INFORMATION AND COMPUTATION **128**, 127–138 (1996)
ARTICLE NO. 0068

Self-Monitoring Machines and an ω^ω Hierarchy of Loops

Mark Changizi

Department of Applied Mathematics, University of Maryland, College Park, Maryland 20742
E-mail: changizi@carnap.umd.edu

An ordinal hierarchy of recursive functions is developed based on the level to which a function requires a machine computing it to monitor and make decisions concerning itself. The major theorem states that the functions with self-monitoring level below ω^ω are precisely the class of loop functions (or primitive recursive functions).

INTRODUCTION

This work is motivated by the desire to model the degree to which one must monitor oneself and make decisions to solve a problem. "Problems" will be modelled by recursive functions, "problem solvers" by Turing Machines, and self-monitoring "degrees" by constructible ordinals. For any constructible ordinal α, an α-self-monitoring machine, or α-machine, (as they will be called) behaves as follows:

— Before seeing the input, it places the initial ordinal α into an ordinal clock. This is the self-monitoring degree by which it must compute the function on *all* inputs.

— After each computation step the ordinal clock is decremented by one.

— If the ordinal clock gets to a limit ordinal, then the machine must immediately decide on an ordinal less than the current one.

— When the ordinal reaches zero, the machine halts its computing.

For example, an $\omega \cdot 2 + 3$-machine can make at most two decisions concerning its own running time—the first after three steps when the clock has been decremented to $\omega \cdot 2$, and the second (if there is one) after the clock has been decremented to ω. In this example, after getting to $\omega \cdot 2$, the machine may pick an ordinal below ω, in which case it does not allow itself a second decision concerning its own running time. As a second example, an ω^2-machine must immediately choose an ordinal below ω^2, and so chooses, say, $\omega \cdot 88 + 7$. In this case, the machine's first decision is to allow itself 7 more computations and at most 88 more decisions concerning its own running time.

The major results of the paper include the following:

1. *SUC*-functions, the fastest growing functions self-monitoring machines can compute, are proved to behave sensibly for all ordinals below ω^2 (Theorem 14, number 7), and their eventual behavior is shown not to depend on the choice of notation for ordinals.

2. However, the nice properties below ω^2 do not generally hold for ordinals $\geqslant \omega^2$. The behavior of *SUC*-functions for ordinals $\geqslant \omega^2$ is heavily dependent on the choice of notations; intuitively, there are at these higher levels systems of notation that exhibit "insane" behavior. There are ω^2-machines that are less powerful than $\omega \cdot 3$-machines (untapped genius machines; Theorem 19, numbers 3, 4, 5), and for any recursive function f there are ω^2-machines that can compute g such that $\forall x(g(x) > f(x))$ (autistic machines; Theorem 19, numbers 7, 8; this is reminiscent of a result proved in [Fef62] about a different hierarchy.) ω^2 is unique in that it is the least ordinal for which the machine is allowed to make second order decisions; i.e., decisions concerning how many decisions it will allow itself.

3. In contradistinction to insane machines, a wise choice of the system of notations can lead to *sane* machines which are well-behaved in the sense that any two *SUC*-functions at the same ordinal level will behave similarly. The class of sane functions that are α-self-monitoring (a function is α-self-monitoring iff a machine that computes it is) for $\alpha < \omega^\omega$ is proved to be the class of functions computable by a loop program (which is equivalent to the primitive recursive functions [Eng73].) Intuitively, an ω^k-machine is capable of kth order decisions (and no higher.) Thus, the loop functions (the primitive recursive functions) are the class of recursive functions which can be computed by a self-monitoring machine with finite order decisions.

4. One interesting specific "sane machine" result is that $\lambda x[x^2]$ and $\lambda x[2^x]$ both have a self-monitoring level of ω^2. Intuitively, no deeper thinking is required to compute $\lambda x[2^x]$ than $\lambda x[x^2]$, even though the former takes exponential time to compute, and the latter only polynomial time.

0890-5401/96 $18.00

1. SELF-MONITORING MACHINES

The self-monitoring machines to be described use ordinals to state the level to which they can make decisions concerning their own running time (their self-monitoring level). As machines, they will manipulate notations for ordinals, not ordinals themselves. It is useful for a quick review of ordinal notations [Rog67].

DEFINITION 1 [Kle38]. A *system of notation for ordinals* S is a mapping v_S from a set of natural numbers D_S (the *notations*) onto a segment of the ordinals such that:

1. there exists a partial recursive function k_S such that $v_S(x)=0 \Rightarrow k_S(x)=0$, $v_S(x)$ is a successor ordinal $\Rightarrow k_S(x)=1$, and $v_S(x)$ is a limit ordinal $\Rightarrow k_S(x)=2$;

2. there exists a partial recursive function p_S such that $v_S(x)$ is a successor $\Rightarrow [p_S(x)\downarrow$ and $v_S(x)=v_S(p_S(x))+1]$; and

3. there exists a partial recursive function q_S such that $v_S(x)$ a limit $\Rightarrow [q_S(x)\downarrow$ and $\phi_{q_S(x)}$ is total and $\langle v_S(\phi_{q_S(x)}(n))\rangle_{n\in\omega}$ is an increasing sequence with $v_S(x)$ as limit].

S will denote the set of ordinals that S has notations for. Note that if $\alpha\in S$, then for all $\beta<\alpha$, $\beta\in S$. The following convention will be useful.

Convention 2. Every system of notations S will satisfy for all x, if $q_S(x)\downarrow$, then $k_S(\phi_{q_S(x)}(0))=0$.

The following definitions and theorems may be found in [Rog67].

DEFINITION 3. Fix a system of notation S. (a) S is *univalent* iff v_S is 1–1. (b) S is *recursive* iff D_S is recursive. (c) S is *recursively related* iff $R_S=\{\langle x, y\rangle \mid x\in D_S \wedge y\in D_S \wedge v_S(x)\leqslant v_S(y)\}$ is recursive. (d) α is a *constructive ordinal* iff there is a system of notations that assigns at least one notation to α. (e) ω_1^{CK} denotes the set of constructive ordinals, and $|\omega_1^{CK}|_S$ denotes the set of ordinal notations of S. (f) S is *maximal* iff S gives a notation to every constructive ordinal.

THEOREM 4. 1. *There is no maximal recursively related system of notations.*

2. *For every constructive ordinal α there is a recursively related, univalent system assigning a notation to α.*

3. [Spe55] *There is a maximal univalent system of notation.*

Only univalent systems of notation will be used in this paper; thus, for any $\alpha\in\omega_1^{CK}$, if a (univalent) system of notations S gives a notation for α, then it will make sense to say "the notation for α," and it will be denoted $|\alpha|_S$. For the remainder of the paper every mention of a system of notations will be assumed to be univalent. Also, the subscripts S will be dropped from the notation when it is clear. In addition to the three "ordinal" partial recursive functions defined above it is useful to define functions able to break $|\alpha|=|\beta+n|$ into a limit ordinal notation $|\beta|$ and a natural number n. For example, from $\omega+n$ obtain a notation for ω and the natural number n. L will denote the limit part, and N the natural number part.

DEFINITION 5 [FS93]. Fix a system of notations S. (The subscripts S are dropped.)

$$L(x)=\begin{cases} x & \text{if } k(x)=0 \text{ or } k(x)=2\\ L(p(x)) & \text{otherwise}\end{cases}$$

$$N(x)=\begin{cases} 0 & \text{if } k(x)=0 \text{ or } k(x)=2\\ 1+N(p(x)) & \text{otherwise.}\end{cases}$$

Given any S, $\alpha\in\omega_1^{CK}$ and natural number n, the following procedure determines $|\alpha+n|_S$.

Begin Program. Compute $L(|\alpha|)$ and $N(|\alpha|)$. Set $s=0$.

Begin stage s. Compute s steps of each of the following: $L(0)$, ..., $L(s)$, and $N(0)$, ..., $N(s)$.

If for some $0\leqslant i\leqslant s$, $L(i)=L(|\alpha|)$ and $N(i)=N(|\alpha|)+n$, then output i and halt.

Otherwise go to stage $s+1$.

End stage s.

DEFINITION 6. For any system of notations S, $\alpha\in\omega_1^{CK}$ and natural number n, let $\bigoplus_S(|\alpha|_S, n)$ *be* $|\alpha+n|_S$, as computed by the program above.

DEFINITION 7. $Z=\lambda x[0]$, $SUC=\lambda x[x+1]$ and for all m and $i\leqslant m$ $P_i^m=\lambda x_0, ..., x_m[x_i]$ are the basic functions.

Note that every recursive function may be built from basic functions by primitive recursion, composition, and unbounded minimization. The reason for defining the basic functions is to decide on what a computing machine can do in "one step." Many of the results of this paper will be generalizable to arbitrary sets of primitive recursive functions as the set of basic functions, so long as the set is not weaker than the one in Definition 7 above. Also, every result proved here would still hold so long as SUC is the fastest growing basic function. Occasionally it will be useful to modify the set of basic functions to include other functions.

The following definition introduces a notion of a machine that is just like an ordinary machine, but in addition to being able to compute one basic function at every step, it keeps track of an ordinal clock through its "higher cognitive faculty" discussed below.

