Zeitschrift: L'Enseignement Mathématique
Herausgeber: Commission Internationale de l'Enseignement Mathématique
Band: 28 (1982)
Heft: 1-2: L'ENSEIGNEMENT MATHÉMATIQUE

Artikel: TURING MACHINES THAT TAKE ADVICE
Autor: Karp, Richard M. / Lipton, Richard J.
Kapitel: 6. The Method of Recursive Definition
DOI: https://doi.org/10.5169/seals-52237

Nutzungsbedingungen
Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Siehe Rechtliche Hinweise.

Conditions d'utilisation
L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. Voir Informations légales.

Terms of use
The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. See Legal notice.

Download PDF: 02.04.2025

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

Since DAG is logspace complete in *NSPACE* (*log n*), it suffices to show that

$$\text{DAG} \in \mathit{DSPACE}\,(\log n)/\log \Rightarrow \text{DAG} \in \mathit{DSPACE}\,(\log n)\ .$$

Suppose that DAG $=$ S : h, where S $\in$ *DSPACE* (*log n*) and

$$|h(n)| \leqslant k \log_2 n\ .$$

Then, guided by the self-reducibility of DAG, we can test whether $(\Psi, s, t) \in$ DAG by performing the following computation for each string w of length $\leqslant k \log_2 n$:

$v := s$;

while v has out-degree 2 do

$v :=$ if $w \cdot (\Psi, v_0, t) \in$ S then v_0 else v_1.

If v is ever set equal to t then accept (Ψ, s, t); otherwise, reject it. It is clear that this method recognizes DAG deterministically within space 0 (*log n*). ■

6. The Method of Recursive Definition

Let K be a subset of $\{0, 1\}^*$, and let $C_K : \{0, 1\}^* \to \{0, 1\}$ be the characteristic function of K. By a recursive definition of C_K we mean a rule that specifies C_K on a "basis set" A $\subseteq \{0, 1\}^*$, and uniquely determines C_K on the rest of $\{0, 1\}^*$ by a recurrence formula of the form

$$C_K(x) = F(x, C_K(f_1(x)), C_K(f_2(x)), \dots, C_K(f_t(x))),$$
$$x \in \{0, 1\}^* - \text{A}\ .$$

Example 1. Let G be a game, as defined in Section 4, and let G be the set of positions from which the player to move can force a win. Then G is uniquely determined by

(i) if $x \in$ W then $x \in$ G

(ii) if $x \in \{0, 1\}^* -$ W then $x \in$ G $\Leftrightarrow F_0(x) \notin$ G or $F_1(x) \notin$ G.

Example 2. Let $(<, \text{A}, G_0, G_1)$ be a self-reducibility structure for the set $K \subseteq \{0, 1\}^*$. Then K is determined uniquely by its intersection with A, together with the recurrence

$$\text{for} \quad x \notin \text{A},\ x \in \text{K} \Leftrightarrow G_0(x) \in \text{K} \cup G_1(x) \in \text{K}\ .$$

The theme of the present section is that, when C_K has a simple enough recursive definition, bounds on the nonuniform complexity of K yield bounds on its uniform complexity. The idea is as follows. Suppose $K = S : h$, and C_K is determined by its values on A, together with the recurrence formula

$$C_K(x) = F(x, C_K(f_1(x)), \ldots, C_K(f_t(x))), \; x \in \{0, 1\}^* - A ,$$

where

$$|f_1(x)| = |f_t(x)| = |x| .$$

For any string w, define $K_w = \{x \mid wx \in S\}$. Then, for $x \in A$, we can make the following assertion:

$$x \in K \Leftrightarrow \exists w [x \in K_w] \wedge \forall y [C_{K_w}(y) = F(y, C_{K_w}(f_1(y)), \ldots, C_{K_w}(f_t(y))] .$$

Here, w ranges over all strings of length $|h(|x|)|$, and y ranges over all strings of the same length as x. The above formula suggests a uniform algorithm to test membership in K by searching through all choices of w and y. Further, the quantifier structure of the formula allows us to conclude that K lies in $\sum_2^p$, provided that $|h(n))|$ is bounded by a polynomial in n, S is in P, and F is computable in polynomial time.

As an illustration of this approach, we prove that, if NP has small circuits, then $\bigcup_i \sum_i^p = \sum_2^p$, i.e., the polynomial-time hierarchy collapses. Originally we proved this with $\sum_2^p$ replaced by $\sum_3^p$. The improvement is due to M. Sipser.

