TAPE- AND TIME-BOUNDED TURING ACCEPTORS AND AFLs: Extended Abstract†

Ronald V. Book
Harvard University

Sheila A. Greibach
University of California at Los Angeles

Ben Wegbreit
Harvard University

Abstract

Complexity classes of formal languages defined by time- and tape-bounded Turing acceptors are studied with the aim of showing sufficient conditions for these classes to be AFLs and to be principal AFLs.

Introduction

Much recent work in automata theory has focused on the computational complexity of functions and of languages. In particular families of languages have been defined by various measures of complexity (among others see [1], [2], [8], [9], [11] - [14], [16], [19]). At the same time researchers in formal language theory have attempted to discover unifying concepts which underlie the study of formal languages ([10], [3]). One approach is to define abstract families of languages or AFLs as collections of languages closed under certain operations common to several families studied extensively in formal language theory. This viewpoint has influenced the study of formal languages to the extent that in studying properties of formal languages, one now asks if these are properties of AFLs or at least of certain types of AFLs. Here we study complexity classes of formal languages as determined by time- and tape-bounded Turing acceptors with the aim of showing sufficient conditions for these classes to be AFLs and to be principal AFLs.

In Section 1 we define the classes of languages to be studied by placing bounds on the amount of time (i.e., number of steps used) or the amount of tape (i.e., number of tape squares visited) in an accepting computation by (deterministic or nondeterministic) multitape Turing acceptors. Certain useful "representation" theorems are established, relating these families to certain homomorphic images of the family Q of quasi-realtime languages, the family CS of context-sensitive languages, and the family $DetLBA$ of languages accepted by deterministic linear bounded automata. From one of these representation theorems and results on Q in [2], it is shown that a time-bounded nondeterministic Turing acceptor need have only two storage tapes, one a pushdown store and the other a stack.

In Section 2 certain lemmas are given which are useful in establishing conditions for the families studied to be AFLs. Some of these lemmas are directed toward clarifying the role of the operator which takes the family $\mathcal{L}_{\mu,f}$ (those languages determined by machines whose μ-complexity is bounded by function f where μ is time or tape) to the family $\mathcal{S}(\mathcal{L}_{\mu,f}) = \bigcup_k \mathcal{L}_{\mu,f_k}$, where for any positive integer k and all real x, $f_k(x) = f(kx)$. It is shown that there are two questions to be asked in order that $\mathcal{L}_{\mu,f}$ be an AFL: (i) what are the conditions such that $\mathcal{S}(\mathcal{L}_{\mu,f})$ is an AFL, and (ii) what are the conditions such that $\mathcal{S}(\mathcal{L}_{\mu,f}) = \mathcal{L}_{\mu,f}$. It is shown that for the families studied here, $\mathcal{S}(\mathcal{L}_{\mu,f})$ is an AFL if f is superadditive $(\forall x \forall y\ (f(x+y) \geq f(x) + f(y))$ and $\mathcal{S}(\mathcal{L}_{\mu,f}) = \mathcal{L}_{\mu,f}$ if f is semihomogeneous $(\forall k_1 > 0\ \exists k_2 > 0\quad \forall x > 0\ (f(k_1 x) \leq k_2 f(x)))$.

Section 3 is concerned with sufficient conditions for the families studied to be principal AFLs. An AFL is principal if it is the smallest AFL containing some particular language, the generator. Each language in a principal AFL can be obtained from the generator by AFL operations (operations under which every AFL is closed). Here we give sufficient conditions for the families defined by time- or tape-bounded Turing acceptors to be principal AFLs. The arguments given are generalizations of those in [17], and the conditions are natural generalizations of the real-time countable functions of [18] or the constructible functions of [16].

Section 1

We begin this section by formally defining multitape Turing acceptors so as to give a precise definition of the families of languages involved. There are many different models of Turing machines available and it is well known that many variations (e.g., one-way vs. two-way) do not affect the computational power even with respect to time and tape bounds. The model given below was chosen to facilitate proofs of some of the results and to be readily comparable to the definition of an

† This research was supported in part by Air Force Cambridge Research Laboratories, Office of Aerospace Research, USAF, under Contract F19628-68-C-0029, by grant from the Milton Fund of Harvard University, and by the Division of Engineering and Applied Physics of Harvard University.

abstract family of automata found in [3].

