A homomorphic characterization of time and space complexity classes of languages†

by

K. Culik II and N.D. Diamond
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

Research Report CS-79-07

February 1979

† This research was supported by the Natural Sciences and Engineering Research Council of Canada, Grant No. A 7403.

Abstract

Recently, it has been shown that for each recursively enumerable language there exists an erasing homomorphism h_0 and homomorphisms h_1, h_2 such that $L = h_0(e(h_1, h_2))$ where $e(h_1, h_2)$ is the set of minimimal words on which h_1 and h_2 agree. Here we show that by restrictions on the erasing h_0 we obtain most time-complexity language classes, and by restrictions on the pair (h_1, h_2) we characterize all space complexity language classes.

CR Categories: 5.23, 5.25

Keywords: homomorphic characterization, complexity classes, time complexity, space complexity, language families.

Introduction

Problems concerning homomorphism equivalence have turned out to be of crucial importance in some recent developments in formal language theory. We mention the decidability of the (ultimate) DOL equivalence problem [2, 5], the homomorphic equivalence on context free languages [7] and its applications [4]. Typically we need to check whether two homomorphisms h_1, h_2 on a free monoid Σ^* are equal for every element of certain subset of Σ^*, or alternatively, to find the language of all w in Σ^* for which $h_1(w) = h_2(w)$. Such languages are called equality sets in [9]. They turn out to be a powerful tool in the characterization of various classes of languages (see [3, 6]).

The main result of [3] is that for every recursively enumerable language L there exist erasing h_0 (a homomorphism either preserving or erasing any symbol) and homomorphisms h_1, h_2 such that $L = h_0(e(h_1, h_2))$ where $e(h_1, h_2)$ is the set of minimal strings on which h_1 and h_2 agree. It was indicated in [3] that by restricting the erasing h_0 we obtain various time-complexity classes of languages and by restrictions on the pair (h_1, h_2) we obtain various complexity classes of languages. The time-complexity characterization was suggested by a referee of [3]. In this paper this "machine independent" characterization of most of the time complexity classes (except for obvious restrictions, only closure under squaring is required) and of essentially all of the space complexity classes is obtained.

The well known notion of k-limited erasing (see [8]) is generalized as follows. For a function f on the integers we say that erasing h is f-bounded on a language L if for each w in L at most $f(|w|)$ consecutive symbols of w may be erased. For any class of "nice" complexity functions C closed under squaring let L_C be the class of languages accepted by nondeterministic Turing machines that operate with time bounds in C . We show that L is in L_C iff there exist homomorphisms h_0, h_1, h_2 such that $h_0(e(h_1, h_2)) = L$ and h_0 is f-bounded erasing on $e(h_1, h_2)$ for some f in C .

As special cases we have, for example, the following. A language L is in NP (is primitive recursive, recursive) iff there exist homomorphisms h_0, h_1, h_2 such that $h_0(e(h_1, h_2)) = L$ and h_0 is polynomial- (primitive recursive-, recursive-) bounded erasing on $e(h_1, h_2)$.

The notion of the balance of a pair of homomorphisms was introduced in [5] and in [3] it was shown that a language L is regular if and only if it can be written in the form $L = h_0(e(h_1, h_2))$, where h_0 is an erasing and the pair of homomorphisms h_1, h_2 has k-bounded balance for some constant k .

Just as we have generalized the notion of k-limited (bounded) erasing, we can similarly generalize k-bounded balance. Given a function f on the integers, a language $L \subseteq \Sigma^*$ and an erasing h on Σ^* , we say that a pair of homomorphisms (h_1, h_2) has f-bounded balance on L with respect to h if for each x in L and each prefix w of x we have $||h_1(w)| - |h_2(w)|| < f(|h(x)|)$. We show for all classes of "nice" complexity functions C that a language L is of space complexity C iff there exist an erasing h_0 and homomorphisms h_1, h_2

such that $L = h_0(e(h_1, h_2))$ and the pair (h_1, h_2) has f-bounded balance on $e(h_1, h_2)$ with respect to h_0 for some f in C . For example, the context sensitive languages are exactly those which can be expressed in the form $h_0(e(h_1, h_2))$ where the pair (h_1, h_2) has linear-bounded balance on $e(h_1, h_2)$ with respect to h_0 .

Similar results to those presented here have also been obtained independently by Book and Brandenburg [1].

1. Preliminaries

We assume familiarity with basic formal language theory (see [8]). We recall some basic definitions and some definitions from [3].

We say that C is a class of complexity functions if C is a class of functions closed under addition of and multiplication by a constant. A language L is of time (space) complexity C if L is accepted by a nondeterministic multitape on-line Turing machine M which operates within time-bound (space-bound) f, for some f in C.

A homomorphism $h : \Sigma^* \to \Delta^*$ is called an erasing if for some subset T of Σ we have $h(a) = a$ if $a \in T$ and $h(b) = \varepsilon$ if $b \in \Sigma - T$ (ε is the empty string).

Let h_1, h_2 be two homomorphisms, $h_1, h_2 : \Sigma^* \to \Delta^*$. Define the equality set of h_1, h_2 as
$E(h_1, h_2) = \{w \in \Sigma^* : h_1(w) = h_2(w)\}$ and the minimal (equality) set of h_1, h_2 as $e(h_1, h_2) = \{w \in \Sigma^+ : h_1(w) = h_2(w)$ and $h_1(u) \neq h_2(u)$ for each proper nonempty prefix u of $w\}$. Note that
$e(h_1, h_2) = \mathrm{Min}(E(h_1, h_2)) - \{\varepsilon\}$ using the notation of [H + U].

