Elementary Arithmetic

G. E. Ostrin
Inst. für Informatik und angewandte Mathematik, Universität Bern,
Neubrückstrasse 10, CH-3012 Bern, Switzerland.
geoff@iam.unibe.ch

S. S. Wainer
Dept. of Pure Mathematics, University of Leeds,
Leeds LS2 9JT, UK.
s.s.wainer@leeds.ac.uk

October 17, 2001

Abstract. *There is a quite natural way in which the safe/normal variable discipline of Bellantoni-Cook recursion (1992) can be imposed on arithmetical theories like PA: quantify over safes and induct on normals. This weakens the theory severely, so that the provably recursive functions become more realistically computable (slow growing rather than fast growing). Earlier results of Leivant (1995) are re-worked and extended in this new context, giving proof-theoretic characterizations (according to the levels of induction used) of complexity classes between Grzegorczyk's E^2 and E^3.*

This is a contribution to the search for "natural" theories, without explicitly-imposed bounds on quantifiers as in Buss [3], whose provably recursive functions form "more feasible" complexity classes (than for example the primitive recursive functions). We develop a quite different, alternative treatment of Leivant's results in [6], where ramified inductions over N are cleverly used to obtain proof-theoretic characterizations of PTIME and Grzegorczyk's classes E^2 and E^3; and we further extend the characterization up to the first level of exponential complexity in the hierarchy between E^2 and E^3. Our emphasis will be on cut elimination in "traditional" theories based on unary numerals, so complexity will in the end be measured in terms of the numerical input itself rather than its binary length, thus distinguishing E^2 from PTIME in the case of polynomial bounds.

Whereas Leivant's ramified theories codify the proof-principles implicit in his equational schemes of "ramified recurrence", the genesis of our theory described below lies in the "normal-safe" recursion schemes of Bellantoni and Cook [1]. They show how the polynomial-time functions can be defined by an amazingly simple, two-sorted variant of the usual primitive recursion schemes, in which (essentially) one is only allowed to substitute for safe variables and do recursion over normal variables. So what if one imposes the same kind of variable separation on Peano Arithmetic ? Then one obtains a theory with two kinds of number variables – "safe" or "output" variables which may be quantified over, and "normal" or "input" variables which control the lengths of inductions and only occur free ! The analogies between this theory and classical arithmetic are striking, the fundamental difference being that the associated hierarchy of bounding functions is now the "slow growing" rather than the "fast growing" one. Thus the functions provably recursive in the Σ_1 inductive fragment are bounded by the slow growing functions below ω^ω, i.e. polynomials; and those provably recursive in the full theory are bounded by slow growing functions below ϵ_0, i.e. exponential polynomials. The theory is therefore only strong enough to prove totality of the elementary (E^3) functions – hence our title.

A very appealing feature of Leivant's "intrinsic" theories, which we too adopt, is that they are based on Kleene's equation calculus, which allows for a natural notion of provable recursiveness, completely free of any coding implicit in the more traditional definition involving the T-predicate. Thus one is allowed to introduce arbitrary partial recursive functions f by means of their equational definitions as axioms, but the logical and inductive power of the theory severely restricts one's ability to prove termination: $f(x)\downarrow$. In Leivant's theory over N (he allows for more abstract data types) this is expressed by $N(x) \to N(f(x))$. In our theory, specific to N though it could be generalised, definedness is expressed by

$$f(x)\downarrow \equiv \exists a(f(x) \simeq a).$$

This highlights the principal logical restriction which must be applied to the $\exists$-introduction and (dually) $\forall$-elimination rules of our theory EA(I;O) described below.

If arbitrary terms t were allowed as witnesses for $\exists$-introduction, then from the axiom $t \simeq t$ we could immediately deduce $\exists a(t \simeq a)$ and hence $f(x)\downarrow$ for every f ! This is clearly not what we want. In order to avoid it we make the restriction that only "basic" terms: variables or 0 or their successors or predecessors, may be used as witnesses. This is not quite so restrictive as it first appears, since from the equality rule

$$t \simeq a,\ A(t)\ \vdash\ A(a)$$

we can derive immediately

$$t\downarrow,\ A(t)\ \vdash\ \exists a A(a).$$

Thus a term may be used to witness an existential quantifier only when it has been proven to be defined. In particular, if f is introduced by a defining equation $f(x) \simeq t$ then to prove $f(x)\downarrow$ we first must prove (compute) $t\downarrow$. Here we can begin to see that, provided we formulate the theory carefully enough, proofs in its Σ_1 fragment will correspond to computations in the equation calculus, and bounds on proof-size will yield complexity measures.

1 The theory EA(I;O)

There will be two kinds of variables: "input" (or "normal") variables denoted $x, y, z, \dots$, and "output" (or "safe") variables denoted $a, b, c, \dots$, both intended as ranging over natural numbers. Output variables may be bound by quantifiers, but input variables will always be free. The *basic terms* are: variables of either kind, the constant 0, or the result of repeated application of the successor S or predecessor P. General *terms* are built up in the usual way from 0 and variables of either kind, by application of S, P and arbitrary function symbols $f, g, h, \dots$ denoting partial recursive functions given by sets E of Herbrand-Gödel-Kleene-style defining equations.

Atomic formulas will be equations $t_1 \simeq t_2$ between arbitrary terms, and formulas $A, B, \dots$ are built from these by applying propositional connectives and quantifiers $\exists a$, $\forall a$ over output variables a. The negation of a formula $\neg A$ will be defined as $A \to \perp$.

It will be convenient, for later proof-theoretic analysis, to work with logic in a sequent-style formalism, and the system G3 (with structural rules absorbed) as set out on page 65 of Troelstra and Schwichtenberg [8] suits us perfectly, except that we write $\vdash$ instead of their $\Rightarrow$. However we shall work in their system G3m of "**minimal**", rather than "classical", logic. This is computationally more natural, and it is not a restriction for us, since (as

Leivant points out) a classical proof of $f(x)\downarrow$ can be transformed, by the double-negation interpretation, into a proof in minimal logic of

$$(\exists a((f(x) \simeq a \rightarrow \perp) \rightarrow \perp) \rightarrow \perp) \rightarrow \perp$$

and since minimal logic has no special rule for $\perp$ we could replace it throughout by the formula $f(x)\downarrow$ and hence obtain an outright proof of $f(x)\downarrow$, since the premise of the above implication becomes provable.

