On the proof theory of type two functionals based on primitive recursive operations

David Steiner[*][†] Thomas Strahm[†]

Version of October 2005

Abstract

This paper is a companion to work of Feferman, Jäger, Glaß, and Strahm on the proof theory of the type two functionals μ and E_1 in the context of Feferman-style applicative theories. In contrast to the previous work, we analyze these two functionals in the context of Schlüter's weakened applicative basis PRON which allows for an interpretation in the primitive recursive indices. The proof-theoretic strength of PRON augmented by μ and E_1 is measured in terms of the two subsystems of second order arithmetic, Π^1_0-CA and Π^1_1-CA, respectively.

1 Introduction

This paper is a contribution to the proof theory of type two functionals in the framework of Feferman-style applicative theories; the latter form the operational core of Feferman's explicit mathematics, which has been introduced in [4, 6, 7]. Apart from providing a basis for constructivism, the explicit framework has gained considerable importance in proof theory in connection with the proof-theoretic analysis of subsystems of second order arithmetic and set theory.

It has turned out that *untyped* first-order applicative theories provide a natural framework for a proof-theoretic approach to abstract computability. The systems considered so far range in strength from rather strong subsystems of analysis (cf. the references below) to theories of feasible strength

[*]Research partly supported by the Swiss National Science Foundation

[†]Institut für Informatik und angewandte Mathematik, Universität Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland. Email: `{steiner,strahm}@iam.unibe.ch`. Homepage: `www.iam.unibe.ch/til/staff`

(cf. [3, 2, 17, 18]). An interesting focus in previous work has been on the study of the proof theory of non-constructive type two functionals from generalized recursion theory.

In this connection, the work of Feferman, Jäger, Glaß, and Strahm on the proof-theoretic analysis of the non-constructive μ-operator ([9, 11, 12, 10]), and Jäger and Strahm on the proof theory of the Suslin operator ([13]) is of relevance. The upshot is that systems based on the μ-operator and Suslin operator can be measured in proof-theoretic terms by subsystems of second order arithmetic based on Δ^1_1 and Δ^1_2 comprehension, respectively. The standard applicative basis underlying these frameworks is the theory BON (cf. [9]) whose crucial axioms are those of an untyped partial combinatory algebra.

The aim of this article is to study the proof theory of the above-mentioned type two functionals in the context of a weakened applicative basis, called PRON, which allows for an interpretation in the primitive recursive functions. The system PRON goes back to Schlüter [14], who - answering a question by Feferman - introduced it as an alternative applicative core for explicit mathematics. In PRON, the axioms of a partial combinatory algebra of BON are replaced by those of a so-called partial enumerative algebra. The latter untyped algebras allow for interpretations in classes (of indices) of functions, whose enumerating function does not necessarily belong to the class itself. A crucial restriction in PRON is that the s combinator is replaced by specific weaker combinators i, a and b.

The weakening from BON to PRON results in a drastic decrease in proof-theoretic strength of the non-constructive μ-operator and the Suslin operator E_1, respectively. More precisely, we will show in this paper that the two considered functionals have the respective strength of arithmetical and Π^1_1 comprehension. If $(\mathrm{L}\text{-}\mathsf{I}_{\mathsf{N}})$ denotes the schema of complete induction on the natural numbers for all formulas in the underlying applicative language, we have the following landscape of proof-theoretic equivalences:

$$\begin{array}{rclcrcl}
\mathsf{BON}(\mu) + (\mathrm{L}\text{-}\mathsf{I}_{\mathsf{N}}) & \equiv & \Delta^1_1\text{-}\mathsf{CA} & \quad & \mathsf{BON}(\mu, \mathsf{E}_1) + (\mathrm{L}\text{-}\mathsf{I}_{\mathsf{N}}) & \equiv & \Delta^1_2\text{-}\mathsf{CA} \\
\mathsf{PRON}(\mu) + (\mathrm{L}\text{-}\mathsf{I}_{\mathsf{N}}) & \equiv & \Pi^1_0\text{-}\mathsf{CA} & & \mathsf{PRON}(\mu, \mathsf{E}_1) + (\mathrm{L}\text{-}\mathsf{I}_{\mathsf{N}}) & \equiv & \Pi^1_1\text{-}\mathsf{CA}
\end{array}$$

The results on the first line in this table are from [9, 13] and the equivalences on the second line will be established in this article.

The plan of this paper is as follows. In the next section we introduce the applicative framework, namely Schlüter's theory PRON and the two type two functionals μ and E_1, giving rise to two applicative systems KLE_{pr} and SUS_{pr}, which both contain the full schema of induction on the natural numbers. Section 3 is devoted to lower proof-theoretic bounds, i.e., we show that $\Pi^1_0\text{-}\mathsf{CA}$

and Π^1_1-CA are contained in KLE_{pr} and SUS_{pr}, respectively. In Section 4 we provide recursion-theoretic models of KLE_{pr} and SUS_{pr}. These models will be formalized in Π^1_0-CA and Π^1_1-CA in Section 5. In the case of SUS_{pr} we will need the so-called inside-outside property to show that our model satisfies the axioms of E_1. We conclude our paper with some scattered remarks on systems with restricted forms of complete induction and additional combinators.

The results of this paper are based on Steiner's Master's thesis [16].

2 The applicative framework

It is the aim of this section to introduce Schlüter's basic applicative theory PRON of primitive recursive operations and numbers, cf. [14]. Moreover, we will provide suitable axiomatizations of the non-constructive μ operator and the Suslin operator E_1.

The language of our applicative theories is a first order language L of partial terms with individual variables $a, b, c, f, g, h, u, v, w, x, y, z \ldots$ (possibly with subscripts). L includes individual constants $\mathsf{i}, \mathsf{k}, \mathsf{a}, \mathsf{b}$ (combinators), $\mathsf{p_0}, \mathsf{p_1}$ (unpairing), 0 (zero), $\mathsf{s_N}$ (numerical successor), $\mathsf{p_N}$ (numerical predecessor), $\mathsf{d_N}$ (definition by numerical cases), r (primitive recursion), μ (non-constructive μ operator), and E_1 (Suslin operator). Further, L has two binary function symbols $\cdot$ (partial term application) and $<>$ (pairing), two unary relation symbols $\downarrow$ (defined) and N (natural numbers), as well as a binary relation symbol $=$ (equality).

The *individual terms* $(r, s, t, r_1, s_1, t_1, \ldots)$ of L are inductively generated as follows:

1. The individual variables and individual constants are individual terms.
2. If s and t are individual terms, then so also are $(s \cdot t)$ and $<s, t>$.

In the following we often abbreviate $(s \cdot t)$ simply as (st) or st; the context will always ensure that no confusion arises. We further adopt the convention of association to the left so that $s_1 s_2 \ldots s_n$ stands for $(\ldots(s_1 s_2) \ldots s_n)$. Further, we put $t' := \mathsf{s_N} t$ and $1 := 0'$. We define general tupling in the expected manner:

$$\begin{array}{rcl} <s> & := & s, \\ <s_1, \ldots, s_{n+1}> & := & <<s_1, \ldots, s_n>, s_{n+1}>. \end{array}$$

Finally, we also use quite frequently the vector notation $\vec{\mathcal{Z}}$ for a finite string of objects $\mathcal{Z}_1, \ldots, \mathcal{Z}_n$ of the same sort. Whenever we write $\vec{\mathcal{Z}}$ the length of this string is either irrelevant or given by the context.

