DISCRETE COMPUTATION:
THEORY AND OPEN PROBLEMS

Notes for the lectures by

Albert R. Meyer

Preceptorial Introduction to Computer Science for Mathematicians

American Mathematical Society

San Francisco
January, 1974

This work was supported in part by the National Science Foundation under research grant GJ-34671.

LECTURE I

1

MULTIPLICATION IN BINARY

$$\begin{array}{r} U = 101100 \\ V = \underline{111111} \\ 101100 \\ 101100 \\ \vdots \\ \hline U \times V = 101011010100 \end{array}$$

2

α = the add and shift multiplication algorithm

$T_\alpha(U,V)$ = time (number of basic operations on digits) to multiply U and V by method α.

$T_\alpha(n) = \max \{T_\alpha(U,V) \mid \ell(U) = \ell(V) = n\}$

Remark: $T_\alpha(n) = 0(n^2)$

3

Recursive algorithm for multiplication

$$U = \begin{array}{|c|c|}\hline U_1 & U_2 \\ \hline\end{array} = U_1 \cdot 2^{n/2} + U_2$$

$$V = \begin{array}{|c|c|}\hline V_1 & V_2 \\ \hline\end{array} = V_1 \cdot 2^{n/2} + V_2$$

$$U \cdot V = U_1V_1 \cdot 2^n + (U_1V_2 + U_2V_1) \cdot 2^{n/2} + U_2V_2$$

① ② ③ ④

4

ρ = recursive algorithm

$T_\rho(n) = 4T\ (n/2)$ + (time to add and shift length n numbers)

$= 4T_\rho(n/2) + 0(n)$

$= 4^kT_\rho(n/2^k) + 0(n)$

$= 0(4^{\log_2 n}) = 0(n^2)$

---

5 β = better recursive algorithm using only three half length multiplications

① $(U_1 + U_2)\cdot(V_1 + V_2)$

② U_1V_1

③ U_2V_2

$$U\cdot V = \textcircled{2}\cdot 2^n + (\textcircled{1} - \textcircled{2} - \textcircled{3})\cdot 2^{n/2} + \textcircled{3}$$

---

6 $$T_\beta(n) = 3T_\beta(n/2) + 0(n)$$

$$= 0(3^{\log_2 n})$$

$$= 0(n^{\log_2 3})$$

$$\approx 0(n^{1\cdot 6})$$

---

7 Best upper bound known for multiplication:

$$O(n \cdot \log n \cdot \log \log n)$$

by Strassen and Schönhage.

Question: What is the fastest possible way to multiply?

Need there even be one?

