive functions* $g,\ g'\colon \mathbb{N} \to \mathbb{N}$ *such that*

(1) $f(n) < g(n)$ *and* $f(n) < g'(n)$ *for all* $n \in \mathbb{N}$*, and*

(2) g *is* T*-constructible, and* g' *is S-constructible.*

Nondeterministic Turing Machines I

Definition 3

A TM $M = [B, Z, A]$ is called *nondeterministic k-tape TM*, $k \geqslant 1$, provided $B \cup Z = \emptyset$, and

(1) $B = \{*, \,|, \ldots\}$ is a finite set such that $|B| \geqslant 2$,

(2) $Z = \{z_s, \, z_f, \ldots\}$ is a finite set such that $|Z| \geqslant 2$,

(3) $A \colon Z \setminus \{z_f\} \times B^k \to \wp\left((B \times \{L, N, R\})^k \times Z\right) \setminus \{\emptyset\}$

and M has an input-tape with read-only head (read-write head iff $k = 1$) and $k - 1$ work-tapes (none iff $k = 1$) each of which possesses exactly one read-write head.

Nondeterministic Turing Machines II

Again, in every step M moves k heads and observes k cells (one on each tape). Also, we make the restriction that M is not allowed to write on its input-tape iff $k > 1$.

Initially, M is in state z_s (the start state) and all heads are observing the first cell located right to position 0. What M is writing into the cells it is observing and which move the heads make is decided nondeterministically by choosing one possibility from the set of allowed possibilities (cf. Condition (3)).

Nondeterministic Turing Machines III

Looking at Definition 6.4 and Definition 3, we see that the main difference to a deterministic TM is Condition (3). For a deterministic TM we had

$$A \subseteq Z \setminus \{z_f\} \times B^k \times (B \times \{L, N, R\})^k \times Z \,,$$

while now we have

$$A : Z \setminus \{z_f\} \times B^k \to \wp\left((B \times \{L, N, R\})^k \times Z\right) \setminus \{\emptyset\} \,.$$

That is, a nondeterministic TM possesses in each step a set of possible continuations. From this set, one possibility is chosen nondeterministically.

Nondeterministic Turing Machines IV

Initially, M is in state z_s (the start state) and all heads are observing the first cell located right to position 0. But now, on one and the same input there are many possible computations that can be performed by M. Thus, we also have to modify the definition of accepting a language. As before, we assume $\Sigma \subseteq B$ to be the input alphabet, where in particular $* \notin \Sigma$.
This is done as follows:

Nondeterministic Turing Machines V

Definition 4

A language $L \subseteq \Sigma^*$ is *accepted* by a nondeterministic k-tape TM M if for every string $w \in \Sigma^*$ the following conditions are satisfied:
If w is written on the empty input-tape of M (beginning in cell 0) and M is started such that the read-only head on the input-tape is put on the leftmost symbol of w and all other heads are put on the first cell located right to position 0 in state z_s then, if $w \in L$, *there is a possible computation path* such that M stops after having executed finitely many steps in state z_f and the cell observed by M on its first work-tape in state z_f contains a $|$. In this case we also write $M(w) = |$.

Nondeterministic Turing Machines VI

In the following, we use NTM as an abbreviation for nondeterministic TMs.

In contrast to a deterministic TM, if $w \notin L$ then an NTM M is *not* supposed to deliver any information. We do not even require M to stop on inputs $w \notin L$. By $L(M)$ we denote the language accepted by M.

Nondeterministic Turing Machines VI

In the following, we use NTM as an abbreviation for nondeterministic TMs.

In contrast to a deterministic TM, if $w \notin L$ then an NTM M is *not* supposed to deliver any information. We do not even require M to stop on inputs $w \notin L$. By $L(M)$ we denote the language accepted by M.

Next, we define time and space complexity for NTMs.

Nondeterministic Turing Machines VII

Definition 5

Let M be an NTM and $w \in \Sigma^*$. We define $T_M(w)$ to be the *minimum number of steps* executed by M on input w among all its accepting computations if $w \in L(M)$, and the *minimum number of steps* executed by M among all its computation executed on input w if $w \notin L(M)$. We define $S_M(w)$ to be the *minimum number of all cells* on M's work tapes visited by the read-write heads of M on input w among all its accepting computations if $w \in L(M)$ and *minimum number of all cells* on M's work tapes visited by the read-write heads of M on input w among all its computations if $w \notin L(M)$. Both functions T and S remain undefined if there is no computation of M on input w that stops. Furthermore, we set

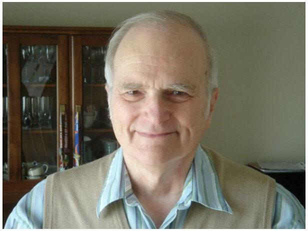
$$\begin{aligned} T_M(n) &=_{df} \max\{T_M(w) \mid |w| = n\}\,, \\ S_M(n) &=_{df} \max\{S_M(w) \mid |w| = n\}\,. \end{aligned}$$

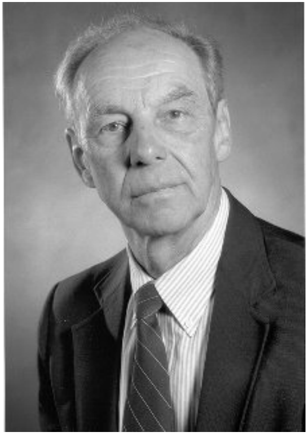
Nondeterministic Turing Machines VIII

Finally, we define the resulting complexity classes as follows:

$$\begin{aligned}
\mathit{NTime}_k(f(n)) &=_{df} \{L(M) \mid M \text{ is a nondeterministic } k\text{-tape TM} \\
&\qquad \text{and } T_M(n) \leqslant f(n) \text{ for all } n \in \mathbb{N}\}\,; \\
\mathit{NSpace}_k(f(n)) &=_{df} \{L(M) \mid M \text{ is a nondeterministic } k\text{-tape TM} \\
&\qquad \text{and } S_M(n) \leqslant f(n) \text{ for all } n \in \mathbb{N}\}\,; \\
\mathit{NTIME}(f(n)) &=_{df} \{L(M) \mid M \text{ is an NTM and} \\
&\qquad T_M(n) \leqslant f(n) \text{ for all } n \in \mathbb{N}\}\,; \\
\mathit{NSPACE}(f(n)) &=_{df} \{L(M) \mid M \text{ is an NTM and} \\
&\qquad S_M(n) \leqslant f(n) \text{ for all } n \in \mathbb{N}\}\,.
\end{aligned}$$

Thank you!


Richard Edwin Stearns



Juris Hartmanis