PROOF THEORETIC COMPLEXITY

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`*)*

Abstract *A weak formal theory of arithmetic is developed, entirely analogous to classical arithmetic but with two separate kinds of variables: induction variables and quantifier variables. The point is that the provably recursive functions are now more feasibly computable than in the classical case, lying between Grzegorczyk's E^2 and E^3, and their computational complexity can be characterized in terms of the logical complexity of their termination proofs. Previous results of Leivant are reworked and extended in this new setting, with quite different proof theoretic methods.*

Keywords: proof theory; computational complexity; equation calculus; arithmetic; (sub-) elementary functions; cut elimination; ordinal bounds.

1. Introduction

The classical methods of proof theory (cut elimination or normalization) enable one to read off, from a proof that a recursively defined function terminates everywhere, a bound on its computational complexity. This was already implicit in the work of Kreisel (1951-52) fifty years ago, where the first recursion theoretic characterization of the provably terminating functions of arithmetic was given (in terms of an algebra of recursions over transfinite well orderings of order-types less than ε_0). However, although the programs defining them are quite "natural" in their structure, the functions themselves are (in general) far away from being realistically or feasibly computable. It took another thirty-five years before Buss (1986) had developed Bounded Arithmetic, and characterized the PTIME functions as those provably terminating in its Σ_1 fragment. More recently, through the work of Bellantoni and Cook (1992) and Leivant (1994) on their different forms of "tiered recursion", there has been a growing interest in developing theories analogous to classical arithmetic, but without the explicit quantifier-bounds of Bounded Arithmetic, whose provably terminating functions form (much) more realistic complexity classes than in the classical case. In particular, Leivant's (1995) theories based on his ramified induction schemes give proof theoretic characterizations of PTIME and the low Grzegorczyk classes E^2 and E^3. What we do here is present a simple re-formulation of arithmetic, motivated by the normal/safe variable separation of Bellantoni and Cook, in which results similar to Leivant's can be developed and extended. Our methods, however, are quite different from his, being based on traditional cut elimination with ordinal bounds. The analogies with the classical case are very strong (cf. Fairtlough and Wainer (1998)).

Bellantoni and Cook made a simple syntactic restriction on the usual schemes of primitive recursion, by inserting a semicolon to separate the variables into two distinct classes: the recursion variables, which they call "normal", and the substitution variables, which they call "safe". The effect of this is quite dramatic: every function so defined is now polynomially bounded! By working on binary number representation, they thus characterize PTIME, and if instead one works with unary representation (as we shall do) one obtains the Grzegorczyk class E^2 (see also Handley and Wainer (1999)). We shall consider the result of imposing a similar variable separation on a formal system of arithmetic. Thus "input" or "normal" variables will control the inductions, and will only occur free, whereas "output" or "safe" variables may be bound by quantifiers. The resulting theory, denoted EA(I;O) to signify this input/output separation of variables, we call Elementary Arithmetic because its provably terminating functions all lie in the Grzegorczyk class E^3 of "elementary" functions (those computable in a number of steps bounded by a finitely iterated exponential). There is a well known complexity hierarchy for E^3, according to the number of times the exponential is iterated in producing the required bound. At the bottom is E^2 with polynomial bounds, at the next level lies EXPTIME with bounds 2^p, p a polynomial, and so on. As we shall show, this hierarchy is closely related to the increasing levels of induction complexity of termination proofs.

The theory EA(I;O) will be based on minimal, not classical, logic. The reason is that from a proof in classical logic of a Σ_1 formula $\exists a A(a)$ we can, by Herbrand's Theorem, extract a finite set of terms t_i such that $\vdash A(t_1) \vee \ldots \vee A(t_n)$. Although one of the t_i will be a correct witness for A, we don't necessarily know which one. Minimal logic allows us to extract one, correct witness, so it provides more precise computational information. This will not be a restriction for us, since we will still be able to prove termination of the same functions in EA(I;O) with minimal logic, as we can using classical logic. On the other hand one needs to exercise more care in measuring the complexity of induction formulas, since minimal or intuitionistic logic is more sensitive to the precise logical structure of formulas than is classical logic (see Burr (2000)).

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) $t \downarrow$.

For example suppose f is introduced by the defining equation:

$$f(x) \simeq g(x, h(x))$$

where g and h have been previously defined. Then the termination proof for $f(x)$ goes like this:

By the substitution axiom (which, interacting with equational cuts, constitutes the only computation rule of the equation calculus),

$$h(x) \simeq b,\ g(x, b) \simeq a\ \vdash\ g(x, h(x)) \simeq a$$

and by the defining equation for f,

$$g(x, h(x)) \simeq a\ \vdash\ f(x) \simeq a\ .$$

Therefore by an equational cut,

$$h(x) \simeq b,\ g(x, b) \simeq a\ \vdash\ f(x) \simeq a$$

and then by applying the $\exists$ rule on the right, followed by an $\exists$ rule on the left (since variable a is then not free anywhere else),

$$h(x) \simeq b,\ g(x, b) \downarrow \vdash\ f(x) \downarrow\ .$$

Now assuming that $g(x, b) \downarrow$ has already been proven, we can cut it to obtain

$$h(x) \simeq b\ \vdash\ f(x) \downarrow\ .$$

Since b does not now occur free anywhere else, we can existentially quantify it to obtain

$$h(x) \downarrow \vdash\ f(x) \downarrow$$

and then, assuming $h(x) \downarrow$ is already proven, another cut gives

$$\vdash\ f(x) \downarrow\ .$$

Note that a bottom-up (goal-directed) reading of the proof determines the order of computation of $f(x)$ thus: first find the value of $h(x)$, call it b, then find the value of $g(x, b)$, and then pass this as the value of $f(x)$.

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.

2. The theory EA(I;O)

There will be two kinds of variables: "input" (or "normal") variables denoted $x, y, z, \ldots$, and "output" (or "safe") variables denoted $a, b, c, \ldots$, 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 $f, g, h, \ldots$ 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 $A, B, \ldots$ 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 $A \rightarrow \perp$.

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 (1996) suits us perfectly, except that we write $\vdash$ instead of their $\Rightarrow$. However, as already mentioned above, 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-∃ and right-∃ 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-∃ rule the output variable b is not free in Γ, B, and in the right-∃ rule the witnessing term t is basic. The left-∀ and right-∀ 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.

3. Elementary Functions are Provably Recursive

We will first look at some examples of how termination proofs work.