---

8 Might have algorithms $\beta_1, \beta_2, \ldots$

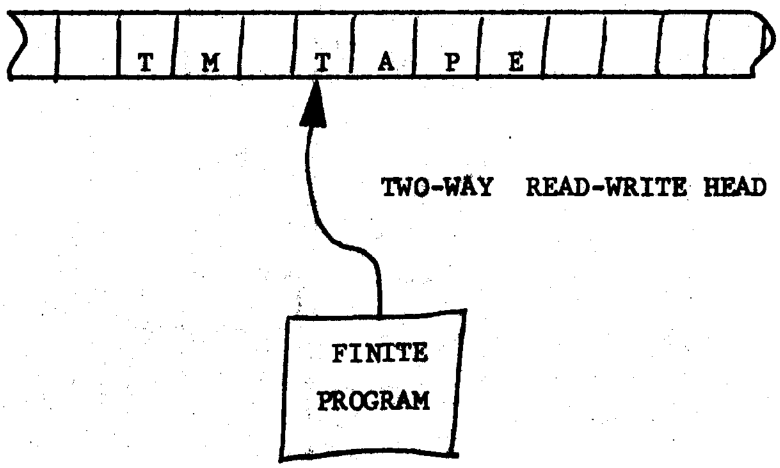
$$T_{\beta_1}(n) = n \log n$$

$$T_{\beta_2}(n) = n \sqrt{\log n}$$

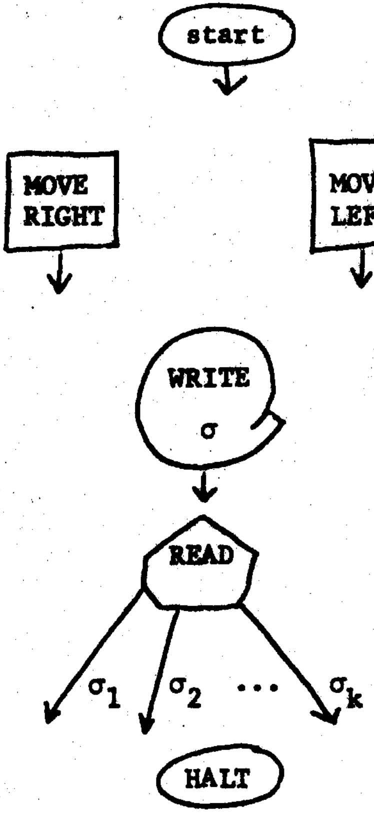
$$T_{\beta_i}(n) = n \cdot (\log n)^{\frac{1}{i}}$$

Then there is no fastest one.

Turing Machine



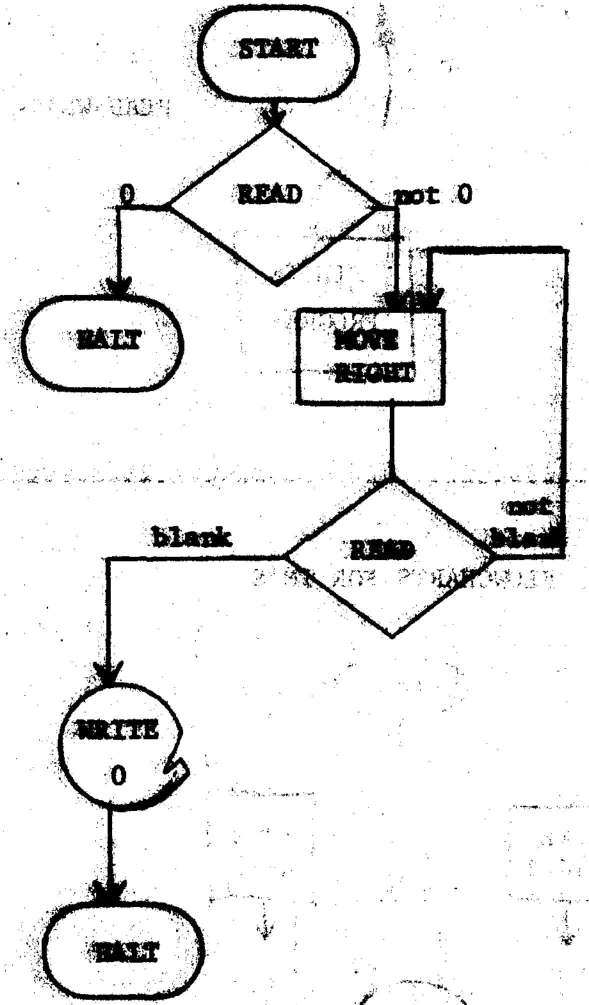
FLOWCHARTS FOR TM'S



$\sigma \in \Sigma = \{$ **tape symbols** $\}$

11 FLOWCHART FOR $F(x) = 2x$

(Input x an integer in binary notation.)



12 $\mathfrak{M}$ a T.M.,

x an input word.

$T_{\mathfrak{M}}(x)$ = number of instructions executed by $\mathfrak{M}$ on x if $\mathfrak{M}$ halts; ∞ if $\mathfrak{M}$ doesn't halt on x.

$S_{\mathfrak{M}}(x)$ = number of tape squares visited by head of $\mathfrak{M}$ with input x if $\mathfrak{M}$ halts; ∞ if $\mathfrak{M}$ does not halt.

$\varphi_{\mathfrak{M}}(x)$ = output of $\mathfrak{M}$ on x, if any; ∞ if no output.

T = <u>t</u>ime S = <u>s</u>pace φ = <u>f</u>unction

13

<u>Church's</u> <u>Thesis</u>:

The effectively (mechanically) computable functions and the Turing machine computable functions are the same.

<u>Extended</u> <u>Church's</u> <u>Thesis:</u>

If a function is computable in time T on any reasonable computer model, then it is computable in time ≤ polynomial (T) on a Turing machine.