1.1 Definition. An n-tape Turing machine is a (n+6)-tuple $M = (K, \Sigma, \Gamma_1, \ldots, \Gamma_n, \delta, q_0, F, n)$ where

(1) $K, \Sigma, \Gamma_1, \ldots, \Gamma_n$ are finite sets, $q_0 \in K$, $F \subseteq K$, n is a non-negative integer, and E is a special symbol not in Γ_i for $1 \leq i \leq n$;

(2) δ is a function from $K \times (\Sigma \cup \{e\}) \times [(\Gamma_1 \cup \{e\}) \times \ldots \times (\Gamma_n \cup \{e\})]$ into the finite subsets of $K \times [(\Gamma_1 \times \{1, 0, -1\}) \cup \Gamma_1^* \cup \{E\}] \times \ldots \times [(\Gamma_n \times \{1, 0, -1\} \cup \Gamma_n^* \cup \{E\}]$.†

M is deterministic if for all $q \in K$, $A_i \in \Gamma_i \cup \{e\}$,

(3) $\delta(q, e, A_1, \ldots, A_n) \neq \phi$ implies $\delta(q, a, A_1, \ldots, A_n) = \phi$ for all $a \in \Sigma$, and

(4) for all $a \in \Sigma \cup \{e\}$, $\#\delta(q, a, A_1, \ldots, A_n) \leq 1$.††

1.2 Definition. A configuration of $M = (K, \Sigma, \Gamma_1, \ldots, \Gamma_n, \delta, q_0, F, n)$ is a $(2n + 2)$-tuple $(q, w, y_1, \ldots, y_n, i_1, \ldots, i_n)$ where

(1) $q \in K$, $w \in \Sigma^*$,

(2) for $1 \leq k \leq n$, $y_k \in \Gamma_k^*$ and $0 \leq i_k \leq |y_k|$.†††

1.3 Notation. We define a relation $\vdash$ between configurations as follows:

Let $(q, aw, y_1, \ldots, y_n, i_1, \ldots, i_n)$ be a configuration, $a \in \Sigma \cup \{e\}$.

Let $(q', \sigma_1, \ldots, \sigma_n)$ be in $\delta(q, a, A_1, \ldots, A_n)$.

Then $(q, aw, y_1, \ldots, y_n, i_1, \ldots, i_n) \vdash (q', w, y_1', \ldots, y_n', j_1, \ldots, j_n)$ if for each k, $1 \leq k \leq n$, either

(1) $A_k \in \Gamma_k$, $\sigma_k = (B, i)$, $y_k = xA_ky$, $y_k' = xBy$, $i_k = |x| + 1$, $0 \leq j_k = i_k + i \leq |y_k|$ or

(2) $A_k = e = y_k$, $\sigma_k = z_k \in \Gamma_k^*$, $y_k' = z_k$, and $j_k = |z_k|$ or

(3) $A_k \in \Gamma_k$, $\sigma_k = z_k \in \Gamma_k^*$, $y_k = xA_k$, $i_k = |x| + 1$, $y_k' = xz_k$, $j_k = |xz_k|$ or

(4) $A_k \in \Gamma_k$, $\sigma_k = E$, $y_k = xA_ky$, $i_k = |xA_k|$, $y_k' = e$, $j_k = 0$.

Define the relations $\vdash^m$ $(m \geq 0)$ and $\vdash^*$ as follows. For any configuration C, $C \vdash^0 C$. If $C_0 \vdash C_1 \vdash \ldots \vdash C_m$, then $C_0 \vdash^m C_m$, and if $C_0 = (q_0, w, e, \ldots, e, 0, \ldots, 0)$ then $C_0, C_1, \ldots, C_m$ is called a m-step computation on w; if $C_m = (q, e, e, \ldots, e, 0, \ldots, 0)$ and $q \in F$, then it is an accepting computation. For C_1, C_2, $C_1 \vdash^* C_2$ if $C_1 \vdash^m C_2$ for some $m \geq 0$. If $C_0 \vdash C_1 \vdash \ldots \vdash C_m$ and $k > 0$ is a constant such that for each configuration $C_j = (q, w_j, y_1, \ldots, y_n, i_1, \ldots, i_n)$, every y_t is such that $|y_t| \leq k$, then M visits no more than k tape squares on any one of its storage tapes in the computation $C_0, \ldots, C_m$.

Finally, define

$$L(M) = \{w \in \Sigma^* \mid \exists q \in F, (q_0, w, e, \ldots, e, 0, \ldots, 0) \vdash^* (q, e, e, \ldots, e, 0, \ldots, 0)\}.$$

The functions which we shall use as upper bounds on the amount of tape or the amount of time used in a computation or on the amount of erasing a homomorphism may perform are total functions which are real-valued functions of a single real variable. Each such function f has the following properties:

(i) if $x \geq 0$, then $f(x) \geq 0$;

(ii) for some constant $t_f \geq 0$ and all $x \geq t_f$, $f(x) \geq x$.

We shall assume that any function used in this paper has these properties.

1.4 Definition. An outline multitape Turing acceptor M operates within time bound f if for each input string w accepted by M, every accepting computation of M on w has no more than $\max(|w|, f(|w|))$ steps. Define $\mathrm{TIME}(f) = \{L(M) \mid M$ is a nondeterministic multitape Turing acceptor which operates within time bound $f\}$ and $\mathrm{DetTIME}(f) = \{L(M) \mid M$ is a deterministic multitape Turing acceptor which operates within time bound $f\}$.

