A New "Feasible" Arithmetic

S. Bellantoni* Martin Hofmann†

January 19, 2000

Abstract

A classical quantified modal logic is used to define a "feasible" arithmetic $\mathcal{A}_2^1$ whose provable functions are exactly the polynomial-time computable functions. Informally, one understands $\Box\,\alpha$ as "α is feasibly demonstrable".

$\mathcal{A}_2^1$ differs from a system $\mathcal{A}_2$ that is as powerful as Peano Arithmetic only by the restriction of induction to ontic (i.e. $\Box$-free) formulas. Thus, $\mathcal{A}_2^1$ is defined without any reference to bounding terms, and admitting induction over formulas having arbitrarily many alternations of unbounded quantifiers. The system also uses only a very small set of initial functions.

To obtain the characterization, one extends the Curry-Howard isomorphism to include modal operations. This leads to a realizability translation based on recent results in higher-type ramified recursion. The fact that induction formulas are not restricted in their logical complexity, allows one to use the Friedman A translation directly.

The development also leads us to propose a new Frege rule, the "Modal Extension" rule: if $\vdash \alpha$ then $\vdash A \leftrightarrow \alpha$ for new symbol A.

1 Introduction

In recent years considerable effort has been dedicated to defining and exploring logical and arithmetic systems in which the reasoning involved is not only constructive but "feasibly constructive". In most cases this is understood to mean that the constructive content of the proof – however that might be defined – is polynomial time computable. In any case, an important litmus test for feasiblity of a first order arithmetic is that the functions for which a suitable convergence statement can be proved, are at most the polynomial time computable functions. For this test to be of any significance, of course, the system must have enough expressive power to discuss a wider class of functions, say all the primitive recursive functions. Buss's system $\mathcal{S}_2^1$ of bounded arithmetic [7] is fundamental to this subject; see Krajicek [21] for a discussion of related work.

*Department of Computer Science, University of Toronto. The assistance of the Fields Institute for Research in Mathematical Sciences is gratefully acknowledged.

†Laboratory for Foundations of Computer Science, University of Edinburgh.

At the same time, researchers in recursion theory have developed systems in which computational complexity is controlled by type information rather than by explicit resource bounds [29], [3], [23], [17], [4]. Each of the various types ι, $\Box\iota$, $\Box\Box\iota$, ... in a ramified system is a different intension for the same extensional values. Typically, one may recurse on a value that is comprehended through a type $\Box\iota$ reference, while one may only access a few low-order bits from a value referred to by a type ι variable. A related area of work is the "descriptive complexity" characterizations such as those of Immerman [19] and others – again, no explicit resource bounds appear.

The two lines of research collide due to the well known functional translation based on the Curry-Howard isomorphism between formulas and types. Logical deductions are like derivations of terms in a lambda calculus: modus ponens corresponds to functional application, and induction corresponds to recursion. The type of the recursion interpreting an induction instance is the type of the formula in the induction — if the formula contains "$\supset$" then it is of higher type. Thus, to generate a feasible logic through research in ramified recursion, one must solve the problem of restricting higher type recursion schemes (e.g. in Gödel's system T) so that only polytime functions are definable. This has recently been achieved [4], [16], [17], [18]. In this work we use a system $T^{\Box}$ which can be translated into one of these [17]. $T^{\Box}$ is a lambda calculus that restricts recursion in higher types by a ramification system defined by a modal operator. A ramification step in this system refers simultaneously to two forms of knowledge: that which allows a higher-type object to be used non-linearly, and that which allows a ground-type object to be used as the pattern for recursion.

Interesting discussions of realizability, intuitionism, knowledge, and difficulties in the application of [3] to arithmetic, were given by Nelson in [26] and [27]. These have guided our reasoning. Nelson also applied predicative concepts to define a constructive arithmetic in [25].

Shapiro [28] used modal logic to define an "epistemic arithmetic" EA. He proved that it is as strong as HA under a Gödel-style mapping, while Goodman [15] used an infinitary cut elimination argument to show its conservativity over HA under a similar mapping. In contrast, we use Friedman's "A" translation to prove a form of conservativity, we do not use a Gödel mapping, and we do not require cut elimination.

One precursor to this work in ramified arithmetic is the "intrinsic theory" of Leivant [22]. There, ramification predicates $N_0, N_1, \ldots$ are used to delineate the tiers. Provability is obtained for the elementary-time computable functions when induction is allowed over formulas referring to tier 0 (i.e. N_0) only. Additional restrictions on the quantifiers in the induction formula lead to provability of polytime functions in Leivant's system.

Another precursor is the ramified arithmetic and corresponding model theory by Bellantoni [2]. The arithmetic in [2] demonstrated the possibility of admitting arbitrarily many alternations of unbounded quantifiers in induction while still obtaining "polytime provability". However, the actual system defined there was inadequate as a working logic e.g. it was awkwardly defined and not closed under modus ponens.

For background in ramified recursion, see the systems of [3], [5], [4], [16] [17], [18], and Leivant [23], [24] and further references cited there.

Although it is carried out more in the tradition of ramified recursion than linear logic, this work has obvious and important connections to linear logic. In linear logics, one introduces special operators to control and track the usage of formulas. See Abramsky [1] and Girard, Scedrov, and Scott [13] for prototypical discussions in linear logic. In the polytime system in [13], one attaches explicit polynomials to the occurrences of the modal operator in order to bound the complexity of a proof. The "light linear logic" of Girard [14] includes a polytime system using new connectives that have modal features. Although the work is relevant, we do not attempt a direct detailed comparison here because of differences in the defining frameworks.

The main result is stated in Corollary 5.6 below.

2 Arithmetic $\mathcal{A}_2^1$

Now we define an arithmetic $\mathcal{A}_2^1$ based on a classical modal logic. Contraction will be admitted for modal formulas; that is, nonlinear usage will be allowed for formulas of the form $\Box\,\phi$. Induction will be admitted only for ontic formulas, i.e. those whose type does not contain $\Box$. Hence the induction hypothesis can be used at most once. As well, property $\Box\,\iota(x)$ will be required of the induction variable.