The *formulas* $(A, B, C, A_1, B_1, C_1, \dots)$ of L are inductively generated as follows:

1. Each atomic formula $\mathsf{N}(t)$, $t{\downarrow}$, and $(s = t)$ is a formula.
2. If A and B are formulas, then so also are $\neg A$, $(A \vee B)$, $(A \wedge B)$, and $(A \rightarrow B)$.
3. If A is a formula, then so also are $(\exists x)A$ and $(\forall x)A$.

Our applicative theories are based on *partial* term application. Hence, it is not guaranteed that terms have a value, and $t{\downarrow}$ is read as "t is defined" or "t has a value". Accordingly, the *partial equality relation* $\simeq$ is introduced by

$$s \simeq t \;:=\; (s{\downarrow} \vee t{\downarrow}) \rightarrow (s = t).$$

In addition, we write $(s \neq t)$ for $(s{\downarrow} \wedge t{\downarrow} \wedge \neg(s = t))$.

Finally, we use the following abbreviations concerning the predicate N ($\vec{t} = t_1, \dots, t_m$, $\vec{x} = x_1, \dots, x_n$):

$$\begin{array}{rcl}
\vec{t} \in \mathsf{N} & := & \mathsf{N}(t_1) \wedge \dots \wedge \mathsf{N}(t_m), \\
(\exists \vec{x} \in \mathsf{N})A & := & (\exists \vec{x})(\vec{x} \in \mathsf{N} \wedge A), \\
(\forall \vec{x} \in \mathsf{N})A & := & (\forall \vec{x})(\vec{x} \in \mathsf{N} \rightarrow A), \\
\vec{t} \in (\mathsf{N}^n \rightarrow \mathsf{N}) & := & (\forall \vec{x} \in \mathsf{N})(t_1{<}\vec{x}{>} \in \mathsf{N} \wedge \dots \wedge t_m{<}\vec{x}{>} \in \mathsf{N}).
\end{array}$$

Of course we will write $\vec{t} \in (\mathsf{N} \rightarrow \mathsf{N})$ instead of $\vec{t} \in (\mathsf{N}^1 \rightarrow \mathsf{N})$. Let us also recall the notion of a *subset of* N from [5, 9]. Sets of natural numbers are represented via their characteristic functions which are total on N. Accordingly, we define

$$f \in \mathcal{P}(\mathsf{N}) \;:=\; (\forall x \in \mathsf{N})(fx = 0 \vee fx = 1)$$

with the intention that a natural number x belongs to the set $f \in \mathcal{P}(\mathsf{N})$ if and only if $(fx = 0)$.

Now we are going to recall Schlüter's basic applicative theory $\mathsf{PEA}^+ + (\mathsf{r})$ of [14]. Following Steiner [16], in this paper we will call this theory PRON. It can be seen as the primitive recursive analogue of the theory BON in Feferman and Jäger [9]. Its underlying logic is the *classical logic of partial terms* due to Beeson [1]; it is also described in Feferman [8] and corresponds to E^+ logic with strictness and equality of Troelstra and Van Dalen [19]. The non-logical axioms of PRON are divided into the following five groups.

I. Partial enumerative algebra.

(1) $\mathsf{i}x = x$,

(2) $\mathsf{k}xy = x$,

(3) $\mathsf{a}{<}x, y{>}\downarrow \,\wedge\, \mathsf{a}{<}x, y{>}z \simeq {<}xz, yz{>}$,

(4) $\mathsf{b}{<}x, y{>}\downarrow \,\wedge\, \mathsf{b}{<}x, y{>}z \simeq x(yz)$.

II. Pairing and projection.

(5) $\mathsf{p}_0{<}x, y{>} = x \,\wedge\, \mathsf{p}_1{<}x, y{>} = y$.

III. Natural numbers.

(6) $0 \in \mathsf{N} \,\wedge\, (\forall x \in \mathsf{N})(x' \in \mathsf{N})$,

(7) $(\forall x \in \mathsf{N})(x' \neq 0 \wedge \mathsf{p_N}x' = x)$,

(8) $(\forall x \in \mathsf{N})(x \neq 0 \,\rightarrow\, \mathsf{p_N}x \in \mathsf{N} \wedge (\mathsf{p_N}x)' = x)$.

IV. Definition by numerical cases.

(9) $u \in \mathsf{N} \wedge v \in \mathsf{N} \,\wedge\, u = v \rightarrow \mathsf{d_N}{<}x, y, u, v{>} = x$,

(10) $u \in \mathsf{N} \wedge v \in \mathsf{N} \,\wedge\, u \neq v \rightarrow \mathsf{d_N}{<}x, y, u, v{>} = y$.

V. Primitive recursion.

(11) $\mathsf{r}{<}f, g{>}\downarrow \,\wedge\, \mathsf{r}{<}f, g{>}{<}x, 0{>} \simeq fx$,

(12) $\mathsf{r}{<}f, g{>}{<}x, y'{>} \simeq g{<}x, y, \mathsf{r}{<}f, g{>}{<}x, y{>}{>}$.

Since PRON is formulated in the restricted partial enumerative algebra setting, it is no longer possible to define abstraction terms $(\lambda x.t)$ for arbitrary terms t. As is observed in Schlüter [14], in the presence of PRON we can form $(\lambda x.t)$ only for those terms t where the variable x occurs in *argument position.* The crucial requirement for a variable x to occur in argument position in t is that x does not occur in r for arbitrary subterms (rs) of t. For a precise definition of this notion and a proof of the following lemma, see [14].

Lemma 1 (Abstraction) *For each* L *term* t *and all variables* x *so that* x *appears in argument position in* t *there exists an* L *term* $(\lambda x.t)$ *whose variables are those of* t*, excluding* x*, so that* PRON *proves*

$$(\lambda x.t)\downarrow \,\wedge\, (\lambda x.t)x \simeq t.$$

For example, we cannot form the terms $(\lambda x.xx)$ or $(\lambda z.xz(yz))$ with their usual properties. On the other hand, the abstraction $(\lambda z.{<}xz, yz{>})$ is legitimate.

Moreover, we have the following restricted form of the recursion or fixed point theorem. Interpreted in the indices for primitive recursive functions, it corresponds to the well-known primitive recursion theorem. Observe that the fixed point may only occur in argument position.

Lemma 2 (Recursion) *There exists an* L *term* rec *so that* PRON *proves*

$$\mathsf{rec} f\downarrow \;\wedge\; \mathsf{rec} fx \simeq f{<}\mathsf{rec} f, x{>}.$$

We now turn to the type two functionals which are of interest in the sequel. These are the non-constructive μ operator, which acts as a quantification operator on the natural numbers (Kleene's E_0), and the Suslin operator E_1 testing for the wellfoundedness of a given binary relation. Due to our restricted applicative setting we axiomatize μ and E_1 in a slightly different manner than in [13].