Examples 3.1.

1. Let E be the following program that defines the function $a + b$,

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

   From an appropriate logical axiom we obtain $\bar{E} \vdash a + Sb \simeq S(a + b)$ as a consequence of left $\forall$ rules. Cutting with appropriate equality axioms we obtain,

$$\bar{E}, \ a + b \simeq c \ \vdash \ a + Sb \simeq Sc$$

   which when read from left to right mirrors what we would expect the order of computation to be. As Sc is a basic term we can existentially quantify on the right. Now that c is no longer free anywhere else we can existentially quantify on the left. What remains is the step premise for an induction over b,

$$\bar{E}, \ a + b \downarrow \ \vdash \ a + Sb \downarrow .$$

   An equivalent sequent for when b is 0 can be trivially obtained. Thus as an induction conclusion we obtain $\bar{E} \vdash a + x \downarrow$.

2. For multiplication we need to augment E by

$$b + c.0 \simeq b, \quad b + c.Sd \simeq (b + c.d) + c.$$

As above by starting from an appropriate axiom and applying the left $\forall$ rule three times we obtain, $\bar{E} \vdash b + x.Sd \simeq (b + x.d) + x$ and hence,

$$\bar{E},\ b + x.d \simeq a \ \vdash\ b + x.Sd \simeq a + x.$$

Under the assumption that $a + x \downarrow$ we can existentially quantify on the right we obtain,

$$\bar{E},\ a + x \downarrow,\ b + x.d \simeq a \ \vdash\ b + x.Sd \downarrow.$$

This assumption can be cut out as it is the result of the previous example. Now we can existentially quantify on the left over the remaining variable a. What remains is an induction step over d, from which the conclusion is $\bar{E} \vdash b + x.y \downarrow$, where y is any input variable. Note that the cut reflects that to prove termination for multiplication we need first have proved it for addition. Further note that we derive $a + x$ for addition not $a + b$. This precipitates our choice of formulating the starting axiom over an input x.

3. If we recount this last derivation whereby the starting axiom is formulated over $x.y$ instead of the input x, then after applying the left $\forall$ rules (the definedness of $x.y$ having been obtained in the previous example by setting b to be 0), we would have,

$$\bar{E} \ \vdash\ b + (x.y).Sd \simeq (b + (x.y).d) + (x.y),$$

from which we obtain,

$$\bar{E},\ b + (x.y).d \simeq a \ \vdash\ b + (x.y).Sd \simeq a + (x.y).$$

We follow the same steps as above. Assuming $a + (x.y) \downarrow$ we can existentially quantify on the right. This assumption is then cut out as it is exactly the result obtained from the previous example. The induction step is obtained once we existentially quantify over a on the left,

$$\bar{E},\ b + (x.y).d \downarrow \ \vdash\ b + (x.y).Sd \downarrow$$

The conclusion is $\bar{E} \vdash b + (x.y).z \downarrow$. Clearly this procedure can be repeated as many times as we would like. In particular, we therefore obtain derivations for $a+x^2 \downarrow, a+x^3 \downarrow, a+x^4 \downarrow, \ldots$, the building blocks for constructing polynomials, over inputs variables. In the following we shall generalise this argument.

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

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

Lemma 3.2. *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 $\forall a(A(a) \rightarrow A(Sa))$ to yield $A(a+b) \rightarrow A(a+Sb)$. 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

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

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.

Notice that $Prog_a A \vdash A(x)$ is equivalent to the induction principle of EA(I;O). The preceding lemma extends this principle to any polynomial in $\vec{x}$ whereby, through inspection, the induction complexity is maintained as an "A" induction. What follows is the extension of this principle to any finitely iterated exponential, the cost of which being ever greater induction complexity. We therefore start by defining a measure for induction complexity.

Definition. A Σ_1 formula is said to be of "level-1" and a "level-(n+1)" formula is one of the form

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

where C and D are level-n. We could allow a string of universal quantifiers $\forall\vec{a}$ in the prefix, but it is not necessary for what follows. Typical examples of level-2 formulas would be $\forall a(f(a)\downarrow)$, $\forall a(f(a)\downarrow) \wedge \forall a(g(a)\downarrow)$ and $\forall c(c \leqslant d \to \forall a(f(a,c)\downarrow))$. The latter two do not fit the definition, but they are both minimally provably equivalent to level-2 formulas, namely $\forall a(f(a)\downarrow\ \wedge\ g(a)\downarrow)$ and $\forall c\forall a(c \leqslant d \to f(a,c)\downarrow)$ respectively. All these formulations play roles in the following examples.

Examples 3.3.

1. Let E contain the the following two equations,

   $$a + 2^0 \simeq Sa, \quad a + 2^{Sb} \simeq (a + 2^b) + 2^b$$

   from which we can now obtain,

   $$\bar{E},\ a + 2^b \simeq c\ \vdash\ a + 2^{Sb} \simeq c + 2^b.$$

   Using the more general existential rule for the right we obtain,

   $$\bar{E},\ c + 2^b \downarrow,\ a + 2^b \simeq c\ \vdash\ a + 2^{Sb} \downarrow.$$

   To apply the induction all that remains is to existentially bind the remaining formula over the variable c. This is inhibited since c appears free in the first formula. The solution is easy for we first universally bind this free occurrence of c, and then apply the left $\exists$ rule to obtain,

   $$\bar{E},\ \ \forall a(\, a + 2^b \downarrow),\ a + 2^b \downarrow \vdash\ a + 2^{Sb} \downarrow.$$

   The simplicity of the solution hides the significance of its consequences; the induction complexity. The induction is going to be over b and as this variable appears throughout all the remaining formulas, we need to first "tidy up", through quantifications. Therefore our induction is no longer a level-1 induction (as for addition, multiplication and all polynomials), but of level-2,

   $$\bar{E},\ \forall a(\, a + 2^b \downarrow)\ \vdash\ \forall a(\, a + 2^{Sb} \downarrow).$$

   The conclusion, $\bar{E}\ \vdash\ \forall a(\, a + 2^x \downarrow)$, completes the derivation. Note that the two uses of left $\forall$ rule, correspond to two applications of the induction hypothesis.