2. Space-Complexity Classes

When considering on-line space complexity we can, clearly, without restriction of generality, consider the following normal form of on-line Turing machine with one storage tape. Our machine is a seven-tuple $M = (K, T, V, B, \delta, q_0, F)$ where:

K is the set of states;

T is the terminal alphabet (on the input tape);

V is the tape alphabet (on the storage tape);

$B \in V$ is the storage tape's blank character;

$\delta : K \times T \times V \to$ finite subsets of $T \times \{N, R\} \times V \times \{L, N, R\} \times K$ is the transition function;

q_0 in K is the initial state;

$F \subseteq K$ is the set of final (accepting) states.

We assume that $T \cap V = \phi$. Furthermore we assume that there are K_T, K_V, δ_T and δ_V such that $K = K_T \cup K_V \cup F$ and $|K| = |K_T| + |K_V| + |F|$ (disjoint union); and

$\delta_T : K_T \times T \to$ finite subsets of K

$\delta_V : K_V \times V \to$ finite subsets of $(V - \{B\}) \times \{L, N, R\} \times K$

such that δ satisfies the following conditions:

(i) $\delta(q, a, A) = \{a\} \times \{R\} \times \{A\} \times \delta_T(q, a)$ for $q \in K_T$,

(ii) $\delta(q, a, A) = \{a\} \times \{N\} \times \delta_V(q, A)$ for $q \in K_V$,

(iii) $\delta(q, a, A) = \phi$ for $q \in F$.

This means that each move of machine M consists of either

(i) reading and advancing the tape head one symbol on the input tape, while ignoring and preserving the storage tape; or

(ii) performing a nondeterministic computation on the storage tape, while ignoring and preserving the input tape (a blank can not be written in this case).

The machine must always halt upon entering an accepting state.

We will now introduce notation describing possible moves of our machine by triples from $V^{(2)}KV^{(2)} \times (T \cup \{\varepsilon\}) \times V^{(2)}KV^{(2)}$ where $V^{(2)} = V^2 \cup V \cup \{\varepsilon\}$. In the following always A, C, D are in $V - \{B\}$.

Let
$$\Omega_T = \{\langle q, a, p\rangle : a \in T , \ p \in \delta_T(q, a)\} ,$$
$$\Omega_N = \{\langle qA, \varepsilon, pC\rangle : (C, N, p) \in \delta_V(q, A)\} ,$$
$$\Omega_{NB} = \{\langle q, \varepsilon, pC\rangle : (C, N, p) \in \delta_V(q, B)\} ,$$
$$\Omega_R = \{\langle qA, \varepsilon, Cp\rangle : (C, R, p) \in \delta_V(q, A)\} ,$$
$$\Omega_{RB} = \{\langle q, \varepsilon, Cp\rangle : (C, R, p) \in \delta_V(q, B)\} ,$$
$$\Omega_L = \{\langle AqC, \varepsilon, pAD\rangle : (D, L, p) \in \delta_V(q, C)\} ,$$
and
$$\Omega_{LB} = \{\langle Aq, \varepsilon, pAC\rangle : (C, L, p) \in \delta_V(q, B)\} .$$

Finally, let $\Omega = \Omega_T \cup \Omega_N \cup \Omega_{NB} \cup \Omega_R \cup \Omega_{RB} \cup \Omega_L \cup \Omega_{LB}$. We also will need the partition $\Omega = \Omega_S \cup \Omega_F$, where

$$\Omega_S = \{\langle u, a, \eta p \zeta\rangle : p \in K - F\}$$
and
$$\Omega_F = \{\langle u, a, \eta p \zeta\rangle : p \in F\} .$$

The above notation will be used in the proof of Theorem 1. To formulate it we need to generalize the notion of bounded balance (see [5]) .

<u>Definition</u> Consider two fixed homomorphisms h_1 and h_2 from Σ^* to Δ^* and a word w in Σ^* . The <u>balance</u> of w is defined by

$$B(w) = |h_1(w)| - |h_2(w)|$$

where $|x|$ denotes the length of x. Given a function f on the integers, we say that a pair of homomorphisms (h_1, h_2) has f-bounded balance on a language L with respect to an erasing h_0, if for each x in L and each prefix w of x we have $|B(w)| \leq f(|h(x)|)$. Given a complexity class C, we say that (h_1, h_2) has C-bounded balance on L with respect to h_0, if the same is true for some $f \in C$.

In [3, Theorem 4] it was shown that a constant bound on the balance of the pair (h_1, h_2) on $e(h_1, h_2)$ in $h_0(e(h_1, h_2))$ characterizes the regular sets ("with respect to erasing h_0" can be omitted for constant bound). We will extend this result for constant space bounds to arbitrary space complexity.

Theorem 1 Let C be any class of complexity functions. Then for each language L, L is of (nondeterministic, on-line) space complexity C iff there exist an erasing h_0 and homomorphisms h_1, h_2 such that $L = h_0(e(h_1, h_2))$ and the pair (h_1, h_2) has C-bounded balance on $e(h_1, h_2)$ with respect to h_0.

Proof

(Only if)

Let L be of space complexity C, that is let L be accepted by Turing machine $M = (K, T, V, B, \delta, q_0, F)$ of the form described above, operating with space bound s_1 for some s_1 in C. We construct homomorphisms h_0, h_1 and h_2 using technique adapted from those in [3].