The non-constructive μ operator

(μ.1) $(\forall x \in \mathsf{N})(f{<}u, x{>} \in \mathsf{N}) \;\leftrightarrow\; \mu fu \in \mathsf{N}$,

(μ.2) $(\forall x \in \mathsf{N})(f{<}u, x{>} \in \mathsf{N}) \;\rightarrow$

$$[(\exists x \in \mathsf{N})(f{<}u, x{>} = 0) \;\rightarrow\; f{<}u, \mu fu{>} = 0].$$

The Suslin operator E_1

(E_1.1) $(\forall x, y \in \mathsf{N})(f{<}u, x, y{>} \in \mathsf{N}) \;\leftrightarrow\; \mathsf{E}_1 fu \in \mathsf{N}$,

(E_1.2) $(\forall x, y \in \mathsf{N})(f{<}u, x, y{>} \in \mathsf{N}) \;\rightarrow$

$$[(\exists g \in \mathsf{N} \rightarrow \mathsf{N})(\forall x \in \mathsf{N})(f{<}u, gx', gx{>} = 0) \;\leftrightarrow\; \mathsf{E}_1 fu = 0].$$

In the following we write $\mathsf{PRON}(\mu)$ for PRON augmented by the axioms $(\mu.1), (\mu.2)$ and $\mathsf{PRON}(\mu, \mathsf{E}_1)$ for $\mathsf{PRON}(\mu)$ with $(\mathsf{E}_1.1), (\mathsf{E}_1.2)$.

With respect to induction on the natural numbers, in the sequel we are mainly concerned with the full induction scheme given as usual; for restricted forms of induction, see the conclusion of this paper.

Formula induction on N (L-$\mathsf{I_N}$). For all formulas $A(x)$ of L:

$$A(0) \wedge (\forall x \in \mathsf{N})(A(x) \rightarrow A(x')) \rightarrow (\forall x \in \mathsf{N})A(x).$$

In the rest of this paper we will be chiefly dealing with the two systems KLE_{pr} and SUS_{pr} which are defined as follows.

$$\mathsf{KLE}_{pr} \;:=\; \mathsf{PRON}(\mu) + (\text{L-}\mathsf{I}_\mathsf{N}) \qquad \mathsf{SUS}_{pr} \;:=\; \mathsf{PRON}(\mu, \mathsf{E}_1) + (\text{L-}\mathsf{I}_\mathsf{N}).$$

In the next section we will establish lower proof-theoretic bounds by embedding suitable subsystems of analysis into these applicative systems.

3 Lower bounds

In the following we establish proof-theoretic lower bounds for KLE_{pr} and SUS_{pr} by embedding suitable subsystems of second order arithmetic. In the first paragraph we introduce the language and axioms of second order arithmetic and recapitulate the Π^1_1 normal form theorem. The second paragraph is devoted to the embeddings of Π^1_0-CA and Π^1_1-CA into KLE_{pr} and SUS_{pr}, respectively.

3.1 Subsystems of analysis

In the following we introduce the two subsystems of second order arithmetic Π^1_0-CA and Π^1_1-CA. Later we will show that these systems are naturally contained in KLE_{pr} and SUS_{pr}, respectively. It will be convenient in the sequel to work with a form of second order arithmetic with set *and* function variables.

Let $\mathcal{L}_2$ denote a language of second order arithmetic with *number variables* $a, b, c, f, g, h, u, v, w, x, y, z, \ldots$, *set variables* $U, V, W, X, Y, Z, \ldots$, and *function variables* $F, G, H, \ldots$ (all possibly with subscripts). In addition, $\mathcal{L}_2$ includes a constant 0 as well as function and relation symbols for all primitive recursive functions and relations. The *number terms* $(r, s, t, r_1, s_1, t_1, \ldots)$ of $\mathcal{L}_2$ and the *formulas* $(A, B, C, A_1, B_1, C_1, \ldots)$ of $\mathcal{L}_2$ are defined as usual.

An $\mathcal{L}_2$ formula is called *arithmetic*, if it does not contain bound set or function variables; let Π^1_0 denote the class of arithmetic $\mathcal{L}_2$ formulas. The Π^1_1 $[\Sigma^1_1]$ formulas of $\mathcal{L}_2$ are obtained from the arithmetic formulas by closing under universal [existential] set and function quantification.

In the following we make use of the usual primitive recursive coding machinery in $\mathcal{L}_2$: $\langle \ldots \rangle$ is a standard primitive recursive function for forming n-tuples $\langle t_1, \ldots, t_n \rangle$; Seq is the primitive recursive set of sequence numbers; $\mathit{lh}(t)$ denotes the length of (the sequence number coded by) t; $(t)_i$ is the ith component of (the sequence coded by) t if $i < \mathit{lh}(t)$, i.e. $t = \langle (t)_0, \ldots, (t)_{\mathit{lh}(t) \dot{-} 1} \rangle$ if t is a sequence number; we will write, e.g., $(s)_{i,j}$ instead of $((s)_i)_j$. Moreover, $*$

denotes the binary primitive recursive operation of sequence concatenation. Finally, we write $s \in (U)_t$ for $\langle s, t \rangle \in U$.

If $\mathcal{F}$ is a collection of $\mathcal{L}_2$ formulas, then $\mathcal{F}$ comprehension ($\mathcal{F}$-CA) is the schema

$$(\exists X)(\forall x)(x \in X \leftrightarrow A(x))$$

for all formulas $A(u)$ in the collection $\mathcal{F}$. The relationship between sets and functions is given by the so-called graph principle ($\mathcal{GP}$),

$$(\forall X)[(\forall x)(\exists! y)\langle x, y \rangle \in X \;\rightarrow\; (\exists F)(\forall x)\langle x, F(x) \rangle \in X].$$

Moreover, $\mathcal{L}_2$ induction on the natural numbers ($\mathcal{L}_2$-$\mathsf{I_N}$) comprises

$$A(0) \wedge (\forall x)(A(x) \rightarrow A(x')) \;\rightarrow\; (\forall x)A(x)$$

for all formulas $A(u)$ of $\mathcal{L}_2$.

Π^1_0-CA is the $\mathcal{L}_2$ theory which contains the usual axioms of Peano arithmetic PA, all instances of Π^1_0 comprehension, the graph principle ($\mathcal{GP}$) as well as formula induction ($\mathcal{L}_2$-$\mathsf{I_N}$). Π^1_1-CA is defined accordingly.

We conclude this subsection by stating a version of the normal form theorem for Π^1_1 formulas tailored for our later purposes. Its proof is more or less folklore and can be found at many places (for example in Simpson [15]).

Theorem 3 (Π^1_1 normal forms) *For every Π^1_1 formula A there exists an arithmetic formula $B_A(u, v)$ which contains the free variables of A plus two fresh variables u and v so that Π^1_0-CA proves*

$$A \;\leftrightarrow\; \neg(\exists F)(\forall x)B_A(F(x'), F(x)).$$

Let us mention that in fact restricted induction is enough to establish the normal form theorem in Π^1_0-CA.

3.2 Embeddings

We work with the natural embedding of $\mathcal{L}_2$ into L so that (i) the number variables of $\mathcal{L}_2$ are interpreted as ranging over N, (ii) the set variables of $\mathcal{L}_2$ as ranging over $\mathcal{P}(\mathsf{N})$, and (iii) the function variables as ranging over $(\mathsf{N} \rightarrow \mathsf{N})$.

In the following we assume that we have a translation of the number, set and function variables of $\mathcal{L}_2$ into the variables of L so that no conflicts arise. For convenience we often simply write, for example, a, x, f for the translations of the number, set and function variable a, X, F, respectively. Furthermore, we can use the recursion operator r to associate a suitable L term to each symbol for a primitive recursive function on the natural numbers and prove the corresponding recursion equations and totality properties in KLE_{pr}. Thus, every $\mathcal{L}_2$ term t has a canonical translation t^{N} in L. Similarly, each symbol for a primitive recursive relation on N can be represented by an L term which represents its characteristic function in the sense above.