THEOREM 6.1. If $NP \subseteq P/poly$ then $\sum_2^p = \bigcup_{i=1}^{\infty} \sum_i^p$.

The proof of this theorem requires the following lemma.

LEMMA 6.2. If $NP \subseteq P/poly$ then $\bigcup_{i=1}^{\infty} \sum_i^p \subseteq P/poly$.

Proof. Let E_i be the set of encodings of true sentences of the form

$$(*) \quad Q_1 \vec{x}_1 Q_2 \vec{x}_2 \ldots Q_i \vec{x}_i F(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_i)$$

where $Q_1 = \exists$, the Q_j are alternately $\exists$ and $\forall$, $\vec{x}_j$ is shorthand for the triple $x_{j_1}, x_{j_2}, \ldots, x_{j,r_j}$ of Boolean variables, and F is a propositional formula. Let A_i be defined in the same way, except that $Q_1 = \forall$. It is known that E_i is logspace complete in $\sum_i^p$, and A_i is logspace complete in

$\prod_i^p$. Also, it is clear that $A_i \in P/poly \Leftrightarrow E_i \in P/poly$. It suffices for the lemma to prove that $E_i \in P/poly$ for all i.

By hypothesis, $E_1 \in P/poly$. We proceed by induction on i. Assume $E_{i-1} \in P/poly$; then $A_{i-1} \in P/poly$. Thus there exists a set $S \in P$, a constant k and a function $h: N \rightarrow \{0, 1\}^*$ such that $|h(n)| \leqslant k + n^k$ and $x \in A_{i=1} \Leftrightarrow h(|x|) \cdot x \in S$.

If y is the encoding of a sentence of the form (*), and $\vec{a}$ is a t_1-tuple of boolean variables, let $y_{\vec{a}}$ denote the encoding of the sentence that results from y by deleting the quantifier Q_1 and substituting $\vec{a}$ for $\vec{x}_i$ in $F(\vec{x}_1, \vec{x}_2, ..., \vec{x}_i)$. We choose our encoding conventions and method of substitution so that the length of $y_{\vec{a}}$ is equal to the length of y.

Since $S \in P$, the following set T is in NP:

$$T = \{wy \mid \text{for some } \vec{a}, w \cdot y_{\vec{a}} \in S\}.$$

By hypothesis $T \in P/poly$, so there exist $S' \in P$, $k' \in N$ and $h': N \rightarrow \{0, 1\}^*$ so that $|h'(n)| \leqslant k' + n^{k'}$ and $x \in T \Leftrightarrow h'(|x|) \cdot x \in S$. Then $y \in A_i \Leftrightarrow$ for some $\vec{a}$, $y_{\vec{a}} \in E_{i-1} \Leftrightarrow$ for some a,

$$h(|y_{\vec{a}}) \cdot y_{\vec{a}} \in S \Leftrightarrow h(|y_{\vec{a}}|) \cdot y \in T \Leftrightarrow h'(|h(|y_{\vec{a}}|) \cdot y|(\cdot h(|y_{\vec{a}}|) \cdot y \in S'.$$

But the prefix $h'(|h(|y_{\vec{a}}|) \cdot y|(\cdot h(|y_{\vec{a}}|)$ is a polynomial-bounded function of $|y|$; also $S' \in P$. These two facts together establish that $A_i \in P/poly$. ■

Proof of Theorem 6.1. It suffices to prove that $NP \subseteq P/poly \Rightarrow \prod_3^p \subseteq \sum_2^p$; for this it is sufficient to prove that the set A_3 is in $\sum_2^p$. Our proof is based on the fact that A_3 has an easty-to-evaluate recursive definition of the form $C_{A_3}(y) = R(y, C_{A_3}(y'), C_{A_3}(y''))$. Consider a sentence y of the form

$$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n F(x_1, x_2, \ldots, x_n)$$

where the string of quantifiers $Q_1 Q_2 \ldots Q_n$ is contained in $\forall^* \exists^* \forall^*$.