---

14 **Infinitely-often Speed-up Theorem:**

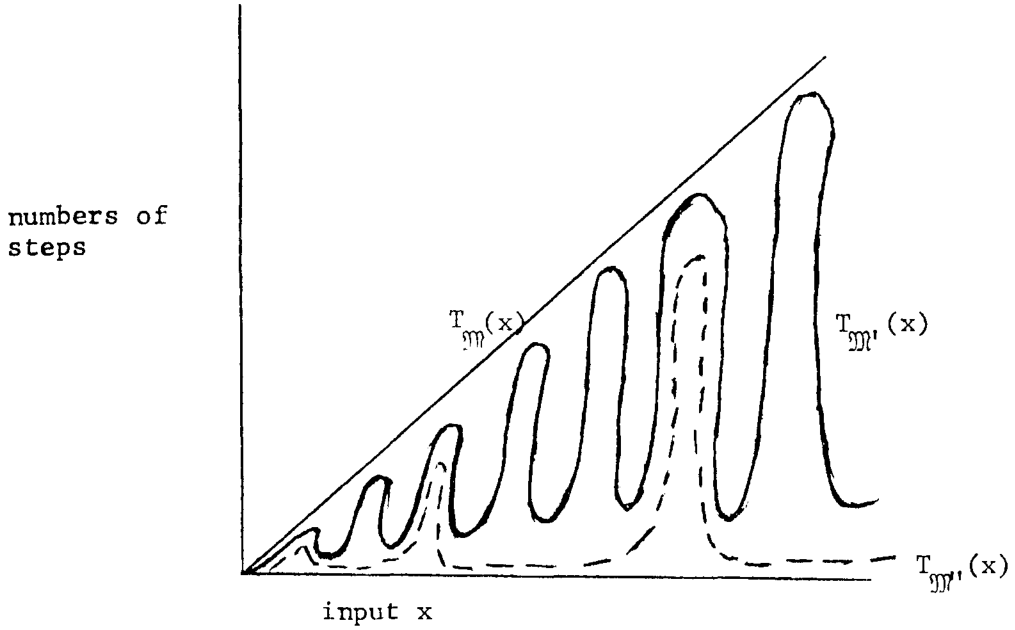
(M. BLUM). Let t: N → N be any computable function. Then there is a computable function $C_t : N \to \{0,1\}$ such that

> given any $\mathfrak{M}$ computing C_t one can construct an $\mathfrak{M}'$ also computing C_t with the property that

$T_{\mathfrak{M}}(x) > t(x)$ and $T_{\mathfrak{M}'}(x) <$ constant

for infinitely many $x \in N$.

15



$\mathfrak{M}'$ is faster than $\mathfrak{M}$ infinitely often,

$\mathfrak{M}''$ is faster than $\mathfrak{M}'$ infinitely often, etc.

16 Let $\mathfrak{M}_0, \mathfrak{M}_1, \ldots, \mathfrak{M}_i$ be an orderly list of of all Turing machines (say in order of the size of their flowcharts).

Let φ_i abbreviate $\varphi_{\mathfrak{M}_i}$,

" T_i " $T_{\mathfrak{M}_i}$.

Universal Machine Theorem:

$\varphi_i(x)$, regarded as a function of both

i and x, is computable.

16a **PADDING LEMMA**:

Given any program, one can "pad" it with instructions which it never uses. Thus, we obtain

a new program with the same behavior as the old one.

More formally,

---

17 **LEMMA:** For any T.M. $\mathbb{M}_e$ there is an infinite set PAD(e) $\subset$ N such that

(1) for any e' $\in$PAD(e)

$$\varphi_e = \varphi_{e'}, \; T_e = T_{e'}, \text{ and } S_e = S_{e'}$$

and (2) there is a T.M. which recognizes elements of PAD(e) in constant time.

(Think of PAD(e) being binary numbers of the form

e	#	irrelevant

)

---

18 **Proof of L.O. Speed-up Thm:**

Let

$$C_t(x) \overset{df}{=} \begin{cases} 1 \dot{-} \varphi_x(x) & \text{if } T_x(x) \le t(x), \\ 0 & \text{otherwise.} \end{cases}$$

$$1 \dot{-} y \overset{df.}{=} \begin{cases} 0 & \text{if } y \ge 1, \\ 1 & \text{if } y = 0. \end{cases}$$

(1) C_t is computable (implicit in the Universal Machine Thm.)