Now let R be a symbol for an n-ary primitive recursive relation and t_R the corresponding L term. If $s, t_1, \ldots, t_n$ are terms of $\mathcal{L}_2$, then the atomic formulas of $\mathcal{L}_2$ are translated into L formulas as follows:

$$(s \in U)^{\mathsf{N}} := ((us^{\mathsf{N}}) = 0); \qquad (t_1 = t_2)^{\mathsf{N}} := (t_1^{\mathsf{N}} = t_2^{\mathsf{N}});$$
$$R(t_1, \ldots, t_n)^{\mathsf{N}} := (t_R{<}t_1^{\mathsf{N}}, \ldots, t_n^{\mathsf{N}}{>} = 0).$$

We extend this translation in the usual way and associate to each $\mathcal{L}_2$ formula $A(\vec{U}, \vec{F}, \vec{v})$ an L formula $A^{\mathsf{N}}(\vec{u}, \vec{f}, \vec{v})$ such that

$$\begin{aligned} ((\exists X)A(X))^{\mathsf{N}} &= (\exists x \in \mathcal{P}(\mathsf{N}))A^{\mathsf{N}}(x), \\ ((\exists F)A(F))^{\mathsf{N}} &= (\exists f \in \mathsf{N} \to \mathsf{N})A^{\mathsf{N}}(f), \\ ((\exists y)A(y))^{\mathsf{N}} &= (\exists y \in \mathsf{N})A^{\mathsf{N}}(y), \end{aligned}$$

and similarly for universal quantifiers. A further convention is that we often identify $\mathcal{L}_2$ terms and arithmetic $\mathcal{L}_2$ formulas with their translation in L as long as no conflict arises.

It is not surprising that the unbounded μ operator can be used in order to eliminate arithmetical quantifiers and, hence, to represent each arithmetic formula of $\mathcal{L}_2$ by means of a term in the applicative language L, cf. Feferman and Jäger [9]. Moreover, the additional presence of the Suslin operator E_1 enables us to find characteristic terms even for Π^1_1 formulas, as is easily seen by making use of the above-mentioned Π^1_1 normal form theorem, cf. Jäger and Strahm [13]. However, in the context of our restricted applicative framework, some additional technicalities arise, for instance, the characteristic terms are in general not closed and applied only to the number parameters of a given formula. The details are not difficult and worked out at full length in Steiner [16].

Lemma 4 *For every arithmetic formula $A(\vec{U}, \vec{F}, \vec{v})$ of $\mathcal{L}_2$ with all its free variables in $\vec{U}, \vec{F}, \vec{v}$ there exists an individual term t_A of* L *with all its free variables in $\vec{u}, \vec{f}$ so that KLE_{pr} proves*

1. $(\forall \vec{u} \in \mathcal{P}(\mathsf{N}))(\forall \vec{f} \in \mathsf{N} \to \mathsf{N})(\forall \vec{v} \in \mathsf{N})(t_A{<}\vec{v}{>} = 0 \vee t_A{<}\vec{v}{>} = 1)$,
2. $(\forall \vec{u} \in \mathcal{P}(\mathsf{N}))(\forall \vec{f} \in \mathsf{N} \to \mathsf{N})(\forall \vec{v} \in \mathsf{N})(A^{\mathsf{N}}(\vec{u}, \vec{f}, \vec{v}) \leftrightarrow t_A{<}\vec{v}{>} = 0)$.

Lemma 5 *For every Π^1_1 formula $A(\vec{U}, \vec{F}, \vec{v})$ of $\mathcal{L}_2$ with all its free variables in $\vec{U}, \vec{F}, \vec{v}$ there exists an individual term t_A of* L *with all its free variables in $\vec{u}, \vec{f}$ so that* SUS_{pr} *proves*

1. $(\forall \vec{u} \in \mathcal{P}(\mathsf{N}))(\forall \vec{f} \in \mathsf{N} \to \mathsf{N})(\forall \vec{v} \in \mathsf{N})(t_A{<}\vec{v}{>} = 0 \vee t_A{<}\vec{v}{>} = 1)$,
2. $(\forall \vec{u} \in \mathcal{P}(\mathsf{N}))(\forall \vec{f} \in \mathsf{N} \to \mathsf{N})(\forall \vec{v} \in \mathsf{N})(A^{\mathsf{N}}(\vec{u}, \vec{f}, \vec{v}) \leftrightarrow t_A{<}\vec{v}{>} = 0)$.

From these two lemmata it is immediate how to deal with (Π^1_0-CA) in KLE_{pr} and with (Π^1_1-CA) in SUS_{pr}. In a nutshell, μ is employed to deal with arithmetic comprehension and E_1 is used to translate each instance of Π^1_1 comprehension. Moreover, the translation of the graph principle can be seen to be valid by means of μ. For details see Steiner [16] and Jäger and Strahm [13].

Theorem 6 *Let $A(\vec{U}, \vec{F}, \vec{v})$ be an $\mathcal{L}_2$ formula with all its free variables in $\vec{U}, \vec{F}, \vec{v}$ and assume that Π^1_0-CA proves $A(\vec{U}, \vec{F}, \vec{v})$. Then we have*

$$\mathsf{KLE}_{pr} \vdash \vec{u} \in \mathcal{P}(\mathsf{N}) \wedge \vec{f} \in (\mathsf{N} \to \mathsf{N}) \wedge \vec{v} \in \mathsf{N} \to A^{\mathsf{N}}(\vec{u}, \vec{f}, \vec{v}).$$

Theorem 7 *Let $A(\vec{U}, \vec{F}, \vec{v})$ be an $\mathcal{L}_2$ formula with all its free variables in $\vec{U}, \vec{F}, \vec{v}$ and assume that Π^1_1-CA proves $A(\vec{U}, \vec{F}, \vec{v})$. Then we have*

$$\mathsf{SUS}_{pr} \vdash \vec{u} \in \mathcal{P}(\mathsf{N}) \wedge \vec{f} \in (\mathsf{N} \to \mathsf{N}) \wedge \vec{v} \in \mathsf{N} \to A^{\mathsf{N}}(\vec{u}, \vec{f}, \vec{v}).$$

We finish this section by mentioning that we cannot define the ω-jump and the ω-hyperjump in KLE_{pr} and SUS_{pr}, respectively. This is due to our restricted applicative setting and, in particular, the absence of the s combinator. In contrast to our present framework, transfinite (hyper-)jump hierarchies are available in the corresponding applicative theories based on BON. For further information about this phenomenon, cf. the discussion at the beginning of Section 5.1.

4 Recursion-theoretic models

In this section we provide models for KLE_{pr} and SUS_{pr}. We proceed in two steps and will first define indices and their evaluations. Secondly, we show how to use the so-obtained evaluation functions in order to set up models for KLE_{pr} and SUS_{pr}. In Section 5 we will formalize these models in order to establish upper bounds for KLE_{pr} and SUS_{pr}.

4.1 Function classes and indices

In this section we are going to introduce the classes of number-theoretic functions $PRIM(\mu)$ and $PRIM(E_1)$. Further, we will introduce a set of indices for both classes and we are going to define the level of an index. Finally, we define evaluation functions for the defined set of indices.