† For a set A, A^* is the monoid with identity e freely generated by A. $A^+ = A^*A$.

†† For a finite set S, let $\#S$ be the cardinality of S.

††† For a string w, $|w|$ is the length of w.

For any function f and any constant k > 0, the methods of [8] and [2] can be applied to show that TIME(kf) = TIME(f). The methods of [8] can be applied to show that

DetTIME($\lceil kf \rceil$ + i) = DetTIME(f)

where i is the identity function, i(x) = x, and for any function g and all x,†

(g + i)(x) = g(x) + i(x).

The languages accepted in quasi-realtime by nondeterministic multitape Turing acceptors form the family Q = TIME(i) [2]. The family of real-time definable languages of [14] is the family DetTIME(i).

1.5 Definition. A multitape Turing acceptor M operates within tape bound f if for each input string w accepted by M, every accepting computation of M on w visits no more than max(|w|, f(|w|)) tape squares on any one of its storage tapes. Define TAPE(f) = {L(M)| M is a nondeterministic multitape Turing acceptor which operates within tape bound f}. and DetTAPE(f) = {L(M)| M is a deterministic multitape Turing acceptor which operates within tape bound f}.

For any function f and any constant k > 0, TAPE(kf) = TAPE(f) and DetTAPE(f) = DetTAPE(kf) [16]. As shown in [11], [16], in considering TAPE(f) or DetTAPE(f) we need only consider Turing acceptors with one storage tape. The context-sensitive languages form the family CS = TAPE(i) and the family of languages accepted by deterministic linear bounded automata is the family DetLBA = DetTAPE(i).

The definition of acceptance by a nondeterministic Turing acceptor differs from the usual one in that we require all accepting computations to meet the appropriate condition instead of simply requiring the existence of some computation which meets that condition. If the bounding function is the appropriate generalization of the real-time countable functions of [18] or the constructible functions of [16] to nondeterministic machines, the definitions are equivalent in the sense that the same families are defined. In Section 2 we discuss the invariance of our results with regard to variations in the definitions of acceptance.

We now establish certain "representation" theorems for the families TIME(f), TAPE(f), and DetTAPE(f).

1.6 Definition. If h: $\Sigma^* \to \Delta^*$ is a homomorphism, $L \subseteq \Sigma^*$, and f is a function such that for some k > 0 and all w ε L, $|w| \leq kf(|h(w)|)$, then h is f-bounded on L. For any family $\mathcal{L}$ of languages and any function f, the image of $\mathcal{L}$ under f-bounded erasing is $H_f[\mathcal{L}] = \{h(L) \mid L \in \mathcal{L}$ and h is a homomorphism which is f-bounded on L$\}$.

1.7 Theorem. For any function f, TIME(f) = H_f[Q]. That is, a language L is accepted by a (nondeterministic) multitape Turing machine which operates within time bound f if and only if there is a quasi-realtime language L' and a homomorphism h which is f-bounded on L' such that h(L') = L.

In the proof of the Theorem 1.7, the construction of the new machines involves only changing the way the input is advanced. No new tapes were added. Thus from the Theorem 1.7 and the results in [2] characterizing the family Q, we obtain the following:

1.8 Theorem. For any function f, the following are equivalent:

(i) L ε TIME(f) ;

(ii) L is the f-bounded homomorphic image of a quasi-realtime language ;

(iii) L is the f-bounded homomorphic image of the intersection of three context-free languages ;

(iv) L is accepted by a nondeterministic one stack, one pushdown store automaton operating within time bound f.

The methods used in the proof of Theorem 1.7 generalize to the families TAPE(f) and DetTAPE(f), and thus we obtain Theorem 1.9 below. Results closely related to Theorem 1.9 were established in [7] but the definition of a homomorphism being f-bounded on a language L differs from Definition 1.6 in that the function f is applied to the other side of the inequality.

1.9 Theorem. For any function f, TAPE(f) = H_f[CS] and DetTAPE(f) = H_f[DetLBA].

The "representation" theorems, 1.7 and 1.9, are very useful since they allow one to use results obtained for families of the form $H_f[\mathcal{L}]$ where $\mathcal{L}$ is an arbitrary family of languages or is an AFL. This is the tactic used here, that is, we shall establish certain results concerning families of languages of the form $H_f[\mathcal{L}]$ and then apply these results to the families TIME(f), TAPE(f), and DetTAPE(f) since they can be expressed as H_f[Q], H_f[CS], and H_f[DetLBA], respectively.