Following Leivant [22] we will admit arbitrary equational programs such as $\forall x\vec{y}.fx\vec{y} = hx\vec{y}(fx\vec{y})$. No ramification type conditions are imposed; this allows definition of all partial recursive functions. Therefore the system is not restricted a priori to predicatively definable functions.

2.1 Language

Terms are constructed as usual from a given first-order signature containing the constant ε (the empty string) and unary function symbols $\mathbf{s}_0$, $\mathbf{s}_1$ (binary string successors). One also admits into the language countably many arbitrary function symbols, of all arities.

Atomic formulas are $s = t$ where s, t are terms and $\iota(t)$ where t is a term.

If ϕ, ψ are formulas so are $\neg\phi, \phi \supset \psi, \phi \wedge \psi, \forall x.\phi$, and $\Box\,\phi$. By convention, $\Box$ and $\neg$ bind stronger than $\wedge$ and $\wedge$ binds stronger than $\supset$. The formula $\forall x.\iota(x) \supset \phi$ is abbreviated by $\forall x^{\iota}.\phi$, and $\forall x.\Box\,\iota(x) \supset \phi$ by $\forall x^{\Box\,\iota}.\phi$. We may write $\mathbf{s}_i x$ for $\mathbf{s}_i(x)$.

A formula is *ontic* if it does not contain the modality $\Box$. A formula is *modal* if it is of the form $\Box\,\phi$. By $\iota(x)$ one understands that x is a number; by $\Box\,\iota(x)$, that x is a "feasible" number.

2.2 System $\mathcal{A}_2^1$

$\mathcal{A}_2^1$ is a classical first-order modal arithmetic with contraction for modal formulas and $\iota(t)$ only; with modal axioms, and restricted induction. The axioms and rules are as follows.
(1) Classical first-order logic with equality but without contraction:

(K)	$\alpha \supset \beta \supset \alpha$
(B)	$(\beta \supset \gamma) \supset (\alpha \supset \beta) \supset (\alpha \supset \gamma)$
(C)	$(\alpha \supset \beta \supset \gamma) \supset \beta \supset \alpha \supset \gamma$
($\wedge$I)	$\alpha \supset \beta \supset (\alpha \wedge \beta)$
($\wedge$E)	$(\alpha \supset \beta \supset \gamma) \supset (\alpha \wedge \beta) \supset \gamma,$
(Class)	$(\neg\alpha \supset \neg\beta) \supset \beta \supset \alpha$
(MP)	From α and $\alpha \supset \beta$, deduce β.
($\forall$I)	From $\alpha \supset \beta$, deduce $\alpha \supset \forall x.\beta$, provided x is not free in α.
($\forall$E)	$(\forall x\, \alpha) \supset (\alpha[r/x])$, provided r is a term free for x in α.
(Refl)	$x = x$
(Sym)	$x = y \supset y = x$
(Trans)	$x = y \supset y = z \supset x = z$
($\text{Sub}_=$)	$x = y \supset r[x/z] = r[y/z]$
(Sub_ι)	$x = y \supset \iota(x) \supset \iota(y)$

(2) Modal axiom and rule schemes:

($\Box$I)	From α deduce $\Box\, \alpha$
($\Box$E)	$\Box\, \alpha \supset \alpha$
($\Box\,\Box$)	$\Box\, \alpha \supset \Box\,\Box\, \alpha$
($\Box \supset$)	$\Box(\alpha \supset \beta) \supset \Box\, \alpha \supset \Box\, \beta$

(3) Contraction for modal formulas and for $\iota(t)$:

($\Box$Contr)	$\Box\, \alpha \supset \Box\, \alpha \wedge \Box\, \alpha$
(ιContr)	$\iota(x) \supset \iota(x) \wedge \iota(x)$

(4) Axioms of generation, separation, and surjectivity in ι:

(Gen_ε)	$\iota(\varepsilon)$
(Gen_0)	$\forall x.\iota(x) \supset \iota(\mathbf{s}_0(x))$
(Gen_1)	$\forall x.\iota(x) \supset \iota(\mathbf{s}_1(x))$
(Sep_ε)	$\forall x^\iota.\neg(\varepsilon = \mathbf{s}_0 x) \wedge \neg(\varepsilon = \mathbf{s}_1 x)$
(Sep_0)	$\forall x^\iota, \forall y^\iota.\mathbf{s}_0 x = \mathbf{s}_0 y \supset x = y$
(Sep_1)	$\forall x^\iota, \forall y^\iota.\mathbf{s}_1 x = \mathbf{s}_1 y \supset x = y$
(Sep_{01})	$\forall x^\iota \neg \mathbf{s}_0 x = \mathbf{s}_1 x$
(Surj)	$\forall x^\iota.\neg\forall y^\iota\ (\neg x = \varepsilon \wedge \neg x = \mathbf{s}_0 y \wedge \neg x = \mathbf{s}_1 y)$

(5) Induction over ontic (i.e. $\Box$-free) formulas α:

$$\text{(IND)} \qquad \alpha(\varepsilon) \supset \Box(\forall x^{\Box\iota}.\alpha(x) \supset \alpha(\mathbf{s}_0(x))) \supset \Box(\forall x^{\Box\iota}.\alpha(x) \supset \alpha(\mathbf{s}_1(x))) \supset \forall x^{\Box\iota}.\alpha(x)$$

If Γ is a (possibly infinite) multiset of formulas and γ is a formula then we write $\Gamma \vdash \gamma$ to mean that $\alpha_1 \wedge \cdots \wedge \alpha_n \supset \gamma$ is derivable for some finite sub-multiset $\{\alpha_1, \dots, \alpha_n\}$ of Γ.