Definition 8 We define the following schemas and classes of functions:

1. Let n, m, k, $\vec{x} = x_0, \ldots, x_{n-1}$ be natural numbers. We define the following basic functions:
   (a) *Successor.* $S(x_0) := x_0 + 1$
   (b) *Constant functions.* $Cs^n_m(\vec{x}) := m$
   (c) *Projections.* If $k < n$ then $Pr^n_k(\vec{x}) := x_k$
2. Let n, m, $\vec{x} = x_0, \ldots, x_{n-1}$, y be natural numbers and $\mathcal{K}$ be a class of number theoretic functions. We define the following closure conditions on functions in $\mathcal{K}$:
   (a) *Composition.* If $m > 0$ and f is an m-ary function of $\mathcal{K}$ and g_0, $\ldots$, g_{m-1} are n-ary functions of $\mathcal{K}$, then the n-ary function
   $$\mathit{Comp}^n(f, g_0, \ldots, g_{m-1})(\vec{x}) \quad := \quad f(g_0(\vec{x}), \ldots, g_{m-1}(\vec{x}))$$
   is an element of $\mathcal{K}$.
   (b) *Primitive recursion.* If f is an n-ary function of $\mathcal{K}$ and g is an $(n+2)$-ary function of $\mathcal{K}$, then the $(n+1)$-ary function
   $$\mathit{Rec}^{n+1}(f,g)(\vec{x},y) := \begin{cases} f(\vec{x}) & \text{if } y = 0 \\ g(\vec{x}, y-1, \mathit{Rec}^{n+1}(f,g)(\vec{x}, y-1)) & \text{if } y > 0 \end{cases}$$
   is an element of $\mathcal{K}$.
   (c) *μ operator.* If f is an $(n+1)$-ary function of $\mathcal{K}$, then the n-ary function
   $$\mathit{Zero}^n(f)(\vec{x}) \quad := \quad \begin{cases} \min\{y \mid f(\vec{x},y) = 0\} & \text{if there is a } y \text{ so} \\ & \quad \text{that } f(\vec{x},y) = 0 \\ 0 & \text{otherwise} \end{cases}$$
   is an element of $\mathcal{K}$.

(d) *Suslin operator.* If f is an $(n+2)$-ary function of $\mathcal{K}$, then the n-ary function

$$Sus^n(f)(\vec{x}) \;:=\; \begin{cases} 0 & \text{if there is a unary function } g\text{, so} \\ & \quad \text{that } f(\vec{x}, g(z'), g(z)) = 0 \\ & \quad \text{for every natural number } z \\ 1 & \text{otherwise} \end{cases}$$

is an element of $\mathcal{K}$.

3. Now we can introduce the following classes of functions:

   (a) The class *PRIM*(μ) consists of the basic functions and is closed under composition, primitive recursion, and the μ operator.

   (b) The class *PRIM*(E_1) consists of the basic functions and is closed under composition, primitive recursion, the μ operator, and the Suslin operator.

In a next step we want to introduce indices for all the functions belonging to the classes *PRIM*(μ) and *PRIM*(E_1). Moreover, we will spell out a corresponding evaluation function for each set of indices. We start by defining the indices for every function of *PRIM*(E_1) and will later obtain the indices in *PRIM*(μ) by a suitable restriction.

Definition 9 *SusPrim* is defined to be the set of indices for functions in *PRIM*(E_1).

$$\begin{aligned}
&s \in \mathit{SusPrim} \;\Leftrightarrow \\
&s \in \mathit{Seq} \wedge [s = \langle 0, 1\rangle \vee \\
&[(s)_0 = 1 \wedge \mathit{lh}(s) = 3] \vee [(s)_0 = 2 \wedge \mathit{lh}(s) = 3 \wedge (s)_1 > (s)_2] \vee \\
&[(s)_0 = 3 \wedge \mathit{lh}(s) = (s)_{2,1} + 3 \wedge (s)_2 \in \mathit{SusPrim} \wedge (s)_{2,1} > 0 \wedge \\
&\quad (\forall k < (s)_{2,1})((s)_{k+3} \in \mathit{SusPrim} \wedge (s)_{k+3,1} = (s)_1)] \vee \\
&[(s)_0 = 4 \wedge \mathit{lh}(s) = 4 \wedge (s)_2 \in \mathit{SusPrim} \wedge (s)_3 \in \mathit{SusPrim} \wedge \\
&\quad (s)_1 = (s)_{2,1} + 1 \wedge (s)_{3,1} = (s)_1 + 1] \vee \\
&[(s)_0 = 5 \wedge \mathit{lh}(s) = 3 \wedge (s)_2 \in \mathit{SusPrim} \wedge (s)_{2,1} = (s)_1 + 1] \vee \\
&[(s)_0 = 6 \wedge \mathit{lh}(s) = 3 \wedge (s)_2 \in \mathit{SusPrim} \wedge (s)_{2,1} = (s)_1 + 2]]
\end{aligned}$$

Note that *SusPrim* is a primitive recursive set. We obtain the natural restriction μ*Prim* of *SusPrim* by dropping the last clause in the inductive definition above. Clearly, μ*Prim* is a primitive recursive set as well.

The next definition provides the expected evaluation function Ψ for indices in *SusPrim*, using the class of functions in *PRIM*(E_1).

Definition 10 Let $s \in \mathit{SusPrim}$ be an index. The function Ψ_s is defined by induction on the build-up of s as follows:

$$\begin{array}{rcl}
\Psi_{\langle 0,1\rangle} & := & S \\
\Psi_{\langle 1,n,m\rangle} & := & \mathit{Cs}^n_m \\
\Psi_{\langle 2,n,k\rangle} & := & \mathit{Pr}^n_k \\
\Psi_{\langle 3,n,f,g_0,\ldots,g_{m-1}\rangle} & := & \mathit{Comp}^n(\Psi_f, \Psi_{g_0}, \ldots, \Psi_{g_{m-1}}) \\
\Psi_{\langle 4,n+1,f,g\rangle} & := & \mathit{Rec}^{n+1}(\Psi_f, \Psi_g) \\
\Psi_{\langle 5,n,f\rangle} & := & \mathit{Zero}^n(\Psi_f) \\
\Psi_{\langle 6,n,f\rangle} & := & \mathit{Sus}^n(\Psi_f)
\end{array}$$

The obvious restriction of Ψ for $s \in \mu\mathit{Prim}$ is denoted by Φ in the following. Moreover, we will sometimes use the notation $[s]$ for Ψ_s or Φ_s, depending on $s \in \mathit{SusPrim}$ or $s \in \mu\mathit{Prim}$, respectively.

The evaluation function *SusPrimEv* is calculating the function Ψ_s, if s is a unary index of *SusPrim*.

Definition 11 *SusPrimEv* is the following function from $\mathbb{N}^2$ to $\mathbb{N}$:

$$\mathit{SusPrimEv}(x,y) \quad := \quad \begin{cases} \Psi_x(y) & \text{if } x \in \mathit{SusPrim} \wedge (x)_1 = 1 \\ 0 & \text{otherwise} \end{cases}$$

The corresponding evaluation function defined in terms of Φ and $\mu\mathit{Prim}$ will be denoted by $\mu\mathit{PrimEv}$ in the sequel.