We note that the representations TIME(f) = H_f[Q], etc., are deceptive at first glance. If L is any recursively enumerable set, then for some L' ε Q and some homomorphism h, h(L') = L [2]. In this case the amount of erasing that h performs on words of L' is not

† For any real number x, $\lceil x \rceil$ is the least integer greater than or equal to x, and $\lfloor x \rfloor$ is the greatest integer less than or equal to x. For any function f, $\lceil f \rceil$ is the function given by $\lceil f \rceil(x) = \lceil f(x) \rceil$ and $\lfloor f \rfloor$ is the function given by $\lfloor f \rfloor(x) = \lfloor f(x) \rfloor$.

restricted. Here we wish to restrict the amount of erasing that h performs on all words of L' by the same function which bounds the number of steps in an accepting computation of a Turing acceptor M such that $L(M) = L$. To obtain the equation $h(L') = L$ where h is f-bounded on L', it is necessary to construct L' in such a way that substrings that will be erased do not get too long. This is the role played by the requirement in Definition 1.4 that for every $w \in L(M)$, every accepting computation of M on w has no more than $\max(|w|, f(|w|))$ steps. In the construction used in the proof of Theorem 1.7, the number of "dummy" symbols in words in L' cannot become too long. Similar remarks hold for the cases of TAPE(f) and DetTAPE(f).

1.10 Definition. A function f is superadditive if for every $x, y \geq 0$,
$$f(x) + f(y) \leq f(x + y).$$
A function f is semihomogeneous if for every $k_1 > 0$ there is a $k_2 > 0$ such that for all $x \geq 0$, $f(k_1 x) \leq k_2 f(x)$.

It is straightforward to verify the following properties of any superadditive function f:

(i) f is nondecreasing ;

(ii) for every integer $k > 0$ and every $x > 0$, $kf(x) \leq f(kx)$;

(iii) for every $x_1, \ldots, x_n \geq 0$,
$$f(x_1) + \ldots + f(x_n) \leq f(x_1 + \ldots + x_n) .$$

Note that a nondecreasing function f is semihomogeneous if and only if there is a $k_1 > 1$ such that for some $k_2 > 0$ and all $x \geq 0$, $f(k_1 x) \leq k_2 f(x)$.

1.11 Notation. For any function f and any integer $k > 0$, f_k is the function given by $f_k(x) = f(kx)$.

It is immediate that for any function f and any constant $k > 0$, if f is superadditive, so is f_k, and if f is semihomogeneous, so is f_k.

1.12 Definition. For any function f, define
$$\mathscr{A}\,\mathrm{TIME}(f) = \bigcup_k \mathrm{TIME}(f_k) ,$$
$$\mathscr{A}\,\mathrm{DetTIME}(f) = \bigcup_k \mathrm{DetTIME}(f_k)$$
$$\mathscr{A}\,\mathrm{TAPE}(f) = \bigcup_k \mathrm{TAPE}(f_k) , \quad \text{and}$$
$$\mathscr{A}\,\mathrm{DetTAPE}(f) = \bigcup_k \mathrm{DetTAPE}(f_k) .$$
For any family $\mathscr{L}$ of languages and any function f, $\mathscr{A}H_f[\mathscr{L}] = \bigcup_k H_{f_k}[\mathscr{L}]$.

From Theorems 1.7 and 1.9, the following is immediate:

1.13 Corollary. For any function f,
$$\mathscr{A}\,\mathrm{TIME}(f) = \mathscr{A}H_f[Q] ,$$
$$\mathscr{A}\,\mathrm{TAPE}(f) = \mathscr{A}H_f[CS] , \quad \text{and}$$
$$\mathscr{A}\,\mathrm{DetTAPE}(f) = \mathscr{A}H_f[\mathrm{DetLBA}] .$$

We note that representation theorems similar to Theorem 1.7 and Corollary 1.13 may be obtained for time-bounded AFA (as defined in [3]).

1.14 Definition. An Abstract Family of Languages (AFL) is defined in [3] to be a nonempty collection $\mathscr{L}$ of languages such that:

(i) at least one language in $\mathscr{L}$ is nonempty;

(ii) if $L \in \mathscr{L}$, there is a finite vocabulary Σ such that $L \subseteq \Sigma^*$;

(iii) $\mathscr{L}$ is closed under union, concatenation, Kleene +, intersection with regular sets, nonerasing homomorphic mappings, and inverse homomorphic mappings.

The smallest AFL containing a family $\mathscr{L}$ of languages is $\mathscr{F}(\mathscr{L})$. An AFL $\mathscr{L}$ is principal if there is a language L such that $\mathscr{L} = \mathscr{F}(\{L\})$ [4].

In Section 2 we investigate the problem of finding sufficient conditions on f so that TIME(f), TAPE(f), etc., will be AFLs. As suggested by results in [3], [7], and [6], it is suitable to carry out this task by asking two questions: (i) what are sufficient conditions for f so that $\mathscr{A}$TIME(f), $\mathscr{A}$TAPE(f), etc., will be AFLs?, and (ii) what are sufficient conditions for f so that TIME(f) = $\mathscr{A}$TIME(f), TAPE(f) = $\mathscr{A}$TAPE(f), etc.?