One has the following derived rules:

Lemma 2.1

1. *Linear deduction: If* $\Gamma, \alpha \vdash \gamma$ *then* $\Gamma \vdash \alpha \supset \gamma$*;*
2. *Generalisation: If* $\Gamma \vdash \alpha$ *and* x *is not free in* Γ *then* $\Gamma \vdash \forall x.\alpha$*;*
3. *Necessitation: If* $\Gamma \vdash \alpha$ *and all formulas in* Γ *are modal then* $\Gamma \vdash \Box\,\alpha$*.*
4. *Contraction: If* $\Gamma, \alpha, \alpha \vdash \gamma$ *and* α *is* $\iota(s)$ *or* $\Box\,\alpha'$ *then* $\Gamma, \alpha \vdash \gamma$
5. $\vdash \Box\,\alpha \wedge \Box\,\beta \supset \Box(\alpha \wedge \beta)$
6. $\vdash \Box(\forall x^{\iota}.\alpha) \supset \forall x^{\Box\iota}.\Box\,\alpha$

Proof. Linear deduction is by induction on derivations similar to the usual proof of deduction lemma for Hilbert style proof systems. The others are direct. $\Box$

A *valuation*, η, is an assignment of a function over the finite binary strings $\{0,1\}^*$ for each function symbol, together with an assignment of a finite binary string for each variable; such that $\eta(\varepsilon)$ is the empty string, and $(\eta(\mathbf{s}_0))(w) = w0$, and $(\eta(\mathbf{s}_1))(w) = w1$. This defines $\eta(t) \in \{0,1\}^*$ for each $\mathcal{A}^1_2$ term t. Given a valuation, η, satisfaction $\eta \models \alpha$ of formula α is defined as for ordinary classical first-order logic with equality, judging that $\Box\,\alpha$ is true iff α is true, and that $\iota(t)$ is always true. In this way, for the purposes of this paper attention has been restricted to valuations over a one-world standard model. An analogous simplification will later be made in the denotations of functional λ terms.

2.3 Equational specifications

An *equational specification* is a finite conjunction of closed universally quantified equations. If Φ is an equational specification then a convergence statement is a formula of the form $\Box\,\Phi \supset \forall \vec{x}^{\Box\iota}\iota(f(\vec{x}))$ where f is a function symbol. We say that Φ defines $\mathbf{f}$ using f if $\{\eta(f) : \ \eta \models \Phi\} = \{\mathbf{f}\}$.

For example, the following defines an exponentially growing function using f (with helper function f^*):

$$\begin{aligned}
&\forall a.f^*(\varepsilon, a) = \mathbf{s}_1 a\\
&\forall x, \forall a.f^*(\mathbf{s}_0 x, a) = f^*(x, f^*(x, a))\\
&\forall x, \forall a.f^*(\mathbf{s}_1 x, a) = f^*(x, f^*(x, a))\\
&\forall x.f(x) = f^*(x, \varepsilon)
\end{aligned}$$

The same function could also be defined (with helper function +) by:

$$\begin{aligned}
&\forall a.\varepsilon + a = a\\
&\forall x, \forall a.\ (\mathbf{s}_0 x) + a = \mathbf{s}_0(x + a)\\
&\forall x, \forall a.\ (\mathbf{s}_1 x) + a = \mathbf{s}_1(x + a)\\
&f(\varepsilon) = \mathbf{s}_1\varepsilon\\
&\forall x. f(\mathbf{s}_0 x) = f(x) + f(x)\\
&\forall x. f(\mathbf{s}_1 x) = f(x) + f(x)
\end{aligned}$$

On the other hand, one may also define a polynomially growing function using $\times$ by:

$$\begin{aligned}
&\forall x.\ x \times \varepsilon = \varepsilon\\
&\forall x, \forall y.\ x \times \mathbf{s}_0 y = x + (x \times y)\\
&\forall x, \forall y.\ x \times \mathbf{s}_1 y = x + (x \times y)
\end{aligned}$$

3 Functional calculus

To carry out the realizability translation we use a simply-typed modal/linear lambda calculus $T^{\Box}$ which is a simplified version of the system [17]; see also [4].

The types of $T^{\Box}$ are given by the grammar $\tau ::= \iota \mid \tau_1 \multimap \tau_2 \mid \tau_1 \otimes \tau_2 \mid \Box\tau$ The constants with their types are:

$$\begin{aligned}
&\varepsilon : \iota\\
&\mathbf{s}_0 : \iota \multimap \iota\\
&\mathbf{s}_1 : \iota \multimap \iota\\
&\mathbf{p} : \iota \multimap \iota\\
&\mathbf{d}_\tau : \tau \multimap (\iota \multimap \tau) \multimap (\iota \multimap \tau) \multimap \iota \multimap \tau\\
&\mathcal{R}_\tau : \tau \multimap \Box(\Box\,\iota \multimap \tau \multimap \tau) \multimap \Box\,\iota \multimap \tau,\ \text{ where } \tau \text{ is } \Box\text{-free}\\
&\langle \cdot, \cdot \rangle_{\sigma,\tau} : \sigma \multimap \tau \multimap \sigma \otimes \tau\\
&\Pi_{\sigma,\tau,\rho} : (\sigma \multimap \tau \multimap \rho) \multimap \sigma \otimes \tau \multimap \rho\\
&\kappa_\tau : \Box\,\tau \multimap \tau\\
&\mathbf{W} : \iota \multimap \iota \otimes \iota\\
&\mathbf{B}_\tau : \Box\,\tau \multimap \Box\,\tau \otimes \Box\,\tau\\
&\Box_\tau : \tau \multimap \Box\,\tau
\end{aligned}$$

A *context* (or type assignment) Θ is a finite partial function from variables to types. A context Θ is *modal* if $\Theta(x)$ is of the form $\Box\,\tau$ for each $x \in \mathit{dom}(\Theta)$.