For every index s of *SusPrim* we inductively define its *level* $\mathit{lev}(s)$ as follows.

Definition 12 The primitive recursive function $\mathit{lev}(s)$ is defined by course-of-value recursion:

$$\mathit{lev}(s) \quad := \quad \begin{cases} 0 & \text{if } (s)_0 \leq 2 \\ \max\{\mathit{lev}((s)_2), \ldots, \mathit{lev}((s)_{\mathit{lh}(s) \dot{-} 1})\} + 1 & \text{if } (s)_0 \geq 3 \end{cases}$$

We will define models of KLE_{pr} and SUS_{pr} in terms of indices of *unary* functions only. Therefore, we need to define some auxiliary functions which change the arity of a given index in a suitable manner. The proof of the following lemma is easy and therefore omitted.

Lemma 13 *There exist primitive recursive functions* $\cdot^\dagger$, $\widetilde{\cdot}$, *and* $\overline{\overline{\cdot}}$ *so that we have for all* $n > 1$, *all natural numbers* $x_0, \ldots, x_{n-1}$, *all* $e \in \mathit{SusPrim}$ *with* $(e)_1 = n$ *and all* $f \in \mathit{SusPrim}$ *with* $(f)_1 = 1$*:*

$$\begin{array}{rcl}
[\, e^\dagger\,](\langle \ldots \langle\langle x_0, x_1\rangle, x_2\rangle, \ldots, x_{n-1}\rangle) & = & [e](x_0, \ldots, x_{n-1}) \\
[\, \widetilde{f}\,](x_0, x_1) & = & [f](\langle x_0, x_1\rangle) \\
[\, \overline{\overline{f}}\,](x_0, x_1, x_2) & = & [f](\langle\langle x_0, x_1\rangle, x_2\rangle)
\end{array}$$

4.2 Model constructions

In this subsection we will define the intended recursion-theoretic models $\mathcal{S}_{pr}$ of SUS_{pr} and $\mathcal{K}_{pr}$ of KLE_{pr}. The universe of these models is the set $\mathbb{N}$ of natural numbers and the function symbol $\cdot$ is interpreted as the previously defined evaluation functions *SusPrimEv* and *μPrimEv*, respectively.

Definition 14 The structure $\mathcal{S}_{pr} = (\mathbb{N}, \mathbb{N}, \mathit{SusPrimEv}, \dots)$ is defined in the following manner:

$$\begin{array}{rcl}
|\mathcal{S}_{pr}| & := & \mathbb{N} \\
\mathsf{N}^{\mathcal{S}_{pr}} & := & \mathbb{N} \\
.^{\mathcal{S}_{pr}} & := & \mathit{SusPrimEv} \\
<>^{\mathcal{S}_{pr}} & := & \text{the primitive recursive function } (x,y) \mapsto \langle x, y\rangle \\
(0)^{\mathcal{S}_{pr}} & := & 0 \\
(\mathsf{s_N})^{\mathcal{S}_{pr}} & := & \langle 0, 1\rangle \\
(\mathsf{i})^{\mathcal{S}_{pr}} & := & \langle 2, 1, 0\rangle \\
(\mathsf{k})^{\mathcal{S}_{pr}} & := & \text{index of the primitive recursive function } x \mapsto \langle 1, 1, x\rangle \\
(\mathsf{p_N})^{\mathcal{S}_{pr}} & := & \text{index of the primitive recursive function } x \mapsto x \dot{-} 1 \\
(\mathsf{p_0})^{\mathcal{S}_{pr}} & := & \text{index of the primitive recursive function } x \mapsto (x)_0 \\
(\mathsf{p_1})^{\mathcal{S}_{pr}} & := & \text{index of the primitive recursive function } x \mapsto (x)_1 \\
(\mathsf{d_N})^{\mathcal{S}_{pr}} & := & \text{index of the primitive recursive function} \\
 & & x \mapsto \begin{cases} (x)_{0,0,0} & \text{if } (x)_{0,1} = (x)_1 \\ (x)_{0,0,1} & \text{otherwise} \end{cases}
\end{array}$$

$$\begin{array}{rcl}
(\mathsf{a})^{\mathcal{S}_{pr}} & := & \langle 3, 1, s_2, \langle 1, 1, <>^{\mathcal{S}_{pr}}\rangle, (\mathsf{p_0})^{\mathcal{S}_{pr}}, (\mathsf{p_1})^{\mathcal{S}_{pr}}\rangle \text{ where } s_2 \text{ is the} \\
 & & \text{index of the primitive recursive function } (x,y,z) \mapsto \\
 & & \begin{cases} \langle 3,1,x,y,z\rangle & \text{if } y \in \mathit{SusPrim} \wedge (y)_1 = 1 \wedge \\ & \quad z \in \mathit{SusPrim} \wedge (z)_1 = 1 \\ \langle 3,1,x,y,\langle 1,1,0\rangle\rangle & \text{if } y \in \mathit{SusPrim} \wedge (y)_1 = 1 \wedge \\ & \quad (z \notin \mathit{SusPrim} \vee (z)_1 \neq 1) \\ \langle 3,1,x,\langle 1,1,0\rangle,z\rangle & \text{if } (y \notin \mathit{SusPrim} \vee (y)_1 \neq 1) \wedge \\ & \quad z \in \mathit{SusPrim} \wedge (z)_1 = 1 \\ \langle 3,1,x,\langle 1,1,0\rangle,\langle 1,1,0\rangle\rangle & \text{otherwise} \end{cases}
\end{array}$$

$$\begin{array}{rcl}
(\mathsf{b})^{\mathcal{S}_{pr}} & := & \langle 3, 1, s_1, (\mathsf{p_0})^{\mathcal{S}_{pr}}, (\mathsf{p_1})^{\mathcal{S}_{pr}}\rangle \text{ where } s_1 \text{ is the} \\
 & & \text{index of the primitive recursive function } (x,y) \mapsto \\
 & & \begin{cases} \langle 3,1,x,y\rangle & \text{if } y \in \mathit{SusPrim} \wedge (y)_1 = 1 \\ \langle 3,1,x,\langle 1,1,0\rangle\rangle & \text{otherwise} \end{cases}
\end{array}$$

$$
\begin{array}{rcl}
(\mathsf{r})^{\mathcal{S}_{pr}} & := & \text{index of the primitive recursive function } x \mapsto \\
& & \begin{cases} \langle 4, 2, (x)_0, \langle 1, 3, 0\rangle\rangle^{\dagger} & \text{if } (x)_0 \in \mathit{SusPrim} \wedge (x)_{0,1} = 1 \wedge \\ & \quad ((x)_1 \notin \mathit{SusPrim} \vee (x)_{1,1} \neq 1) \\ \langle 4, 2, \langle 1, 1, 0\rangle, \overline{\overline{(x)_1}}\rangle^{\dagger} & \text{if } ((x)_0 \notin \mathit{SusPrim} \vee (x)_{0,1} \neq 1) \wedge \\ & \quad (x)_1 \in \mathit{SusPrim} \wedge (x)_{1,1} = 1 \\ \langle 4, 2, (x)_0, \overline{\overline{(x)_1}}\rangle^{\dagger} & \text{otherwise} \end{cases} \\
(\mu)^{\mathcal{S}_{pr}} & := & \text{index of the primitive recursive function } x \mapsto \langle 5, 1, \widetilde{x}\rangle \\
(\mathsf{E}_1)^{\mathcal{S}_{pr}} & := & \text{index of the primitive recursive function } x \mapsto \langle 6, 1, \overline{\overline{x}}\rangle
\end{array}
$$

Note that in the definition of the constants a, b and r above, we have to make case distinctions in order to deal with the fact that some of the arguments may not be unary indices.