Let

$$y' = Q_2 x_2 \ldots Q_n x_n F(0, x_2, \ldots, x_n)$$

and

$$y'' = Q_2 x_2 \ldots Q_n x_n F(1, x_2, \ldots, x_n).$$

Then

$$C_{A3}(y) = (\text{if } Q_1 = \forall \text{ then } C_{A3}(y') \wedge C_{A3}(y'') \text{ else } C_{A3}(y') \cup C_{A3}(y'')).$$

C_{A_3} is uniquely determined by this recursive definition which is of the form $C_{A_3}(y) = R((y, C_{A_3}(y'), C_{A_3}(y''))$, coupled with its values on the "basis set" consisting of sentences without quantifiers.

By Lemma 6.2, $A_3 \in P/poly$. Thus $A_3 = S:h$ where $S \in P$ and $|h(n)| \leqslant k + n^k$. For each $w \in \{0,1\}^*$ define $f_w: \{0,1\}^* \to \{0,1\}$ by $f_w(x) = 1 \Leftrightarrow wx \in S$. Then membership of y in A_3, in the case where y contains at least one quantifier, is expressed by the following formula:

$$(**) \qquad \exists w \forall z \left[f_w(y) = 1 \wedge f_w(z) = R\left(z, f_w(z'), f_w(z'')\right)\right] .$$

Here w ranges over all strings of length $\leqslant k + |y|^k$, and z ranges over all strings of length $|y|$. Also, with the help of a polynomial-time algorithm to test membership in S, the property $f_w(y) = 1$ and

$$f_w(z) = R\left(z, f_w(z'), f_w(z'')\right)$$

can be tested in polynomial time. Thus the $\exists\,\forall$ form of (**) establishes that $A_3 \in \sum_2^p$. ■

Theorem 6.1 has a number of corollaries.

COROLLARY 6.3. If $R = NP$ then $\bigcup_i \sum_i^p = \sum_2^p$.

This follows immediately from the observation [1] that every set in R has small circuits.

The next corollary concerns sparse sets. A set S is *sparse* [6, 7] if

$$\exists\, c\, \forall n \geqslant 2, |S \cap \{0,1\}^n| \leqslant n^c .$$

COROLLARY 6.4. If there is a sparse set S that is complete in NP with respect to polynomial time Turing reducibility (cf. Cook [4]), then

$$\bigcup_i \sum_i^p = \sum_2^p .$$

This corollary follows immediately from Theorem 6.1 once it is noted that the existence of such an S implies that every set in NP has small circuits. Corollary 6.4 should be compared with results of Mahaney [11] and Fortune [6] which show that, if there exists a sparse or co-sparse set which is complete in NP with respect to many-one polynomial-time reducibility (Karp [8]) then $P = NP$. Note that Corollary 6.4 has a weaker conclusion than the results of Mahaney and Fortune, but also a weaker hypothesis.

Let *ZEROS* denote the following decision problem: given a prime q and a set $\{p_1(x), p_2(x), \ldots, p_n(x)\}$ of sparse polynomials with integer coefficients, to determine whether there exists an integer x such that, for $i = 1, 2, \ldots, n$, $p_i(x) \equiv 0 \mod q$.

COROLLARY 6.5. If $ZEROS \in P/poly$, then $\cup \sum_i^p = \sum_2^p$.

This is based on Plaisted's result [15] that every problem in NP can be solved in polynomial time with the help of an oracle for $ZEROS$ together with a polynomial-bounded number of advice bits. Thus $NP \subseteq P/poly$ if $ZEROS \in P/poly$.

THEOREM 6.6. (Meyer) $EXPTIME \subseteq P/poly \Leftrightarrow EXPTIME = \sum_2^p$.

Proof. Let G be the set of strings representing positions from which the first player can win in the *EXPTIME*-complete game mentioned in FACT 1. It suffices to prove that

$$G \in P/poly \Rightarrow G \in \sum_2^p .$$

Suppose $G = S : h$ where $S \in P$ and h is polynomial-bounded. Then

$$x \in G \Leftrightarrow \exists w \forall z [x \in W \cup z \in W \cup (wz \in S \Leftrightarrow wF_0(z) \notin S \cup wF_1(z) \notin S)]$$

Here w ranges over all strings of length $| h(|x|)$ and z ranges over all strings of the same length as x. Since membership in S or membership in W can be tested in polynomial time, it tollows that $G \in \sum_2^p$. ■

COROLLARY 6.7. $EXPTIME \subseteq P/poly \Rightarrow P \neq NP$.