(2) If $\varphi_{e'} = C_t$, then $T_{e'}(e') > t(e')$ and $C_t(e') = 0$ (by def. of C_t).

(3) Say $\varphi_e = C_t$. Then for any $e' \in PAD(e)$,

$$t(e') < T_{e'}(e') = T_e(e')$$

and $C_t(e') = 0$.

So speed-up $\mathfrak{M}_e$ by always testing if the input is in PAD(e), and if so immediately print output 0.

19 **Def.** $\mathrm{Time}(t) = \{\, \varphi_i : N \to N \;\mid\; T_i(x) \le t(x)$

almost everywhere$\}$

$\mathrm{Space}(t) = \;\cdots\; \qquad S_i \;\cdots$

Corollary. $C_t \notin \mathrm{Time}(t)$ for any computable t.

Remark: Time to compute C_t depends on time to compute t.

Convention: $n = \mathrm{length}(x) = \ell(x) \approx \log_2 x$. Thus,

$\mathrm{Time}\ (2^n) = \mathrm{Time}\ (2^{\ell(x)}) = \mathrm{Time}\ (x)$,

$\mathrm{Time}\ (2^{2n}) = \mathrm{Time}\ (x^2)$, etc.

20 **Def.** A computable $t: N \to N$ is time-honest iff $t \in \mathrm{Time}(t^3)$ and $t(x) \geq \ell(x)$.

Cor. (Compression Theorem, Hartmanis-Stearns) For any time-honest t, $C_t \in \mathrm{Time}(t^4) - \mathrm{Time}(t)$.

Remark: Lots of time-honest fcns.

$$\lceil \log_2 x \rceil,\ x,\ 2^x,\ \left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\} x$$

closed under +, •, exp, composition.

Z1 Is more time better than less?

Is

$$\mathrm{Time}(t^4) - \mathrm{Time}(t) \neq \emptyset$$

for all computable t?

NOT NECESSARILY!

Z2

Gap Theorem: (Trachtenbrot, Borodin) For any computable g, there exist arbitrarily large computable t such that

$$\mathrm{Time}(t) = \mathrm{Time}(g \circ t)$$

Z2a Proof of Gap Theorem:

Given g, define

t(x) = the least z such that

$(\forall i \leq x)[\, T_i(x) < z$ or $T_i(x) > g(z)\,]$.

---

23 **Honesty Theorem** (McCreight, Meyer)

For every computable t, there is a time-honest t' such that

$$\mathrm{Time}(t) = \mathrm{Time}(t')$$

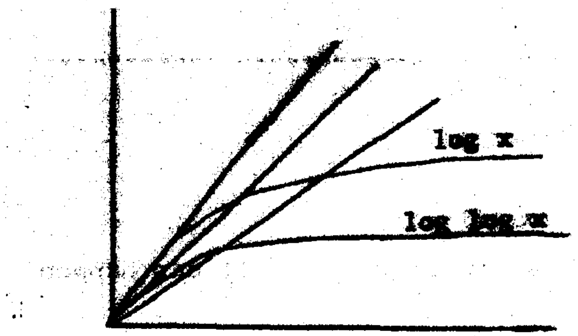
24

Summary:

For arbitrarily large t, t' it can happen that

$$\mathrm{Time}(2^{2^{t}}) \underset{\substack{\nearrow \\ \text{GAP}}}{=} \mathrm{Time}(t) \underset{\substack{\uparrow \\ \text{HONESTY}}}{=} \mathrm{Time}(t') \underset{\substack{\nwarrow \\ \text{COMPRESSION}}}{\subsetneqq} \mathrm{Time}((t')^{4})$$

MNEMONIC PICTURE



Lines(f) = radial lines almost everywhere under f

Compression: For any line $L \neq 0$,

$$\text{Lines}(2L) \supsetneq \text{Lines}(L)$$

Honesty: For any function f, there is a line L,

Lines(f) = Lines(L).

Gap: Lines(t) = Lines (2^t) = {zero line}

for t = loglog.

26 *Def.* Let f be a computable function. A sequence $t_1, t_2, \ldots$ of functions is a (space) *complexity sequence for f* iff

(1) If $\varphi_e = f$, then $S_e \geq t_i$ almost every where for some i,

and (2) For every i, there is a $\varphi_e = f$ such that

$$t_i \geq S_e \text{ almost everywhere.}$$

---

27 *Def.* A sequence of functions $p_1, p_2, \ldots$ is an *r.e. complexity sequence* (for space) iff

(1) $p_{i+1} \leq \left\lceil \frac{1}{2} \cdot p_i \right\rceil$ for all i,

(2) for each i there is a j such that,

$$p_i = S_j$$

and (3) $p_i(x)$ is a computable function of i and x.