By dropping the interpretation of E_1 and replacing *SusPrimEv* by μ*PrimEv* and *SusPrim* by μ*Prim*, respectively, we obtain the structure

$$\mathcal{K}_{pr} = (\mathbb{N}, \mathbb{N}, \mu\mathit{PrimEv}, \dots).$$

Indeed, it is not difficult to see that $\mathcal{K}_{pr}$ validates all the axioms of the theory KLE_{pr}. For a detailed proof, see Steiner [16] and Schlüter [14]. We summarize these observations in the following theorem.

Theorem 15 $\mathcal{K}_{pr} \models \mathsf{KLE}_{pr}$.

The question arises whether $\mathcal{S}_{pr} \models \mathsf{SUS}_{pr}$. In fact, this is the case, but we will only prove this in detail in Section 5, where formalizations of $\mathcal{K}_{pr}$ and $\mathcal{S}_{pr}$ in appropriate subsystems of analysis are presented.

It is not immediate that $\mathcal{S}_{pr}$ is a model of SUS_{pr}, because we have quantifiers ranging over arbitrary functions in the evaluation function *SusPrimEv*. Suppose that f is a unary index of *SusPrim*. Then the following equation holds:

$$(*) \qquad (\mathsf{E}_1 f x)^{\mathcal{S}_{pr}} = \begin{cases} 0 & \text{if } (\exists G)(\forall z)([\overline{\overline{f}}](x, G(z'), G(z)) = 0) \\ 1 & \text{otherwise} \end{cases}$$

However, in order to validate the axioms of E_1 as spelled out in Section 2, we need the following equation to hold:

$$(**) \quad (\mathsf{E}_1 f x)^{\mathcal{S}_{pr}} = \begin{cases} 0 & \text{if } (\exists g \in \mathit{SusPrim})(\forall z)([\overline{\overline{f}}](x, [g](z'), [g](z)) = 0) \\ 1 & \text{otherwise} \end{cases}$$

Observe that if we tried to define the extension of $\mathsf{E}_1 fx$ inductively in order to satisfy $(**)$, a quantification over *all extensions* of indices $g \in \mathit{SusPrim}$ does not make sense, because some of these extensions might only be defined at later stages of the inductive definition. Clearly, we are confronted with a typical *impredicative* phenomenon.

After having formalized the application relation in Π^1_1-CA, we will see that both conditions $(*)$ and $(**)$ are equivalent (cf. Theorem 34, "inside-outside property") and, hence, it will follow that $\mathcal{S}_{pr}$ validates the axioms of SUS_{pr}.

Theorem 16 $\mathcal{S}_{pr} \models \mathsf{SUS}_{pr}$.

Observe that, indeed, the models provided in this section satisfy the two axioms (i) $(\forall x)\mathsf{N}(x)$ and (ii) $(\forall x, y)xy\downarrow$. This is radically different in the context of a stronger applicative setting such as BON, which refutes (i) under the additional assumption of totality of application, (ii).

This concludes our discussion on models of KLE_{pr} and SUS_{pr}. Some details of these model constructions will be needed in the exact upper bound computations in the next section of this paper.

5 Upper bounds

In this section we provide upper bound computations for KLE_{pr} and SUS_{pr}. In particular, we carry out formalized model constructions for KLE_{pr} in Π^1_0-CA and SUS_{pr} in Π^1_1-CA.

5.1 Embedding KLE_{pr} into Π^1_0-CA

Our goal is to formalize the expression $[e](\vec{b}) = c$ in Π^1_0-CA for any n-ary index e of $\mu\mathit{Prim}$. For this purpose, we will work with triples

$$s = \langle e, \langle b_0, \ldots, b_{n-1}\rangle, c\rangle$$

and define for every natural number l a set X so that for all $i \leq l$, $(X)_i$ contains all triples with $\mathit{lev}(e) \leq i$. Recall that the slices $(X)_i$ of X are defined by $s \in (X)_i \leftrightarrow \langle s, i\rangle \in X$.

In the case of $\mathsf{BON}(\mu)$ a hierarchy with finite levels is not sufficient. This is witnessed by the fact that the ω-jump (and much more) is definable in models of $\mathsf{BON}(\mu)$. This latter fact is due to the presence of the s combinator, which allows for diagonalisation at limit ordinals; this diagonalisation, however, cannot be obtained by primitive recursive means. For the model construction in the case of $\mathsf{BON}(\mu)$, cf. Feferman and Jäger [9].

The following formulas $\mathcal{A}$ and $\mathcal{B}$ will prepare the construction of the intended hierarchy.

Definition 17 For describing the triples s with indices of level 0 we define the Π^0_0 formula $\mathcal{A}(s)$ as follows:

$$\begin{aligned}
\mathcal{A} &:= \mathcal{A}_0 \wedge (s)_0 \in \mu \mathit{Prim} \wedge \mathcal{A}_1 \\
\mathcal{A}_0 &:= s \in \mathit{Seq} \wedge \mathit{lh}(s) = 3 \wedge (s)_1 \in \mathit{Seq} \wedge \mathit{lh}((s)_1) = (s)_{0,1} \\
\mathcal{A}_1 &:= [(s)_{0,0} = 0 \wedge (s)_2 = (s)_{1,0} + 1] \vee \\
&\quad\ [(s)_{0,0} = 1 \wedge (s)_2 = (s)_{0,2}] \vee \\
&\quad\ [(s)_{0,0} = 2 \wedge (s)_2 = (s)_{1,(s)_{0,2}}]
\end{aligned}$$

Definition 18 For describing the triples s with indices of level less than or equal to $l{+}1$ we define the arithmetic $\mathcal{L}_2$ formula $\mathcal{B}(U, s, l)$ as follows:

$$\begin{aligned}
\mathcal{B} &:= s \in U \vee [\mathcal{B}_0 \wedge (s)_0 \in \mu \mathit{Prim} \wedge (\mathcal{B}_1 \vee \mathcal{B}_2 \vee \mathcal{B}_3)] \\
\mathcal{B}_0 &:= \mathcal{A}_0 \wedge \mathit{lev}((s)_0) = l{+}1 \\
\mathcal{B}_1 &:= (s)_{0,0} = 3 \wedge (\exists a)[\langle (s)_{0,2}, a, (s)_2 \rangle \in U \wedge \\
&\qquad (\forall b < \mathit{lh}((s)_0) \dot{-} 3)(\langle (s)_{0,b+3}, (s)_1, (a)_b \rangle \in U)] \\
\mathcal{B}_2 &:= (s)_{0,0} = 4 \wedge (\exists a)(\exists b \leq s)[a \in \mathit{Seq} \wedge \\
&\qquad \mathit{lh}(a) = (s)_{1,\mathit{lh}((s)_1) \dot{-} 1} + 1 \wedge (s)_2 = (a)_{\mathit{lh}(a) \dot{-} 1} \wedge \\
&\qquad (s)_1 = b * \langle (s)_{1,\mathit{lh}((s)_1) \dot{-} 1} \rangle \wedge \langle (s)_{0,2}, b, (a)_0 \rangle \in U \wedge \\
&\qquad (\forall c < \mathit{lh}(a) \dot{-} 1)(\langle (s)_{0,3}, b * \langle c, (a)_c \rangle, (a)_{c+1} \rangle \in U)] \\
\mathcal{B}_3 &:= (s)_{0,0} = 5 \wedge [[(\forall a)(\langle (s)_{0,2}, (s)_1 * \langle a \rangle, 0 \rangle \notin U) \wedge (s)_2 = 0] \vee \\
&\qquad [\langle (s)_{0,2}, (s)_1 * \langle (s)_2 \rangle, 0 \rangle \in U \wedge \\
&\qquad (\forall a)(\langle (s)_{0,2}, (s)_1 * \langle a \rangle, 0 \rangle \in U \rightarrow a \geq (s)_2)]]
\end{aligned}$$