Proof. Assume for contradiction that $EXPTIME \subseteq P/poly$ and $P = NP$. The first hypothesis implies that $EXPTIME = \sum_2^p$, and the second implies that $P = \sum_2^p$. Hence $P = EXPTIME$. But this contradicts the result that $P \subsetneq EXPTIME$, which is easily proved by diagonalization. ■

Figure 1. MAIN RESULTS

$PSPACE \subseteq P/poly \Rightarrow PSPACE = \sum_2^p \cap \sum_2^p$

$PSPACE \subseteq P/log \Leftrightarrow PSPACE = P$

$EXPTIME \subseteq PSPACE/poly \Leftrightarrow EXPTIME = PSPACE$

$P \subseteq DSPACE((\log n)^l)/log \Leftrightarrow P \subseteq DSPACE((\log n)^l)$

$NSPACE(\log n) \subseteq DSPACE(\log n)/log$
$\Leftrightarrow NSPACE(\log n) = DSPACE(\log n)$

$$NP \subseteq P/log \Leftrightarrow P = NP \quad (^1)$$

$$NP \subseteq P/poly \Rightarrow \cup \textstyle\sum_i^p = \sum_2^p \quad (^2)$$

$$EXPTIME \subseteq P/poly \Rightarrow EXPTIME = \textstyle\sum_2^p \Rightarrow P \neq NP \quad (^3)$$

REFERENCES

[1] ADLEMAN, L. Two Theorems on Random Polynomial Time. *Proc. 19th IEEE Symp. on Foundations of Computer Science*, pp. 75-83 (1978).

[2] ALELIUNAS, R., R. M. KARP, R. J. LIPTON, L. LOVÁSZ and C. RACKOFF. Random Walks, Universal Sequences, and the Complexity of Maze Problems. *Proc. 20th IEEE Symp. on Foundations of Computer Science*, pp. 218-223 (1979).

[3] CHANDRA, A. K. and L. J. STOCKMEYER. Alternation. *Proc. 17th IEEE Symp. on Foundations of Computer Science*, pp. 98-108 (1976).

[4] COOK, S. A. The Complexity of Theorem-Proving Procedures. *Proc. 3rd ACM Symp. on Theory of Computing*, pp. 151-158 (1971).

[5] —— Towards a Complexity Theory of Synchronous Parallel Computation. *Technical Report 141/80*, Computer Science Department, University of Toronto (1980).

[6] FORTUNE, S. A Note on Sparse Complete Sets. *SIAM J. Computing 8*, pp. 431-433 (1979).

[7] HARTMANIS, J. and L. BERMAN. On Isomorphisms and Density of *NP* and Other Complete Sets. *Proc. 8th ACM Symp. on Theory of Computing*. pp. 30-40 (1976).

[8] KARP, R. M. Reducibility Among Combinatorial Problems. I: *Complexity of Computer Computations* (R. E. Miller and J. W. Thatcher, eds.), Plenum, New York (1972).

[9] KARP, R. M. and R. J. LIPTON. Some Connections Between Nonuniform and Uniform Complexity Classes. *Proc. 12th Annual ACM Symposium on Theory of Computing*, pp. 302-309 (1980).

[10] KOZEN, D. On Parallelism in Turing Machines. *Proc. IEEE Symp. on Foundations of Computer Science*, pp. 89-97 (1976).

[11] MAHANEY, S. R. Sparse Complete Sets for NP: Solution of a Conjecture of Berman and Hartmanis. *Proc. 21st IEEE Symp. on Foundations of Computer Science*, pp. 54-60 (1980).

[12] MEYER, A. R. and M. S. PATERSON. With What Frequency are Apparently Intractable Problems Difficult, *M.I.T. Tech. Report*, Feb. 1979.

[13] MEYER, A. R. and L. J. STOCKMEYER. The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space. *Proc. 13th IEEE Symp. on Switching and Automata Theory*, pp. 125-129 (1972).

[14] PIPPENGER, N. On Simultaneous Resource Bounds. *Proc. 20th IEEE Symp. on Foundations of Computer Science*, pp. 307-311 (1979).

[15] PLAISTED, D. A. New NP-hard and NP-complete Polynomial and Integer Divisibility Problems. *Proc. 18th IEEE Symp. on Foundations of Computer Science*, pp. 241-253 (1977).

---

(1) Obtained jointly with Ravindran Kannan.
(2) An improvement by Michael Sipser of an early result of ours.
(3) Due to Albert Meyer.