Let

$$\Gamma = \Omega \cup V \cup \{\#\} \quad ,$$

$$\overline{\Gamma} = \{\overline{\gamma} : \gamma \in \Gamma\} \quad ,$$

$$\hat{V} = \{\hat{A} : A \in V\} \quad ,$$

$$\Sigma = \Gamma \cup \overline{\Gamma} \cup \hat{V} \cup \{\vdash, \overline{\vdash}, \$, 0, 2, 3\} \ ,$$

and

$$\Delta = \Gamma \cup \overline{\Gamma} \cup \{0,1,2,3,\vdash,\dashv\} \quad .$$

Define homomorphisms h_1 , h_2 from Σ^* to Δ^* , and h_0 from Σ^* to T^* , by the following table:

ξ	$\vdash$	$\overline{\vdash}$	$\langle\alpha q\beta,\tilde{a},\eta p_S\zeta\rangle$	$\overline{\langle\alpha q\beta,\tilde{a},\eta p_S\zeta\rangle}$	A	$\overline{A}$	$\#$	$\overline{\#}$	$\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle$	$\overline{\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle}$	$\$$	$\hat{A}$	0	2	3
$h_1(\xi)$	$\vdash$	$\vdash$	$\alpha q\beta$	$\overline{\alpha}\overline{q}\overline{\beta}$	A	$\overline{A}$	$\#$	$\overline{\#}$	$\alpha q\beta$	$\overline{\alpha}\overline{q}\overline{\beta}$	$\#$	$\overline{A}$	10	$\dashv$	123
$h_2(\xi)$	$\vdash\overline{q}_0\overline{\#}$	$\vdash q_0\#$	$\overline{\eta}\overline{p}_S\overline{\zeta}$	$\eta p_S\zeta$	$\overline{A}$	A	$\overline{\#}$	$\#$	$\overline{\eta}\overline{\zeta}$	$\eta\zeta$	$\dashv 1$	01	ε	2	3
$h_0(\xi)$	ε	ε	$\tilde{a}$	$\tilde{a}$	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε	ε

for all $\langle\alpha q\beta,\tilde{a},\eta p_S\zeta\rangle \in \Omega_S$ $\quad$ ($\overline{\langle\alpha q\beta,\tilde{a},\eta p_S\zeta\rangle}$ accordingly) ,

$\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle \in \Omega_F$ $\quad$ ($\overline{\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle}$ accordingly) ,

and $A \in V$ ($\overline{A}$, $\hat{A}$ accordingly).

Then each element σ of $e(h_1, h_2)$ looks like either

$$(h_1)\quad \vdash \quad \overbrace{\overline{q_0}}^{\overline{w_0}} \quad \overbrace{\bar{\#} u_1 \quad \alpha_1 q_1 \beta_1 \quad v_1}^{w_1} \overbrace{\#\overline{u_2} \quad \overline{\alpha_2}\overline{q_2}\overline{\beta_2} \quad \overline{v_2}\bar{\#}}^{\overline{w_2}}$$

$$(\sigma)\quad \vdash \quad \overline{<q_o,\tilde{a}_o,\eta_o q_1 \zeta_o>}\bar{\#}u_1<\alpha_1 q_1 \beta_1,\tilde{a}_1,\eta_1 q_2 \zeta_1>v_1\#\overline{u_2}\overline{<\alpha_2 q_2 \beta_2,\tilde{a}_2,\eta_2 q_3 \zeta_2>}\overline{v_2}\bar{\#}$$

$$(h_2)\quad \vdash \underbrace{\overline{q_o}\bar{\#}}_{\overline{w_o}} \quad \underbrace{\eta_o q_1 \zeta_o}_{w_1} \quad \underbrace{\#\overline{u_1} \quad \overline{\eta_1}\overline{q_2}\overline{\zeta_1} \quad \overline{v_1}}_{\overline{w_2}} \underbrace{\bar{\#}u_2 \quad \eta_2 q_3 \zeta_2 \quad v_2\#}_{w_3}$$

$$(h_1)\quad \cdots \overbrace{\#\overline{u_{n-2}} \quad \overline{\alpha_{n-2}}\,\overline{q_{n-2}}\,\overline{\beta_{n-2}} \quad \overline{v_{n-2}}\#}^{\overline{w_{n-2}}}$$

$$(\sigma)\quad \cdots \#\overline{u_{n-2}}\overline{<\alpha_{n-2}q_{n-2}\beta_{n-2},\tilde{a}_{n-2},\eta_{n-2}q_{n-1}\zeta_{n-2}>}\overline{v_{n-2}}\#$$

$$(h_2)\quad \cdots \underbrace{\bar{\#}u_{n-2} \quad \eta_{n-2}q_{n-1}\zeta_{n-2} \quad v_{n-2}\#}_{w_{n-1}}$$

$$(h_1)\quad \overbrace{u_{n-1} \quad \alpha_{n-1}q_{n-1}\beta_{n-1} \quad v_{n-1}}^{w_{n-1}} \quad \#\overbrace{\overline{w}_n}^{\overline{w}_n} \qquad \dashv 10\ 10\ \ldots\ 10\ 123$$

$$(\sigma)\quad u_{n-1}<\alpha_{n-1}q_{n-1}\beta_{n-1},\tilde{a}_{n-1},\eta_{n-1}q_n\zeta_{n-1}>v_{n-1} \quad \$\hat{w}_n \qquad 2\ 0\ \ 0\ \ \ldots\ 0\ \ 3$$

$$(h_2)\quad \underbrace{\overline{u_{n-1}} \quad \overline{\eta_{n-1}}\ \overline{\zeta_{n-1}} \quad \overline{v_{n-1}}}_{\overline{w}_n}\dashv\ 1\ 0101\ldots01\ 2\ \varepsilon\ \ \varepsilon\ \ \ldots\ \varepsilon\ \ 3$$