DEFINITION 8. Fix a system of notations S. A machine M computing f is an S-machine iff for all x it satisfies the following:

1. *Regular computation.* On every step s, $M(x)$ computes exactly one basic function.

2. *Higher memory*. In addition to the regular memory there are three special storage cells whose contents are denoted by $cnt(x)$, $lim(x)$, and $num(x)$. Regular computations may not "peek" at these special memory cells. A subscript s on any storage cell name (e.g., $lim_s(x)$) denotes the contents of the cell after s steps of $M(x)$. When x is clear, the storage cells are written without the parameters (e.g., *lim*). Also, if any storage cell has a value at step s, then it retains that value for all $t \geqslant s$, unless it is changed.

cnt is the *ordinal clock*; it will at each step denote a stored notation for an ordinal and will be decremented to zero in self-monitoring machines. Also intended for use in self-monitoring machines are *lim* and *num*: *lim* will denote a stored natural number that, when *cnt* denotes a notation for a limit ordinal, dictates what limit ordinal α below the current one the clock is to drop to. *num* will denote a natural number n that dictates how (finitely) far above α to make the new clock, which will become $\alpha + n$.

3. *Higher cognitive functions.* On every step s,

(a) If $k(cnt_{s-1}) = 1$, then set $cnt_s = p(cnt_{s-1})$.

(b) If $k(cnt_{s-1}) = 2$, then cnt_s must be effectively set such that $k(cnt_{s-1}) \in \{0, 1\}$ and $v(cnt_s) < v(cnt_{s-1})$. This may be achieved by arbitrarily many times assigning *lim* and *num* values from (regular) memory and then setting $cnt = |\beta + v(L(\phi_{q(cnt)}(lim))) + num|$, where $|\beta|$ is some hardwired parameter. (Note that if $lim = num = 0$, then $cnt_s = |\beta|$.)

(c) If $cnt_s = |0|$, then output $f(x)$ and halt.

Intuitively, in part 3b of Definition 8, the ordinal β is one of finitely many ordinals the machine decides on *before* beginning to solve the problem.

DEFINITION 9. Fix S. S-machine M is *simple* if it is always the case that $\beta = 0$ in Definition 8 part 3b.

S-machine M will sometimes be written M^S. Given M_e^S, ϕ_e^S is called an S-function. When S is clear or doesn't matter, the superscript will be dropped. Note that all activities specified in part 3 of Definition 8 take no time to compute; i.e., arbitrarily many higher cognitive functions may be computed on any step. However, only one basic function may be computed by the regular machine at each step. Intuitively, S-machines are just like regular machines but they also have a higher cognitive faculty that will be used below by self-monitoring machines to manipulate ordinals, where these higher cognitive functions do not affect the regular machine computations occurring, except that they *can* determine when the machine halts. Other than this ability to halt, the higher cognitive faculty can only *monitor* what the regular machine is doing. What the higher cognitive faculty can think about is limited by the values computed so far. With the regular memory, the S-machine can fill *lim* and *num*, and with *lim* and *num* a new value for the ordinal clock *cnt* will be obtained. These machines will be central to the definition of a self-monitoring machine below.

Before presenting the following definition, it is useful to consider the intuition it is intended to formalize. A machine that has a self-monitoring level of, say, $\omega^2 + \omega \cdot 3 + 72$, will be allowed 72 steps before it must decide on a new ordinal lower than $\omega^2 + \omega \cdot 3$. Supposing the machine chooses $\omega^2 + 4$, then it has 4 more steps before it must decide on how many more decisions it will allow itself and how many more steps before its next decision must be made. Choosing, say, $\omega \cdot 11 + 13$ means that the machine allows itself at most 11 more decisions, the first of which must be made after 13 more steps. After 13 steps, suppose the machine drops to $\omega \cdot 12$, then immediately, since it is not allowed any steps, to $\omega \cdot 2$, and finally to the ordinal 0, where the machine halts having made $72 + 4 + 13 = 89$ steps in total.

DEFINITION 10. Fix a recursive function f, $\alpha \in \omega_1^{CK}$, S such that $\alpha \in S$, and M^S.

1. Fix x. $M(x)\downarrow$ *in* α *steps* iff $cnt_0 = |\alpha|$. (Note that for $n \in \omega$, $M(x)\downarrow$ in n steps means the usual.)

2. M is α-*self-monitoring* iff for all x, $M(x)\downarrow$ in α steps. Sometimes such a machine will be called an S-α-*machine*, or said to *halt in* α *steps.*

3. Function f is α-*self-monitoring* in S ($f \in SM[\alpha](S)$) iff there is S-α-machine e such that $\phi_e = f$. Sometimes such a function will be called an S-α-*function*, or said to *halt in* α *steps in* S.

4. $SM[\alpha] = \{F \mid \exists S(F \subseteq SM[\alpha](S))\}$.

5. $SM(S) = \{F \mid \exists \alpha(F \subseteq SM[\alpha](S))\}$.

6. $SM = \bigcup \{SM[\alpha] \mid \alpha \in \omega_1^{CK}\}$.

If cnt_i, lim_i, num_i are defined, it is useful to let $\bigoplus_i$ denote $\bigoplus (L(\phi_{q(cnt_i)}(lim_i)), num_i)$. If at the ith step the ordinal clock gets to a limit ordinal notation, then $\bigoplus_i$ is often the ordinal dropped to. $\bigoplus$ without a subscript will denote $\bigoplus (L(\phi_{q(cnt)}(lim)), num)$.

Note that every S-α-function is total recursive. The reader may wonder if a machine being an α-machine is any constraint at all on its running time. For example, for any single variable recursive function f there might seem to be an ω-machine M_e such that for all x, $f(x) < \phi_e(x)$ because there may seem to be the ability for an ω-machine to pick higher and higher ordinals below ω without bound. This intuition is false, and it will be proven later that each ordinal α in fact bounds the running time of α-machines.

It is useful to explain why the bound exists. Intuitively, for any fixed single variable ω-machine M and input x, M must *immediately* choose an ordinal below ω—it is not

allowed any computations from the basic set—and so must choose x or some hardwired parameter. More formally, for all x, M must either set $lim_0 = num_0 = x$, or one or both of lim_0 and num_0 to a hardwired parameter. Since there are only finitely many hardwired parameters, $\overset{\infty}{\forall} x[\, cnt_1 = \bigoplus_0 = \bigoplus (L(\phi_{q(cnt_0)}(lim_0)),\ num_0) = \bigoplus (|0|,\ num_0) = num_0 = x\,]$. But then $\overset{\infty}{\forall} x(M(x)\downarrow$ in x steps). The following example exhibits a particular ω-machine and shows the ordinal clock values in brackets at each line.

EXAMPLE 1. Fix S such that $\omega \in S$ and consider the following S-machine M_{e_ω} defined by the following program.

Begin Program.

1. Input(x). Set $cnt_0 = (|\omega|)$ and $z = x$. $\langle \omega \rangle$
2. Set $lim_0 = num_0 = l = z$ and $cnt_0 := \bigoplus_0$. $\langle x \rangle$
3. If $cnt_0 = |0|$, then output(0) and halt. Otherwise continue.
4. For $s := 1$ to l
   do $z := SUC(z)$ and set $cnt_s = p(cnt_{s-1})$. $\langle x-1, ..., 0 \rangle$
5. Output(z) and halt.

End Program.

$\phi_{e_\omega} = \lambda x[\,2x\,]$ and M_{e_ω} is ω-self-monitoring. It will be seen later that ϕ_{e_ω} is the largest function that can be computed in ω steps.

As a second example of how α-machines are bounded, consider the somewhat trickier case of $\omega \cdot 2$-machines. Any single variable $\omega \cdot 2$-machine M must immediately choose an ordinal below $\omega \cdot 2$ (if it were a $\omega \cdot 2 + n$-machine, then it would have n computations from the basic set before it must choose an ordinal below $\omega \cdot 2$.) Forgetting about hardwired parameters for the moment, M must set $lim_0 = num_0 = x$, and then set the second ordinal in the ordinal clock to $cnt_1 = \bigoplus_0$. If $L(\phi_{q(|\omega \cdot 2|)}(x)) = |0|$, then $\bigoplus_0 = |x|$ and M has only x more steps. Otherwise, $\bigoplus_0 = |\omega + x|$, and M has one more decision after x more steps. In this "otherwise" case, M has x steps to compute a higher number, say, $2x$, from the basic set. It can now use this number to drop below ω to $2x$, getting $2x$ more steps, or a total of $4x$ steps. The following example exhibits a particular $\omega \cdot 2$-machine.

EXAMPLE 2. Fix S such that $\omega \cdot 2 \in S$. Consider the S-machine $M_{e_{\omega \cdot 2}}$ computed by the following program.

Begin Program.