2. Let $Fib(a, b)$ be the function that gives a plus the b'th entry in the Fibonacci sequence, and let E be the following program program defining it:

   $$Fib(a, 0) \simeq Sa, \quad Fib(a, 1) \simeq Sa, \quad Fib(a, SSb) \simeq Fib(Fib(a, b), Sb).$$

   When unravelled the recursion step becomes,

   $$\bar{E},\ Fib(a, b) \simeq c\ \vdash\ Fib(a, SSb) \simeq Fib(c, Sb).$$

   Existentially quantifying on the right, under the assumption that witness $Fib(c, Sb)$ is defined, we obtain,

   $$\bar{E},\ Fib(a, b) \simeq c,\ Fib(c, Sb) \downarrow\ \vdash\ Fib(a, SSb) \downarrow.$$

Notice that c appears twice (reflecting the substitution that increases the function's complexity) and that an existential quantifier on the left would have to bind both occurrences, something that would be undesired. Universally quantifying the righter most occurrence solves the problem and after further quantifications we now have,

$$\bar{E},\ \forall a(\,Fib(a,b)\downarrow),\ \forall a(\,Fib(a,Sb)\downarrow)\ \vdash\ \forall a(\,Fib(a,SSb)\downarrow).$$

Along with $\forall a(\,Fib(a,Sb)\downarrow) \vdash \forall a(\,Fib(a,Sb)\downarrow)$, (easily obtained from a logical axiom) we have after a right $\wedge$ rule following immediately by a left $\wedge$ rule,

$$\bar{E},\forall a(Fib(a,b)\downarrow)\wedge\forall a(Fib(a,Sb)\downarrow)\vdash\forall a(Fib(a,Sb)\downarrow)\wedge\forall a(Fib(a,SSb)\downarrow).$$

This is now an induction step, from which we could extract the result that $\forall a(\,Fib(a,x)\downarrow)$. Note that although the induction formula, $\forall a(\,Fib(a,b)\downarrow)\wedge\forall a(\,Fib(a,Sb)\downarrow)$, is not formally a level-2 formula, it is provably equivalent to one, namely $\forall a(\,Fib(a,b)\downarrow\wedge Fib(a,Sb)\downarrow)$.

3. A more complex example suggested by H. Jervell: let E contain the following three equations,

$$f(a,0,c)\simeq a+Sc,$$
$$f(a,Sb,0)\simeq f(a,b,b),$$
$$f(a,Sb,Sc)\simeq f(f(a,Sb,c),b,b).$$

describing the function $a+Fac(b).Sc$ as a double recursion, where $Fac(b)$ is the factorial of b, the function we are interested in. Following in the usual way, we obtain,

$$\bar{E},\ f(a,Sb,c)\simeq d\ \vdash\ f(a,Sb,Sc)\simeq f(d,b,b).$$

Applying a right $\exists$ rule, under the assumption that $f(d,b,b)\downarrow$, gives,

$$\bar{E},\ f(d,b,b)\downarrow,\ f(a,Sb,c)\simeq d\ \vdash\ f(a,Sb,Sc)\downarrow.$$

Again we are obliged to universally quantify on the left, to enable the appropriate left existential quantification. From this, after further $\forall$ rules, first on the left then finally on the right, we have,

$$\bar{E},\ \forall a(\,f(a,b,b)\downarrow),\ \forall a(\,f(a,Sb,c)\downarrow)\ \vdash\ \forall a(\,f(a,Sb,Sc)\downarrow).$$

We trivially obtain $\bar{E},\forall a(\,f(a,b,b)\downarrow)\ \vdash\ \forall a(\,f(a,Sb,0)\downarrow)$ from the second equation of E. Therefore we have,

$$\bar{E},\ \forall a(\,f(a,b,b)\downarrow)\ \vdash\ Prog_c(\,\forall a(\,f(a,Sb,c)\downarrow)\,).$$

Looking ahead, we know that if we were to apply the induction rule here, we will be prevented from applying the second induction at a later stage. Instead, we need a stronger form of induction, one that enables us to derive the following conclusion,

$$\bar{E},\ \forall a(\, f(a,b,b)\downarrow) \ \vdash\ \forall c(\, c \leqslant x \rightarrow \forall a(\, f(a,Sb,c)\downarrow)\,).$$

For from this sequent after inverting the outermost universal quantifier on the right at Sb and introducing an implication on the left with a derivation of $\bar{E}, Sb \leqslant x \vdash b \leqslant x$ we arrive at,

$$\bar{E},\ b \leqslant x \rightarrow \forall a(\, f(a,b,b)\downarrow) \ \vdash\ Sb \leqslant x \rightarrow \forall a(\, f(a,Sb,Sb)\downarrow).$$

Now we may apply an induction over b to obtain,

$$\bar{E} \ \vdash\ x \leqslant x \rightarrow \forall a(\, f(a,x,x)\downarrow),$$

from which we may extract the result that $a + Fac(Sx)$ is defined.

There remain two outstanding parts of the proof. The first is the stronger form of induction, that from the premise $\bar{E},\Gamma \vdash Prog_a A(a)$ we may derive $\bar{E},\Gamma \vdash \forall a(\, a \leqslant x \rightarrow A(a)\,)$. Thus we augment E by the following equations that define the predecessor and modified minus functions,

$$P0 \simeq 0, \quad P(Sb) \simeq b$$

$$a \dot{-} 0 \simeq a, \quad a \dot{-} Sb \simeq Pa \dot{-} b$$

and we define $t \leqslant r$ by $t \dot{-} r \simeq 0$. From these extra equations alone we derive $\bar{E},\ a \leqslant Sb \ \vdash\ Pa \leqslant b$. Further, from the premise of progression and that $\bar{E} \vdash \forall a(\, a \simeq 0 \vee a \simeq S(Pa)\,)$ (easily obtained from cases) we derive $\bar{E},\ A(Pa) \ \vdash\ A(a)$. Putting these together with the implication rules we have,

$$\bar{E},\ Pa \leqslant b \rightarrow A(Pa) \ \vdash\ a \leqslant Sb \rightarrow A(a).$$

Universally quantifying on the left, over witness Pa, and then on the right over variable a we have the induction step. The case for when b is 0 is trivially obtained and the result then follows from an induction.