(if n is even); or

$$(h_1)\quad \vdash \quad \overbrace{q_o}^{w_o} \quad \overbrace{\#\overline{u}_1 \quad \overline{\alpha}_1\overline{q}_1\overline{\beta}_1 \quad \overline{v}_1}^{\overline{w}_1}\overbrace{\overline{\#}u_2 \quad \alpha_2 q_2 \beta_2 \quad v_2\#}^{w_2}$$

$$(\sigma)\quad \vdash \quad <q_o,\tilde{a}_o,\eta_o q_1\zeta_o>\#\overline{u}_1\overline{<\alpha_1 q_1\beta_1,\tilde{a}_1,\eta_1 q_2\zeta_1>}\overline{v}_1\overline{\#}u_2<\alpha_2 q_2\beta_2,\tilde{a}_2,\eta_2 q_3\zeta_2>v_2\#$$

$$(h_2)\quad \underbrace{\vdash q_o\#}_{w_o} \quad \underbrace{\overline{\eta}_o\overline{q}_1\overline{\zeta}_o}_{\overline{w}_1} \quad \underbrace{\overline{\#}u_1 \quad \eta_1 q_2\zeta_1 \quad v_1}_{w_2}\underbrace{\#\overline{u}_2 \quad \overline{\eta}_2\overline{q}_3\overline{\zeta}_2 \quad \overline{v}_2\overline{\#}}_{\overline{w}_3}$$

$$(h_1)\quad \ldots\overbrace{\#\overline{u_{n-2}} \quad \overline{\alpha_{n-2}}\,\overline{q_{n-2}}\,\overline{\beta_{n-2}} \quad \overline{v_{n-2}}\overline{\#}}^{\overline{w_{n-2}}}$$

$$(\sigma)\quad \ldots\#\overline{u_{n-2}}\overline{<\alpha_{n-2}q_{n-2}\beta_{n-2},\tilde{a}_{n-2},\eta_{n-2}q_{n-1}\zeta_{n-2}>}\overline{v_{n-2}}\overline{\#}$$

$$(h_2)\quad \ldots\underbrace{\#u_{n-2} \quad \eta_{n-2}q_{n-1}\zeta_{n-2} \quad v_{n-2}\#}_{w_{n-1}}$$

$$(h_1)\quad \overbrace{u_{n-1} \quad \alpha_{n-1}q_{n-1}\beta_{n-1} \quad v_{n-1}}^{w_{n-1}} \quad \overbrace{\#\overline{w_n}}^{\overline{w_n}} \qquad \dashv\ 10\ 10\ldots10\ 123$$

$$(\sigma)\quad u_{n-1}<\alpha_{n-1}q\beta_{n-1},\tilde{a}_{n-1},\eta_{n-1}q_n\zeta_{n-1}>v_{n-1} \quad \$\hat{w}_n \qquad 2\ 0\ 0\ \ldots0\ \ 3$$

$$(h_2)\quad \underbrace{\overline{u_{n-1}} \quad \overline{\eta_{n-1}}\,\overline{\zeta_{n-1}} \quad \overline{v_{n-1}}}_{\overline{w_n}}\dashv 1\ 01\ 01\ldots01\ 2\ \varepsilon\ \ \varepsilon\ \ldots\varepsilon\ \ 3$$

(if n is odd); where $q_n \in F$; <u>each</u> $u_i, v_i \in V^*$; and $w_n \in V^*$.

Notice that each of $w_0, \ldots, w_{n-1}$ gives the storage tape, head position, and state, while w_n gives only the final storage tape (after acceptance). We have $w_i \underset{M}{\Rightarrow} w_{i+1}$ for $i = 0,\ldots,n-2$; case $i = n-1$ is similar but for the vanishing of the final state q_n . Taking $h_0(\sigma)$ produces $\tilde{a}_0\ \tilde{a}_1 \ldots \tilde{a}_{n-1}$, where each $a_i \in T \cup \{\varepsilon\}$. This string is the input tape accepted by the Turing machine computation described by the element of $e(h_1, h_2)$.

Let the Turing machine operate with space bound $S_1 \in C$. Let $S_2 = 2S_1 + 2$ (then by hypothesis $S_2 \in C$ also). It needs to be shown that the pair (h_1, h_2) has S_2-bounded balance on $e(h_1, h_2)$ with respect to h_0 . We do this by "traversing" an arbitrary element σ from let to right, showing that each prefix π of σ has the property

$$||h_1(\pi)| - |h_2(\pi)|| \leq S_2(|h_0(\sigma)|) .$$

Since $h_0(\sigma)$ is the input string read by the Turing machine with computation represented by the sequence $w_0,\ldots,w_n$, the space complexity of the computation is the longest that the storage tape becomes; i.e.

$$\text{"}S_1\text{"}(h_0(\sigma)) = \max\{|w_0|-1, |w_1|-1, \ldots, |w_{n-2}|-1, |w_{n-1}|-1, |w_n|\} ;$$

therefore

$$S_1(|h_0(\sigma)|) \geq \max\{|w_0|-1, |w_1|-1,\ldots,|w_{n-2}|-1, |w_{n-1}|-1, |w_n|\} .$$

(Since each string $w_0,\ldots,w_{n-1}$ contains a symbol for the state, we subtract 1 to get the length of the storage tape.)