It is not necessary to list the propositional rules as they are quite standard, and the cut rule (with "cut formula" C) is:

$$\frac{\Gamma \;\vdash\; C \quad \Gamma,\, C \;\vdash\; A}{\Gamma \;\vdash\; A}$$

where, throughout, Γ is an arbitrary finite multiset of formulas. However, as stressed above, the quantifier rules need restricting. Thus the minimal left-$\exists$ and right-$\exists$ rules are:

$$\frac{\Gamma,\, A(b) \;\vdash\; B}{\Gamma,\, \exists a A(a) \;\vdash\; B} \qquad \frac{\Gamma \;\vdash\; A(t)}{\Gamma \;\vdash\; \exists a A(a)}$$

where, in the left-$\exists$ rule the output variable b is not free in Γ, B, and in the right-$\exists$ rule the witnessing term t is basic. The left-$\forall$ and right-$\forall$ rules are:

$$\frac{\Gamma,\, \forall a A(a),\, A(t) \;\vdash\; B}{\Gamma,\, \forall a A(a) \;\vdash\; B} \qquad \frac{\Gamma \;\vdash\; A(b)}{\Gamma \;\vdash\; \forall a A(a)}$$

where, in the left-hand rule the term t is basic, and in the right-hand rule the output variable b is not free in Γ.

The logical axioms are, with A atomic,

$$\Gamma,\, A \;\vdash\; A$$

and the equality axioms are $\Gamma \;\vdash\; t \simeq t$ and, again with $A(.)$ atomic,

$$\Gamma,\, t_1 \simeq t_2,\, A(t_1) \;\vdash\; A(t_2)$$

The logic allows these to be generalised straightforwardly to an arbitrary formula A and the quantifier rules then enable us to derive

$$\Gamma,\, t\downarrow,\, A(t) \;\vdash\; \exists a A(a)$$

$$\Gamma,\ t\downarrow,\ \forall a A(a)\ \vdash\ A(t)$$

for any terms t and formulas A.

Two further principles are needed, describing the data-type N, namely induction and cases (a number is either zero or a successor). We present these as rules rather than their equivalent axioms, since this will afford a closer match between proofs and computations. The induction rule (with "induction formula" $A(.)$) is

$$\frac{\Gamma\ \vdash\ A(0) \quad \Gamma,\ A(a)\ \vdash\ A(Sa)}{\Gamma\ \vdash\ A(x)}$$

where the output variable a is not free in Γ and where, in the conclusion, x is an input variable, or a basic term on an input variable.

The cases rule is
$$\frac{\Gamma\ \vdash\ A(0) \quad \Gamma\ \vdash\ A(Sa)}{\Gamma\ \vdash\ A(t)}$$

where t is any basic term. Note that with this rule it is easy to derive $\forall a(a \simeq 0 \vee a \simeq S(Pa))$ from the definition: $P(0) \simeq 0$ and $P(Sa) \simeq a$.

Definition. Our notion of Σ_1 formula will be restricted to those of the form $\exists\vec{a}A(\vec{a})$ where A is a conjunction of atomic formulas. A typical example is $f(\vec{x})\downarrow$. Note that a conjunction of such Σ_1 formulas is provably equivalent to a single Σ_1 formula, by distributivity of $\exists$ over $\wedge$.

Definition. A k-ary function f is *provably recursive* in EA(I;O) if it can be defined by a system E of equations such that, with input variables $x_1,\ldots,x_k$,

$$\bar{E}\ \vdash\ f(x_1,\ldots,x_k)\downarrow$$

where $\bar{E}$ denotes the set of universal closures (over output variables) of the defining equations in E.

2 Elementary Functions are Provably Recursive

Let E be a system of defining equations containing the usual primitive recursions for addition and multiplication:

$$a+0 \simeq a, \quad a+Sb \simeq S(a+b)$$

$$a.0 \simeq 0, \quad a.Sb \simeq (a.b) + a$$

and further equations of the forms

$$p_0 \simeq S0, \quad p_i \simeq p_{i_0} + p_{i_1}, \quad p_i \simeq p_{i_0}.b$$

defining a sequence $\{p_i : i = 0, 1, 2 \ldots\}$ of polynomials in variables $\vec{b} = b_1, \ldots, b_n$. Henceforth we allow $p(\vec{b})$ to stand for any one of the polynomials so generated (clearly all polynomials can be built up in this way).

Definition. The *progressiveness* of a formula $A(a)$ with distinguished free variable a, is expressed by the formula

$$Prog_a A \;\equiv\; A(0) \wedge \forall a(A(a) \rightarrow A(Sa))$$

thus the induction principle of EA(I;O) is equivalent to

$$Prog_a A \;\vdash\; A(x).$$

The following lemmas derive extensions of this principle, first to any polynomial in $\vec{x}$, then to any finitely iterated exponential. In the next section we shall see that this is the most that EA(I;O) can do.

Lemma 2.1 *Let $p(\vec{b})$ be any polynomial defined by a system of equations E as above. Then for every formula $A(a)$ we have, with input variables substituted for the variables of p,*

$$\bar{E},\; Prog_a A \;\vdash\; A(p(\vec{x}))$$

Proof. Proceed by induction over the build-up of the polynomial p according to the given equations E. We argue in an informal natural deduction style, deriving the succedent of a sequent from its antecedent.

If p is the constant 1 (that is $S0$) then $A(S0)$ follows immediately from $A(0)$ and $A(0) \rightarrow A(S0)$, the latter arising from substitution of the defined, basic term 0 for the universally quantified variable a in $\forall a(A(a) \rightarrow A(Sa))$.