The typing judgment $\Theta \vdash e : \tau$ (read e is a term given type τ by context Θ) is inductively defined by the following rules.

$$\frac{c : \tau \text{ a constant other than any } \Box_\sigma}{\Theta \vdash c : \tau} \qquad (\textsc{Const})$$

$$\frac{x \in \mathit{dom}(\Theta)}{\Theta \vdash x : \Theta(x)} \qquad (\textsc{Var})$$

$$\frac{\Theta, x:\sigma \vdash e:\tau}{\Theta \vdash \lambda x^{\sigma}.e : \sigma \multimap \tau} \quad (\textsc{Lam})$$

$$\frac{\Theta_1 \vdash e_1 : \sigma \multimap \tau \qquad \Theta_2 \vdash e_2 : \sigma \qquad dom(\Theta_1) \cap dom(\Theta_2) = \emptyset}{\Theta_1, \Theta_2 \vdash e_1 e_2 : \tau} \quad (\textsc{App})$$

$$\frac{\Theta \vdash e:\tau \qquad \Theta \text{ modal}}{\Theta \vdash \Box_{\tau}(e) : \Box\,\tau} \quad (\textsc{Nec})$$

By a *closed term* $e:\tau$ we mean a closed term e together with a typing judgment concluding $\vdash e:\tau$. To each type τ we associate a set $[\![\tau]\!]$ by

$$\begin{aligned}
&[\![\iota]\!] = \{0,1\}^* \\
&[\![\tau_1 \multimap \tau_2]\!] = [\![\tau_1]\!] \to [\![\tau_2]\!] \\
&[\![\tau_1 \otimes \tau_2]\!] = [\![\tau_1]\!] \times [\![\tau_2]\!] \\
&[\![\Box\,\tau]\!] = [\![\tau]\!]
\end{aligned}$$

To each constant $c:\tau$ an element $[\![c]\!] \in [\![\tau]\!]$ is associated by

$$\begin{aligned}
&[\![\varepsilon]\!] = \varepsilon \\
&[\![\mathbf{s}_0]\!](w) = w0 \\
&[\![\mathbf{s}_1]\!](w) = w1 \\
&[\![\mathbf{p}]\!](\varepsilon) = \varepsilon \\
&[\![\mathbf{p}]\!](w0) = w \\
&[\![\mathbf{p}]\!](w1) = w \\
&[\![\mathbf{d}_{\tau}]\!](g, h_0, h_1, \varepsilon) = g \\
&[\![\mathbf{d}_{\tau}]\!](g, h_0, h_1, w0) = h_0(w) \\
&[\![\mathbf{d}_{\tau}]\!](g, h_0, h_1, w1) = h_1(w) \\
&[\![\mathcal{R}_{\tau}]\!](g, h, \varepsilon) = g \\
&[\![\mathcal{R}_{\tau}]\!](g, h, w0) = h(w0, [\![\mathcal{R}_{\tau}]\!](g, h, w)) \\
&[\![\mathcal{R}_{\tau}]\!](g, h, w1) = h(w1, [\![\mathcal{R}_{\tau}]\!](g, h, w)) \\
&[\![\langle\cdot,\cdot\rangle_{\sigma,\tau}]\!](x, y) = [x, y] \\
&[\![\Pi_{\sigma,\tau,\rho}]\!](f, [x, y]) = f(x, y) \\
&[\![\kappa_{\tau}]\!](x) = x \\
&[\![\mathbf{W}]\!](x) = [x, x] \\
&[\![\mathbf{B}]\!](x) = [x, x]
\end{aligned}$$

Using this definition we obtain $[\![t]\!]$ for all closed terms t.

The translation into the system in [17] interprets ι as N with the understanding that only numbers greater than 0 are taken on by the translation to avoid the fact that $\mathbf{s}_0$ is non-injective in that system. Accordingly, ε is interpreted as the constant 1; the translations of $\mathbf{d}$ and $\mathcal{R}$ must be appropriately changed to provide default values for the values not taken on by the translation. Alternatively, one can apply the semantic technique described in [17] directly to the present system. Whichever route is taken, it follows that whenever $e : \Box\,\iota \multimap \Box\,\iota \multimap \cdots \multimap \Box\,\iota \multimap \iota$ is a closed term then $[\![e]\!]$ is a polynomial time computable function.

4 Realisability

Although our end result holds for $\mathcal{A}_2^1$, the functional realizability translation requires an intuitionistic logic. Let $\mathcal{I}_2^1$ be the logic defined in the same way as $\mathcal{A}_2^1$ but replacing classical negation (Class) by intuitionistic negation (Int): $\neg\alpha \supset \alpha \supset \beta$. The terms and formulas of $\mathcal{I}_2^1$ are the same as those of $\mathcal{A}_2^1$.

We will now interpret proofs in the intuitionistic system by terms in the functional system.

To each formula ϕ a type $\mathsf{t}(\phi)$ is assigned by

$$
\begin{aligned}
&\mathsf{t}(s{=}t) = \iota \\
&\mathsf{t}(\iota(t)) = \iota \\
&\mathsf{t}(\neg\phi) = \iota \\
&\mathsf{t}(\phi \supset \psi) = \mathsf{t}(\phi){\multimap}\mathsf{t}(\psi) \\
&\mathsf{t}(\phi \wedge \psi) = \mathsf{t}(\phi){\otimes}\mathsf{t}(\psi) \\
&\mathsf{t}(\forall x.\phi) = \mathsf{t}(\phi) \\
&\mathsf{t}(\Box\,\phi) = \Box\,\mathsf{t}(\phi)
\end{aligned}
$$

If $w \in \{0,1\}^*$, then write $\eta[x \mapsto w]$ for the valuation which maps x to w and behaves like η otherwise. When e is a term of type $\tau_1{\otimes}\tau_2$, introduce the abbreviations $\pi_1 e$ for $\Pi_{\tau_1,\tau_2,\tau_1}(\lambda x^{\tau_1}.\lambda y^{\tau_2}.x)(e)$ and $\pi_2 e$ for $\Pi_{\tau_1,\tau_2,\tau_1}(\lambda x^{\tau_1}.\lambda y^{\tau_2}.y)(e)$. We have $\pi_i e : \tau_i$.