As the machine is not allowed to write blanks on the tape, it is clear that

$$|w_0|-1 \le |w_1|-1 \le |w_2|-1 \le \ldots \le |w_{n-2}|-1 \le |w_{n-1}|-1 \le |w_n| .$$

Hence $$S_1(|h_0(\sigma)|) \ge |w_n| .$$

Since $S_2 = 2S_1 + 2$, we get

$$2|w_n| + 2 \le S_2(|h_0(\sigma)|) .$$

Thus in order to prove, for each prefix π of σ , $||h_1(\pi)| - |h_2(\pi)|| \le S_2(|h_0(\sigma)|)$, it suffices to show, for each π , $||h_1(\pi)| - |h_2(\pi)|| \le 2|w_n| + 2$. In fact we show, for each π , $0 \le |h_2(\pi)| - |h_1(\pi)| \le 2|w_n| + 2$. We use induction on the length of π .

If $\pi = \varepsilon$, then $|h_2(\pi)| - |h_1(\pi)| = 0 - 0 = 0$. (This is the base of the induction.

Now we check the induction by considering all prefixes up to the character 2. Suppose the inequalities hold for $\tilde{\pi}$ of length $k-1$, and consider prefixes π of length k .

1. If $\pi = \tilde{\pi} \vdash$,

   then $$|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| < |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| + 2$$
   $$= |h_2(\tilde{\pi}) \vdash \overline{q_0 \#}| - |h_1(\tilde{\pi}) \vdash| = |h_2(\pi)| - |h_1(\pi)| .$$

2. If $\pi = \tilde{\pi} \overline{\vdash}$,

   then $$|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| < |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| + 2$$
   $$= |h_2(\tilde{\pi}) \vdash q_0\#| - |h_1(\tilde{\pi}) \vdash| = |h_2(\pi)| - |h_1(\pi)| .$$

3. If $\pi = \tilde{\pi}A$ for $A \in V$,

   then $$|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| = |h_2(\tilde{\pi})\overline{A}| - |h_1(\tilde{\pi})A| = |h_2(\pi)| - |h_1(\pi)| .$$

4. If $\pi = \tilde{\pi}\overline{A}$ for $A \in V$,

then $|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| = |h_2(\tilde{\pi})A| - |h_1(\tilde{\pi})\overline{A}| = |h_2(\pi)| - |h_1(\pi)|$.

5. If $\pi = \tilde{\pi}\langle\alpha q\beta,\tilde{a},\eta p_S\zeta\rangle$ for $\langle\alpha q\beta,\tilde{a},\eta p_S\zeta\rangle \in \Omega_S$,

then $|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| \le |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| + (|\eta\zeta| - |\alpha\beta|)$

$= |h_2(\tilde{\pi})\overline{\eta p_S \zeta}| - |h_1(\tilde{\pi})\alpha q\beta| = |h_2(\pi)| - |h_1(\pi)|$.

6. If $\pi = \tilde{\pi}\overline{\langle\alpha q\beta,\tilde{a},\eta p_S\zeta\rangle}$ for $\langle\alpha q\beta,\tilde{a},\eta p_S\zeta\rangle \in \Omega_S$,

then $|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| \le |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| + (|\eta\zeta| - |\alpha\beta|)$

$= |h_2(\tilde{\pi})\eta p_S\zeta| - |h_1(\tilde{\pi})\overline{\alpha q\beta}| = |h_2(\pi)| - |h_1(\pi)|$.

7. If $\pi = \tilde{\pi}\#$,

then $|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| = |h_2(\tilde{\pi})\overline{\#}| - |h_1(\tilde{\pi})\#| = |h_2(\pi)| - |h_1(\pi)|$.

8. If $\pi = \tilde{\pi}\overline{\#}$,

then $|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| = |h_2(\tilde{\pi})\#| - |h_1(\tilde{\pi})\overline{\#}| = |h_2(\pi)| - |h_1(\pi)|$.

9. If $\pi = \tilde{\pi}\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle$ for $\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle \in \Omega_F$,

then $|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| \le |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| + (|\eta\zeta| - |\alpha\beta|)$

$= |h_2(\tilde{\pi})\bar{\eta}\bar{\zeta}| - |h_1(\tilde{\pi})\alpha q\beta| + 1 = |h_2(\pi)| - |h_1(\pi)| + 1$.

10. If $\pi = \tilde{\pi}\overline{\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle}$ for $\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle \in \Omega_F$,

then $|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| \le |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| + (|\eta\zeta| - |\alpha\beta|)$

$= |h_2(\tilde{\pi})\eta\bar{\zeta}| - |h_1(\tilde{\pi})\alpha q\beta| + 1 = |h_2(\pi)| - |h_1(\pi)| + 1$.

11. If $\pi = \tilde{\pi}\$$,

then $|h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| + 1 = |h_2(\tilde{\pi})\dashv 1| - |h_1(\tilde{\pi})\#|$

$= |h_2(\pi)| - |h_1(\pi)|$.

12. If $\pi = \tilde{\pi}\hat{A}$ for $A \in V$,

$$\text{then } |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| < |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| + 1$$
$$= |h_2(\tilde{\pi})01| - |h_1(\tilde{\pi})A| = |h_2(\pi)| - |h_1(\pi)| \quad .$$

13. If $\pi = \tilde{\pi}2$,

$$\text{then } |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| = |h_2(\tilde{\pi})2| - |h_1(\tilde{\pi})\dashv|$$
$$= |h_2(\pi)| - |h_1(\pi)| \quad .$$

Furthermore, if $\pi = \tilde{\pi}2$, then clearly

$$h_2(\pi) = h_1(\pi)1(01)^{|\hat{w}_n|}2 = h_1(\pi)1(01)^{|w_n|}2 \text{ , i.e.}$$