---

28 Theorem (Meyer, Schnorr) Every computable function has an r.e. complexity sequence.

Every r.e. complexity sequence is a complexity sequence for some 0-1 valued computable function.

29

Example:

$$\text{Let } t_i(x) = \left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\} x \dot{-} i$$

So $t_{i+1} = \log_2 t_i$ almost everywhere.

Cor. Almost everywhere Speed-up

(Blum) There is a 0-1 valued computable function, c, such that for any T.M. computing c there is another T.M. computing c which uses exponentially less space at almost all arguments.

LECTURE II

1

Σ = finite set called the <u>alphabet</u> or
<u>vocabulary</u>,

an element $\sigma \in \Sigma$ is called a <u>letter</u>.

Σ^* = set of all finite sequence of
letters,

an element $x \in \Sigma^*$ is called a <u>word</u>.

2

Binary operation <u>concatenation</u>, written " • "
on Σ^*:

$$x \cdot y = xy = \text{word } x \text{ followed by word } y.$$

<u>Example</u>: $001 \cdot 01 = 00101$

$\ell(x)$ = length (number of occurrences of letter) of the word x.

$\ell(001) = 3$

$\ell(x \cdot y) = \ell(x) + \ell(y).$

---

3 $\underline{\lambda} \in \Sigma^*$ acts as an identity element under concatenation.

$$\lambda \cdot x = x \cdot \lambda = x \quad \text{for all } x \in \Sigma^*,$$

$$\ell(\lambda) = 0.$$

Remark 1: $<\Sigma^*, \cdot>$ is the free monoid generated by Σ with identity λ.

Remark 2: Remark 1 is irrelevant.

Remark 3: λ is introduced as a technical convenience and could be eliminated in what follows at the expense of some minor awkwardness.

4

A set $L \subset \Sigma^*$ is called a language. Extending concatenation to languages in the usual way:

$L \cdot M$, also written $LM \overset{\text{def.}}{=}$

$$\{x \cdot y \mid x \in L \text{ and } y \in M\}$$

Example: $\{0\} \cdot \{0,1\} = \{00,01\}$

$$\{0,00\} \cdot \{1,01\} = \{01,001,0001\}$$

$\{0,1,\lambda\} \cdot \{0,1,\lambda\} \cdot \{0,1,\lambda\}$ = all binary words of length ≤ 3 (including λ).

5

For $x \in \Sigma^*$, $n \in N$,

$$x^n = \underbrace{x \cdot x \cdots x}_{n}$$

$$x^0 = \lambda$$

Example: $(01)^3 = 010101$

Similarly for $A \subset \Sigma^*$

$$A^n = \underbrace{A \cdot A \cdots A}_{n}$$

$$A^0 = \{\lambda\}$$

6

Example:

$\{0,1\}^4$ = all binary words of length <u>exactly</u> 4.

$\{0,1,\lambda\}^4$ = all binary words of length ≤ 4.

$(((\{0,1\}^2)^2)^2)^2 = \{0,1\}^{2^4}$ = all binary words of length 16.

---

7 Important example:

$$(01)^{n} = \underbrace{0101\cdots01}_{2n} =$$

$$= \{0,1\}^{2n} - (1\cdot\{0,1,\lambda\}^{2n} \cup$$
$$\{0,1,\lambda\}^{2n}\cdot 0 \cup$$
$$\{0,1,\lambda\}^{2n}\cdot\{00,11\}\cdot\{0,1,\lambda\}^{2n})$$

= all binary words of length 2n which do not

(1) start wrong

or (2) end wrong

or (3) move wrong (contain a forbidden local pattern)

8

<u>Problem</u>: Given two <u>expressions</u> involving letters in Σ,λ, and operations

" $\cdot$ "	concatenation
" $\cup$ "	union
" 2 "	squaring
" $\cap$ "	intersection
" - "	set difference

is there a way to tell if they describe the same language?

---

9 YES!

BUT NO GOOD WAY!!