In Section 3 we investigate the problem of finding sufficient conditions on f so that TIME(f), TAPE(f), etc., will be principal AFLs. In this case it is appropriate to ask for sufficient conditions on f such that if $\mathscr{A}$TIME(f) (resp., $\mathscr{A}$TAPE(f), etc.) is an AFL, then $\mathscr{A}$TIME(f) (resp., $\mathscr{A}$TAPE(f), etc.) is a principal AFL.

We may summarize the main results of this investigation as follows:

(i) If f is a superadditive function, then $\mathscr{A}$TIME(f), $\mathscr{A}$TAPE(f), and $\mathscr{A}$DetTAPE(f) are AFLs (Theorem 2.9).

(ii) If f is a semihomogeneous function, then $\mathscr{A}$TIME(f) = TIME(f), $\mathscr{A}$TAPE(f) = TAPE(f), and $\mathscr{A}$DetTAPE(f) = DetTAPE(f) (Cor. 2.6).

(iii) If f is a superadditive time **constructible** (resp., tape **constructible**, deterministic-tape **constructible**), then $\mathscr{A}$TIME(f) (resp., $\mathscr{A}$TAPE(f), $\mathscr{A}$DetTAPE(f)) is a principal AFL. (Theorem 3.3).

Section 2

In this Section we shall extablish results connecting the families TIME(f), TAPE(f), DetTAPE(f) with AFLs. We begin with a general result about families of languages of the form $H_f[\mathscr{L}]$.

2.1 Lemma. For any function f and any nonempty family $\mathcal{L}$ of languages, $\mathcal{A}H_f[\mathcal{L}] \subseteq \mathcal{F}(H_f[\mathcal{L}])$.

2.2 Corollary. For any function f and any nonempty family of languages,

$$\mathcal{F}(\mathcal{A}H_f[\mathcal{L}]) = \mathcal{F}(H_f[\mathcal{L}]) ,$$

that is, the smallest AFL containing $\mathcal{A}H_f[\mathcal{L}]$ is the smallest AFL containing $H_f[\mathcal{L}]$.

2.3 Corollary. For any function f and any nonempty family of languages, if $H_f[\mathcal{L}]$ is an AFL, then $\mathcal{A}H_f[\mathcal{L}] = H_f[\mathcal{L}]$.

As shown by Theorems 1.7 and 1.9 and by Corollary 1.13, the results of Lemma 2.1 and Corollaries 2.2 and 2.3 could be restated in terms of TIME(f) and $\mathcal{A}$TIME(f), TAPE(f) and $\mathcal{A}$TAPE(f), and DetTAPE(f) and $\mathcal{A}$DetTAPE(f). Thus in obtaining conditions for TIME(f), TAPE(f), or DetTAPE(f) to be AFLs, the families $\mathcal{A}$TIME(f), $\mathcal{A}$TAPE(f), and $\mathcal{A}$DetTAPE(f), respectively, play a vital role.

2.4 Lemma. For any semihomogeneous function f and any family $\mathcal{L}$ of languages,

$$\mathcal{A}H_f[\mathcal{L}] \subseteq H_f[\mathcal{L}] .$$

2.5 Corollary. For any semihomogeneous function f and any family of languages,

$$\mathcal{A}H_f[\mathcal{L}] = H_f[\mathcal{L}] .$$

We use Theorems 1.7 and 1.8 to express the preceding corollary in terms of $\mathcal{A}$TIME(f) and TIME(f), etc.

2.6 Corollary. For each semihomogeneous function f, $\mathcal{A}$TIME(f) = TIME(f), $\mathcal{A}$TAPE(f) = TAPE(f), and $\mathcal{A}$DetTAPE(f) = DetTAPE(f).

In order to show that TIME(f), etc., forms an AFL when f is sufficiently well-behaved, we rely on a result of [6]. The result as stated here is in a slightly different form from that stated in [6] in that we use the "$\mathcal{A}$" operator here and we define the operator "H_f" in a somewhat different manner. However, it is immediate that the proofs in [6] can be altered to the form of the result stated below. Alternatively, it is not difficult to verify this result directly.

2.7 Theorem. For any superadditive function f and any AFL $\mathcal{L}$, $\mathcal{A}H_f[\mathcal{L}]$ is an AFL.

2.8 Corollary. For any superadditive semihomogeneous function f and any AFL $\mathcal{L}$, $H_f[\mathcal{L}]$ is an AFL.

From Corollary 1.13 and Theorem 2.7 and from the fact that the families Q, CS, and DetLBA form AFLs, the following is immediate.

2.9 Theorem. For any superadditive function f, the families $\mathcal{A}$TIME(f), $\mathcal{A}$TAPE(f) and $\mathcal{A}$DetTAPE(f) are AFLs.