$|h_2(\pi)| - |h_1(\pi)| = |1(01)^{|w_n|}2| = 1 + 2|w_n| + 1 = 2|w_n| + 2$. Therefore, if π is any of the above cases, then $|h_2(\pi)| - |h_1(\pi)| \leq 2|w_n| + 2$. Additionally, if $\pi = \varepsilon$, then $0 \leq |h_2(\pi)| - |h_1(\pi)|$. If $\pi \neq \varepsilon$ but no character $\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle$ or $\overline{\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle}$ (for $\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle \in \Omega_F$) is included in π , then $2 \leq |h_2(\pi)| - |h_1(\pi)|$. Then if one character $\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle$ or $\overline{\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle}$ (for $\langle\alpha q\beta,\tilde{a},\eta p_F\zeta\rangle \in \Omega_F$) is included in π , but not \$, then still $1 \leq |h_2(\pi)| - |h_1(\pi)|$. Finally, if the character \$ is included in π , then again $2 \leq |h_2(\pi)| - |h_1(\pi)|$. Therefore in all of the above cases, $0 \leq |h_2(\pi)| - |h_1(\pi)|$. Thus in all of the above cases, $0 \leq |h_2(\pi)| - |h_1(\pi)| \leq 2|w_n| + 2$.

We now show these bounds apply to the remaining cases.

(1) If $\pi = \tilde{\pi}0$,

then $|h_2(\pi)| - |h_1(\pi)| = |h_2(\tilde{\pi})\varepsilon| - |h_1(\tilde{\pi})10|$

$= |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| - 2 < |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})|$.

(2) If $\pi = \tilde{\pi}3$,

then $|h_2(\pi)| - |h_1(\pi)| = |h_2(\tilde{\pi})3| - |h_1(\tilde{\pi})123|$

$= |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})| - 2 < |h_2(\tilde{\pi})| - |h_1(\tilde{\pi})|$.

But of course if $\pi = \tilde{\pi}3$, then $\pi = \sigma$, and $|h_2(\pi)| - |h_1(\pi)| = 0$ because $h_2(\sigma) = h_1(\sigma)$. Therefore in these remaining two cases, $0 \le |h_2(\pi)| - |h_1(\pi)| \le 2|w_n| + 2$. Hence for every prefix π of σ , $0 \le |h_2(\pi)| - |h_1(\pi)| \le 2|w_n| + 2$. Therefore the pair (h_1, h_2) has S_2-bounded balance on $e(h_1, h_2)$ with respect to h_0 ; and since $s_2 \in C$, the pair (h_1, h_2) has C-bounded balance on $e(h_1, h_2)$ with respect to h_0 .

(<u>If</u>)

Let $L = h_0(e(h_1, h_2))$ for aribtrary homomorphisms h_0, h_1, h_2 satisfying the hypothesis that the pair (h_1, h_2) has S_2-bounded balance on $e(h_1, h_2)$ with respect to h_0 (with $S_2 \in C$) . It is necessary to show that there exists a Turing machine, online with one storage tape with space bound $S_1 \in C$ (for some $S_1 \in C$) which accepts L . In fact, S_1 takes the form S_2+m , where m is the constant equal to the maximum length of the right hand-sides of h_0, h_1 and h_2 . Say $h_1, h_2 : \Sigma \to \Delta^*$; $h_0 : \Sigma \to T^*$, $L \subseteq T^*$ $(L = h_0(e(h_1, h_2)).)$ Let

$m = \max\{|h_i(x)| : i \in \{0,1,2\}, x \in \Sigma\}$. Let $w \in T^*$ be given on the input tape. We construct 3 tracks on the storage tape - one for inverse images of w (under h_0^{-1}) and one each for images under h_1 and h_2 . The machine is described informally below. It is easy to see that the machine works as required.

0. If the input tape has been completely read; and tracks 1, 2, and 3 on the storage tape are all empty; then halt and accept. Otherwise continue to step 1.

1. If the input tape is not empty, but tracks 1, 2, and 3 on the storage tape are all empty, then halt and reject. (If a prefix of $h_0^{-1}(w)$ is in $e(h_1, h_2)$, then the entire string $h_0^{-1}(w)$ cannot be.)

2. Non-deterministically choose an element $x \in \Sigma$ and branch to state $\langle 3, x\rangle$.

$\langle 3, x\rangle$. Write $h_0(x)$ on the track 1 of the storage tape. (This may or may not be the empty string.) Continue to step $\langle 4, x\rangle$.

$\langle 4, x\rangle$. Read characters from the input tape, comparing to $h_0(x)$. (If $h_0(x) = \varepsilon$, then no characters are read and the comparison is successful.) If the entire string matching $h_0(x)$ is successfully read, then continue to state $\langle 5, x\rangle$. Otherwise halt without accepting. (Reall $|h_0(x)| \le m$.)

$\langle 5, x\rangle$. Erase track 1. (If it is preferred not to overwrite $h_0(x)$ with blanks, then pseudo-blanks are equally acceptable.) Continue to state $\langle 6, x\rangle$.

$\langle 6, x\rangle$. Append $h_1(x)$ to the right of the current contents of track 2. Continue to state $\langle 7, x\rangle$.

$\langle 7, x\rangle$. Append $h_2(x)$ to the right of the current contents of track 3. Continue to state 8. (x need no longer be retained.)
[At the beginning of state $\langle 6, x\rangle$, either track 2 or track 3 was empty; and the other track contained a string of length at most $S_2(|w|)$ if $w \in L$, because of the hypothesized bound on the balance of (h_1, h_2) . Now the longer of tracks 2 or 3 has length at most $S_2(|w|)+m$, since $|h_1(x)| \leq m$ and $|h_2(x)| \leq m$. We now shorten tracks 2 and 3 again, by an amount which is sufficient if $w \in L$.]