Suppose p is $p_0 + p_1$ where, by the induction hypothesis, the result is assumed for each of p_0 and p_1 separately. First choose $A(a)$ to be the formula $a\downarrow$ and note that in this case $Prog_a A$ is provable. Then the induction hypothesis applied to p_0 gives $p_0(\vec{x})\downarrow$. Now again with an arbitrary formula A, we can easily derive

$$\bar{E},\; Prog_a A,\; A(a) \;\vdash\; Prog_b(a + b\downarrow \wedge A(a + b))$$

because if $a + b$ is assumed to be defined, it can be substituted for the universally quantified a in $\forall a(A(a) \to A(Sa))$ to yield $A(a{+}b) \to A(a{+}Sb))$. Therefore by the induction hypothesis applied to p_1 we obtain

$$\bar{E},\ Prog_a A,\ A(a)\ \vdash\ a + p_1(\vec{x}) \downarrow \wedge\, A(a + p_1(\vec{x}))$$

and hence

$$\bar{E},\ Prog_a A\ \vdash\ \forall a(A(a) \to A(a + p_1(\vec{x}))).$$

Finally, substituting the defined term $p_0(\vec{x})$ for a, and using the induction hypothesis on p_0 to give $A(p_0(\vec{x}))$ we get the desired result

$$\bar{E},\ Prog_a A\ \vdash\ A(p_0(\vec{x}) + p_1(\vec{x})).$$

Suppose p is $p_1.b$ where b is a fresh variable not occurring in p_1. By the induction hypothesis applied to p_1 we have as above, $p_1(\vec{x}) \downarrow$ and

$$\bar{E},\ Prog_a A\ \vdash\ \forall a(A(a) \to A(a + p_1(\vec{x})))$$

for any formula A. Also, from the defining equations E and since $p_1(\vec{x}) \downarrow$, we have $p_1(\vec{x}).0 \simeq 0$ and $p_1(\vec{x}).Sb \simeq (p_1(\vec{x}).b) + p_1(\vec{x})$. Therefore we can prove

$$\bar{E},\ Prog_a A\ \vdash\ Prog_b(p_1(\vec{x}).b \downarrow \wedge\, A(p_1(\vec{x}).b))$$

and an application of the EA(I;O)-induction principle on variable b gives, for any input variable x,

$$\bar{E},\ Prog_a A\ \vdash\ p_1(\vec{x}).x \downarrow \wedge\, A(p_1(\vec{x}).x)$$

and hence $\bar{E},\ Prog_a A\ \vdash\ A(p(\vec{x}))$ as required.

Definition. Extend the system of equations E above by adding the new recursive definitions:

$$f_1(a,0) \simeq Sa, \qquad f_1(a,Sb) \simeq f_1(f_1(a,b),b)$$

and for each $k = 2, 3, \ldots$,

$$f_k(a, b_1, \ldots, b_k) \simeq f_1(a, f_{k-1}(b_1, \ldots, b_k))$$

so that $f_1(a,b) = a + 2^b$ and $f_k(a,\vec{b}) = a + 2^{f_{k-1}(\vec{b})}$. Finally define

$$2_k(p(\vec{x})) \simeq f_k(0, \ldots, 0, p(\vec{x}))$$

for each polynomial p given by E.

Lemma 2.2 *In EA(I;O) we can prove, for each k and any formula $A(a)$,*

$$\bar{E},\ Prog_a A\ \vdash\ A(2_k(p(\vec{x}))).$$

Proof. First note that by a similar argument to one used in the previous lemma (and going back all the way to Gentzen) we can prove, for any formula $A(a)$,

$$\bar{E},\ Prog_a A\ \vdash\ Prog_b \forall a(A(a) \to f_1(a,b)\downarrow \wedge\ A(f_1(a,b)))$$

since the $b := 0$ case follows straight from $Prog_a A$, and the induction step from b to Sb follows by appealing to the hypothesis twice: from $A(a)$ we first obtain $A(f_1(a,b))$ with $f_1(a,b)\downarrow$, and then (by substituting the defined $f_1(a,b)$ for the universally quantified variable a) from $A(f_1(a,b))$ follows $A(f_1(a,Sb))$ with $f_1(a,Sb)\downarrow$, using the defining equations for f_1.

The result is now obtained straightforwardly by induction on k. Assuming $\bar{E}$ and $Prog_a A$ we derive

$$Prog_b \forall a(A(a) \to f_1(a,b)\downarrow \wedge\ A(f_1(a,b)))$$

and then by the previous lemma,

$$\forall a(A(a) \to f_1(a,p(\vec{x}))\downarrow \wedge\ A(f_1(a,p(\vec{x}))))$$

and then by putting $a := 0$ and using $A(0)$ we have $2_1(p(\vec{x}))\downarrow$ and $A(2_1(p(\vec{x})))$, which is the case $k = 1$. For the step from k to $k+1$ do the same, but instead of the previous lemma use the induction to replace $p(\vec{x})$ by $2_k(p(\vec{x}))$.

Theorem 2.3 *Every elementary (E^3) function is provably recursive in the theory EA(I;O), and every sub-elementary (E^2) function is provably recursive in the fragment which allows induction only on Σ_1 formulas.*

Proof. Any elementary function $g(\vec{x})$ is computable by a register machine M (working in unary notation with basic instructions "successor", "predecessor", "transfer" and "jump") within a number of steps bounded by $2_k(p(\vec{x}))$ for some fixed k and polynomial p. Let $r_1(c), r_2(c), \ldots, r_n(c)$ be the values held in its registers at step c of the computation, and let $i(c)$ be the number of the machine instruction to be performed next. Each of these functions depends also on the input parameters $\vec{x}$, but we suppress mention of these for brevity. The state of the computation $\langle i, r_1, r_2, \ldots, r_n\rangle$ at step $c+1$ is obtained from the state at step c by performing the atomic act dictated by the instruction $i(c)$. Thus the values of $i, r_1, \ldots, r_n$ at step $c+1$ can be defined from their values at step c by a simultaneous recursive definition involving only the successor S, predecessor P and definitions by cases C. So now, add these defining equations for $i, r_1, \ldots, r_n$ to the system E above, together with the equations for predecessor and cases:

$$P(0) \simeq 0, \quad P(Sa) \simeq a$$

$$C(0,a,b) \simeq a, \quad C(Sd,a,b) \simeq b$$

and notice that the cases rule built into EA(I;O) ensures that we can prove $\forall d \forall a \forall b \; C(d,a,b) \downarrow$. Since the passage from one step to the next involves only applications of C or basic terms, all of which are provably defined, it is easy to convince oneself that the Σ_1 formula

$$\exists \vec{a} \; (i(c) \simeq a_0 \wedge r_1(c) \simeq a_1 \wedge \ldots \wedge r_n(c) \simeq a_n)$$

is provably progressive in variable c. Call this formula $A(\vec{x}, c)$. Then by the second lemma above we can prove

$$\bar{E} \vdash A(\vec{x}, 2_k(p(\vec{x})))$$

and hence, with the convention that the final output is the value of r_1 when the computation terminates,

$$\bar{E} \vdash r_1(2_k(p(\vec{x}))) \downarrow .$$

Hence the function g given by $g(\vec{x}) \simeq r_1(2_k(p(\vec{x})))$ is provably recursive.

In just the same way, but using only the first lemma above, we see that any sub-elementary function (which, e.g. by Rödding [7], is register machine computable in a number of steps bounded by just a polynomial of its inputs) is provably recursive in the Σ_1-inductive fragment. This is because the proof of $A(\vec{x}, p(\vec{x}))$ by the first lemma only uses inductions on substitution instances of A, and here, A is Σ_1.

3 Provably Recursive Functions are Elementary

Suppose we have a derivation of $\bar{E} \vdash f(\vec{x}) \downarrow$ in EA(I;O), and suppose (arbitrary, but fixed) numerals $\bar{n}_1, \bar{n}_2, \ldots$ are substituted for the input variables

$\vec{x} = x_1, x_2, \dots$ throughout. In the resulting derivation, each application of induction takes the form:

$$\frac{\Gamma \vdash A(0) \quad \Gamma,\, A(a) \vdash A(Sa)}{\Gamma \vdash A(t(\bar{n}_i))}$$

where $t(x_i)$ is the basic term appearing in the conclusion of the original (unsubstituted) EA(I;O)-induction. Let m denote the value of $t(\bar{n}_i)$, so m is not greater than n_i plus the length of term t. Furthermore, let ℓ denote the length of the binary representation of m. Then, given the premises, we can unravel the induction so as to obtain a derivation of

$$\Gamma \vdash A(\bar{m})$$

by a sequence of cuts on the formula A, with proof-height $\ell + 1$. To see this we first induct on ℓ to derive

$$\Gamma,\, A(a) \vdash A(S^m a) \quad \text{and} \quad \Gamma,\, A(a) \vdash A(S^{m+1} a)$$

by sequences of A-cuts with proof-height ℓ. This is immediate when $\ell = 1$, and if $\ell > 1$ then either $m = 2m_0$ or $m = 2m_0 + 1$ where m_0 has binary length less than ℓ. So from the result for m_0 we get

$$\Gamma,\, A(a) \vdash A(S^{m_0} a) \quad \text{and} \quad \Gamma,\, A(S^{m_0} a) \vdash A(S^m a)$$

by substitution of $S^{m_0}a$ for the free variable a, and both of these derivations have proof-height $\ell - 1$. Therefore one more cut yields

$$\Gamma,\, A(a) \vdash A(S^m a)$$

as required. The case $A(S^{m+1}a)$ is done in just the same way.

Therefore if we now substitute 0 for variable a, and appeal to the base case of the induction, a final cut on $A(0)$ yields $\Gamma \vdash A(\bar{m})$ with height $\ell + 1$ as required.

Definition. For each number n let EA(n;O) be the theory obtained from EA(I;O) by discarding the induction rule and replacing all input variables by numerals for numbers no greater than n.

Lemma 3.1 *If $\bar{E} \vdash f(\vec{x}) \downarrow$ in EA(I;O) then there is a fixed number k determined by this derivation, such that: for all $n_1, n_2, \dots \leq n$ of binary length $\leq \ell$, there is a derivation of $\bar{E} \vdash f(\bar{n}_1, \bar{n}_2, \dots) \downarrow$ in EA(n;O), with proof-height $\leq \ell.k$. Furthermore the non-atomic cut-formulas in this EA(n;O) derivation are the induction-formulas occurring in the original EA(I;O) derivation.*

Proof. First, by standard "free cut"-elimination arguments (as below) eliminate from the given EA(I;O) derivation all non-atomic cut-formulas which are not induction formulas. Then pass through the resulting free-cut-free proof, substituting the numerals for the input variables and unravelling the inductions in the manner described above. Note that if the conclusion of an induction rule is $A(t(x_i))$ with t a basic term then, upon substitution of $\bar{n}_i$ for x_i, we obtain $t(\bar{n}_i) \simeq \bar{m}$ and $A(\bar{m})$, and hence $A(t(\bar{n}_i))$ with proof-height bounded by a fixed linear function of ℓ determined by the term t and formula A, both of which appear in the original EA(I;O) derivation.

Lemma 3.2 *(Cut elimination) Define the "cut rank" of a derivation to be the maximum size of cut formulas appearing in it, where the "size" of a formula is zero if it's atomic and at least one greater than that of its subformulas otherwise.*

(i) Let C be a fixed formula of size $r+1$ and suppose we have derivations in EA(n;O) of $\Gamma \vdash C$ and $\Gamma', C \vdash A$ of proof-heights h_1, h_2 respectively, both with cut rank $\leq r$. Then a derivation of $\Gamma', \Gamma \vdash A$ can be obtained, with proof-height $\leq h_1 + h_2$ and again with cut rank $\leq r$.

(ii) Hence any derivation in EA(n;O) with height h and cut rank $r+1$ can be transformed into a derivation of the same end sequent, with height $\leq 2^h$ and cut rank $\leq r$.

Proof. (i) Suppose for example, that C is a formula of the form $\forall a D(a)$ with D of size r. In this case proceed by induction on the proof-height h_2 of $\Gamma', \forall a D(a) \vdash A$, showing, with each rule applied, that we can drop $\forall a D(a)$ in favour of Γ. Note first that if $\Gamma', \forall a D(a) \vdash A$ is an axiom then so is the result of replacing $\forall a D(a)$ by Γ.