2.10 Corollary. For any superadditive semihomogeneous function f, the families TIME(f), TAPE(f), DetTAPE(f) are AFLs.

It is not difficult to verify Theorem 2.9 and Corollary 2.10 directly, and in doing so the superadditivity of f is not necessary in the cases of TAPE(f) and DetTAPE(f). However, in order to show that TIME(f) is closed under Kleene + , the superadditivity of f appears to be necessary.

In [1] it is shown that results similar to Theorem 2.9 and Corollary 2.10 hold for the families of languages generated by time-bounded grammars.

There are several variations on the definitions of the families TIME(f), TAPE(f), and DetTAPE(f) such that Theorem 2.9 and Corollary 2.10 still hold. In particular these results hold if we define TIME(f) to be any of the following families of languages:

(i) $\{L(M) |$ for each $w \in L(M)$, there exists an accepting computation of M on w which has no more than $\max(|w|, f(|w|))$ steps$\}$;

(ii) $\{L(M) |$ for each input string w, every computation of M on w has no more than $\max(|w|, f(|w|))$ steps$\}$;

If TIME(f) is defined by either (i) or (ii) above, it is straightforward to verify that Theorem 2.9 and Corollary 2.10 hold for the families TIME(f) and $\mathcal{A}$TIME(f). However, these families may not be equal to the family $H_f[Q]$ so that the various results concerning $H_f[Q]$ (or $H_f[\mathcal{L}]$) may not be used. If f is the appropriate generalization of a real-time countable function [18], then defining TIME(f) either by (i) or by (ii) yields the family $H_f[Q]$ so that all the proofs hold in the form given here.

In the definitions (i) or (ii) above, if the bound on the number of steps is replaced by the same bound on the number of tape squares visited, then we obtain the corresponding definitions of TAPE(f) or DetTAPE(f). In this case it is straightforward to verify that Theorem 2.9 and Corollary 2.10 hold for the families TAPE(f) and $\mathcal{A}$TAPE(f), and for DetTAPE(f) and $\mathcal{A}$DetTAPE(f). These families may not be equal to the families $H_f[CS]$ or $H_f[DetLBA]$, but equality can be assured if f is the appropriate generalization of a constructible function [16].

We now consider the case of DetTIME(f). If f is superadditive, then DetTIME(f) is closed under marked union, marked concatenation, marked Kleene + , and intersection with regular sets [5], and $\mathcal{A}$DetTIME(f) is closed under all of these operations as well as inverse homomorphism.

Thus if f is superadditive, then $\mathscr{J}\text{DetTIME}(f)$ is a pre-AFL [5], so that $\mathscr{J}\text{DetTIME}(f)$ is an AFL if and only if

$$H(\mathscr{J}\,\text{DetTIME}(f)) \subseteq \mathscr{J}\,\text{DetTIME}(f),$$

where $H = H_i$ (again, i is the identity function). It is straightforward to show that for any function f, any integer $k \geq 1$, and any constant $t > 1$,

$$H(\text{DetTIME}(f_k)) \subseteq \mathscr{J}\,\text{DetTIME}([t, f])$$

where for all x, $[t, f](x) = t^x f(x)$. Thus for any $t > 1$,

$$H(\mathscr{J}\,\text{DetTIME}(f)) \subseteq \mathscr{J}\,\text{DetTIME}([t, f]).$$

Suppose f has the property that for some $\epsilon > 0$ and all sufficiently large x,

$$(f(2x))/(f(x)) \geq (1 + \epsilon)^x.$$

Then for any $t > 1$ there is an integer $k_t \geq 1$ such that for all sufficiently large x

$$f_{k_t}(x) = f(k_t x) \geq t^x f(x) = [t, f](x),$$

and hence

$$\text{DetTIME}([t,f]) \subseteq \text{DetTIME}(f_{k_t}) \subseteq \mathscr{J}\,\text{DetTIME}(f).$$

Thus if f is a superadditive function such that for some $\epsilon > 0$ and all sufficiently large x,

$$(f(2x))/(f(x)) \geq (1 + \epsilon)^x$$

then $\mathscr{J}\,\text{DetTIME}(f)$ is an AFL.

Consider a superadditive function f such that for some $\epsilon > 0$ and all sufficiently large x,

$$(f(2x))/(f(x)) \geq (f(x))^{\epsilon}.$$

Then for any integer $t \geq 1$, there is an integer k such that for all sufficiently large x,

$$f_k(x) = f(kx) \geq (f(x))^t.$$

Thus if for any integer $t \geq 1$, the function f^t is defined for all x by $f^t(x) = (f(x))^t$, then

$$\text{DetTIME}(f^t) \subseteq \mathscr{J}\,\text{DetTIME}(f),$$

$$\text{TIME}(f^t) \subseteq \mathscr{J}\,\text{TIME}(f),$$

$$\text{DetTAPE}(f^t) \subseteq \mathscr{J}\,\text{DetTAPE}(f), \text{ and}$$

$$\text{TAPE}(f^t) \subseteq \mathscr{J}\,\text{TAPE}(f)$$

and so

$$\mathscr{J}\,\text{DetTIME}(f^t) = \mathscr{J}\,\text{DetTIME}(f),$$

$$\mathscr{F}(\text{TIME}(f^t)) = \mathscr{J}\,\text{TIME}(f^t) = \mathscr{J}\,\text{TIME}(f) = \mathscr{F}(\text{TIME}(f)),$$