1. *Input*(x). Set $cnt_0 = |\omega \cdot 2|$ and $z = x$.
2. Set $lim_0 = num_0 = x$ and $cnt_0 = \bigoplus_0$.
3. If $cnt_0 = |0|$, then output(0) and halt. Otherwise continue.
4. For $s := 1$ to x
   do $z := SUC(z)$ and set $cnt_s = p(cnt_{s-1})$.
5. If $cnt_s = |0|$, then output(z) and halt. Otherwise continue. (Otherwise, $cnt_s = |\omega|$.)
6. Set $lim_s = num_s = l = z$ and $cnt_s = \bigoplus_s$.
7. For $t := s$ to $s + l$
   do $z := SUC(z)$ and set $cnt_t = p(cnt_{t-1})$.
8. Output(z) and halt.

End Program.

Let N be the least natural number such that for all $x \geqslant N$, $L(\phi_{q(|\omega \cdot 2|)}(x)) = |\omega|$. Then

$$\phi_{e_{\omega \cdot 2}}(x) = \begin{cases} 2x & \text{if } x < N \\ 4x & \text{if } x \geqslant N \end{cases}$$

and $M_{e_{\omega \cdot 2}}$ is a 2ω-machine. It will be seen later that $\phi_{e_{\omega \cdot 2}}$ is the largest function that can be computed in $\omega \cdot 2$ steps.

2. INSANE SELF-MONITORING FUNCTIONS

The following defines a special sort of α-function that will be studied first. Let $H(e, x)$ be the least step s such that $\phi_{e,s}(x)\downarrow$.

DEFINITION 11. Fix S. For each $\alpha \in S$, let e_α denote a single-variable simple α-machine such that for all x, $\phi_{e_\alpha}(x) = x + H(e_\alpha, x)$ (i.e., SUC is the only basic function used), and for all s, M_{e_α} sets $num_s = lim_s = x + s$.

ϕ_{e_α} is sometimes called the *SUC-function for* α. Intuitively, SUC-functions jump, at each limit ordinal, as high as it can below the limit ordinal.

DEFINITION 12. Fix S. $\alpha \in S$ is a *limit of limits* iff $k(|\alpha|) = 2$ and there is an increasing sequence of ordinals $\langle \gamma_i \rangle_{i \in \omega}$ such that (a) $sup\{\gamma_i \mid i \in \omega\} = \alpha$ and (b) for all i, $k(|\gamma_i|) = 2$.

The following is a simple consequence of Definition 12 and is stated without proof.

LEMMA 13. *Fix S and $\alpha \in S$. If $k(|\alpha|) = 2$ and α is not a limit of limits, then there is $\gamma < \alpha$ such that* (a) $k(|\gamma|) = 2$ *and* (b) $sup\{\gamma + n \mid n \in \omega\} = \alpha$.

For limit ordinals β that are not a limit of limits, let the *limit predecessor of* β denote γ determined by Lemma 13. Also, for all ordinals α, let the *limit successor of* α be the $sup\{\alpha + n \mid n \in \omega\}$. Also, call a function f *majorizing* if $\overset{\infty}{\forall} x(f(x) < f(x+1))$.

The following theorems state some basic results concerning SUC-functions.

THEOREM 14. *Fix S.*

1. $\forall x(\phi_{e_0}(x) = x)$.
2. *Fix* $\alpha \in S$. $\forall x\, \forall n(\phi_{e_{\alpha+n}}(x) = \phi_{e_\alpha}(x+n))$.
3. *If* $\alpha \in S$ *is a limit ordinal and* $\forall x(L(\phi_{q_S(|\alpha|)}(x)) = |\alpha_x|)$, *then* $\forall x(\phi_{e_\alpha}(x) = \phi_{e_{\alpha_x}+x}(x))$.
4. *If* α *is a limit ordinal but not a limit of limits, and has limit predecessor* $\gamma \in S$, *then* $\overset{\infty}{\forall} x(\phi_{e_\alpha}(x) = \phi_{e_{\gamma+x}}(x))$.
5. *Fix* α *such that for all* n, $\alpha + n \in S$. *If* ϕ_{e_α} *is majorizing, then for all* n, (a) ϕ_{e_α} *is majorized by* $\phi_{e_{\alpha+n}}$, *and* (b) $\phi_{e_{\alpha+n}}$ *is majorizing.*
6. *Fix* $\beta \in S$ *a limit ordinal that is not a limit of limits, and let* γ *be the limit predecessor of* β. *If* ϕ_{e_γ} *is majorizing, then* (a) ϕ_{e_β} *majorizes* ϕ_{e_γ}, *and* (b) ϕ_{e_β} *is majorizing.*
7. *Suppose* S *is such that for all* $\alpha < \omega^2$, $\alpha \in S$. (a) *For all* $\alpha < \omega^2$, ϕ_{e_α} *is majorizing, and* (b) *for all* $\alpha < \beta < \omega^2$, ϕ_{e_α} *is majorized by* ϕ_{e_β}.
8. *Fix* S *such that for all* $\alpha < \omega^2$, $\alpha \in S$. *For all* $a, b \in \omega$, $\overset{\infty}{\forall} x(\phi_{e_{\omega\cdot a+b}}(x) = 2^a(x+b))$.

Proofs. The proofs of 1, 2, and 3 follow directly from Definitions 7 and 8. To prove 4, notice that $\overset{\infty}{\forall} x(\alpha_x = \gamma)$, and the result follows.

Proving 5, by number 2, $\phi_{e_{\alpha+1}}(x) = \phi_{e_\alpha}(x+1)$. Since ϕ_{e_α} is majorizing, ϕ_{e_α} is majorized by $\phi_{e_{\alpha+1}}$.

To show that $\phi_{e_{\alpha+1}}$ is majorizing, note that it has just been shown that $\phi_{e_{\alpha+1}}$ majorizes ϕ_{e_α}. But, $\phi_{e_{\alpha+1}}(x) = \phi_{e_\alpha}(x+1)$. Since $\overset{\infty}{\forall} x(\phi_{e_{\alpha+1}}(x+1) > \phi_{e_\alpha}(x+1))$, then $\overset{\infty}{\forall} x(\phi_{e_{\alpha+1}}(x+1) > \phi_{e_{\alpha+1}}(x))$, or $\phi_{e_{\alpha+1}}$ is majorizing.

This argument may be repeated n times to obtain ϕ_{e_α} is majorized by $\phi_{e_{\alpha+n}}$ and $\phi_{e_{\alpha+n}}$ majorizing.

Proving 6, for β, $\overset{\infty}{\forall} x$ ($\bigoplus_0 = |\gamma + x|$), so $\overset{\infty}{\forall} x(\phi_{e_\beta}(x) = \phi_{e_{\gamma+x}}(x))$. Fix $n \in \omega$ such that $\bigoplus_0 = |\gamma + n|$. Then for all $x \geqslant n$, $\phi_{e_{\gamma+x}}(x) \geqslant \phi_{e_{\gamma+n}}(x)$. By Theorem 5, $\phi_{e_{\gamma+n}}$ majorizes ϕ_{e_γ}, so ϕ_{e_β} majorizes ϕ_{e_γ}.

Also, since $\overset{\infty}{\forall} x(\phi_{e_\beta}(x) = \phi_{e_{\gamma+x}}(x) \geqslant \phi_{e_{\gamma+n}}(x))$ and ϕ_{e_γ} is majorizing, then, by Theorem 5, so is $\phi_{e_{\gamma+n}}$, and therefore ϕ_{e_β} is majorizing.

To prove 7, trivially, ϕ_{e_0} is stage-majorizing. Noting that every limit ordinal $< \omega^2$ is not a limit of limits, the result follows from 5 and 6.

8 is proved by transfinite induction.

Base case. $\phi_{e_0} = \lambda x[x]$. *Successor case.* Suppose for some $a, b \in \omega$, $\overset{\infty}{\forall} x(\phi_{e_{\omega\cdot a+b}}(x) = 2^a(x+b))$. However, $\phi_{e_{\omega\cdot a+b+1}}(x) = \phi_{e_{\omega\cdot a+b}}(x+1)$, and so $\overset{\infty}{\forall} x(\phi_{e_{\omega\cdot a+b+1}}(x) = 2^a(x+b+1))$. *Limit case.* Fix $a \in \omega$ arbitrarily and suppose for all $n \in \omega$, $\overset{\infty}{\forall} x(\phi_{e_{\omega\cdot a+n}}(x) = 2^a(x+n))$. Notice that $\overset{\infty}{\forall} x(\phi_{e_{\omega\cdot(a+1)}}(x) = \phi_{e_{\omega\cdot a+x}}(x) = 2^a(x+x) = 2^{a+1}(x))$. ∎

The last theorem in Theorem 14 is independent of the system of notations S. However, what ϕ_{e_α} is *exactly* for $\alpha < \omega^2$ does depend on the system of notations; namely, the system of notations determines the least number N such that for all $x \geqslant N$, $\phi_{e_{\omega\cdot a+b}}(x) = 2^a(x+b)$. For all S, the functions $f \in \bigcup \{SM[\alpha](S) \mid \alpha < \omega^2\}$ are the linear functions. As already mentioned, $\phi_{e_{\omega^2}}$ and the *SUC*-functions for higher ordinals are not so simple. It will turn out that the precise nature of the hierarchy from ω^2 on up depends greatly on the system of notations.

The following definitions and lemmas prepare for the statement of Theorem 19.

DEFINITION 15. Fix S.