The second, and final, outstanding part of the original proof is the derivation of $\bar{E}, Sb \leqslant x \vdash b \leqslant x$. From the axiom

$$Pa \dot{-} Sb \simeq P(a \dot{-} Sb) \ \vdash\ Pa \dot{-} Sb \simeq P(a \dot{-} Sb)$$

we obtain,

$$\bar{E},\ PPa \dot{-} b \simeq P(Pa \dot{-} b)\ \vdash\ Pa \dot{-} Sb \simeq P(a \dot{-} Sb),$$

by substitution since $Pa \dot{-} Sb \simeq PPa \dot{-} b$ and $a \dot{-} Sb \simeq Pa \dot{-} b$. Universally quantifying, first on the left with witness Pa and then on the right over variable a we obtain,

$$\bar{E},\ \forall a(\, Pa \dot{-} b \simeq P(a \dot{-} b)\,)\ \vdash\ \forall a(\, Pa \dot{-} Sb \simeq P(a \dot{-} Sb)\,).$$

For when b is 0 we can trivially derive $\bar{E} \vdash \forall a(\, Pa \dot{-} 0 \simeq P(a \dot{-} 0)\,)$, thus we obtain from an induction $\bar{E} \vdash \forall a(Pa \dot{-} x \simeq P(a \dot{-} x))$. Substituting the term Sb for the universally quantified a, we obtain

$$\bar{E} \vdash PSb \dot{-} x \simeq P(Sb \dot{-} x)$$

and from the definition of predecessor this is the same as

$$\bar{E} \vdash b \dot{-} x \simeq P(Sb \dot{-} x).$$

Thus if $Sb \dot{-} x \simeq 0$, (i.e. $Sb \leqslant x$), then $b \dot{-} x \simeq 0$, (i.e. $b \leqslant x$), since $P0 \simeq 0$, and we are therefore done.

Note that the induction takes the form $\forall c(\, c \leqslant d \rightarrow \forall a(\, f(a, Sb, c)\downarrow)\,)$, which although is not formally a level-2 formula is indeed provably equivalent to one, namely, $\forall c \forall a(\, c \leqslant d \rightarrow f(a, Sb, c)\downarrow)$.

The three functions, exponential, Fibonacci and factorial, have two properties in common; firstly, they all need level-2 induction to prove their termination and secondly they are all register machine computable in an exponential number of steps. This relationship is of no coincidence, as it is a result that follows from the following more general results.

Definition. Extend the system of equations E that includes the equations for constructing polynomials, by adding the new recursive definitions:

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

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 3.4. *In EA(I;O) we can prove, for each k, for any polynomial p 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 $A(a)$,

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

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

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

and then by the previous lemma,

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

and then by putting a to be 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 3.5. *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. Every function computable in number of steps bounded by an exponential function of its inputs is provably recursive in the "level-2" inductive fragment of EA(I;O).*

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 (1968), 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 $A(\vec{x}, p(\vec{x}))$ in lemma 3.2 only uses inductions on substitution instances of A, and here, A is Σ_1.

To see that a function computable in $\leqslant 2^{p(\vec{x})}$ steps is provably recursive in the level-2 inductive fragment, we need only verify that the proof of $\bar{E} \vdash A(\vec{x}, 2_1(p(\vec{x})))$ uses inductions that are at most level-2. Therefore, all we need to do is analyse the proofs of lemmas 3.2 and 3.4 a little more carefully. For any polynomial term p let $B(p)$ be the formula

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

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 3.2 we can prove $\bar{E} \vdash B(p(\vec{x}))$ for any polynomial p, and by setting $c := 0$ we obtain the desired result. It only remains to check that this application of lemma 3.2 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.

4. 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 $\vec{x} = x_1, x_2, \ldots$ 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.

What we have just done is unravelled the induction up to $A(\bar{m})$, replacing the single application of the induction rule by a sequence of cuts. Thus if we make any fixed numerical instantiation of the input variables, a proof in EA(I;O) can be unravelled into one in which all the inductions have been removed, and replaced by sequences of cuts on the induction formulas. The Cut-Elimination process then allows us to successively reduce the logical complexity of cut formulas, but the price paid is an exponential increase in the height of the proof. Once we get down to the level of Σ_1 cut formulas, what remains is essentially a computation (from the given system of equations E), and its complexity will correspond to the number of proof-steps. However, the size of the resulting proof (or computation) will vary with the size of numerical inputs originally substituted for the input variables. What we need is some way of giving a *uniform* bound on the proof-size, independent of the given input numbers. This is where *ordinals* enter into proof theory.

4.1. ORDINAL BOUNDS FOR RECURSION

For each fixed number k, we inductively generate an infinitary system of sequents

$$E,\ n : N,\ \Gamma \ \vdash^{\alpha} \ A$$

where (i) E is a (consistent) set of Herbrand-Gödel-Kleene defining equations for partial recursive functions $f, g, h, \ldots$; (ii) n is a bound on the numerical inputs (or more precisely, its representing numeral); (iii) A is a closed formula, and Γ a finite multiset of closed formulas, built up from atomic equations between arbitrary terms t involving the function symbols of E; and (iv) α, β denote ordinal bounds which we shall be more specific about later (for the time being think of β as being smaller than α "modulo k").

Note that we do not explicitly display the parameter k in the sequents below, but if we later need to do this we shall insert an additional declaration $k : I$ in the antecedent thus:

$$E,\ k : I,\ n : N,\ \Gamma \ \vdash^{\alpha} \ A\ .$$

Intuitively, k will be a bound on the lengths of "unravelled inductions".

The first two rules are just the input and substitution axioms of the equation calculus, the next two are computation rules for N, and the rest are essentially just formalised versions of the truth definition, with Cut added.

E1 $E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ e(\vec{n})$ where e is either one of the defining equations of E or an identity $t \simeq t$, and $e(\vec{n})$ denotes the result of substituting, for its variables, numerals for numbers $\leqslant n$.

E2 $E,\ n : N,\ \Gamma,\ t_1 \simeq t_2,\ e(t_1)\ \vdash^{\alpha}\ e(t_2)$ where $e(t_1)$ is an equation between terms in the language of E, with t_1 occurring as a subterm, and $e(t_2)$ is the result of replacing an occurrence of t_1 by t_2.