8. If either track 2 or track 3 (or both) is empty, go to step 0. If the first character on track 2 does not equal the first character on track 3, then halt without accepting. Otherwise containue to step 9.

9. Delete the first characters of tracks 2 and 3, by shifting the entire storage tape (after the first character) one character left, and writing a blank (or, if preferred , pseudo-blank) at the end. Then branch to state 8. [Thus the matching portions of h_1 and h_2 are deleted, leaving the balance string if $w \in L$.]

<u>Note</u>: If $w \notin L$ then clearly the machine does not accept w . However, the machine might in this case "want" to write more than $S_1(w)$ characters on tracks 2 or 3. This is because the pair (h_1, h_2) has S_2-bounded balance only on $e(h_1, h_2)$ with respect to h_0 ; and (h_1, h_2) has

unstated (possibly unbounded) balance on Σ^* (in particular, on $\Sigma^* - e(h_1, h_2)$) with respect to h_0. Therefore when speaking of space limitations, we must mean that the machine may write tapes of any length, but must "guarantee" that, once exceeding the space bound, it will never accept. We cannot mean that the machine guarantees a priori never to write more than S_1 characters.

In some specialized cases (e.g. $S_1(n) = n$, i.e. LBA's) there is no difference between the above meanings because it is trivial to limit the space used. In general, the machine can limit its own space to $S_1(n)$ _if_ S_1 is at least linear (since the input string must be read to know what n is! -- and that string may then be stored in a fourth track on the storage tape, for use in the computation) _and_ $S_1(n)$ is itself computable in space $S_1(n)$ (on, say, a fifth track). This still provides a large set of space bounds, but does not appear to be entirely as useful as choosing the first definition of space-bounded. [In these cases, the machine bounds itself by the obvious manner of initially writing boundary markers (e.g. ¢) in tracks 2 and 3 at S_1 locations from the left, and rejecting if steps $<6, x>$ or $<7, x>$ would attempt to overwrite the markers]

The first part of the proof (the _only if_ part) has been written in a manner that works for either of the above definitions for space bounds.

Note: The theorem applies equally well if we restrict h_0 to be purely erasing. We do this in the (_only if_) part of the proof by requiring the Turing machine, instead of being able to read input characters in several states in K_T, to "decide" in advance when and what it expects to read.

Whenever it is intended to read a character $a \in T$ (any of several characters may be "chosen" by the power of non-determinism), the machine must encode its current state into the current cell on the storage tape and then switch to state a' or a''. From a', the only legal computation shall be to leave the storage tape unchanged and switch to state a''. From a'', the only legal computation shall be to read a from the input tape and switch to state a'''. From a''', computations shall consist of decoding the previous state from the storage tape and (non-deterministically) choose a new state. Defining the set of triples, Ω, as earlier, we notice that $\Omega_T = \{<a'', a, a'''> : a \in T\}$, i.e. $|\Omega_T| = |T|$. Therefore we may instead define $\Omega_T = T$, leaving the remaining constituents of Ω as triples. Then h_1, h_2, and h_0 are defined as before; except for each $a \in \Omega_T$, where $h_1(a) = a''$, $h_2(a) = a'''$, and $h_0(a) = a$. (The reason for including state a' is to permit elements of Ω_T to appear in σ without bars - otherwise we would also need to map some symbols $\bar{a}$ from σ as $h_1(\bar{a}) = \overline{a''}$, $h_2(\bar{a}) = a'''$, and $h_0(\bar{a}) = a$, and h_0 would no longer be purely erasing.) Similarly we need to include additional penultimate states which do nothing but delay for one step before transferring to an accepting state, so that w_n still has the correct parity. Then h_0 is of the form $h_0(a) = a$ if $a \in T$ and $h_0(b) = \varepsilon$ if $b \in \Sigma - T$, and the theorem is still true.

Corollary 1.1 A language L is context sensitive iff there exists erasing h_0 and homomorphisms h_1, h_2 such that $L = h_0(e(h_1, h_2))$ and the pair (h_1, h_2) has linear-bounded balance on $e(h_1, h_2)$ with respect to h_0.

3. Time Complexity Classes

To characterize the time complexity language classes we need first to generalize the notion of k-limited erasing [H + U] as follows

Definition For a function f on the integers we say that erasing h is f-bounded on a language L if for each w in L , $w = xyz$ and $h(y) = \varepsilon$ implies $|y| \leq f(|w|)$, that is at most $f(|w|)$ consecutive symbols of w may be erased. We say that h is C-bounded, for a class C of complexity functions, if h is f-bounded for some f from C .

Theorem 2 Let C be a class of complexity functions closed under squaring. Then for each language L , L is of time complexity C iff there exist an erasing h_0 and homomorphisms h_1, h_2 such that $L = h_0(e(h_1, h_2))$ and h_0 is C-bounded erasing on $e(h_1, h_2)$.