10

Lemma. An expression containing n operation symbols describes a subset of

$$(\Sigma \cup \lambda)^{2^n}$$

Proof. By induction on n:

If $n=0$, the expression must consist of a single letter or λ.

11

If E is an expression containing n+1 operations, then E is of the form

$$E_1 \cdot E_2$$
$$E_1 \cup E_2$$
$$(E_1)^2$$
$$E_1 \cap E_2$$
$$E_1 - E_2$$

where E_1, E_2 are expressions containing $\leq$ n operation symbols. Proof follows immediately.

12

For any expression E, let

$\mathcal{L}(E) \subset \Sigma^*$ be the language described by E.

<u>Remark</u>: Formally, $E_1 = E_2$ means that E_1 and E_2 are identical expressions. E_1 and E_2 are <u>equivalent</u> (written $E_1 \equiv E_2$) iff $\mathcal{L}(E_1) = \mathcal{L}(E_2)$.

13

$E_1 \equiv E_2$ iff $(E_1 - E_2) \cup (E_2 - E_1) = \emptyset$

Hence sufficient to test whether an expression describes the empty set.

14

To test if $\mathcal{L}(E) = \emptyset$, convert E to a list of the words in $\mathcal{L}(E)$ beginning at the "innermost" subexpressions of E and working out.

See if the list is empty when you finish.

<u>Difficulty</u>: The list for

$$(\cdots(((0 \cup 1)^{2}\underbrace{)^{2})\cdots)}_{n}{}^{2}$$

contains

---

15 $2^n \cdot 2^{2^n}$ bits

16

Theorem 1. There is (for any finite Σ) a constant $k > 0$ and a Turing machine $\mathfrak{M}$ such that

(1) $\mathfrak{M}$ accepts an input w iff w is a well-formed expression and $\mathcal{L}(w) = \phi$.

(2) $T_{\mathfrak{M}}(n) \overset{\text{df.}}{=} \max\{T_{\mathfrak{M}}(x) \mid \ell(x) = n\} \leq$

$$\leq 2^{2^{kn}}$$

17

Theorem 2. There is a finite Σ and a constant $k > 1$ such that

if $\mathfrak{M}$ is *any* T.M. accepting precisely the expressions over Σ describing the empty set, then

$$T_{\mathfrak{M}}(n) > 2^{\sqrt[k]{n}}$$

for infinitely many n.

(That is, $\{E \mid \mathcal{L}(E) = \phi\} \notin \text{Time}(2^{\sqrt[k]{n}})$)

---

18 To prove Theorem 2:

(i) Define a relation on languages

$$L_1 \prec L_2$$

with intuitive meaning that L_1 is easy to decide given L_2.

(ii) Show that for <u>any</u> $L \in \text{Time}(2^n)$

$$L \prec \{E \mid \mathcal{L}(E) = \emptyset\}.$$

(iii) Deduce from the Compression Theorem that there is an $L \in \text{Time}(2^n)$ which is hard to decide.

(iv) Conclude that $\{E \mid \mathcal{L}(E) = \emptyset\}$ is hard to decide.

19

Def. For $L_1 \subset \Sigma_1^*$, $L_2 \subset \Sigma_2^*$ we say $L_1 \prec L_2$ (L_1 is polynomial time reducible to L_2) iff there exists a function $f: \Sigma_1^* \to \Sigma_2^*$

(1) f is computable in time bounded by a polynomial in the length of its argument ($f \in \text{Time}(\, p \circ \ell\,)$ where $p: N \to N$ is a polynomial and $\ell: \Sigma_1^* \to N$ is the length function).

(2) $x \in L_1 \Leftrightarrow f(x) \in L_2$ for all $x \in \Sigma_1^*$.

---

20 **Lemma.** Let $t: N \rightarrow N$ be nondecreasing, and $t(n) \geq n$.