1. ω^2 is *moderate in* S iff $\forall x(L(\phi_{q(|\omega^2|)}(x)) = |\omega \cdot x|)$.
2. ω^2 is *fast in* S iff $\forall x(L(\phi_{q(|\omega^2|)}(x)) \neq L(\phi_{q(|\omega^2|)}(x+1)))$.

Notice that moderate implies fast. Also, if ω^2 is fast in S, then $\forall x(\nu(L(\phi_{q(|\omega^2|)}(x))) \geqslant \omega \cdot x)$. Note further that if ω^2 is moderate in S, then $L(\phi_{q_S(|\omega^2|)})$ provides a recursive enumeration of the notations for limit ordinals $< \omega^2$. The following lemma says that there are S such that ω^2 is moderate, and thus fast, in S.

LEMMA 16. *There is* S *such that* ω^2 *is moderate.*

Proof. Let $(\,,\,)$ be a recursive bijection from $\omega \times \omega$ to $\omega - \{0\}$. Also, if $(x, y) = z$, let $(z)_1 = x$ and $(z)_2 = y$. (x, y) is a code for the ordinal $\omega \cdot x + y$. Define S as follows:

$$\nu_S(x) = \begin{cases} \omega \cdot (x)_1 + (x)_2 & \text{if} \quad x \neq 0 \\ \omega^2 & \text{if} \quad x = 0 \end{cases}$$

$$k_S(x) = \begin{cases} 0 & \text{if} \quad x = (0,0) \\ 1 & \text{if} \quad (x)_2 \neq 0 \\ 2 & \text{if} \quad (x)_2 = 0 \text{ and } (x)_1 \neq 0 \end{cases}$$

$$p_S(x) = \begin{cases} ((x)_1, (x)_2 - 1) & \text{if} \quad (x)_2 > 0 \\ \uparrow & \text{otherwise} \end{cases}$$

$$q_S(x) = \begin{cases} e_1 & \text{if} \quad x = 0, \text{ where } \forall y(\phi_{e_1}(y) = (y, 0)) \\ e & \text{if} \quad (x)_1 \geqslant 1 \text{ and } (x)_2 = 0, \\ & \text{where} \quad \forall y(\phi_e(y) = ((x)_1 - 1,\ y) \\ \uparrow & \text{otherwise.} \end{cases}$$

Now $\forall x(L(\phi_{q_S(|\omega^2|)}(x)) = |\omega \cdot x|)$, so ω^2 is moderate in S. ∎

LEMMA 17. *For all* $g \in R$ *there is* f *such that* $\lambda x[\lceil \log_2 (f(x)/x) \rceil]$ *is increasing and* f *majorizes* g.

DEFINITION 18. Fix S, limit $\alpha \in S$, and $f \in R$. (a) α is *enumeration bounded* (*en-bounded*) *in* S *by* f iff $\forall y\, \forall z(\phi_{q(|\alpha|)}(z) = y \Rightarrow z \leqslant f(y))$. (b) α is *en-bounded in* S iff there is $g \in R$ such that α is en-bounded in S by g.

The following theorem displays some of the "insane" properties of general self-monitoring functions. 1 and 2 say that there are *SUC*-functions that infinitely often obtain less

steps given larger inputs than smaller inputs. 3 and 4 say that having higher self-monitoring level does not guarantee a long running time. 5 and 6 give lower bounds on how "slow" a *SUC*-machine can be. Finally, 7 and 8 show that an ω^2-machine can be arbitrarily smart (autistic). Let R, P denote, respectively, the set of recursive, partial recursive functions.

THEOREM 19. 1. *There is recursively related S and $\alpha \in S$ such that $\phi^S_{e_\alpha}$ is not majorizing.*

2. *Fix maximal S. For all $\alpha \in \omega_1^{CK}$, if α is en-bounded in S, then there is maximal S'' such that $\phi^{S''}_{e_\alpha}$ is not majorizing.*

3. *There is recursively related S and α, $\beta \in S$ such that $\beta < \alpha$, α is a limit of limits, for all $\beta < \gamma < \alpha$, ϕ_{e_γ} is majorizing and majorizes ϕ_{e_β}, and (yet) ϕ_{e_β} majorizes ϕ_{e_α}.*

4. *Fix maximal S. If ω^2 is en-bounded in S, then there is maximal S'' and $\beta < \omega^2$ such that for all $\beta < \gamma < \omega^2$, $\phi^{S''}_{e_\gamma}$ is majorizing and majorizes $\phi^{S''}_{e_\beta}$, and (yet) $\phi^{S''}_{e_\beta}$ majorizes $\phi^{S''}_{e_{\omega^2}}$.*

5. *For all S and limit of limits $\alpha \in S$, $\overset{\infty}{\forall} x(\phi^S_{e_\alpha}(x) \geqslant 4x)$. And there are S and limit of limits $\alpha \in S$ such that $\overset{\infty}{\forall} x(\phi^S_{e_\alpha}(x) = 4x)$.*

6. *Fix S and $\alpha \in S$ such that $k(|\alpha|) = 2$, α is not a limit of limits, $\alpha \geqslant \omega^2 + \omega$, and $\alpha = \beta + \omega \cdot a$ for some limit of limits β and $a \in \omega$. If $\phi_{e_\beta} = f \in R$, then $\overset{\infty}{\forall} x(\phi_{e_\alpha}(x) = f(2^a x) \geqslant 2^{a+2}x)$.*

7. *Fix $f \in R$ such that $\lambda x[\lceil \log_2(f(x)/x) \rceil]$ is increasing and recursively related S such that $\omega^2 \in S$. There is recursively related S^3 with $v_{S^3} = v_S$ such that $\phi^{S^3}_{e_{\omega^2}}$ majorizes f.*

8. *Fix f and S as in Theorem 7. If ω^2 is moderate in S, then there is S^3 such that $\overset{\infty}{\forall} x(\phi^{S^3}_{e_{\omega^2}}(x) = x2^{\lceil \log_2(f(x)/x) \rceil})$ (i.e., $\phi^{S^3}_{e_{\omega^2}}$ acts very similar to f.)*

Proofs. To prove 1, fix recursively related S such that there is a limit of limits $\alpha \in S$. A system of notations S'' will be built from S. v, k, p, N, L remain the same; only q will be altered. Let $r \in P$ (P is the set of partial recursive functions) be defined as follows: $\phi_{r(e)}(x)$ is obtained by the program below.

Begin Program.

1. Input(x) and set $j_0 = 0$, $a_0 = L(\phi_e(0))$.
2. If $x = 0$, then output($|0|$) and halt. Otherwise continue.
3. For $s := 1$ to x do
   (a) Set j_s be the least $y > j_{s-1}$ such that $L(\phi_e(y)) \neq a_{s-1}$.
   (b) Set $a_s = L(\phi_e(j_s))$.
4. Output(a_x).

End Program.

Fix $\alpha \in S$ a limit of limits ordinal. Let

$$q_{S'}(y) = \begin{cases} r(q_S(y)) & \text{if } \ y = |\alpha| \\ q_S(y) & \text{otherwise.} \end{cases}$$

This modifies q_S to $q_{S'}$ in such a way that ordinal drops from α go to an ordinal above a *different* limit ordinal for each x. This may not have been the case for q_S, which may have spent a lot of time dropping to a point above the same limit ordinal. More precisely, in S', for all x, $v(L(\phi_{q_{S'}(|\alpha|)}(x))) < v(L(\phi_{q_{S'}(|\alpha|)}(x+1)))$. Notice that $\phi_{r(q_S(y))}$ is total iff y is a notation for a limit of limits, since, informally, if y is *not* a notation for a limit of limits, then any sequence of ordinals with supremum equal to $v(y)$ will eventually run out of limit ordinals, and so $q_{S'}$ will not be able to find a next limit ordinal. S' is recursively related, but note that a maximal S' could be built as just described from a maximal S for the recursive relatedness of S has not yet been used.

Toward building S'', let $h \in P$ be defined as follows: $\phi_{h(e)}(x)$ is obtained by the following program.

Begin Program.

1. Input(x) and set l_e = least y such that $q_{S'}(L(\phi_{q_{S'}(|\alpha|)}(y))) = e$.
2. If $x = 0$, then output($|0|$) and halt. Otherwise continue.
3. If l_e even, then
   If $x \leqslant 2l_e$, then output($|0|$) and halt.
   If $x > 2l_e$, then output($\phi_e(x)$) and halt.
4. If l_e odd, then
   If $x \leqslant 2l_e$, then output($|\omega|$) and halt.
   If $x > 2l_e$, then (a) let l be the least $y > 2l_e$ such that $v(L(\phi_e(y))) \geqslant \omega$, and (b) output($\phi_e(l + x - 2l_e)$) and halt.

End Program.