Secondly, suppose $\Gamma', \forall a D(a) \vdash A$ comes about by a final application of any rule in which $\forall a D(a)$ is an inactive "side-formula". Then (possibly renaming a free output variable in a premise if it happens to clash with one in Γ) we can apply the induction hypothesis to the premise(s), replacing $\forall a D(a)$ by Γ and resulting in a derivation (or two) of proof-height less than $h_1 + h_2$. Then by re-applying this final rule one gets the desired result.

The only case remaining is where $\Gamma', \forall a D(a) \vdash A$ comes from the premise $\Gamma', \forall a D(a), D(t) \vdash A$ by application of the lefthand $\forall$-rule, with t some basic term. Then, applying the induction hypothesis to this premise yields a derivation of $\Gamma', \Gamma, D(t) \vdash A$ with height less than $h_1 + h_2$ and cut rank $\leq r$. Furthermore the given derivation of $\Gamma \vdash \forall a D(a)$ can be inverted and weakened to a derivation of $\Gamma', \Gamma \vdash D(t)$ without increasing the height or cut rank. Therefore by a final cut on the formula $D(t)$ we retrieve $\Gamma', \Gamma \vdash A$ with height no greater than $h_1 + h_2$ and cut rank r as required.

For other kinds of formulas C the proof is similar, but one may need to induct on h_1 instead of h_2, for example when C is $\exists a D(a)$.

(ii) The proof of this is now easy, given part (i). By induction on the height h of a EA(n;O) derivation with cut rank $r + 1$, we only need show how cuts:

$$\frac{\Gamma \vdash C \quad \Gamma, C \vdash A}{\Gamma \vdash A}$$

with cut formula C of size $r+1$, can be replaced by cuts of rank r at the expense of an exponential increase in proof-height. So suppose we encounter such a cut. First apply the induction hypothesis to each premise, reducing their cut ranks to r. Then the resulting derivations will have height no greater than 2^{h-1}. We can then apply part (i) above to these, in order to obtain a derivation of $\Gamma, \Gamma \vdash A$ with cut rank r and proof-height $\leq 2^{h-1} + 2^{h-1} = 2^h$. The system allows contraction of the two occurrences of Γ into one, without increasing the rank or height, and so this completes the proof.

These two lemmas now provide the following crucial result:

Theorem 3.3 *If $\bar{E} \vdash f(\vec{x}) \downarrow$ in EA(I;O) then there are fixed numbers k and r determined by its derivation, such that: for all numerical inputs $\vec{n} = n_1, n_2, \ldots \leq n$ of binary length $\leq \ell$, there is a derivation of $\bar{E} \vdash f(\vec{n}) \downarrow$ in EA(n;O) with proof-height at most $2_r(\ell.k)$, and in which all the cut formulas are Σ_1. If the original EA(I;O) derivation contains only Σ_1 induction formulas then we can take $r = 0$.*

Proof. The first lemma provides a EA(n:O) derivation of $\bar{E} \vdash f(\vec{n}) \downarrow$ with height $\ell.k$ in which the cut formulas are the induction formulas appearing in the EA(I:O) proof. If they are all Σ_1 nothing more needs to be done. Otherwise they have fixed bounded size r modulo Σ_1 formulas (counted here as having size 0). Then cut elimination uniformly reduces them to Σ_1 cuts at the expense of r successive exponential increases in the height.

Why is this important ? Because a EA(n:O) derivation of $f(\vec{n})$ with only Σ_1 cut formulas *is* a computation of $f(\vec{n})$!

Definition. Given a system of defining equations E, call an equation $t_1 \simeq t_2$ "E-true" if $\bar{E} \vdash t_1 \simeq t_2$. By the equality axioms, the relation "$t_1 \simeq t_2$ is E-true" is an equivalence between terms, and a congruence with respect to the function symbols. A set or conjunction of equations is E-true if each one is.

Lemma 3.4 *Suppose that in EA(n;O) we have a derivation, with proof-height h and involving only Σ_1 cut formulas, of*

$$\bar{E},\ \Gamma(\vec{a}) \vdash \exists\vec{b}A(\vec{a},\vec{b})$$

where Γ (respectively A) is a multiset (respectively conjunction) of equations $t_1 \simeq t_2$ with output variables among those displayed, and all function symbols come from the given system of defining equations E. We assume that any input variables have already been substituted by numerals $\bar{n}_1, \bar{n}_2, \dots$ with $n_i \leq n$. Let $m > 1$ be a fixed number, greater than the length of any basic term occurring as an existential witness or in the conclusion of a cases rule.

Then from any numerical assignment to the variables $\vec{a}$, and from the given inputs $n_1, n_2, \dots$, we can compute (via E) numerical witnesses $\vec{b}$ such that if $\Gamma(\vec{a})$ is E-true, so is $A(\vec{a},\vec{b})$. Furthermore the computation of each witness b_i takes no more than m^h register machine steps.

Proof. By induction on h with a case-analysis according to the last rule applied in deriving the given sequent

$$\bar{E},\ \Gamma(\vec{a}) \vdash \exists\vec{b}A(\vec{a},\vec{b})$$

which we shall refer to as S. We assume that, in addition to the inputs $n_1, n_2, \dots$, an arbitrary assignment of numerals to the variables $\vec{a}$ has been made. Then we must show how to compute values for $\vec{b}$ so that $A(\vec{a},\vec{b})$ is E-true if $\Gamma(\vec{a})$ is. For brevity we simply say in this case that the sequent S is E-true. The variables $\vec{a}, \vec{b}$ play the role of working-registers in the computation.

If S is an axiom, or if there are no existential quantifiers $\exists\vec{b}$, then there is nothing to compute, and the sequent is automatically E-true. In particular, for the equality axioms we appeal to the fact that "$t_1 \simeq t_2$ is E-true" is a congruence relation.