N1

$$E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ m : N \ \text{ provided } m \leqslant n+1$$

N2

$$\frac{E,\ n : N,\ \Gamma\ \vdash^{\beta}\ n' : N \qquad E,\ n' : N,\ \Gamma\ \vdash^{\beta'}\ A}{E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ A}$$

Cut

$$\frac{E,\ n : N,\ \Gamma\ \vdash^{\beta}\ C \qquad E,\ n : N,\ \Gamma,\ C\ \vdash^{\beta'}\ A}{E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ A}$$

∃L

$$\frac{E,\ \max(n,i) : N,\ \Gamma,\ B(i)\ \vdash^{\beta_i}\ A \ \text{ for every } i \in N}{E,\ n : N,\ \Gamma,\ \exists b B(b)\ \vdash^{\alpha}\ A}$$

∃R

$$\frac{E,\ n : N,\ \Gamma\ \vdash^{\beta}\ m : N \qquad E,\ n : N,\ \Gamma\ \vdash^{\beta'}\ A(m)}{E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ \exists a A(a)}$$

∀L

$$\frac{E,\ n : N,\ \Gamma\ \vdash^{\beta}\ m : N \qquad E,\ n : N,\ \Gamma,\ \forall b B(b),\ B(m)\ \vdash^{\beta'}\ A}{E,\ n : N,\ \Gamma,\ \forall b B(b)\ \vdash^{\alpha}\ A}$$

∀R

$$\frac{E,\ \max(n,i) : N,\ \Gamma\ \vdash^{\beta_i}\ A(i) \ \text{ for every } i \in N}{E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ \forall a A(a)}$$

In addition, there are of course two rules for each propositional symbol, but it is not necessary to list them since they are quite standard. However it should be noted that the rules essentially mimic the truth definition for arithmetic, but as with EA(I;O) they are formalized in the style of **minimal**, not classical, logic. They provide a system within which the inductive proofs of EA(I;O) can be unravelled in a uniform way.

Ordinal Assignment à la Buchholz (1987).

The ordinal bounds on sequents above are intensional, "tree ordinals", generated inductively by: 0 is a tree ordinal; if α is a tree ordinal so is $\alpha+1$; and if $\lambda_0, \lambda_1, \lambda_2, \ldots$ is an ω-sequence of tree ordinals then the function $i \mapsto \lambda_i$ denoted $\lambda = \sup \lambda_i$, is itself also a tree ordinal. Thus tree ordinals carry a specific choice of fundamental sequence to each "limit" encountered in their build-up, and because of this the usual definitions of primitive recursive functions lift easily to tree ordinals. For example addition is defined by:

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

and multiplication is defined by:

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

and exponentiation is defined by:

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

For ω we choose the specific fundamental sequence $\omega = \sup(i + 1)$. For ε_0 we choose the fundamental sequence $\varepsilon_0 = \sup 2_i(\omega^2)$ where $2_i(\beta)$ is defined to be β if $i = 0$ and $2^{2_{i-1}(\beta)}$ if $i > 0$.

Definitions. For each integer i there is a predecessor function given by:

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

and by iterating P_i we obtain, for each non-zero tree ordinal α, the finite set $\alpha[i]$ of all its "i-predecessors" thus:

$$\alpha[i] = \{P_i(\alpha), P_i^2(\alpha), P_i^3(\alpha), \ldots, 0\}.$$

Call a tree ordinal α "structured" if every sub-tree ordinal of the form $\lambda = \sup \lambda_i$ (occurring in the build-up of α) has the property that $\lambda_i \in \lambda[i+1]$ for all i. Then if α is structured, $\alpha[i] \subset \alpha[i + 1]$ for all i, and each of its sub-tree ordinals β appears in one, and all succeeding, $\alpha[i]$. Thus we can think of a structured α as the directed union of its finite sub-orderings $\alpha[i]$. The basic example is $\omega[i] = \{0, 1, \dots, i\}$. All tree ordinals used here will be structured ones.

Ordinal Bounds. The condition on ordinal bounds in the above sequents is to be as follows:

- In rules E1, E2, N1, the bound α is arbitrary.
- In all other rules, the ordinal bounds on the premises are governed by $\beta, \beta', \beta_i \in \alpha[k]$ where k is the fixed parameter.

Lemma 4.1. *(Weakening) If $E,\ n : N,\ \Gamma\ \vdash^{\alpha}\ A$ where $n \leqslant n'$ and $\alpha[k] \subset \alpha'[k]$ then $E,\ n' : N,\ \Gamma\ \vdash^{\alpha'}\ A$.*

4.2. BOUNDING FUNCTIONS

Definition. The bounding functions $B_\alpha(k; n)$ are given by the recursion:

$$B_0(k;n) = n + 1,\ \ B_{\alpha+1}(k;n) = B_\alpha(k; B_\alpha(k;n)),\ \ B_\lambda(k;n) = B_{\lambda_k}(k;n)\ .$$

Lemma 4.2. *$m \leqslant B_\alpha(k;n)$ if and only if, using the N1, N2 rules, we can derive $k : I,\ n : N\ \vdash^{\alpha}\ m : N$.*

Proof. For the "only if" proceed by induction on α. If $\alpha = 0$ the result is immediate by rule N1. If $\alpha > 0$ then it's easy to see that $B_\alpha(k;n) = B_\beta(k; B_\beta(k;n))$ where $\beta = P_k(\alpha)$. So if $m \leqslant B_\alpha(k;n)$ then $m \leqslant B_\beta(k;n')$ where $n' = B_\beta(k;n)$. Therefore by the induction hypothesis we have both $k : I,\ n : N\ \vdash^{\beta}\ n' : N$ and $k : I,\ n' : N\ \vdash^{\beta}\ m : N$. Hence we obtain by the N2 rule $k : I,\ n : N\ \vdash^{\alpha}\ m : N$.

The "if" part again follows by induction on α. The result is immediate if the sequent $k : I,\ n : N\ \vdash^{\alpha}\ m : N$ comes about by the N1 rule. If it comes about by an application of N2 from premises $k : I,\ n : N\ \vdash^{\beta}\ n' : N$ and $k : I,\ n' : N\ \vdash^{\beta'}\ m : N$ then by the induction hypothesis $n' \leqslant B_\beta(k;n)$ and $m \leqslant B_{\beta'}(k;n')$. Thus $m \leqslant B_{\beta'}(k; B_\beta(k;n))$ and this is bounded by $B_\alpha(k;n)$ since $\beta, \beta' \in \alpha[k]$.

Lemma 4.3. *$B_\alpha(k;n) = n + 2^{G(\alpha,k)}$ where $G(\alpha,k)$ is the size of $\alpha[k]$. Furthermore, if $\alpha = 2_i(\omega.d) \prec \varepsilon_0$ then $G(\alpha,k)$ is simply obtained by substituting $k+1$ for ω thus $G(\alpha,k) = 2_i((k+1).d)$.*

Proof. The first part follows directly from the recursive definition of B_α, by noting also that $G(0,k) = 0$, $G(\alpha+1,k) = G(\alpha,k)+1$ and $G(\lambda,k) = G(\lambda_k,k)$.