Let ϕ be a formula, η a valuation, and let e be a closed term of type $\mathsf{t}(\phi)$. The relation $e\,\mathbf{rz}_\eta\,\phi$ read e realises ϕ under valuation η is defined inductively as follows:

$$
\begin{array}{lcl}
e\,\mathbf{rz}_\eta\, s{=}t & \Leftrightarrow & \eta(s) = \eta(t) \\
e\,\mathbf{rz}_\eta\, \iota(t) & \Leftrightarrow & [\![e]\!] = \eta(t) \\
e\,\mathbf{rz}_\eta\, \neg\phi & \Leftrightarrow & \text{there does not exist } e' \text{ s.t. } e'\,\mathbf{rz}_\eta\,\phi \\
e\,\mathbf{rz}_\eta\, \phi \supset \psi & \Leftrightarrow & \text{for all } e' \text{ s.t. } e'\,\mathbf{rz}_\eta\,\phi \text{ one has } ee'\,\mathbf{rz}_\eta\,\psi \\
e\,\mathbf{rz}_\eta\, \phi \wedge \psi & \Leftrightarrow & \pi_1 e\,\mathbf{rz}_\eta\,\phi \text{ and } \pi_2 e\,\mathbf{rz}_\eta\,\psi \\
e\,\mathbf{rz}_\eta\, \forall x.\phi & \Leftrightarrow & e\,\mathbf{rz}_{\eta[x\mapsto w]}\,\phi \text{ for all } w \in \{0,1\}^* \\
e\,\mathbf{rz}_\eta\, \Box\,\phi & \Leftrightarrow & \kappa_{\mathsf{t}(\phi)}(e)\,\mathbf{rz}_\eta\,\phi
\end{array}
$$

Notice that $\mathsf{t}(\forall x^\iota.\phi) = \iota{\multimap}\phi$ and $e\,\mathbf{rz}_\eta\,\forall x^\iota.\phi$ iff $ew\,\mathbf{rz}_{\eta[x\mapsto[\![w]\!]]}\,\phi$ for each closed term $w : \iota$. Similarly for $\forall x^{\Box\,\iota}.\phi$.

Lemma 4.1 (Isomorphism) *If formula α is provable in $\mathcal{I}_2^1$ then there is a closed term $e : \mathsf{t}(\alpha)$ such that $e\,\mathbf{rz}_\eta\,\alpha$ for every valuation η.*

Proof. Define: $\varepsilon^\iota = \varepsilon$; $\varepsilon^{\sigma\multimap\tau} = \lambda x^\sigma.\varepsilon^\tau$; $\varepsilon^{\Box\,\sigma} = \Box_\sigma\,\varepsilon^\sigma$; and $\varepsilon^{\sigma\otimes\tau} = \langle\varepsilon^\sigma, \varepsilon^\tau\rangle$. To prove the lemma, one proceeds by induction on the $\mathcal{I}_2^1$ proof concluded by α. Omitting type information, the closed term $e : \mathsf{t}(\alpha)$ in each case is as follows.

In each case of this structural induction one proves that $\forall\eta.e\,\mathbf{rz}_\eta\,\alpha$.

(K)	$\lambda x, y.x$
(B)	$\lambda x, y, z.x(yz)$
(C)	$\lambda x, y, z.(xz)y$
($\wedge$I)	$\lambda x, y.\langle x, y\rangle$
($\wedge$E)	Π
(MP)	Given $e_0 : \mathsf{t}(\alpha \supset \beta)$ and $e_1 : \mathsf{t}(\alpha)$, generate $(e_0 e_1)$
($\forall$I)	Given $e : \mathsf{t}(\alpha \supset \beta)$, the same term is $e : \mathsf{t}(\alpha \supset \forall x.\beta)$
($\forall$E)	$\lambda x.x$
(SUB$_\iota$)	$\lambda x, y.y$
($\Box$I)	Given $e : \mathsf{t}(\alpha)$ obtain $\Box\, e : \mathsf{t}(\Box\,\alpha)$
($\Box$E)	$\lambda x.\kappa x$
($\Box\,\Box$)	$\lambda x.\Box\, x$
($\Box \supset$)	$\lambda x, y.\Box((\kappa x)(\kappa y))$
($\Box$CONTR)	$\mathbf{B}$
(ιCONTR)	$\mathbf{W}$
(GEN$_\varepsilon$)	ε
(GEN$_0$)	$\lambda x.\mathbf{s}_0 x$
(GEN$_1$)	$\lambda x.\mathbf{s}_1 x$
(IND)	$\mathcal{R}$

In the cases (INT), (REFL), (SYM), (TRANS), (SUB$_=$), (SEP$_\varepsilon$), (SEP$_0$), (SEP$_1$), (SEP$_{01}$) and (SURJ), the realizing term is $\varepsilon^{\mathsf{t}(\alpha)}$. $\Box$

Theorem 4.2 *Let Φ be an equational specification for $\mathbf{f}$ using f, such that $\mathcal{I}_2^1$ proves $\Box\,\Phi \supset \forall \vec{x}^{\Box\,\iota}.\iota(f(\vec{x}))$. Then there is a closed term $e : \Box\,\iota \multimap \cdots \multimap \Box\,\iota \multimap \iota$ such that $[\![e]\!] = \mathbf{f}$ (and therefore $\mathbf{f}$ is polynomial-time computable).*

Proof. If Φ is satisfied by η then any term e_0 of type $\mathsf{t}(\Box\,\Phi)$ satisfies $e_0\,\mathbf{rz}_\eta\,\Box\,\Phi$. So, by the isomorphism lemma 4.1 we obtain a term e such that $e\,\mathbf{rz}_\eta\,\forall\vec{x}^{\Box\,\iota}.\iota(f(\vec{x}))$. Unravelling the definition reveals that $[\![e]\!](\vec{w}) = (\eta(f))(\vec{w}) = \mathbf{f}(\vec{w})$ for all $\vec{w}$. The polytime property is a feature of e typeable in $T^{\Box}$. $\Box$

It is instructive to see what happens if we were to allow arbitrary specifications not merely equational ones. Consider for instance the specification Φ given by

$$\forall x.(f(x) = 0 \vee f(x) = 1) \wedge (f(x) = 0 \equiv K(x))$$

where $K(x)$ is a Σ_1-formula defining the halting set and $\vee$ (disjunction) and $\equiv$ (biimplication) are defined as usual from $\wedge, \supset, \neg$. We can easily prove in $\mathcal{A}_2^1$ that $\forall x^{\Box\,\iota}.\iota(f(x))$, however the function f defined by Φ is not computable.