For all y, let $z_y =$ the least u such that $v(\phi_{q_{S'}(|\alpha|)}(u)) \geqslant v(y)$. For the same fixed limit of limits ordinal α, let

$$q_{S''}(y) = \begin{cases} h(q_{S'}(y)) & \text{if } \ v(y) < \alpha \text{ and} \\ & v(\phi_{q_{S'}(|\alpha|)}(z_y)) = v(y) \\ q_{S'}(y) & \text{otherwise.} \end{cases}$$

Notice that S'' is also recursively related and that this definition of $q_{S''}$ is possible only because S' is recursively related. Consider the S''-machine ϕ_{e_α}. If an input x to ϕ_{e_α} is even, then the ordinal drops first to some ordinal $\beta + x$ (β a limit ordinal below α), and after stepping down to the limit ordinal β, ϕ_{e_α} will have computed $2x$, and so drops to the

ordinal $\bigoplus(|0|+2x)=2x<\omega$. If an input to ϕ_{e_α} is odd, then the ordinal drops first to some ordinal $\beta+x$ (β a limit ordinal), and after stepping down to the limit ordinal β, drops to the ordinal $\omega+2x$. Thus,

$$\phi_{e_\alpha}(x)=\begin{cases}4x & \text{if } x \text{ even}\\ 8x & \text{if } x \text{ off.}\end{cases}$$

So, for all odd $x>4$, $\phi_{e_\alpha}(x)>\phi_{e_\alpha}(x+1)$. Further, there are infinitely many γ, $\omega\cdot 3<\gamma<\alpha$, such that the S''-machine ϕ_{e_γ} majorizes the S''-machine ϕ_{e_α}. For example, consider $\alpha=\omega^2$, which is a limit of limits. 1 has been proven.

To prove 2, fix maximal S. As in the proof of 1, a system S'', this time maximal, will be constructed from S. Construct S' exactly as in the proof of 1. Toward constructing S'', define h exactly as in the other proof. Now, to define $q_{S''}$ it is not possible as before to use the properties of recursive relatedness. Let f be such that α is en-bounded in S by f. It is easy to see that $g\in R$ can be built such that α is en-bounded in S' by g. Let

$$q_{S''}(y)=\begin{cases}h(q_{S'}(y)) & \text{if } \exists z\leqslant g(y)(\phi_{q_{S'}(|\alpha|)}(z)=y)\\ q_{S'}(y) & \text{otherwise.}\end{cases}$$

The theorem follows, as well as the remaining observations in the proof of 1.

Proving 3, consider $\beta=\omega\cdot 4$ and $\alpha=\omega^2$. Note that $\omega\cdot 4<\omega^2$ and ω^2 is a limit of limits. Also, by Theorem 14, number 7, for all $\omega\cdot 4<\gamma<\omega^2$, ϕ_{e_γ} is majorizing and majorizes $\phi_{e_{\omega\cdot 4}}$. It is sufficient to show that there is a recursively related system of notations S such that $\phi_{e_{\omega\cdot 4}}$ majorizes $\phi_{e_{\omega^2}}$. Reconsider the proof of 1 above, with α from there set to ω^2. It was seen that in S'' developed there, the S''-machine

$$\phi_{e_{\omega^2}}(x)=\begin{cases}4x & \text{if } x \text{ even}\\ 8x & \text{if } x \text{ odd.}\end{cases}$$

However, from Theorem 14, number 9, $\overset{\infty}{\forall}x(\phi_{e_{\omega\cdot 4}}(x)=2^4x=16x)$. Thus, the S''-machine $\phi_{e_{\omega\cdot 4}}$ majorizes the S''-machine ϕ_{e_α}.

4 may be proved just like number 3 but "reconsidering" the proof of 2 instead of 1.

To prove 5, fix limit of limits $\alpha\in\omega_1^{CK}$. The technique used in the proof of number 1 may be modified, informally, to always drop the second time below ω. More formally, modify the second program in the proof of number 1 by deleting step three and modifying step two to apply to *all* (both even and odd) natural numbers l_e. Now, S'' results in $\phi^{S''}_{e_{\omega^2}}=\lambda x[4x]$.

It is necessary to show that for all S it is not the case that $\overset{\infty}{\exists}x(\phi^S_{e_\alpha}(x)<4x)$. Fix S such that $\alpha\in S$ and suppose by way of contradiction that $\overset{\infty}{\exists}x(\phi^S_{e_\alpha}(x)<4x)$. Then there is an increasing sequence $\langle x_i\rangle_{i\in\omega}$ such that for all $i\in\omega$, $\phi^S_{e_\alpha}(x_i)<4x_i$. Fix $i\in\omega$, and suppose $L(\phi_{q_{S(|\alpha|)}}(x_i))=|\beta|$, where $\beta\geqslant\omega$ a limit ordinal. Then $\overset{\infty}{\forall}x(\phi^S_{e_\alpha}(x_i)=\phi^S_{e_{\beta+x_i}}(x_i)=\phi^S_{e_\beta}(2x_i))$. Noticing that $\nu(L(\phi_{q_{S(|\beta|)}}(2x_i)))\geqslant 0$, then $\overset{\infty}{\forall}x(\phi^S_{e_\alpha}(x_i)=\phi^S_{e_\beta}(2x_i)\geqslant\phi^S_{e_{2x_i}}(2x_i)=4x_i)$ contradicting the supposition. So, it must be the case that for all $i\in\omega$, $L(\phi_{q_{S(|\alpha|)}}(x_i))=|0|$. But this is impossible since $\phi_{q_{S(|\alpha|)}}$ must generate a sequence with supremum α.

Proving 6, $\forall x(\phi^S_{e_\alpha}(x)=\phi^S_{e_{\beta+\omega\cdot a}}(x))$ and $\overset{\infty}{\forall}x(\phi^S_{e_{\beta+\omega\cdot a}}(x)=\phi^S_{e_\beta}(2^ax)=f(2^ax))$. By 5, $\overset{\infty}{\forall}x(f(x)\geqslant 4x)$, so $\overset{\infty}{\forall}x(\phi^S_{e_\alpha}(x)\geqslant 4(2^ax)=2^{a+2}x)$. Note that the speed of *SUC*-functions on ordinals that are not a limit of limits depends on the speed of the *SUC*-function of the limit of limits below it.

To prove 7, construct S' from S as in the the proof of 1 with $\alpha=\omega^2$. Now, ω^2 is fast in S', so $\forall x(\nu(L(\phi_{q_{S'(|\omega^2|)}}(x)))\geqslant\omega\cdot x)$. Let $a_x=L(\phi_{q_{S'(|\omega^2|)}}(x))$. To avoid the problem seen in 5 of a slow *SUC*-function at ω^2 it is necessary to make sure that for S'' (the next system to be built) there is no $a\in\omega$ such that $\overset{\infty}{\forall}x(L(\phi_{q_{S''}(a_x)}(2x))=|\omega\cdot a|)$. For if there is such an a, then $\overset{\infty}{\forall}x(\phi^{S''}_{e_{\omega^2}}(x)=2^{a+2}x)$ and so $\phi^{S''}_{e_{\omega^2}}$ would be linear (in number 5, $a=0$ and $\overset{\infty}{\forall}x(\phi^{S''}_{e_{\omega^2}}(x)=4x)$).

Toward building S'', let $h\in P$ be defined as follows: $\phi_{h(e)}$ is obtained by the following program.

Begin Program.

1. Input(x) and set $l_e=$ least y such that $q_{S'}(a_y)=e$.
2. If $x=0$, then output($|0|$) and halt. Otherwise continue.
3. Let l be the least y such that $\nu(L(\phi_e(y)))\geqslant\nu(a_{l_e-1})$ (this can be done since S is recursively related.)
4. Output($\phi_e(l+x)$) and halt.

End Program.

For all y let $z_y=$ the least u such that $\nu(L(\phi_{q_{S'(|\omega^2|)}}(u)))\geqslant\nu(y)$. Now define

$$q_{S''}(y)=\begin{cases}h(q_{S'}(y)) & \text{if } \nu(y)<\omega^2 \text{ and } \nu(L(\phi_{q_{S'(|\omega^2|)}}(z_y)))=\nu(y)\\ q_{S'}(y) & \text{otherwise.}\end{cases}$$

S'' has been built so that, for example, if $\nu(a_3)=\omega\cdot 10$ and $\nu(a_4)=\omega\cdot 13$, then $\nu(L(\phi_{q_{S''}(|\omega\cdot 13|)}(1)))\geqslant\omega\cdot 10$. More generally, $\forall x\ \forall y(\nu(L(\phi_{q_{S''}(a_{x+1})}(1)))\geqslant\nu(a_x)>\nu(L(\phi_{q_{S''}(a_x)}(y))))$. Since ω^2 is fast in S', it is still in S'', which means that for all x, $a_x\neq a_{x+1}$. Therefore, S'' has indeed avoided the "problem" from 1. Also, for all x $\phi^{S''}_{e_{\omega^2}}(x)$ will drop first, and then be allowed $\geqslant x$ more drops; thus, $\overset{\infty}{\forall}x(\phi^{S''}_{e_{\omega^2}}(x)\geqslant 2^x(x+x)=x2^{x+1})$.