If S arises by a left-$\forall$ rule from

$$\bar{E},\ e(t),\ \Gamma(\vec{a})\ \vdash\ \exists\vec{b}A(\vec{a},\vec{b})$$

where $e(t)$ is one of the defining equations with basic term t substituted, this merely expresses the fact that $e(t)$ is used in the computation of $\vec{b}$. Since $e(t)$ is E-true, so must be S.

If S arises by the cases rule then the values of $\vec{b}$ are computed by a jump instruction: test the appropriate a_i to see if its value is zero or not, and then (by the induction hypothesis) proceed with the computation (in no more than m^{h-1} steps) of the appropriate $\vec{b}$ according to the left or right premise of the rule. S will then be E-true since the premises are, and the computation of $\vec{b}$ takes no more than $m + m^{h-1} \leq m^h$ steps in all.

If S arises by an $\exists$ rule from the premise (with proof-height $h-1$)

$$\bar{E},\ \Gamma(\vec{a})\ \vdash\ \exists\vec{b}A(\vec{a},t_0,\vec{b})$$

where t_0 witnesses the outermost quantifier $\exists b_0$ in S, then we can already compute the values of $\vec{b}$ in m^{h-1} steps so as to make the premise E-true. The conclusion S will be made E-true by computing $b_0 := t_0$ and since t_0 is a basic term (constructed out of successor and predecessor symbols only) of length less than m, this requires a further m register-machine steps at most. So the whole computation of $b_0, \vec{b}$ in this case takes no more than $m + m^{h-1} \leq m^h$ steps.

Finally suppose S arises by a Σ_1 cut, from premises

$$\bar{E},\ \Gamma(\vec{a})\ \vdash\ \exists\vec{c}\,C(\vec{a},\vec{c})$$

and

$$\bar{E},\ \Gamma(\vec{a}),\ \exists\vec{c}\,C(\vec{a},\vec{c})\ \vdash\ \exists\vec{b}A(\vec{a},\vec{b})$$

of maximum proof-height $h-1$. Without increasing proof-height, we can invert the $\exists\vec{c}$ in the antecedent of the second premise, leaving the $\vec{c}$ as fresh free variables, and invert the conjunction C so that it becomes an added set of equations. Then, by the induction hypothesis, the first premise allows us to compute values of $\vec{c}$ (in m^{h-1} steps) making it E-true, and the second premise allows computation of $\vec{b}$ from the values of $\vec{a}$ and $\vec{c}$, making it E-true also. Sequencing these two computations gives a computation of $\vec{b}$ from $\vec{a}$ alone, in $m^{h-1} + m^{h-1} \leq m^h$ steps, and making S E-true. This final case completes the proof.

Theorem 3.5 *If $f(\vec{x})$ is provably recursive in EA(I;O) then it is elementary (E^3). If it is provably recursive in the Σ_1-inductive fragment then it is sub-elementary (E^2).*

Proof. By the preceding theorem, if in EA(I;O) we have $\bar{E} \vdash f(\vec{x}) \downarrow$, then there are fixed numbers k and r such that for all inputs $\vec{n}$ with maximum binary length ℓ: $\bar{E} \vdash \exists b\, (f(\vec{n}) \simeq b)$ is derivable in EA($\max \vec{n}$;O) with proof-height $\leq 2_r(\ell.k)$, and with only Σ_1 cut formulas. By the lemma, the correct value b of $f(\vec{n})$ is then register-machine computable (via E) in a number of steps bounded by m to the power $2_r(\ell.k)$, where m is a fixed number (independent of $\vec{n}$) greater than the length of any basic term occurring in the original EA(I;O) derivation. Therefore f is elementary.

If the given termination proof of f takes place in the Σ_1-inductive fragment of EA(I;O), then we can take r to be zero. Thus f is computable in a number of steps bounded by $m^{\ell.k}$, which is less than some fixed polynomial of $\vec{n}$. This means f is sub-elementary.

3.1 PolyTime Functions

If, instead, the theory EA(I;O) were formulated on the basis of *binary* (rather than unary) number-representation, with two successors $S_0(a) = 2a$, $S_1(a) = 2a + 1$, one predecessor $P(0) = 0$, $P(S_0(a)) = a$, $P(S_1(a)) = a$, and an induction rule of the form

$$\frac{\Gamma \vdash A(0) \quad \Gamma,\, A(a) \vdash A(S_0 a) \quad \Gamma,\, A(a) \vdash A(S_1 a)}{\Gamma \vdash A(x)}$$

then the number of induction steps needed to "climb up" to $A(x)$ would be n where n is now the binary length of x. Thus a similar analysis to that given here, but with n corresponding to the binary length of the input, rather than the actual input itself, would show that the functions provably recursive in the Σ_1 inductive fragment of this binary theory are now those with complexity bounds polynomial in the binary length of their inputs, i.e. PTIME. cf. Leivant [6].

4 Exponential Complexity

In this section we begin to study the complexity hierarchy, between E^2 and E^3, that is induced by increasing levels of induction complexity. Here, we shall only attend to the "next" level after Σ_1 induction, by showing that the functions computable in exponential time $2^{p(x)}$ are exactly those provably recursive in the fragment of EA(I;O) corresponding to Π_2 induction. (The first author plans a more complete investigation of successive levels in a later paper). We must be careful however, about just what is meant by a Π_2 formula, since our theory is based on minimal (not classical) logic, and constructive logics are of course more sensitive to the precise logical structure of formulas (see e.g. Wehmeier [9]). We slightly modify Troelstra and Schwichtenberg (page 265) in making the following definition.

Definition. A Σ_1 formula is said to be of *level-1* and a *level-2* formula is one of the form

$$\forall a(C(a) \to D(a))$$

where C and D are Σ_1. We could allow a string of universal quantifiers $\forall\vec{a}$ in the prefix, but don't need to for what follows.

Note that level-2 formulas are classically equivalent to Π_2 formulas. The work of Burr [2] suggests that, if we were to work with classical logic and Π_2 induction, then the provably recursive functions might be the same as those provable in EA(I;O) with level ≤ 2 induction. However it remains to be seen whether his results do in fact carry over to the present, more restrictive, setting.