The second part uses $G(\omega, k) = G(k+1, k) = k+1$ together with the easily verified fact that if $\varphi(\alpha, \beta)$ denotes either $\alpha + \beta$ or $\alpha.\beta$ or α^β then

$$G(\varphi(\alpha, \beta), k) = \varphi(G(\alpha, k), G(\beta, k)) \ .$$

Lemma 4.4. *If α is structured, $\beta \in \alpha[k]$, $k \leqslant k'$ and $n \leqslant n'$ then $B_\beta(k; n) < B_\alpha(k; n) \leqslant B_\alpha(k'; n')$.*

Proof. Immediate from the above.

4.3. EMBEDDING EA(I;O), AND CUT REDUCTION

Lemma 4.5. *(Embedding) If $\bar{E} \vdash f(\vec{x}) \downarrow$ in EA(I;O) there is a fixed number d determined by this derivation, such that: for all inputs $\vec{n}$ of binary length $\leqslant k$, we can derive*

$$E,\ k : I,\ 0 : N\ \vdash^{\omega.d}\ f(\vec{n}) \downarrow$$

in the infinitary system. Furthermore the non-atomic cut-formulas in this derivation are the induction-formulas occurring in the EA(I;O) proof.

Proof. First, by standard "free cut"-elimination arguments we can 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 translating each proof-rule into the corresponding infinitary rule with an appropriate ordinal bound. The non-inductive steps are quite straightforward, and each induction is unravelled in the manner described earlier. For suppose the two premises of the induction have been embedded with ordinal bound $\omega.d$. If the conclusion is $A(t(x_i))$ with t a basic term then, upon substitution of n_i for x_i, we obtain $t(n_i)$ with value $m \leqslant n_i + c$ where c measures the length of the term t (determined by the given EA(I;O) proof). The binary length of m is then $\ell < k + c$ and we thus obtain a derivation of

$$E,\ k : I,\ m : N\ \vdash^{\omega.d+k+c}\ A(m)$$

whose ordinal bound we can manipulate so that

$$E,\ k : I,\ m : N\ \vdash^{\omega.(d+c)+k}\ A(m) \ .$$

Now $m \leqslant c + 2^{k+1} = B_\omega(k; c)$ and $c \leqslant B_{\omega.c}(k; 0)$. So $m \leqslant B_{\omega.(d+c)}(k; 0)$ and by the lemma about B,

$$E,\ k : I,\ 0 : N\ \vdash^{\omega.(d+c)}\ m : N \ .$$

Therefore by applying the N2 rule,

$$E,\ k : I,\ 0 : N\ \vdash^{\omega.(d+c)+k+1}\ A(m)$$

and then since $\omega.(d + c + 1)[k] = \omega.(d + c) + k + 1[k]$ we have

$$E,\ k : I,\ 0 : N\ \vdash^{\omega.(d+c+1)}\ A(m)$$

as required.

Lemma 4.6. *(Cut Reduction) If $\bar{E} \vdash f(\vec{x}) \downarrow$ in EA(I;O) then there is a tree ordinal $\alpha \prec \varepsilon_0$ such that: for all inputs $\vec{n}$ of binary length $\leqslant k$, we can derive*

$$E,\ k : I,\ 0 : N\ \vdash^{\alpha}\ f(\vec{n}) \downarrow$$

by an infinitary derivation in which all the cut formulas are at worst Σ_1.

Proof. The Embedding Lemma above gives a derivation of

$$E,\ k : I,\ 0 : N\ \vdash^{\omega.d}\ f(\vec{n}) \downarrow$$

wherein the cut formulas might be of great logical complexity, depending on the inductions used in the original EA(I;O) proof. However Gentzen-style cut-reduction applies to the infinitary system (whereas it doesn't apply directly to EA(I;O)). Thus the "sizes" of cut formulas can be successively reduced, one level at a time, so that all that remains are cut formulas of shape Σ_1. But each time the cuts are reduced in size, there will be an exponential increase in the ordinal bound of the resulting derivation. Therefore, if the maximum size of induction formula used in the EA(I;O) proof is r, then we obtain the desired derivation

$$E,\ k : I,\ 0 : N\ \vdash^{\alpha}\ f(\vec{n}) \downarrow$$

with at most Σ_1 cuts and ordinal bound $\alpha \prec 2_r(\omega.d) \prec \varepsilon_0$.

We now describe briefly how the cut reduction process works. First define the *size* of a formula to be 0 if it is a conjunction of atoms, 1 if it is Σ_1, and $r + 1$ otherwise, where r is the maximum size of its subformulas. Then define the *cut rank* of a derivation to be the maximum size of all cut formulas appearing in it. There are two steps to the process:

Step (i) Suppose we have two derivations in the infinitary system:

$$E,\ k : I,\ n : N,\ \Gamma\ \vdash^{\beta}\ C$$

and

$$E,\ k : I,\ n : N,\ \Gamma,\ C\ \vdash^{\gamma}\ A$$

both with cut rank $\leqslant r$, and where C is of size $r+1$. Suppose also that $\beta, \gamma \in \alpha[k]$ for some fixed α. Obviously we could apply the Cut Rule, with cut formula C, to derive

$$E,\ k:I,\ n:N,\ \Gamma\ \vdash^{\alpha}\ A$$

but the cut rank would then be $r+1$. The point is we can do better than this, and replace this cut by another one of rank r. However the ordinal bound will increase to either $\beta+\gamma$ or $\gamma+\beta$, depending on the form of cut formula C.

As an example, suppose C is of the form $\forall a D(a)$. Then by induction on γ we show that we can remove C and derive

$$E,\ k:I,\ n:N,\ \Gamma\ \vdash^{\beta+\gamma}\ A$$

with cut rank r.