Now it is possible to construct a recursively related system S^3 such that $\phi^{S^3}_{e_{\omega^2}}$ majorizes f. Let c_x denote $\lceil \log_2(f(x)/x) \rceil - 1$. Define g by $\phi_{g(y)}(x) = \phi_y(c_x)$ for $x > 0$, $\phi_{g(y)}(0) = |0|$, and let

$$q_{S^3}(z) = \begin{cases} g(q_{S''}(z)) & \text{if } z = |\omega^2| \\ q_{S''}(z) & \text{otherwise.} \end{cases}$$

Since c_x is increasing, then $\langle \phi_{q_{S^3}(|\omega^2|)}(x) \rangle_{x \in \omega}$ is a sequence of notations of ordinals with supremum ω^2. Let $b_x = L(\phi_{q_{S^3}(|\omega^2|)}(x))$. b_x is the notation for the limit below the c_xth notation in the sequence generated by $\phi_{q_{S''}(|\omega^2|)}$. By virtue of the definition of S'', there are $\geqslant c_x$ more drops allowed, so $\overset{\infty}{\forall} x(\phi^{S^3}_{e_{\omega^2}}(x) \geqslant 2^{c_x}(2x) = 2x 2^{\lceil \log_2(f(x)/x) \rceil - 1} \geqslant x 2^{\log_2(f(x)/x)} = x(f(x)/x) = f(x))$. 7 has been proved.

Proving 8, suppose ω^2 is moderate in S. Let $S = S'$. Then build S'' and S^3 as in the proof of 7. Now, the last line of that proof may be tightened to $\overset{\infty}{\forall} x(\phi^{S^3}_{e_{\omega^2}}(x) = 2^{c_x}(x + x) = x 2^{\lceil \log_2(f(x)/x) \rceil})$. ∎

For any fixed S and $\alpha \in S$, it *is* possible to prove that for all x there is $y > x$ such that $\phi_{e_\alpha}(x) < \phi_{e_\alpha}(y)$. The section is ended with a natural function which has a well-behaved self-monitoring level independent of the system of notations. The next theorem says roughly that $f(x_0, ..., x_m) = x_0 + \cdots + x_m$ can be computed in no less than $\omega \cdot m$ steps. For a formula $\psi(x_0, ..., x_m)$, $\overset{\infty}{\forall} x_0, ..., x_m \psi(x_0, ..., x_m)$ means that for all $0 \leqslant i \leqslant m$ and $k_0, ..., k_{i-1}, k_{i+1}, ..., k_m \in \omega$, $\overset{\infty}{\forall} x_i \psi(k_0, ..., k_{i-1}, x_i, k_{i+1}, ..., k_m)$. $\overset{\infty}{\exists} x_0, ..., x_m \psi(x_0, ..., x_m)$ is defined the same way but replacing $\forall$ with $\exists$.

THEOREM 20. *Fix S such that for all $\alpha < \omega^2$, $\alpha \in S$. For all m there is (simple) e such that $\overset{\infty}{\forall} x_0, ..., x_m(\phi_e(x_0, ..., x_m) = x_0 + \cdots + x_m)$, and $\phi_e \in SM[\omega \cdot m](S)$. And there is no such e such that $\phi_e \in \bigcup \{SM[\alpha](S) \mid \alpha < \omega \cdot m\}$.*

Proof. Fix S and m. The following program, call it M_e, is an S-$\omega \cdot m$-machine.

Begin Program.

1. Input$(x_0, ..., x_m)$ and set $cnt = |\omega \cdot m|$.
2. If $cnt = |0|$, then output(x_0) and halt. Otherwise continue.
3. $lim := max\{x_0, ..., x_m\} = x_l$. Swap x_l and x_m (i.e., $x_l := x_m$ and $x_m := lim$).
4. For $i := 0$ to $m - 1$ do
   (a) Set $num = x_i$ and $cnt := \bigoplus$.
   (b) If $cnt = |0|$, then output(0) and halt. Otherwise continue.
   (c) For $j := 1$ to x_i
   do (i) $x_{i+1} := SUC(x_{i+1})$, and (ii) $cnt := p(cnt)$.
5. Output(x_m) and halt.

End Program.

Intuitively, the program takes x_0 successors of x_1 (getting $x_0 + x_1$), then takes $x_0 + x_1$ successors of x_2 (getting $x_0 + x_1 + x_2$), then takes $x_0 + x_1 + x_2$ successors of x_3, etc. $\phi_e \in SM[\omega \cdot m](S)$.

It is necessary to show that $\overset{\infty}{\forall} x_0, ..., x_m(\phi_e(x_0, ..., x_m) = x_0 + \cdots + x_m)$. Pick $0 \leqslant i \leqslant m$ and $k_0, ..., k_{i-1}, k_{i+1}, ..., k_m \in \omega$. For $x_i = k > k_0, ..., k_{i-1}, k_{i+1}, ..., k_m$, line 3 sets $lim = k$ and swaps k and k_m. Write $(k_0, ..., k_{i-1}, k_m, k_{i+1}, ..., k)$ as $(k'_0, ..., k'_m)$. Note that $lim = k = k'_m$ throughout the program since k is the largest input. For sufficiently large k, each drop from a limit ordinal goes to somewhere above its limit predecessor. Thus, $\overset{\infty}{\forall} x(\phi_e(k_0, ..., k_{i-1}, x, k_{i+1}, ..., k_m) = k_0 + \cdots + k_{i-1} + x + k_{i+1} + \cdots + k_m)$.

That this is the "least" ordinal machine that can compute such a function, notice informally that an $\omega \cdot m$-machine can make at most m decisions, but there are $m + 1$ independent variables, each generally requiring a separate decision. More formally, notice that there is no machine M_e that on infinitely many x_0, x_1 computes $x_0 + x_1$ in $\leqslant 0$ steps, and suppose by way of proof by induction there is no machine M_e that on infinitely many $x_0, ..., x_m$ computes $x_0 + \cdots + x_m$ in $\leqslant \omega \cdot (m - 1)$ steps. Now suppose by way of contradiction there is a machine M_e that on infinitely many $x_0, ..., x_m$, x_{m+1} computes $x_0 + \cdots + x_m + x_{m+1}$ in $\leqslant \omega \cdot m$ steps. For this machine M_e, lim_0, num_0 can either always be set to some fixed hardwired parameter or always to the ith input x_i for some $i \leqslant m + 1$. There is no advantage to setting num_0 to a hardwired parameter because for any fixed i almost every x_i will be greater. Suppose without loss of generality that $num_0 = x_{m+1}$. Then on all inputs $cnt_0 := |\omega \cdot k + x_{m+1}|$ for some $k \leqslant m - 1$, and $cnt_{x_{m+1}} = |\omega \cdot k|$. After x_{m+1} steps M_e will have taken at most x_{m+1} successors of some input, say x_m, and will be supposed without loss of generality to have computed $x_m + x_{m+1}$. For infinitely many $x'_0, ..., x'_m$, M_e computes $x'_0 + \cdots + x'_m$, where $x'_0 = x_0, ..., x'_{m-1} = x_{m-1}$, and $x'_m = x_m + x_{m+1}$, and does this in less than $\omega \cdot (m - 1)$ steps, contradicting the induction hypothesis. ∎

3. SANE SELF-MONITORING MACHINES

The last section has presented extreme, or "insane," behaviour of self-monitoring machines. It will be useful to concentrate on a special type of self-monitoring machine, "sane" machines, as they will be called, in order to prove nice results concerning the placement of certain natural functions in the ordinal hierarchy. Otherwise, as Theorem 19, number 7, roughly showed, almost anything can be computed at ω^2 (by some S).

DEFINITION 21. The *exponential polynomials in* 0, 1,..., ω, denoted C_p, is the set of ordinals that can be obtained from finite ordinals and ω by finitely many applications

of ordinal addition, multiplication, and exponentiation. Also,

$$\varepsilon_0 = \omega^{\omega^{\cdot^{\cdot^{\cdot}}}}.$$

THEOREM 22 (Normal-Form Theorem) [Rog67]. *For every $\alpha \in C_p$ there is a unique finite sequence of smaller ordinals $\beta_1 > \beta_2 > \cdots > \beta_k$ and a unique finite sequence of nonzero finite ordinals $n_1, n_2, ..., n_k$ such that $\alpha = \omega^{\beta_1} \cdot n_1 + \omega^{\beta_2} \cdot n_2 + \cdots + \omega^{\beta_k} \cdot n_k$.*

The following definition is similar to one in [Wai70].

DEFINITION 23. Let $P(\alpha, n)$ be defined as provided by the following:

1. If $\alpha = \omega^{\alpha_1} \cdot a_1 + \cdots + \omega^{\alpha_r} \cdot a_r + \omega^{k+1} \cdot (a_{r+1} + 1)$, where $\alpha > \alpha_1 > \cdots > \alpha_r > k+1$, then for all n, $P(\alpha, n) = \omega^{\alpha_1} \cdot a_1 + \cdots + \omega^{\alpha_r} \cdot a_r + \omega^{k+1} \cdot (a_{r+1}) + \omega^k \cdot n$.

2. If $\alpha = \omega^{\alpha_1} \cdot a_1 + \cdots + \omega^{\alpha_r} \cdot a_r + \omega^{\beta+1} \cdot (a_{r+1} + 1)$, where $\alpha > \alpha_1 > \cdots > \alpha_r > \beta + 1$, then for all n, $P(\alpha, n) = \omega^{\alpha_1} \cdot a_1 + \cdots + \omega^{\alpha_r} \cdot a_r + \omega^{\beta+1} \cdot (a_{r+1}) + \omega^\beta \cdot n$.