But notice that Φ is not an equational specification and hence does not have the property of being arbitrarily realisable if true.

Theorem 4.3 (Adequacy) *For every polynomial-time computable function $\mathbf{f}$, there is a closed equational program Φ defining $\mathbf{f}$ using f, such that both $\mathcal{I}_2^1$ and $\mathcal{A}_2^1$ prove $\Box\,\Phi \supset \forall x^{\Box\,\iota}.\iota(f(\vec{x}))$.*

Proof. If Φ is an equational specification corresponding to a function definition in the system of [3] then for each function symbol $f(\vec{x};\vec{y})$ with normal variables $\vec{x}$ and safe variables $\vec{y}$, the convergence statement $\Box\,\Phi \supset \forall \vec{x}^{\Box\,\iota} \forall \vec{y}^{\iota}.\iota(f(\vec{x};\vec{y}))$ is provable. For example, to prove $\forall x^{\Box\,\iota}, \forall y^{\Box\,\iota}.\iota(x \times y)$ under the definition given earlier, one proves $\Box\, x^{\Box\,\iota} \supset A \wedge B \wedge C$ where A, B, C are the antecedents of an induction on y having the conclusion $\forall y^{\Box\,\iota}.\iota(x \times y)$. Cutting A, B and C with (Ind) leads to the proof of convergence.

The result follows since all polynomial time computable functions are expressible in the system of [3]. $\Box$

It is interesting to see what happens when one tries to prove convergence of the exponentially-growing function f for which two definitions were given earlier. If one tries to prove $(\forall a^{\iota}.\iota(f^*(x,a))) \supset (\forall a^{\iota}.\iota(f^*x, f^*(x,a)))$ then two uses of the antecedent are required. This is ruled out by the absence of contraction (nonlinearity) for the non-modal formula $(\forall a^{\iota}.\iota(f^*(x,a)))$. On the other hand, if one tries to prove $\iota(f(x,a)) \supset \iota(f(x,a)+f(x,a))$ then one finds that a required antecedent is $\Box\,\iota(f(x,a))$ due to the fact that convergence of $+$ was proved by induction. But this modalised antecedent is not available in an ontic ($\Box$-free) induction statement.

5 Friedman Translation

The most interesting aspect of $\mathcal{A}_2^1$ distinguishing it e.g. from Bounded Arithmetic is that the logical complexity of induction formulas is not restricted.

One pleasing application of this is that the classical system $\mathcal{A}_2^1$ can be easily translated into the intuitionistic $\mathcal{I}_2^1$ system using Friedman's translation. This translation was originally developed to provide a simple proof that Peano arithmetic is Π_2^0-conservative over its intuitionistic version (see Coquand [9] for an exposition). Since the translation increases the logical complexity of formulas it cannot be applied to systems with induction restricted to Σ_1-formulas as is the case in Bounded Arithmetic.

Definition 5.1 (A-translation) *Let A be an arbitrary ontic formula of $\mathcal{I}_2^1$, and let $\neg_A(\phi)$ stand for $\phi \supset A$. For arbitrary formula ϕ, a formula ϕ^A called the A-translation of ϕ, is given inductively by the following clauses:*

$$
\begin{array}{lcll}
\phi^A & = & \neg_A \neg_A \phi & \textit{when } \phi \textit{ is atomic} \\
(\neg\phi)^A & = & \phi^A \supset A & \\
(\phi \wedge \psi)^A & = & \neg_A \neg_A (\phi^A \wedge \psi^A) & \\
(\phi \supset \psi)^A & = & \phi^A \supset \psi^A & \\
(\forall x.\phi)^A & = & \forall x.\phi^A & \\
(\Box\,\phi)^A & = & \neg_A \neg_A \Box\,\phi^A &
\end{array}
$$

This translation is usually (e.g. in Troelstra and van Dalen [30, Def. 3.4, Def. 5.2]) presented in two steps: the Gödel-Gentzen *negative translation* which corresponds to our A-translation for $A = \bot$ and another translation which replaces $\bot$ by A, i.e. more generally replaces atomic ψ by $\psi \vee A$. Our combined version also follows Coquand [9]. A notable difference from this standard presentation is our treatment of conjunction; see the proof of the next Lemma.

Lemma 5.2

$$\mathcal{I}_2^1 \vdash (\neg\neg\phi \supset \phi)^A$$

for arbitrary formula ϕ and ontic A.

Proof. By structural induction on ϕ. The cases (atomic),$\neg, \wedge, \Box$ are instances of the formula $\neg_A\neg_A\neg_A\alpha \supset \neg_A\alpha$ which is provable in intuitionistic propositional calculus without contraction (LIPC). The "realiser" which may provide an intuition is $\lambda H^{\neg_A^3\psi}\lambda p^{\psi} H(\lambda q^{\neg_A\psi}qp)$.

When ϕ is $\phi_1 \supset \phi_2$ we use the induction hypothesis on ϕ_2 plus an instance of the proposition $\neg_A\neg_A(\alpha \supset \beta) \supset \alpha \supset \neg_A\neg_A\beta$ which again is provable in LIPC the "realiser" being $\lambda H^{\neg_A^2(\alpha\supset\beta)}\lambda x^{\alpha}\lambda k^{\neg_A\beta}H(\lambda f^{\alpha\supset\beta}k(fx))$

When ϕ is $\forall x.\psi$ we first establish $\neg_A\neg_A\forall x.\psi^A \supset \neg_A\neg_A\psi^A$ from ($\forall$E) and LIPC. Then the induction hypothesis for ψ and an instance of ($\forall$I) give the result.

Notice our treatment of conjunction: had we defined $(\phi\wedge\psi)^A = \phi^A \wedge \psi^A$ as is usually done (e.g. by Coquand [9]) then, together with the induction hypothesis, we would need an instance of the formula $\neg_A\neg_A(\alpha\wedge\beta) \supset \neg_A\neg_A\alpha\wedge\neg_A\neg_A\beta$ which does not seem to be provable without contraction. $\square$

The main property of the A-translation is that it models classical reasoning.