$$\mathscr{F}(\text{TAPE}(f^t)) = \mathscr{J}\,\text{TAPE}(f^t) = \mathscr{J}\,\text{TAPE}(f) = \mathscr{F}(\text{TAPE}(f)),$$

$$\mathscr{F}(\text{DetTAPE}(f^t)) = \mathscr{J}\,\text{DetTAPE}(f^t) =$$

$$\mathscr{J}\,\text{DetTAPE}(f) = \mathscr{F}(\text{DetTAPE}(f)).$$

From this we see the following:

(i) The results of [8] on imitating multitape on-line Turing acceptors with single tape off-line Turing acceptors can be applied to show that L is in $\mathscr{J}\,\text{DetTIME}(f)$ if and only if L is accepted by an off-line deterministic single tape Turing acceptor M such that for some $k \geq 1$, M operates within time bound f_k. Clearly this result can be extended to nondeterministic machines so that L is in $\mathscr{F}(\text{TIME}(f))$ if and only if L is accepted by an off-line nondeterministic single tape Turing acceptor M such that for some $k \geq 1$, M operates within time bound f_k.

(ii) Suppose that in addition f is constructible [16] by a deterministic Turing machine (or f is "deterministic-tape constructible" as in Definition 3.1). The results of [15] can be applied to show that $\text{TAPE}(f) \subseteq \text{DetTAPE}(f^2)$. Since

$$\text{DetTAPE}(f^2) \subseteq \text{TAPE}(f^2) \subseteq \mathscr{J}\,\text{TAPE}(f),$$

we see that

$$\mathscr{J}\,\text{TAPE}(f) = \mathscr{F}(\text{TAPE}(f)) =$$

$$\mathscr{F}(\text{DetTAPE}(f)) = \mathscr{J}\,\text{DetTAPE}(f).$$

Now the question of whether the inclusion $\text{DetLBA} \subseteq \text{CS}$ is proper is a long-standing open question. If $\text{DetLBA} = \text{CS}$, then for any f, $\text{DetTAPE}(f) = \text{TAPE}(f)$, since $H_f[\text{DetLBA}] = H_f[\text{CS}]$. If $\text{DetLBA} \neq \text{CS}$, then we have shown that for certain functions at least the AFL defined by $\text{TAPE}(f)$ is exactly the AFL defined by $\text{DetTAPE}(f)$, that is, $\mathscr{J}\,\text{TAPE}(f) = \mathscr{J}\,\text{DetTAPE}(f)$.

Finally, note that the family DetLBA is an AFL which is closed under intersection and which contains the context-free languages. Hence $Q \subseteq \text{DetLBA}$ [2], and so for any f,

$$\text{TIME}(f) = H_f[Q] \subseteq H_f[\text{DetLBA}] = \text{DetTAPE}(f) \text{ and}$$

$$\mathscr{J}\,\text{TIME}(f) \subseteq \mathscr{J}\,\text{DetTAPE}(f).$$

Section 3

In this section we show that the families $\text{TIME}(f)$, $\text{TAPE}(f)$ and $\text{DetTAPE}(f)$, are principal AFLs if the bounding function f has certain properties. As in Section 2 the basic results are first established for the families $\mathscr{J}\,\text{TIME}(f)$, etc., and then extended. The proof of the basic result is a generalization of the arguments given in [17] showing that the families CS and DetLBA are principal.

3.1 Definition. A function f is tape constructable (deterministic-tape constructible) if there is a multitape Turing machine (deterministic multitape Turing machine) M such that for any input w to M any resulting computation of M on w visits precisely $f(|w|)$ tape squares on at least one of its storage tapes and visits no more than $f(|w|)$ tape squares on any one of its storage tapes. The machine M is said to tape-construct f.

3.2 Definition. A function f is said to be time constructible (deterministic-time constructible) if there is a multitape Turing machine (deterministic multitape Turing machine) M such that for any input w to M, any resulting computation of M on w requires precisely f(|w|) steps. The machine M is said to time-construct f.

The generalizations of the real-time functions of [18] and the constructible tape functions of [16] which we shall use here are the time constructible, deterministic-time constructible, tape constructible, and deterministic-tape constructible functions. While it is the case that still weaker conditions can be placed on the functions in order to obtain the following theorem, doing so would obscure the proof.

3.3 Theorem. Let f be a superadditive function.

(i) If f is tape constructible, then $\mathscr{A}$TAPE(f) is a principal AFL.

(ii) If f is deterministic-tape constructible, then $\mathscr{A}$DetTAPE(f) is a principal AFL.