3. If $\alpha = \omega^{\alpha_1} \cdot a_1 + \cdots + \omega^{\alpha_r} \cdot a_r + \omega^\sigma \cdot (a_{r+1} + 1)$, where $\alpha > \alpha_1 > \cdots > \alpha_r > \sigma$, and σ a limit ordinal, then for all n, $P(\alpha, n) = \omega^{\alpha_1} \cdot a_1 + \cdots + \omega^{\alpha_r} \cdot a_r + \omega^\sigma \cdot (a_{r+1}) + \omega^{P(\sigma, n)}$.

EXAMPLE 3. Fix $a \geqslant 1$. $P(\omega \cdot a, n) = \omega \cdot (a-1) + n$, $P(\omega^a, n) = \omega^{a-1} \cdot n$, $P(\omega^\omega, n) = \omega^n$, *and* $P(\omega^{\omega^{\omega^{\omega \cdot 4} \cdot 3} \cdot 2}, n) = \omega^{\omega^{\omega^{\omega \cdot 4} \cdot 3}} + \omega^{(\omega^{\omega^{\omega \cdot 4} \cdot 2} + \omega^{(\omega^{\omega \cdot 3} + \omega^n)})}$.

DEFINITION 24. 1. A sequence of ordinals $\langle \beta_n \rangle_{n \in \omega}$ with limit $\alpha \in C_p$ is *normal* iff $\beta_n = P(\alpha, n)$ for all $n > 0$, $\beta_0 = 0$.

2. A system of notations S with $\sup\{S\} \in C_p$ is *sane* iff for all limit ordinals $\alpha \in S$, $\langle \nu_S(\phi_{q_S(|\alpha|)}(x)) \rangle_{x \in \omega}$ is normal.

3. A function f is α-*sane* iff there is a sane S such that $f \in SM[\alpha](S)$ and for all $\beta < \alpha$, $f \notin SM[\beta](S)$.

4. For $\alpha \in C_p$, $SANE[\alpha] = \{ f \mid \exists \beta < \alpha \ (f \text{ is } \beta\text{-sane}) \}$.

THEOREM 25 [Wai70]. *For each $\alpha \in C_p$ there exists uniform sane S such that $\alpha \in S$.*

Let $t(x, y) = 2^{2^{\cdot^{\cdot^{\cdot^{y}}}}}$, where there are x 2's. $\approx$ is used below only informally. The example shows that the *SUC*-functions get very complex very fast.

EXAMPLE 4.

- $\phi_{e_{\omega^2}}(x) = \phi_{e_{\omega \cdot x + x}}(x) = 2^x(x+x) = x2^{x+1} \approx t(1, x)$.
- $\phi_{e_{\omega^2 \cdot 2}}(x) = \phi_{e_{\omega^2 + \omega \cdot x + x}}(x) = \phi_{e_{\omega^2}}(x2^{x+1})$
  $= (x2^{x+1})\, 2^{(x2^{x+1})+1} = x2^{x2^{x+1}+x+1} \approx t(2, x)$.
- $\phi_{e_{\omega^2 \cdot 3}}(x) = (x2^{x+1})\, 2^{((x2^{x+1})\, 2^{(x2^{x+1}+1)} + x2^{x+1}+1)}$
  $= x2^{(x2^{(x2^{x+1}+x+2)} + x2^{x+1} + x + 2)} \approx t(3, x)$.
- Similarly, $\phi_{e_{\omega^2 \cdot n}}(x) \approx t(n, x)$.
- $\phi_{e_{\omega^3}}(x) = \phi_{e_{\omega^2 \cdot x + x}}(x) = \phi_{e_{\omega^2 \cdot x}}(2x) \approx t(x, 2x) \approx t(x, x)$
  $\equiv t^{(1)}(x)$.
- $\phi_{e_{\omega^3 \cdot 2}}(x) = \phi_{e_{\omega^3 + \omega^2 \cdot x + x}}(x) \approx \phi_{e_{\omega^3}}(t(x, x))$
  $= t(t(x, x), t(x, x)) \equiv t^{(2)}(x)$.
- $\phi_{e_{\omega^3 \cdot 3}}(x) \approx t(t(t(x, x), t(x, x)), t(t(x, x), t(x, x)))$
  $\equiv t^{(3)}(x)$.
- Generally, $\phi_{e_{\omega^3 \cdot (n+1)}}(x) \approx t^{(n)}(t(x, x))$.

The following theorem states some general properties of sane functions, which shows that they really are sane. The proof is not difficult and is omitted.

THEOREM 26. 1. *Fix $f \in SM[\alpha](S)$. Then $\overset{\infty}{\forall} x(f(x) \leqslant \phi_{e_\alpha}(x))$.*

2. *$\alpha < \beta$ iff $\forall x(\phi_{e_\alpha}(x) < \phi_{e_\beta}(x))$.*

3. *$\forall x(\phi_{e_\alpha}(x) < \phi_{e_\alpha}(x+1))$.*

The following theorem states the self-monitoring level of some natural functions for sane systems of notation, including the interesting result that x^2 and 2^x have the same self-monitoring level. It is convenient in the proof of 4 to have as a basic function R where $R(x, y) = 0$ if $x < y$, $R(x, y) = 1$ otherwise. Let $\log^{(m)}(x)$ denote $\log(\log(\cdots(\log(x))\cdots)$ with m log's.

THEOREM 27. 1. *For all m, $\lambda x_0, ..., x_m[x_0 + \cdots + x_m]$ is $\omega \cdot m$-sane.*

2. *For all m, $\lambda x_0, ..., x_m[x_0 \times \cdots \times x_m]$ is $\omega^2 \cdot m$-sane.*

3. *$\lambda x[x^2]$ and $\lambda x[2^x]$ are ω^2-sane.*

4. *Fix sane S such that $\omega^2 \in S$. $\lambda x[\lceil \log_2 x \rceil] \in SM[\omega^2](S)$.*

5. *Fix m and sane S such that $\omega^2 \cdot m \in S$. $\lambda x[\lceil \log_2^{(m)}(x) \rceil]$, $\lambda x[x^{(2^m)}]$, $\lambda x[t(m, x)] \in SM[\omega^2 \cdot m](S)$.*

Proof. The proof of 1 is similar to the proof Theorem 20.

Proving 2, fix m. The following program, call it M_e, is an $\omega^2 \cdot m$-machine.

Begin Program.

1. Input$(x_0, ..., x_m)$ and set $cnt = |\omega^2 \cdot m|$.
2. If $cnt = |0|$, then output(x_0) and halt. Otherwise continue.
3. For $i := 1$ to m do
   a. $z := x_{i-1}$, $num := 0$, $lim := z$, $cnt := \bigoplus$, $cnt := \bigoplus$ (again), and $num := x_i$.
   b. If $num = |0|$, then output(0) and halt. Otherwise continue.
   c. For $j := 2$ to z do
      i. $cnt := \bigoplus$.
      ii. For $k := 1$ to num do $x_i := SUC(x_i)$ and $cnt := p(cnt)$.
4. Output(x_m) and halt.

End Program.

$\phi_e \in SM[\omega^2 \cdot m](S)$. M_e first computes $x_0 \times x_1$, then $(x_0 \times x_1) \times x_2$, then $(x_0 \times x_1 \times x_2) \times x_3$, etc., each time dropping the ordinal clock to the next lower ordinal of multiple of ω^2.

To show that ϕ_e cannot be computed with a less self-monitoring machine, consider first when $m = 1$. Suppose by way of contradiction that $\lambda x, y[x \times y] \in SM[\omega \cdot a + b](S)$ for some sane S and $a, b \in \omega$. Then $\lambda x[x^2] \in SM[\omega \cdot a + b](S)$. By Theorem 26, number 1, and Theorem 14, number 8, $\lambda x[2^{a}(x+b)]$ majorizes $\lambda x[x^2]$, which is impossible. For $m > 1$ notice informally that there are $m+1$ independent variables, so any machine will need at least m 2nd-order decisions.

Proving 3, note that the proof of 2 above shows that $\lambda x[x^2]$ is ω^2-sane. The following program, M_e, is an ω^2-machine and computes $\lambda x[2^x]$.

Begin Program.

1. Input(x) and set $cnt = |\omega^2|$.
2. If $cnt = |0|$, then output(1) and halt. Otherwise continue.
3. $y := 0$, $lim := x$, $num := 2$, and $cnt := \bigoplus$.
4. For $i := 1$ to 2 do
   $y := SUC(y)$ and $cnt := p(cnt)$.
5. For $i := 1$ to x do
   a. $num := y$, $cnt := \bigoplus$.
   b. For $j := 1$ to y do
      $y := SUC(y)$ and $cnt := p(cnt)$.
6. Output(x) and halt.

End Program.

$\phi_e \in SM[\omega^2](S)$. M_e first takes two successors of zero getting 2, then 2 successors of 2 getting 4, 4 successors of 4 getting 8, 8 successors of 8 getting 16, etc. By Lemma 14, number 8, and Theorem 26, number 1, $\lambda x[2^x]$ is too big too compute below ω^2. So it is ω^2-sane.