Proposition 5.3 *If $\mathcal{A}_2^1 \vdash \phi$ then $\mathcal{I}_2^1 \vdash \phi^A$.*

Proof. By induction on the proof of ϕ using Lemma 5.2 in various places. Most cases are standard; we only treat those which significantly differ from the case of Peano arithmetic.

Rule ($\Box$I): if $\mathcal{A}_2^1 \vdash \alpha$ then by hypothesis $\mathcal{I}_2^1 \vdash \alpha^A$, so $\mathcal{I}_2^1 \vdash \neg_A\neg_A\Box\,\alpha^A$ by ($\Box$I) and LIPC.

For the translation of axiom ($\Box$E) we start with $\Box\,\alpha^A \supset \alpha^A$ ($\Box$E) from which we conclude $\neg_A\neg_A(\Box\,\alpha^A) \supset \neg_A\neg_A\alpha^A$ using LIPC. The desired result then follows with Lemma 5.2 on α.

The translation of axiom $\Box\,\Box$ is $\neg_A\neg_A\Box\,\alpha^A \supset \neg_A\neg_A\Box\neg_A\neg_A\Box\,\alpha^A$. To prove this, we start from $\neg_A\neg_A\Box\,\alpha^A \supset \neg_A\neg_A\Box\,\Box\,\alpha^A$ which is a LIPC consequence of an instance of $\Box\,\Box$. The result then follows by combining this with $\neg_A\neg_A\Box\,\Box\,\alpha^A \supset \neg_A\neg_A\Box\neg_A\neg_A\Box\,\alpha^A$ which in turn follows from LIPC and necessitation.

The translations of ($\Box\supset$) and ($\Box$Contr), (ιContr) are direct (LIPC) consequences from their companions in $\mathcal{I}_2^1$. In the latter two cases we rely again on our nonstandard treatment of conjunction.

Finally, for (Case) and (Ind) we use their $\mathcal{I}_2^1$-companions with formula α^A (which is ontic if α is) and Lemma 5.2. $\square$

Lemma 5.4 *If Φ is an equational specification then $\mathcal{I}_2^1 \vdash \Box\,\Phi \supset (\Box\,\Phi)^A$*

Proof. Omitted. $\square$

Theorem 5.5 *If $\mathcal{A}_2^1$ proves convergence of f, then so does $\mathcal{I}_2^1$; that is, if $\mathcal{A}_2^1 \vdash \Box\Phi \supset \forall\vec{x}^{\Box\iota}.\iota(f(\vec{x}))$ then $\mathcal{I}_2^1 \vdash \Box\Phi \supset \forall\vec{x}^{\Box\iota}.\iota(f(\vec{x}))$.*

Proof. Let $A = \iota(f(x))$. By Proposition 5.3 we have $\mathcal{I}_2^1 \vdash (\Box\Phi)^A \supset A^A$, hence $\mathcal{I}_2^1 \vdash \Box\Phi \supset (\iota(f(x)) \supset \iota(f(x))) \supset \iota(f(x))$ by Lemma 5.4 and expansion of the definition. The desired conclusion follows by LIPC. □

The main result now follows.

Corollary 5.6 *Let Φ be an equational specification for* **f** *using f. Then $\mathcal{A}_2^1$ proves the convergence statement $\Box\Phi \supset \forall\vec{x}^{\Box\iota}.\iota(f(\vec{x}))$ if and only if* **f** *is a polynomial-time computable function.*

Proof. One direction follows from Theorem 5.5 and 4.2. The other follows by Theorem 4.3. □

6 Remarks

6.1 Strength of $\mathcal{A}_2^1$

$\mathcal{A}_2^1$ proves convergence of the same functions as Buss's $\mathcal{S}_2^1$. It remains to be seen what other relationships hold between these systems.

If the ramification hierarchy explored in [5] is any guide, one might speculate that generalizations $\mathcal{A}_2^i$ lead to systems at the various levels of the Grzegorczyk hierarchy. One would also expect that replacing induction-on-notation and binary successors, with ordinary primitive induction and the ordinary successor, would result in a linear-space system.

Consider the system A_2 obtained by removing the restriction that α must be ontic in IND. $\mathcal{A}_2$ is as strong as Peano arithmetic under a translation which places $\Box$ in front of each subformula of the proof (and which accounts for the difference between induction on notation and Peano induction).

6.2 Propositional System

A fundamental principle underlying this work is that when a formula has been proved, then it can be used as many times as one would like; but when a formula is a hypothesis (and the hypothesis does not include the assertion that the formula is provable) then it can only be used once.

The number of uses of a formula has been extensively explored in the context of propositional proof systems. The *extension rule* is: $\vdash A \leftrightarrow \alpha$, where A is a new propositional letter not used earlier in the proof. Essentially, A is a new name for α. By using A repeatedly, one can refer repeatedly to α without incurring the cost (in formula size) associated with α. In this sense, the extension rule is analogous to the principle that formulas can be used as many times as desired. The Extended Frege system is defined like the Frege system but including the extension rule. For a discussion of Frege proof systems see Krajicek [21].

In the Frege and Extended Frege systems the extension principle is either withheld or admitted for all formulas at once. In contrast, a guiding idea in the present work is that free re-use (i.e. contraction) should be allowed just for formulas which have been proved. This suggests a new definition.

The *modal extension rule* is: if $\vdash \alpha$, then $\vdash A \leftrightarrow \alpha$, where A is a new propositional letter not used earlier in the proof. The *Modal Extended Frege* system is defined like the Frege system but including the modal extension rule.

This definition is open for investigation. Of course one can try directly to prove various statements about the strength of Modal Extended Frege. Another approach is to try to adapt the mapping between uniform Frege proofs and first-order proofs, to get a mapping between uniform Modal Extended Frege proofs and proofs in $\mathcal{A}_2^1$. This might imply complexity results in the uniform case.