For suppose the derivation with bound γ arises by an application of the $\forall$L rule on formula C. Then, suppressing E, $k:I$ and Γ in the antecedent, we have premises $n:N,\ \vdash^{\gamma_0}\ m:N$ and $n:N,\ C,\ D(m)\ \vdash^{\gamma_1}\ A$, and by applying the induction hypothesis to remove C from the second of these, we obtain $n:N,\ D(m)\ \vdash^{\beta+\gamma_1}\ A$. Also, by inverting the derivation with bound β we obtain $\max(n,m):N\ \vdash^{\beta_m}\ D(m)$ for some $\beta_m \in \beta[k]$ and then by the N2 rule, $n:N\ \vdash^{\beta+\max(\gamma_0,\gamma_1)}\ D(m)$ provided that γ_0 and γ_1 are not both zero. We can then derive $n:N\ \vdash^{\beta+\gamma}\ A$ by a cut on the formula $D(m)$ and the cut rank will be just r as required, since the size of D is one less than the size of C. If, on the other hand, $\gamma_0=\gamma_1=0$ then $n:N,\ C,\ D(m)\ \vdash^{0}\ A$ is an axiom, and therefore so is $n:N,\ D(m)\ \vdash^{\beta}\ A$ since C is not atomic and can thus be deleted from any axiom. Hence using N2 to give $n:N\ \vdash^{\beta}\ D(m)$ we again derive $n:N\ \vdash^{\beta+\gamma}\ A$ with cut rank r by a cut on $D(m)$ as before.

If the derivation with bound γ arises by an application of any other rule, then the C will appear in the premises as an inactive “side formula” which can be removed simply by applying the induction hypothesis, and adding β to their ordinal bounds. Then that rule can be re-applied to give the desired result. This completes our example of Step (i). The other cases are handled in a similar way, though sometimes the induction must be on β instead of γ, producing an ordinal bound $\gamma+\beta$.

Step (ii) Transform any derivation with ordinal bound α and cut rank $r+1$ into a derivation of the same sequent, but with cut rank $\leqslant r$ and ordinal bound 2^α, as follows using Step (i):

Proceed by induction on α with cases according to the last rule applied. If this last rule is a cut with cut formula C of size $r+1$ then the premises (with all "side formulas" suppressed) will be of the form $\vdash^\beta C$ and $C \vdash^\gamma A$ with $\beta, \gamma \in \alpha[k]$. By increasing the smaller of these bounds if necessary, using weakening, we may assume that they are both the same. Applying the induction hypothesis to each premise, one obtains derivations $\vdash^{2^\beta} C$ and $C \vdash^{2^\beta} A$ both now with cut rank $\leqslant r$. Then Step (i) applies to give a derivation of A with cut rank r and ordinal bound $2^\beta + 2^\beta$, which is either equal to 2^α or can be weakened to it.

If the last rule applied is anything other than a cut of rank $r+1$, then simply apply the induction hypothesis to the premises in order to reduce their cut rank, and then re-apply that last rule, noting that if $\beta \in \alpha[k]$ then $2^\beta \in 2^\alpha[k]$. This completes the proof of the Cut Reduction Lemma.

4.4. COMPLEXITY BOUNDS

Notation. We signify that an infinitary derivation involves only Σ_1 cut formulas C, by attaching a subscript 1 to the proof-gate thus: $E, n : N, \Gamma \vdash_1^\alpha A$. If all cut formulas are atomic equations (or possibly conjunctions of them) we attach a subscript 0 instead.

Lemma 4.7. *(Bounding) Let Γ, A consist of (conjunctions of) atomic formulas only.*

1. *If $E,\ k : I,\ n : N,\ \Gamma\ \vdash_1^\alpha\ m : N$ then $m \leqslant B_\alpha(k; n)$.*
2. *If $E,\ k : I,\ n : N,\ \Gamma\ \vdash_1^\alpha\ \exists \vec{a} A(\vec{a})$ then there are numbers $\vec{m} \leqslant B_\alpha(k; n)$ such that $E,\ k : I,\ n : N,\ \Gamma\ \vdash_0^{2.\alpha}\ A(\vec{m})$.*

Proof. Both parts are dealt with simultaneously by induction on α. Since only Σ_1 cuts are involved, it is only the E, N, Cut and $\exists$ rules which come into play. The E rules require no action at all, and if N1 is applied (in case 1) then we have immediately $m \leqslant n+1 \leqslant B_\alpha(k; n)$.

(N2) Suppose the sequent in 1 or 2 comes about by the N2 rule. Then one of the premises is of exactly the same form, but with n replaced by n' and α replaced by a $\beta' \in \alpha[k]$. Therefore by the induction hypothesis, and re-application of N2 to reduce $n' : N$ to $n : N$, the desired result follows since $2.\beta' \in 2.\alpha[k]$. But the bound on existential witnesses that we have at this stage is $B_{\beta'}(k; n')$. However the other premise is a sequent of the form 1 with m replaced by n' and α replaced by a $\beta \in \alpha[k]$, so by the induction hypothesis $n' \leqslant B_\beta(k;n)$. Thus, substituting $B_\beta(k;n)$ for n' we obtain the required bound

$$B_{\beta'}(k;n') \leqslant B_{\beta'}(k;B_\beta(k;n)) \leqslant B_\alpha(k;n)$$

using the definition of B_α and its majorization properties.

($\exists$) The $\exists$L rule does not apply. If the sequent in 2 comes about by an application of $\exists$R, introducing the outermost quantifier in $\exists\vec{a}A(\vec{a})$, then the induction hypothesis applied to the first premise produces a witness $m \leqslant B_\beta(k;n)$. The induction hypothesis applied to the second premise yields the desired result, but with bound $B_{\beta'}(k;n)$ on witnesses for the remaining quantifiers. However $B_{\beta'}(k;n)$ and $B_\beta(k;n)$ are both less than the required bound $B_\alpha(k;n)$ since $\beta, \beta' \in \alpha[k]$.