Lemma 4.1 *Every function computable in a number of steps bounded by $2^{p(\vec{x})}$ with p a polynomial, is provably recursive in the fragment of EA(I;O) allowing induction only on level-1 or level-2 formulas.*

Proof. This is just the particular case of Lemma 2.2 and Theorem 2.3 where $k = 1$. As in the proof of 2.3, suppose $g(\vec{x})$ is computable by a register machine working in unary notation on registers $r_1, r_2, \ldots, r_n$ within a number of steps bounded by $2_1(p(\vec{x}))$. Let E be the system of equations defining the state of the computation at step c:

$$\langle\ i(c), r_1(c), r_2(c), \ldots, r_n(c)\ \rangle$$

and let $A(\vec{x}, c)$ be the Σ_1 formula

$$\exists\vec{a}\ (i(c) \simeq a_0 \wedge r_1(c) \simeq a_1 \wedge \ldots \wedge r_n(c) \simeq a_n)$$

which is provably progressive in variable c. Then we must prove, by level ≤ 2 induction,

$$(\star) \qquad \bar{E}\ \vdash\ A(\vec{x}, 2_1(p(\vec{x})))$$

from which $\bar{E} \vdash g(\vec{x}) \downarrow$ follows by the definition $g(\vec{x}) \simeq r_1(2_1(p(\vec{x})))$.

To prove $(\star)$ by level ≤ 2 induction, we simply have to analyse the proofs of lemmas 2.1 and 2.2 a little more carefully. For any polynomial term p let $B(p)$ be the formula

$$\forall c\, (A(\vec{x}, c) \to p \downarrow \wedge f_1(c, p) \downarrow \wedge A(\vec{x}, f_1(c, p)))$$

and notice that although it isn't quite a level-2 formula, it is trivially and provably equivalent to one. (Recall that our notion of Σ_1 formula is restricted to an existentially quantified conjunction of equations, and the conjunction occurring after the implication inside B is equivalent to a single Σ_1 formula by distribution of $\exists$ over $\wedge$). Notice also that $B(b)$ is provably progressive since $A(\vec{x}, c)$ is. Hence by lemma 2.1 we can prove $\bar{E} \vdash B(p(\vec{x}))$ for any polynomial p, and by setting $c := 0$ we obtain $(\star)$ as required. It only remains to check that this application of lemma 2.1 requires nothing more than level-2 induction. In fact the inductions required are on formulas of shape $q \downarrow \wedge B(p)$ with q other polynomial terms, but since we can prove $A(\vec{x}, 0)$ the subformulas $q \downarrow$ can also be shifted after the implication inside B, yielding provably equivalent level-2 forms. Thus level-2 induction suffices, and this completes the proof.

Lemma 4.2 *(Cut Reduction) A derivation in EA(n;O) in which all cut formulas are of level ≤ 2 can be transformed into one with only Σ_1 cuts. Furthermore, if the original derivation has proof-height h then the new one has height $\leq 3^h$.*

Proof. First, suppose B is any fixed level-2 formula, say $\forall a(C(a) \to D(a))$, and suppose we are given derivations in EA(n;O) of $\Gamma \vdash B$ and $\Gamma,\ B,\ \{C(t_i) \to D(t_i)\}_{i<k} \vdash A$, of heights h_1 and h_2 respectively, both containing only Σ_1 cut formulas, and where the terms t_i are basic. (k may be zero, in which case the $C(t_i) \to D(t_i)$ don't appear). Then we can derive $\Gamma \vdash A$ with proof-height $\leq h_1 + 2.h_2$ and the derivation involves only Σ_1 cuts.

The proof is by induction over the second given derivation of height h_2. The only crucial case is where one of the implications $C(t_j) \to D(t_j)$ is introduced on the left of $\vdash$. Then the immediate premises are

$$\Gamma,\ B,\ \{C(t_i) \to D(t_i)\}_{i<k} \vdash C(t_j)$$

and

$$\Gamma,\ B,\ \{C(t_i) \to D(t_i)\}_{i<k},\ D(t_j)\ \vdash\ A$$

so by the induction hypothesis we have derivations of $\Gamma\ \vdash\ C(t_j)$ and $\Gamma,\ D(t_j)\ \vdash\ A$ of heights $\leq h_1 + 2.h_2 - 2$. But by inverting the other given derivation we have also $\Gamma,\ C(t_j)\ \vdash\ D(t_j)$ with proof-height h_1. Therefore by two successive cuts on $C(t_j)$ and then $D(t_j)$, both of which are Σ_1, we obtain the required derivation of $\Gamma\ \vdash\ A$ with height $\leq h_1 + 2.h_2$. All other cases follow immediately by applying the induction hypothesis to the premises and then (if necessary) re-applying the final rule. In the case of axioms the formulas $B,\ \{C(t_i) \to D(t_i)\}$ can be deleted since they are not atomic.

Now let $\Gamma\ \vdash\ A$ be derived with height h, and suppose all the cuts are on level ≤ 2 formulas. Then by induction on h we obtain another derivation of height $\leq 3^h$ in which all cut formulas are Σ_1. For if the last rule applied is a cut with level-2 cut formula B then the premises are $\Gamma\ \vdash\ B$ and $\Gamma,\ B\ \vdash\ A$. By the induction hypothesis these are both derivable with only Σ_1 cuts and with proof-height $\leq 3^{h-1}$. But then the result above gives a derivation of $\Gamma\ \vdash\ A$ with only Σ_1 cuts and height $\leq 3^h$. For all other rule applications, simply apply the induction hypothesis to the premises and then re-apply that rule. This completes the proof.

Theorem 4.3 *A function is computable within a number of steps bounded by $2^{p(\vec{x})}$ for some polynomial p if and only if it is provably recursive in EA(I;O) by a derivation involving induction only on formulas of level ≤ 2.*

Proof. One half is already done in lemma 4.1. For the converse suppose $\bar{E}\ \vdash\ f(\vec{x})\downarrow$ by a derivation involving only level ≤ 2 inductions. Then for inputs $\vec{n}$ of binary length at most ℓ, $\bar{E}\ \vdash\ f(\vec{n})\downarrow$ is derivable in EA($\max\vec{n}$;O) with height at most $\ell.k$ for some fixed k, and with only level ≤ 2 cut formulas. By the lemma just above, this derivation can be transformed into one with only Σ_1 cuts, but its height will then be at most $3^{\ell.k}$, which is less than $p(\vec{n})$ for some polynomial p. Now apply lemma 3.4 to obtain an exponential bound on the register-machine computation of $f(\vec{n})$.