6.3 Summary

An extension of the Curry-Howard isomorphism to include a modal $\Box$ operator has led to a functional interpretation of a modal arithmetic. The simple structure of the inductions, with no bounding terms, has allowed the use of a Friedman translation. We thereby have proved that the classical quantified modal logic $\mathcal{A}_2^1$ has polynomial strength (Corollary 5.6).

The absence of any bounds or restrictions at all on the quantifiers, seems remarkable. The "feasible" system $\mathcal{A}_2^1$ is obtained from $\mathcal{A}_2$ — a system as strong as Peano arithmetic — simply by excluding $\Box$ from the induction formula.

In addition to the areas mentioned above, one might investigate $\mathcal{A}_2^1$ with respect to the provability interpretation (cf. Boolos and Sambin, [6]).

References

[1] S. Abramsky, "Computational interpretations of linear logic", in *Theoretical Computer Science*, v. 111, p. 3-57, 1993.

[2] S. Bellantoni, "Ranking Arithmetic Proofs by Implicit Ramification", in *Proof Complexity and Feasible Arithmetics*, P. Beame and S. Buss, eds., DIMACS Series in Discrete Mathematics, v. 39, 1998.

[3] S. Bellantoni and S. Cook, "A New Recursion-Theoretic Characterization of the Polytime Functions", in *Computational Complexity*, v. 2, p. 97-110, 1992.

[4] S. Bellantoni, K.H. Niggl, and H. Schwichtenberg, "Ramification, Modality, and Linearity in Higher Type Recursion", to appear in *Annals of Pure and Applied Logic.*

[5] S. Bellantoni, K.H. Niggl, "Ranking Recursions: the Low Grzegorczyk Hierarchy Reconsidered", to appear in *SIAM Journal of Computing.*

[6] G. Boolos, G. Sambin, "Provability: the emergence of a mathematical modality", in *Studia Logica*, l. 1, p. 1-23, 1991.

[7] S. Buss, *Bounded Arithmetic*, 1986, Naples, Bibliopolis.

[8] S. Buss, “The propositional pigeonhole principle has polynomial size bounded Frege proofs”, in *Journal of Symbolic Logic*, 1987, v. 52, p. 916-927.

[9] T. Coquand, “Computational Content of Classical Proofs”, in A. Pitts and P. Dybjer, eds.,*Semantics and Logics of Computation*, Cambridge University Press, 1997.

[10] T. Coquand and M. Hofmann, “A new method for establishing conservativity of classical systems over their intuitionistic version”, in *Math. Struct. Comp. Sci.*, to appear.

[11] J. Garson, “Quantification in Modal Logic”, in *Handbook of Philosophical Logic, Vol. II*, p. 249-307, D. Gabbay and F. Guenthner, eds., D. Reidel Publishing Co., 1984.

[12] K. Gödel, “An Interpretation of the Intuitionistic Sentential Logic”, in *The Philosophy of Mathematics*, J. Hintikka, ed., series *Oxford Readings in Philosophy*, Oxford University Press, 1969.

[13] J.Y. Girard, A. Scedrov, and P.J. Scott, “Bounded linear logic: a modular approach to polynomial-time computability”, in *Theoretical Computer Science*, v. 97, p. 1-66, 1992.

[14] J.Y. Girard, “Light linear logic” in *Information and Computation*, 143, 1998.

[15] Nicolas Goodman, “Epistemic Arithmetic is a Conservative Extension of Intuitionistic Arithmetic”, in *Journal of Symbolic Logic*, v. 49, n. 1, March 1984.

[16] Martin Hofmann, “A mixed modal/linear lambda calculus with applications to Bellantoni-Cook safe recursion”, in *CSL '97*, Springer Lecture Notes in Computer Science 1414, 1998, pp. 275–294.

[17] Martin Hofmann, “Safe recursion with higher types and BCK-algebras”, to appear in *Annals of Pure and Applied Logic.*

[18] Martin Hofmann, “Type systems for polynomial-time computation”. Habilitationsschrift. Darmstadt University of Technology. 1999. Available as Edinburgh University LFCS Technical Report ECS-LFCS-99-406 or via www.dcs.ed.ac.uk/home/mxh/.

[19] N. Immerman, “Languages that capture complexity classes”, in *SIAM Journal of Computing*, v. 4, n. 16, August 1987.

[20] S. C. Kleene, *Introduction to Metamathematics*, North-Holland, Amsterdam, 1971 (first published 1952).

[21] Jan Krajicek, *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, Cambridge University Press, 1995.

[22] D. Leivant, "Intrinsic theories and computational complexity", in *Logic and Computational Complexity, International Workshop LCC '94, Indianapolis (D. Leivant, editor)* Springer LNCS 960, 1995, p. 177-194.

[23] D. Leivant, "Ramified Recurrence and Computational Complexity I: Word Recurrence and Poly-time", in *Feasible Mathematics II*, P. Clote and J. Remmel, eds., p. 320-343, series Perspectives in Computer Science, Birkhauser, 1994.

[24] D. Leivant and J.Y. Marion, "Ramified Recurrence and Computational Complexity IV: Predicative Functionals and Poly-space", in *Information and Computation*, to appear.

[25] Edward Nelson, *Predicative Arithmetic*, Princeton University Press, Princeton, N.J., 1986.

[26] Edward Nelson, "Understanding Intuitionism", presented at the *Rencontre du Reseau Georges Reeb*, March 24-28, 1997.

[27] Edward Nelson, "Ramified Recursion and Intuitionism", manuscript, Princeton University, 1997.

[28] S. Shapiro, "Epistemic and Intuitionistic Arithmetic", in *Intensional Mathematics*, S. Shapiro, ed., Studies in Logic and The Foundations of Mathematics v. 113, North-Holland, 1985.

[29] Harold Simmons, "The Realm of Primitive Recursion", in *Archive for Mathematical Logic* v. 27, p. 177+, Springer Verlag, 1988.

[30] A.S. Troelstra and D. van Dalen, *Constructivism in Mathematics*, vol. I, North-Holland, 1988.