(iii) If f is time constructible, then $\mathscr{A}$TIME(f) is a principal AFL.

3.4 Corollary. Let f be a superadditive function.

(i) If f is tape constructible and TAPE(f) is an AFL, then it is principal.

(ii) If f is deterministic-tape constructible and DetTAPE(f) is an AFL, then it is principal.

(iii) If f is time constructible and TIME(f) is an AFL, then it is principal.

3.5 Corollary. Let f be a superadditive semihomogeneous function.

(i) If f is tape constructible, then TAPE(f) is a principal AFL.

(ii) If f is deterministic-tape constructible, then DetTAPE(f) is a principal AFL.

(iii) If f is time constructible, then TIME(f) is a principal AFL.

As pointed out in [2], the family Q is a principal AFL. In [17] it was shown that both CS and DetLBA are principal AFLs. Thus we have provided answers for three special cases of the following question: If $\mathscr{L}$ is a principal AFL, what are sufficient conditions on f in order that $H_f[\mathscr{L}]$ be a principal AFL?

Suppose f is a superadditive function which is also deterministic-time constructible. Further, suppose that for some $\varepsilon > 0$ and all sufficiently large x,

$$(f(2x))/(f(x)) \geq \max((1+\varepsilon)^x, (f(x))^\varepsilon).$$

As pointed out in Section 2, $\mathscr{A}$DetTIME(f) is then an AFL and $\mathscr{A}$DetTIME(f) = {L | L is accepted by an off-line deterministic single tape Turing acceptor M such that for some $k \geq 1$, M operates within time bound f_k}. In this case the methods used to establish Theorem 3.3 can be applied to yield the fact that $\mathscr{A}$DetTIME(f) is a principal AFL.

References

1. R. Book, "Grammars with time functions", Ph.D. thesis, Harvard University, 1969, Also appears as Mathematical Linguistics and Automatic Translation, Report No. NSF-23, Aiken Computation Laboratory, Harvard University, 1969.

2. R. Book and S. Greibach, "Quasi-realtime languages", in Math. Systems Theory, 4 (1970). An extended abstract appears in the Proceedings of the First ACM Symposium on the Theory of Computing, Marina del Rey, California, May, 1969.

3. S. Ginsburg and S. Greibach, "Abstract families of languages", Studies in Abstract Families of Languages, Memoir No. 87, Amer. Math. Soc., 1-32 (1969).

4. S. Ginsburg and S. Greibach, "Principal AFL", SDC Technical Report, 1969.

5. S. Ginsburg, S. Greibach, and J. Hopcroft, "Pre-AFL", Studies in Abstract Families of Languages, Memoir No. 87, Amer. Math. Soc., 41-51 (1969).

6. S. Ginsburg and J. Hopcroft, "Images of AFL under certain families of homomorphisms", SDC Technical Report, 1969.

7. S. Greibach and J. Hopcroft, "Independence of AFL operations", Studies in Abstract Families of Languages, Memoir No. 87, 33-40 (1969).

8. J. Hartmanis and R. Stearns, "On the computational complexity of algorithms", Trans. Amer. Math. Soc., 117, 285-306 (1965).

9. J. Hartmanis and R. Stearns, "Automata-based computational complexity", Information Sciences, 1, 173-184 (1969).

10. J. Hopcroft and J. Ullman, "An approach to a unified theory of automata", Bell System Technical Journal, 46, 1763-1829 (1967).

11. J. Hopcroft and J. Ullman, Formal Languages and their Relation to Automata, Addison-Wesley, 1969.

12. J. Hopcroft and J. Ullman, "Some results on tape-bounded Turing machines", J. Assoc. Computing Mach., 16, 168-177 (1969).

13. P. M. Lewis, II, R. Stearns, and J. Hartmanis, "Memory bounds for recognition of context-free and context-sensitive languages", 1965 IEEE Conference Record on Switching Circuit Theory and Logical Design.

14. A. L. Rosenberg, "Real-time definable languages" J. Assoc. Computing Mach., 14, 645-662 (1967).

15. W. Savitch, "Relationships between nondeterministic and deterministic tape complexities", submitted for publication. An extended abstract appears in the Proceedings of the First ACM Symposium on the Theory of Computing, Marina del Rey, California, May, 1969.

16. R. Stearns, J. Hartmanis, and P. M. Lewis, II, "Hierarchies of memory limited computations", 1965 IEEE Conference Record on Switching Circuit Theory and Logical Design, 179-190.

17. B. Wegbreit, "A generator of context-sensitive languages", J. Computer and System Sciences, 3, 456-462 (1969).

18. H. Yamada, "Real-time computation and recursive functions not real-time computable", IEEE Transactions on Electronic Computers, 11, 753-760 (1962).

19. P. R. Young, "Toward a theory of enumeration", J. Assoc. Computing Mach., 16, 328-348 (1969).