(Cut) Finally, suppose the sequent in 1 or 2 arises by an application of Cut with cut formula $C \equiv \exists\vec{c}D(\vec{c})$. By applying the induction hypothesis to the first premise we obtain witnesses $\vec{\ell}$ no larger than $B_\beta(k;n)$ such that

$$E,\ k:I,\ n:N,\ \Gamma\ \vdash_0^{2.\beta}\ D(\vec{\ell})$$

and by "inverting" the $\exists\vec{c}$ in the second premise we obtain

$$E,\ k:I,\ \max(n,\vec{\ell}):N,\ D(\vec{\ell}),\ \Gamma\ \vdash_1^{\beta'}\ F$$

where $F \equiv m:N$ or $F \equiv \exists\vec{a}A(\vec{a})$. Applying the induction hypothesis to this last sequent, in the case $F \equiv \exists\vec{a}A(\vec{a})$, we obtain numerical witnesses $\vec{m}$ bounded by:

$$B_{\beta'}(k;\max(n,\vec{\ell})) \leqslant B_{\beta'}(k;B_\beta(k;n)) \leqslant B_\alpha(k;n)$$

and

$$E,\ k:I,\ \max(n,\vec{\ell}):N,\ D(\vec{\ell}),\ \Gamma\ \vdash_0^{2.\beta'}\ A(\vec{m})\ .$$

Since $\vec{\ell} \leqslant B_\beta(k;n) \leqslant B_{2.\beta}(k;n)$ we have, by a lemma above,

$$E,\ k:I,\ n:N,\ D(\vec{\ell}),\ \Gamma\ \vdash_0^{2.\beta}\ \max(n,\vec{\ell}):N$$

and so by the N2 rule, with $\gamma = \max(\beta,\beta')$, we obtain

$$E,\ k:I,\ n:N,\ D(\vec{\ell}),\ \Gamma\ \vdash_0^{2.\gamma+1}\ A(\vec{m})\ .$$

Then by a cut on $D(\vec{\ell})$, with a weakening, we obtain the required

$$E,\ k:I,\ n:N,;\Gamma\ \vdash_0^{2.\alpha}\ A(\vec{m})$$

since $\gamma \in \alpha[k]$ implies $2.\gamma+1 \in 2.\alpha[k]$. The other case $F \equiv m:N$ is much simpler. This completes the proof.

Theorem 4.8. *(Complexity) Suppose f is defined by a system of equations E and $\bar{E} \vdash f(\vec{x}) \downarrow$ in EA(I;O) with induction formulas of size at most r. Then there is an $\alpha = 2_{r-1}(\omega.d)$ such that: for all inputs $\vec{n}$ of binary length at most k we have*

$$E,\ k : I,\ 0 : N\ \vdash_0^{2.\alpha}\ f(\vec{n}) \simeq m \quad \textit{where}\ \ m \leqslant 2_r((k+1).d)\ .$$

This is a computation of $f(\vec{n})$ from E, and the number of computation steps (nodes in the binary branching tree) is less than $4^{2_{r-1}((k+1).d)}$.

Proof. First apply the Embedding Lemma to obtain a number d such that for all inputs $\vec{n}$ of binary length $\leqslant k$,

$$E,\ k : I,\ 0 : N\ \vdash^{\omega.d}\ f(\vec{n}) \downarrow$$

with cut rank r. Then apply Cut Reduction $r-1$ times, to bring the cut rank down to the Σ_1 level. This gives $\alpha = 2_{r-1}(\omega.d)$ such that

$$E,\ k : I,\ 0 : N\ \vdash_1^{\alpha}\ f(\vec{n}) \downarrow\ .$$

Then apply the Bounding Lemma to obtain $m \leqslant B_\alpha(k;0) = 2^{G(\alpha,k)} = 2_r((k+1).d)$ such that

$$E,\ k : I,\ 0 : N\ \vdash_0^{2.\alpha}\ f(\vec{n}) \simeq m\ .$$

This derivation uses only the E axioms, the N rules and equational cuts, so it is a computation in the equation calculus. Since all the ordinal bounds belong to $2.\alpha[k]$, the height of the derivation tree is no greater than $G(2.\alpha, k)$ and the number of nodes (or computation steps) is therefore $\leqslant 2^{G(2.\alpha,k)} = 4^{G(\alpha,k)} = 4^{2_{r-1}((k+1).d)}$.

Theorem 4.9. *The functions provably recursive in EA(I;O) are exactly the elementary E^3 functions. The functions provably recursive in the Σ_1 inductive fragment of EA(I;O) are exactly the subelementary (E^2) functions. The functions provably recursive in the "level-2" inductive fragment of EA(I;O) are exactly those computable in a number of steps bounded by an exponential function of their inputs.*

Proof. By the above, if f is provably recursive in EA(I;O) then it is computable in a number of steps bounded by a finitely iterated exponential function of its inputs. This means it is elementary.

If f is provably recursive in the Σ_1 inductive fragment then we can take $r = 1$ in the above. So f is computable in a number of steps bounded by $4^{(k+1).d}$. But k is the maximum binary length of the inputs $\vec{n}$, so this bound is just a polynomial in $\max \vec{n}$. Therefore by e.g. Rödding (1967), f is subelementary.

If f is provably recursive using "level-2" inductions then, taking $r = 2$, we obtain a computational bound $4^{2_1((k+1).d)}$ which is $\leqslant 2^{p(\vec{n})}$ for some polynomial p. (For a more detailed treatment of this case see Ostrin and Wainer (2001)).

4.5. POLYTIME FUNCTIONS

If the theory EA(I;O) were instead 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 $P(0) = 0$, $P(S_0 a) = P(S_1 a) = a$, 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 it only takes $\log n$ many induction steps to "climb up" to $A(n)$. Thus a similar analysis to that given here, but with n replaced by $\log n$, would show that the functions provably recursive in the Σ_1 inductive fragment of this binary theory are just those with complexity bounds polynomial in the binary lengths of their inputs, i.e. PTIME, see Leivant (1995).

5. 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. Buchholz, *An independence result for* $\Pi^1_1 - CA + BI$, Annals of Pure and Applied Logic Vol. 23 (1987) pp. 131 - 155.

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

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

[5] 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.

[6] 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.

[7] W.G. Handley and S.S. Wainer, *Complexity of primitive recursion*, in U. Berger and H. Schwichtenberg (Eds) Computational Logic, Springer-Verlag ASI Series F (1999), pp. 28.

[8] G. Kreisel, *On the interpretation of non-finitist proofs, Parts I, II*, Journal of Symbolic Logic Vol. 16 (1951) pp. 241 - 267, Vol. 17 (1952) pp. 43 - 58.

[9] D. Leivant, *A foundational delineation of poly-time*, Information and Computation Vol. 110 (1994) pp. 391 - 420.

[10] 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.

[11] G.E. Ostrin, *Proof theories of low subrecursive classes*, Ph.D. Leeds (1999).

[12] G.E. Ostrin and S.S. Wainer, *Elementary arithmetic*, Leeds preprint (2001).

[13] 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.

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

[15] A. Weiermann, *How to characterize provably total functions by local predicativity*, Journal of Symbolic Logic Vol. 61 (1996) pp. 52 - 69.