Proof

(if)

If the language L is accepted by an S_1-time-bounded multi-tape Turing machine, then it is accepted by a single tape $(S_1)^2$-time-bounded single-tape Turing machine. Then the language is accepted by an on-line machine with one storage tape in time bound $(S_1(\ell))^2 + 2\ell$ (where ℓ is the length of the input string) by copying the input to the storage tape, returning to the left-hand side, and performing the computation on the one storage tape. By encoding this machine's transitions into homomorphisms h_0, h_1, and h_2 as in Theorem 1, we see that for any string $\sigma \in \Sigma^*$,

$$\sigma \in e(h_1, h_2) \Rightarrow |\sigma| \leq \text{(time used)}(1 + \text{maximum length of tape used}) + 1 + \text{(maximum length)} + 1 + \text{(maximum length)} + 1 \ .$$

Since writing a character in a tape cell requires at least one time unit, the maximum length of tape used is at most equal to the time used. And since the time used is bounded by $(S_1(\ell))^2 + 2\ell$, we have

$$\begin{aligned}\sigma \in e(h_1, h_2) \Rightarrow |\sigma| &\le [(S_1(\ell))^2 + 2\ell][1 + (S_1(\ell))^2 + 2\ell] \\ &\qquad + 2[1 + (S_1(\ell))^2 + 2\ell] + 3 \\ &= (S_1(\ell))^4 + 4\ell(S_1(\ell))^2 + 4\ell^2 + 3(S_1(\ell))^2 + 6\ell + 5 \\ &\le (S_1(\ell))^4 + 4(S_1(\ell))^4 + 4(S_1(\ell))^4 + 3(S_1(\ell))^4 \\ &\qquad + 6(S_1(\ell))^4 + 5(S_1(\ell))^4 \\ &\le 23(S_1(\ell))^4 \quad .\end{aligned}$$

(This of course assumes $\ell \le S_1(\ell)$, which is natural since we may expect the machine to read the entire string; and $1 \le S_1(\ell)$.)
Of course, $S_1(\ell) \in C \Rightarrow (S_1(\ell))^2 \in C \Rightarrow (S_1(\ell))^4 \in C \Rightarrow 23(S_1(\ell))^4 \in C$. Let $S_2 = 23(S_1)^4$. Thus even if h_0 erases the entire string σ (which it cannot actually do, since it must leave exactly ℓ symbols unerased), then h_0 is S_2-bounded erasing. Therefore $L = h_0(e(h_1, h_2))$ and h_0 is C-bounded erasing on $e(h_1, h_2)$.

(Only if)

Suppose $L = h_0(e(h_1, h_2))$, where h_0 is S_2-bounded erasing on $e(h_1, h_2)$, with $S_2 \in C$.

Construct a Turing machine to accept L , in a manner similar to that in Theorem 1. In fact, we need only produce the entire strings $h_1(\sigma)$ and $h_2(\sigma)$ on separate storage tapes, for σ such that the input string equals $h_0(\sigma)$; and we need not worry about erasing the tapes.

After producing $h_1(\sigma)$ and $h_2(\sigma)$ in full, they are compared from left-to-right to check for equality. For some constant C_1, at most C_1 time units are required to read each character of the input string, or to write each character of $h_1(\sigma)$ or $h_2(\sigma)$. Therefore each accepting computation is done in time at most

$$C_1|h_0(\sigma)| + C_1|h_1(\sigma)| + C_1|h_2(\sigma)| + 2(|h_1(\sigma)| + |h_2(\sigma)|)$$

(the last term being for returning to the left and performing the comparison). For some constant C_2, every RHS of h_1 and of h_2 has length at most C_2. Therefore, letting $\ell = |h_0(\sigma)|$ = the length of the input string, the time taken by the Turing machine is bounded by

$$C_1\ell + C_1C_2|\sigma| + C_1C_2|\sigma| + 4C_2|\sigma|$$

$$\leq C_1C_2|\sigma| + C_1C_2|\sigma| + C_1C_2|\sigma| + 4C_1C_2|\sigma|$$

$$= 7C_1C_2|\sigma|$$

$\leq 7C_1C_2\ S_2(\ell)$ since at most $S_2(\ell)$ characters in σ are erased by h_0 between each of the ℓ characters which are not erased

$$\leq 7C_1C_2(S_2(\ell))^2 \quad .$$

Thus, letting $S_1 = 7C_1C_2(S_2)^2$, the Turing machine accepts L with time bound $S_1 \in \mathcal{C}$.

<u>Corollary 2.1</u> Each language L is in NP iff there exist homomorphisms h_0, h_1, h_2 such that $h_0(e(h_1, h_2)) = L$ and h_0 is polynomial-bounded

erasing on $e(h_1, h_2)$.

Corollary 2.2 A language L is primitive recursive (recursive) iff there exist homomorphisms h_0, h_1, h_2 such that $h_0(e(h_1, h_2)) = L$ and h_0 is primitive recursive- (recursive-) bounded erasing on $e(h_1, h_2)$.

References

1. R.V. Book and F.J. Brandenburg, Equality sets, fixed-point languages, and complexity classes, Proceedings of the Six International Colloquium on Automata, Languages and Programming, Graz, Austria, to appear.

2. K. Culik II, The ultimate equivalence problem for DOL systems, Acta Informatica 10, 79-84 (1978).

3. K. Culik II, A purely homomorphic characterization of recursively enumerable sets, JACM to appear, also University of Waterloo Research Report CS-78-08 (January 1978).

4. K. Culik II, Some decidability results about regular and pushdown translations, Information Processing Letters, 8, 5-8 (1979).

5. K. Culik II and I. Fris, The decidability of the equivalence problem for DOL-systems. Information and Control 35 (1977), 20-39.

6. K. Culik II and H.A. Maurer, On simple representations of language families, Revue d'Automatique, Informatique et Recherche Operationelle, to appear.

7. K. Culik II and A. Salomaa, On the decidability of homomorphism equivalence for languages, JCSS 17, 163-175 (1978).

8. J.E. Hopcroft and J.D. Ullman, Formal languages and their relation to automata. Addison-Wesley, Reading, Mass., 1969.

9. A. Salomaa, Equality sets for homomorphisms of free monoids, Acta Cybernetica, to appear.