If $L_1 < L_2$ and $L_2 \in \text{Time}(t(n))$, then $L_1 \in \text{Time}(t(p(n)))$ for some polynomial $p: N \rightarrow N$

Contrapositive. If $L_1 \notin \text{Time}(\ 2^{n/4}\ )$ and $L_1 < L_2$, then $\exists\ k > 1$ such that $L_2 \notin \text{Time}(\ 2^{\sqrt[k]{n}}\ )$

---

21 Because $2^{\ell(x)/4}$ is time-honest, the compression theorem implies $\exists\ L_1$ such that

$$L_1 \in \text{Time}(2^n) - \text{Time}(2^{n/4}).$$

Thm. 2 follows immediately from the preceding contrapositive if we show

---

22 Main Construction for Theorem 2.

Lemma. For any $L \in \text{Time}(2^n)$, there is an alphabet Σ such that

$$L < \{E \mid E \text{ is an expression over } \Sigma \text{ and } \mathcal{L}(E) = \emptyset\}.$$

---

23 Choose any $L \in \mathrm{Time}(2^n)$

Say $L \subset \Delta^*$ for some alphabet Δ.

Let M be a Turing machine which

(i) halts on any input of length n in $\leq 2^n$ steps, and

(ii) halts scanning a symbol " 1 " on its tape iff the input is in L.

24 Let Q be the states (boxes in the flowchart) of $\mathfrak{M}$,

let W be the tape symbols of $\mathfrak{M}$ including $b \in W$ for the blank tape symbol, let # be still another symbol.

$$\Sigma \overset{\text{def.}}{=} Q \cup W \cup \{\#\}.$$

25

For $x \in \Delta^*$, $\ell(x)=n$,

Comp$(x) \in \Sigma^*$ is to be:

$$\# \, b^{2^n} \cdot \boxed{\text{start}} \cdot x \, b^{2^n} \#\text{(tape after one step)}\#\cdots$$

$$\cdots \# \text{ (tape after k steps) } \# \text{ (tape after k+1 steps)} \#\cdots$$

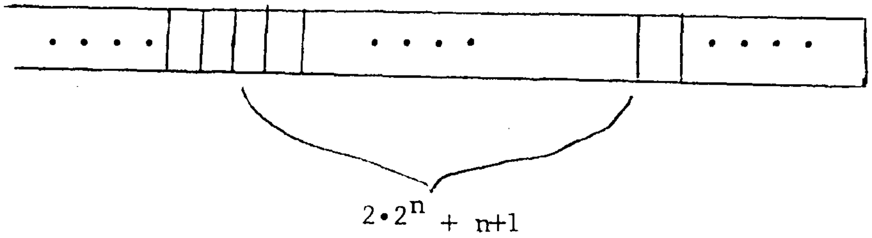
$$\cdots \# \text{ tape } \boxed{\text{halt}} \text{ symbols } \#$$

Exactly $2 \cdot 2^n + n+1$ symbols between successive #'s.

$$\ell(\text{Comp}(x)) \leq 2^{3(n+1)} \overset{\text{df.}}{=} N$$

---

26 Comp(x) has the property that any four consecutive letters determine the letter $2\cdot 2^n + n$ to their right:



Let $F = \{(\sigma_1,\sigma_2,\sigma_3,\sigma_4,\sigma_5) \mid \sigma_5$ is not the letter determined by $\sigma_1\sigma_2\sigma_3\sigma_4\}$. This follows from the fact that at any step the next move of $\mathfrak{M}$ is determined by the state and the tape symbol being scanned.

27

$$\text{Comp}(x) = (\text{starts right}) \cap (\text{ends right}) \cap ((\Sigma \cup \lambda)^N - (\text{moves wrong}))$$

---

starts right: $\#b^{2^n} \cdot \boxed{\text{start}} \cdot x \cdot b^{2^n} \cdot \# \cdot (\Sigma \cup \lambda)^N$

ends right: $(\Sigma \cup \lambda)^N \cdot \boxed{\text{halt}} \cdot (\Sigma \cup \lambda)^N \cdot \#$

---

28 moves wrong:

$$(\Sigma \cup \lambda)^N \cdot \left(\bigcup_F (\sigma_1\sigma_2\sigma_3\sigma_4 \; \Sigma^{2\cdot 2^n+n-1} \cdot \sigma_5)\right) \cdot (\Sigma \cup \lambda)^N$$

---

29 Let $\text{Rejects}(x) =$

$$(\Sigma \cup \lambda)^N \cdot \boxed{\text{halt}} \cdot (\Sigma - \{1\}) \cdot (\Sigma \cup \lambda)^N.$$

Then

$x \in L \Leftrightarrow \mathfrak{M}$ halts reading a 1

$\Leftrightarrow \text{Comp}(x) \cap \text{Rejects}(x) = \emptyset.$

But expressions for Comp(x) and Rejects(x) can be constructed in polynomial time in $\ell(x)$, so $L < \{E \text{ over } \Sigma \mid \mathcal{L}(E) = \emptyset\}$.