5 Slow Growing Bounding Functions

Though technically unnecessary, it is nevertheless interesting to bring out the connection here, with the slow growing hierarchy. Its use comes about by analogy with a standard procedure for reading off "fast growing" bounds in the case of classical Peano Arithmetic (see e.g. Fairtlough and Wainer [5]). However in the present situation, where the induction variables are kept separate from the quantified ones, things are much simpler.

Let us consider the collection of "ordinal structures" (below ϵ_0) built up from 0, 1 and ω by repeated application of addition:

$$\alpha + 0 = \alpha, \quad \alpha + (\beta + 1) = (\alpha + \beta) + 1, \quad \alpha + \lambda = \sup_i \, (\alpha + \lambda_i)$$

and exponentiation to any fixed finite base, for example

$$2^0 = 1, \quad 2^{\beta+1} = 2^\beta + 2^\beta, \quad 2^\lambda = \sup_i \, 2^{\lambda_i}$$

where $\lambda = \sup_i \, \lambda_i$ signifies the assignment of the fixed fundamental sequence $\lambda_0, \lambda_1, \lambda_2, \ldots$ to the limit λ. Note however, that we do not choose the obvious fundamental sequence for ω, but rather the weakly increasing one given by $\omega_i = \log i$, the length of the binary representation of i.

Furthermore, for each fixed i let $\prec_i$ be the transitive closure of

$$0 \preceq_i \alpha, \quad \alpha \prec_i \alpha + 1, \quad \lambda_i \prec_i \lambda$$

where $\lambda = \sup_i \, \lambda_i$ denotes any limit so generated.

Now "dress up" the theory EA(n;O) defined above, by allowing ordinal superscripts as bounds on the proof-height thus $\vdash^\alpha$, and adding a new rule of "accumulation":

$$\frac{\Gamma \vdash^\alpha A}{\Gamma \vdash^\beta A}$$

under the condition $\alpha \prec_n \beta$. Then lemma 3.1 becomes

Lemma 5.1 *If $\bar{E} \vdash f(\vec{x}) \downarrow$ in EA(I;O) then there is a fixed k such that for all inputs $n_1, n_2, \ldots \leq n$, $\bar{E} \vdash^{\omega.k} f(\bar{n}_1, \bar{n}_2, \ldots) \downarrow$ in EA(n;O).*

The cut elimination lemma 3.2 works in just the same way with ordinal bounds, and then theorem 3.3 becomes

Theorem 5.2 *If $\bar{E} \vdash f(\vec{x}) \downarrow$ in EA(I;O) then there are fixed numbers k and r such that: for all numerical inputs $n_1, n_2, \ldots \leq n$ there is a derivation in EA(n;O) of $\bar{E} \vdash^\alpha f(\bar{n}_1, \bar{n}_2, \ldots) \downarrow$ in which all cut formulas are Σ_1, and where $\alpha = 2_r(\omega.k)$.*

This supplies a *uniform* ordinal bound on the derivations, for all inputs $\vec{n}$, and it then only remains to retrieve the finite bounds so as to apply lemma 3.4 and then obtain theorem 3.5.

The point is that these finite bounds arise immediately by collapsing the ordinal bound under the Slow Growing Function G_n given by

$$G_n(0) = 0, \quad G_n(\alpha + 1) = G_n(\alpha) + 1, \quad G_n(\lambda) = G_n(\lambda_n)$$

and since G_n homomorphically maps ordinal arithmetic down onto its number-theoretic part, we have $G_n(\omega) = \log n$ and hence

$$G_n(\ 2_r(\omega.k)\ ) = 2_r(\ell.k)$$

where, as before, ℓ is the maximum binary length of the inputs.

The use of G is brought out in more detail in [4], where a preliminary treatment of the results of section 3 is given, but in terms of a functional interpretation rather than the more direct cut-elimination arguments used here.

References

[1] S. Bellantoni and S. Cook, *A new recursion theoretic characterization of the polytime functions*, Computational Complexity Vol. 2 (1992) pp. 97 - 110.

[2] W. Burr, *Fragments of Heyting arithmetic*, Journal of Symbolic Logic Vol. 65 (2000) pp. 1223 - 1240.

[3] S.R. Buss, *Bounded arithmetic*, Bibliopolis (1986).

[4] N. Cagman, G. E. Ostrin and S.S. Wainer, *Proof theoretic complexity of low subrecursive classes*, in F. L. Bauer and R. Steinbrueggen (Eds) Foundations of Secure Computation, NATO ASI Series F Vol. 175, IOS Press (2000) pp. 249 - 285.

[5] M. Fairtlough and S.S. Wainer, *Hierarchies of provably recursive functions*, in S. Buss (Ed) Handbook of Proof Theory, Elsevier Science BV (1998)

pp. 149 - 207.

[6] D. Leivant, *Intrinsic theories and computational complexity* , in D. Leivant (Ed) Logic and Computational Complexity, Lecture Notes in Computer Science Vol. 960, Springer-Verlag (1995) pp. 177 - 194.

[7] D. Rödding, *Klassen rekursiver funktionen*, in M. H. Löb (Ed) Proc. of Summer School in Logic, Leeds 1967, Lecture Notes in Mathematics Vol. 70, Springer-Verlag (1968) pp. 159 - 222.

[8] A. S. Troelstra and H. Schwichtenberg, *Basic proof theory*, Cambridge Tracts in Theoretical Computer Science Vol. 43, CUP (1996).

[9] K.F. Wehmeier, *Fragments of HA based on* Σ_1*- induction*, Archive for Mathematical Logic Vol. 37 (1997) pp. 37 - 49.