Proving 4, the following program, M_e, is an ω^2-machine and computes $\lambda x[\lceil \log_2 x \rceil]$.

Begin Program.

1. Input(x) and set $cnt = |\omega^2|$.
2. If $cnt = |0|$, then output(0) and halt. Otherwise continue.
3. $l := 0$, $y := 2$, $lim := x$, $num := 2$, and $cnt := \bigoplus (L(\phi_{q(|\omega^2|)}(lim)), num + 2)$. Call this formula for $cnt \bigoplus^*$.
4. For $i := 1$ to x do
   a. $num := y$, $cnt := \bigoplus^*$.
   b. If $R(y, x) = 1$, then output(l) and halt. Otherwise continue and $cnt := p(cnt)$.
   c. $l := SUC(l)$, $cnt := p(cnt)$.
   d. For $j := 1$ to y do
      $y := SUC(y)$ and $cnt := p(cnt)$.
5. Output(x) and halt.

End Program.

This program starts doubling, from 1, until it gets larger than x, keeping track how many doublings were needed. The number of doublings will be $\lceil \log_2 x \rceil$.

5 follows directly from 3 and 4. ∎

The following theorem characterizes $SANE[\omega^\omega]$. *PRIM* denotes the set of primitive recursive functions and *LOOP* denotes the set of loop programs (see [Eng73, p. 138]).

THEOREM 28. $SANE[\omega^\omega] = PRIM = LOOP$. (*The latter equality is well known*; *see* [Eng73].)

Proof. ($\Leftarrow$) (a) *Zero function Z*:

Begin Program.

1. Input(x) and set $cnt = |0|$.
2. Output(0) and halt.

End Program.

(b) *Successor function SUC*: ϕ_{e_1}.

(c) *Projection functions* P_i^m:

Begin Program.

1. Input($x_0, ..., x_m$) and set $cnt = |0|$.
2. Output(x_i).

End Program.

(d) *Composition*: Suppose h is n-ary, and that g_i, $i = 1, ..., n$, are $m+1$-ary primitive recursive functions computed, respectively, by the self-monitoring functions $s, t_i \in SANE[\omega^\omega]$. Suppose that in sane S, s is α-self-monitoring and that each t_i is β_i-self-monitoring, $\alpha, \beta_i \in \omega^\omega$. For any recursive function f, let $cnt(f)$ denote cnt in the self-monitoring program for f. Let γ be the largest of all the terms occurring in the normal-form representations of α, $\beta_1, ..., \beta_n$.

Begin Program.

1. Input($x_0, ..., x_m$) and set $cnt = |\gamma \cdot (n+2)|$.
2. For $i := 1$ to n do
   - Set $cnt = |\gamma \cdot (n + 2 - i) + \beta_i|$ (from hard-wired parameter; thus this program is not simple).
   - Compute $t_i(x_0, ..., x_m)$. At each step of the computation of $t_i(x_0, ..., x_m)$, set $cnt = |\gamma \cdot (n + 2 - i) + v_S(cnt(t_i))|$ (i.e., use *lim* and *num* from $t_i(x_0, ..., x_m)$ to drop the ordinal clock).

3. Set $cnt = |\alpha|$.
4. Compute $s(t_1(x_0, ..., x_m), ..., t_n(x_0, ..., x_m))$. At each step of the computation, set $cnt = cnt(s)$ (i.e., use *lim* and *num* from s to decrement cnt).

End Program.

At step 3 all of the t_i's have been computed, and cnt will equal either $\gamma \cdot 2$ or $\gamma \cdot 3$, depending on whether γ is a term in one of the t_i's. Because γ was chosen the largest term occurring in any of the ordinals, cnt at step 3 is high enough to make sure there is enough self-monitoring level enough to compute s. Also, since $\alpha, \beta_i \in \omega^\omega$, $\gamma \cdot (n+2) \in \omega^\omega$. In fact, if $\gamma = \omega^{l_0}$, then the program above is $\omega^{l_0} \cdot (n+2)$-self-monitoring. So, composition corresponds to multiplying the self-monitoring ordinal level by a natural number. The exponent of ω is, then, not increased by composition.

(e) *Recursion*: Suppose g and h are, respectively, $m-1$-ary and $m+1$-ary primitive recursive functions computed by self-monitoring functions s, $t \in SANE[\omega^\omega]$, respectively. Let f be the m-ary function defined by

$$f(x_0, ..., x_{m-1}, 0) = g(x_0, ..., x_{m-1})$$

$$f(x_0, ..., x_{m-1}, SUC(x_n)) = h(f(x_0, ..., x_m), x_0, ..., x_m).$$

Suppose that in sane S, s is α-self-monitoring and that t is β-self-monitoring, $\alpha, \beta \in \omega^\omega$. Let β_1 be the largest term occurring in the normal-form representation of β. Let γ be the largest of all the terms occurring in the normal-form representations of α and $\beta_1 \cdot \omega$. Let α' be $\alpha - \gamma$ if γ occurs in α, and α otherwise. Let β' be $\beta - \gamma$ if $\gamma = \beta_1$, and β otherwise.

Begin Program.

1. Input$(x_0, ..., x_m)$ and set $cnt = |\gamma \cdot 2|$.
2. Set $cnt = |\gamma + \alpha'|$.
3. Compute $s(x_0, ..., x_{m-1})$. At each step of the computation, set $cnt = |\gamma + v_S(cnt(s))|$.
4. Set $y = s(x_0, ..., x_{m-1})$.
5. Set $cnt = |\beta_1 \cdot \omega|$.
6. For $i := 0$ to x_m do
   - Set $cnt = |\beta_1 \cdot (x_m - i) + \beta'|$.
   - Compute $t(y, x_0, ..., x_m)$. At each step of the computation of $t(y, x_0, ..., x_m)$, set $cnt = |\beta_1 \cdot (x_m - i) + v_S(cnt(t))|$.
   - Set $y = t(y, x_0, ..., x_m)$.
7. Output(y).

End Program.

Since $\alpha, \beta \in \omega^\omega$, $\gamma \cdot 2 \in \omega^\omega$. In fact, if $\beta_1 = \omega^{l_0}$, then the program above is at least ω^{l_0+1}-self-monitoring. So, recursion corresponds to multiplying the self-monitoring ordinal level by ω. The exponent of ω is, then, increased by one in recursion.

($\Rightarrow$) It is necessary to show that every $t \in SANE[\omega^\omega]$ is primitive recursive. Notice that any ω^n-self-monitoring machine can have a loop hierarchy level of at most n. The theorem is finished via the equivalence of the loop hierarchy and *PRIM*. ▮

The proof can be seen to show that for all $n \in \omega$, $SANE[\omega^n]$ is the set of functions computable by a program with $\leqslant n$ loops, that composition corresponds to multiplying the ordinal by a natural number, and that recursion corresponds to multiplying the ordinal by ω. Also, $\phi_{e_{\omega^\omega}} \approx$ Ackermann's function in one variable. The following theorem says that the sane hierarchy does not capture all of the recursive functions.

THEOREM 29. *There is recursive* $f \notin SANE[\varepsilon_0]$.

Proof. Let

$$\omega^{(x)} \text{ denote } \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}},$$

where there are x ω's above the bottom one. Consider the following program.

Begin Program.

1. Input(x).
2. Create a system of notations S such that S is sane and $\omega^{(x)} \in S$ (there is one by Theorem 25).
3. Compute $\phi^S_{e_{\omega^{(x)}}}(x)$.

End Program.

The function f computed is not in $SANE[\varepsilon_0]$ because it is harder than any function in it. ▮

CONCLUSION

"Insane" and sane self-monitoring machines have been introduced and examined, motivated by the desire to model the degree to which one must monitor oneself and make decisions in order to solve a problem. The self-monitoring hierarchy captures in a natural fashion an interesting part of the recursive functions, including a proper superset of the primitive recursive functions, or loop functions. The loop functions are shown to have a natural ω^ω ordering of "deeper" and "deeper" functions.

ACKNOWLEDGMENTS

Many thanks go to Professors Bill Gasarch and Carl Smith for their comments. I also thank Dr. Tim Barber for much discussion at the inception of the paper.

Received December 8, 1995; final manuscript received May 15, 1996

REFERENCES

[Eng73] Engeler, E. (1973), "Introduction to the Theory of Computation," Academic Press, New York.

[Fef62] Feferman, S. (1962), Classifications of recursive functions by means of hierarchies, *Trans. Amer. Math. Soc.* **104**, 101–122.

[FS93] Freivalds, R., and Smith, C. (1993), On the role of procrastination in machine learning. *Inform. and Comput.* **107**, 237–271.

[Kle38] Kleene, S. (1938), On notation for ordinal numbers, *J. Symbolic Logic* **3**, 150–155.

[Rog67] Rogers, H. Jr., (1967), "Theory of Recursive Functions and Effective Computability," McGraw–Hill, New York.

[Spe55] Spector, C. (1955), Recursive well-orderings, *J. Symbolic Logic* **20**, 151–163.

[Wai70] Wainer, S. S. (1970), A classification of the ordinal recursive functions, *Arch. Math. Logik* **13**, 136–153.