Q.E.D.

30

Remarks: (1) Thm. 2 holds for expressions using only " $\cdot$ ", " $\cup$ ", " 2" and letters 0,1 .

(2) If we allow "$\{0,1\}^*$" to be used in expressions Stockmeyer has shown that

$$\{E \text{ with } \{0,1\}^* \mid \mathcal{L}(E) = \emptyset\} \in \text{Time } \left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\} n$$

$$\text{but } \notin \text{Time } \left.2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}\right\} \epsilon \cdot \log_2 n$$

for some fixed $\epsilon > 0$.

(3) If we allow only " $\cup$ ", " $\cdot$ " , the equivalence problem is complete in $\mathcal{NP}$ (discussed in Karp's lecture).

---

31 <u>Remark:</u> Most decidable theories studied in mathematical logic require exponential time or worse. (An important exception being the propositional calculus, for which lower bounds larger than a polynomial are unknown.)

32

<u>Open problems</u>:

(1) Can the satisfiable formulas of the propositional calculus be recognized in polynomial time? (This is the $\mathcal{P} = \mathcal{NP}$ question of Cook and Karp).

(2) Can a multi-tape Turing machine multiply integers (in binary notation) in linear time?

---

33 (3) What is the relation between time and space?

Known: $S_{\mathfrak{M}}(n) \leq T_{\mathfrak{M}}(n) \leq c^{S_{\mathfrak{M}}(n)}$

($c > 1$ depends on $\mathfrak{M}$)

Open: If $L \in \text{Time}(2^n)$ is $L \in \text{Space}(n)$?

(4) Is Space(n) = Nondeterministic Space(n)?
(The LBA problem of Myhill)

---

(5) Are linear time 3 tape T.M.'s more powerful than linear time 2-tape T.M.'s?

(6) Can the primes (represented in binary) be recognized in linear time?

Can the context-free languages?

(7) Can two nxn matrices be multiplied in proportional to $n^{2.8}$ arithmetic operations?

($n^{2.9}$ is known to be possible.)

References:

Abstract Complexity

1. Borodin, A. Computational Complexity: Theory and Practice, in *Currents in the Theory of Computing*, A. Aho, ed., Prentice-Hall, Englewood Cliffs, N.J., 1973, pp.35-89.

2. Hartmanis, J. and J. Hopcroft, An overview of the theory of computational complexity, *Jour. ACM*, *18*, 3 (1971), pp.444-475.

Fast arithmetic

1. Knuth, D. *The Art of Computer Programming*: Vol. 2, *Seminumerical Algorithms*, Addison-Wesley, Reading, Mass., 1969.

BIBLIOGRAPHIC DATA SHEET	1. Report No. NSF-OCA-GJ34671 - TM-39	2.	3. Recipient's Accession No.

4. Title and Subtitle

Discrete Computation: Theory and Open Problems

5. Report Date : Issued January 1974

6.

7. Author(s)

Albert R. Meyer

8. Performing Organization Rept. No. MAC TM-39

9. Performing Organization Name and Address

PROJECT MAC; MASSACHUSETTS INSTITUTE OF TECHNOLOGY:
545 Technology Square, Cambridge, Massachusetts 02139

10. Project/Task/Work Unit No.

11. Contract/Grant No.

GJ34671

12. Sponsoring Organization Name and Address

Associate Program Director
Office of Computing Activities
National Science Foundation
Washington, D. C. 20550

13. Type of Report & Period Covered: Interim Scientific Report

14.

15. Supplementary Notes

Preceptorial Introduction to Computer Science for Mathematicians

16. Abstracts

17. Key Words and Document Analysis. 17a. Descriptors

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement

Unlimited Distribution
Write Project MAC Publications

19. Security Class (This Report) UNCLASSIFIED

20. Security Class (This Page UNCLASSIFIED

21. No. of Pages 36

22. Price

THIS FORM MAY BE REPRODUCED

MIT/LCS/TM-39

DISCRETE COMPUTATION: THEORY AND OPEN PROBLEMS

Albert R. Meyer

January 1974