Now we are able to define the hierarchy formula $\mathcal{H}_0$ which describes sets of triples $\langle e, \langle \vec{b} \rangle, c \rangle$ satisfying $[e](\vec{b}) = c$ below a certain level u.

Definition 19 $\mathcal{H}_0(W, u)$ is the formula specifying that the slices $(W)_l$ contain the triples with indices of level l for all $l \leq u$:

$$\begin{aligned}
\mathcal{H}_0 &:= (\forall x)[(x \in (W)_0 \leftrightarrow \mathcal{A}(x)) \wedge \\
&\qquad (\forall l < u)(x \in (W)_{l+1} \leftrightarrow \mathcal{B}((W)_l, x, l))]
\end{aligned}$$

Indeed, by exploiting Σ^1_1 complete induction on the natural numbers, one derives the existence of arbitrary finite segments of the hierarchy w.r.t. $\mathcal{H}_0$.

Lemma 20 Π^1_0-CA *proves* $(\forall z)(\exists X)\mathcal{H}_0(X,z)$.

Proof We establish this claim by Σ^1_1 induction on z. In the base case $z = 0$ we take X to be the set $\{(x,0) : \mathcal{A}(x)\}$, which exists by arithmetic comprehension since $\mathcal{A}$ is Π^0_0.

For the induction step, assume that we are given a set Y so that $\mathcal{H}_0(Y,z)$. By arithmetic comprehension, we are allowed to form the set X,

$$X \ := \ Y \cup \{(x, z+1) : \mathcal{B}((Y)_z, x, z)\}$$

and, moreover, one easily verifies that $\mathcal{H}_0(X, z+1)$ holds. The induction step and the proof of this lemma are now complete. $\square$

By construction, our hierarchy with respect to $\mathcal{H}_0$ is unique as is stated without proof in the following lemma. Of course, $\mathcal{H}_0$ hierarchies are also increasing.

Lemma 21 Π^1_0-CA *proves*

$$\mathcal{H}_0(W_0,u_0) \wedge \mathcal{H}_0(W_1,u_1) \wedge u_0 \leq u_1 \rightarrow (\forall l \leq u_0)((W_0)_l = (W_1)_l)$$

The stage is now set in order to define the interpretation of the application operation $\cdot$ of L for the embedding of KLE_{pr} into Π^1_0-CA. We will use an $\mathcal{L}_2$ formula $\mu\mathsf{App}$ which is our formalization of $\mu PrimEv$ in Π^1_0-CA.

Definition 22 We define $\mu\mathsf{App}(u,v,w)$ to be the following formula:

$$\begin{aligned}\mu\mathsf{App} \ := \ & [u \in \mu Prim \wedge (u)_1 = 1 \wedge \\ & \quad (\exists X)(\mathcal{H}_0(X, lev(u)) \wedge \langle u, \langle v\rangle, w\rangle \in (X)_{lev(u)})] \vee \\ & [(u \notin \mu Prim \vee (u)_1 \neq 1) \wedge w = 0]\end{aligned}$$

It is crucial that the so-obtained application relation is functional in its third argument, a fact that can be formally verified in Π^1_0-CA.

Lemma 23 Π^1_0-CA *proves*

$$\mu\mathsf{App}(u,v,w_0) \wedge \mu\mathsf{App}(u,v,w_1) \rightarrow w_0 = w_1$$

Proof If $u \notin \mu Prim$ or $(u)_1 \neq 1$ then $\mu\mathsf{App}(u,v,w)$ yields $w = 0$ and we are done. If u is a unary index of $\mu Prim$, we prove the claim by induction on the level $l := lev(u)$ of u:

In the case $l = 0$ we have to deal with the three cases $(u)_0 = 0$ (successor), $(u)_0 = 1$ (constant function), and $(u)_0 = 2$ (projection), which are readily handled. In the induction step from l to $l+1$ we need to take into account the cases $(u)_0 = 3$ (composition) and $(u)_0 = 5$ (non-constructive μ operator) and can easily obtain the desired result by induction hypothesis. In the case $(u)_0 = 4$ (primitive recursion) we again apply the induction hypothesis, but, in addition, we need to do a subsidiary induction on the recursion argument of the given primitive recursion. $\square$

We are now going to define a translation $\star$ of formulas from L to $\mathcal{L}_2$. In a first step, we will assign to each L term t an $\mathcal{L}_2$ formula $V_t(u)$ which expresses that t has the value u with respect to the application relation $\mu\mathsf{App}$. The definition of $V_t(u)$ is by induction on the build-up of t. For the translation of the constants c of L we make use of the numeral corresponding to the interpretation of c in the model $\mathcal{K}_{pr}$. As usual, we let $\overline{n}$ denote the numeral associated to the natural number n.

Definition 24 For every L term t we define the $\mathcal{L}_2$ formula $V_t(u)$, so that the variable u does not occur in t, by induction on the build-up of t as follows:

1. If t is a variable of L then $V_t(u) := (u = t)$.
2. If t is a constant of L then $V_t(u) := (u = \overline{t^{\mathcal{K}_{pr}}})$.
3. If t is the L term $<r, s>$ then
$$V_t(u) := (\exists x, y)[V_r(x) \wedge V_s(y) \wedge u = \langle x, y \rangle].$$
4. If t is the L term (rs) then
$$V_t(u) := (\exists x, y)[V_r(x) \wedge V_s(y) \wedge \mu\mathsf{App}(x, y, u)].$$

Let us mention that for any L term t we have that Π^1_0-CA proves $(\exists! x)V_t(x)$. This can be accomplished by an easy (meta-)induction on the build-up of t.

Definition 25 For every L formula A we define the $\mathcal{L}_2$ formula $A^\star$ by induction on the build-up of A as follows:

1. If A is the formula $\mathsf{N}(t)$ or $t\downarrow$ then $A^\star$ is $(\exists x)V_t(x)$.
2. If A is the formula $(s = t)$ then $A^\star$ is $(\exists x)(V_s(x) \wedge V_t(x))$.
3. If A is the formula $\neg B$ then $A^\star$ is $\neg B^\star$.
4. If A is the formula $B\ j\ C$ then $A^\star$ is $(B^\star\ j\ C^\star)$ for $j \in \{\vee, \wedge, \rightarrow\}$.
5. If A is the formula $(Qx)B$ then $A^\star$ is $(Qx)B^\star$ for $Q \in \{\exists, \forall\}$.

The following lemma shows that our translation has natural properties.

Lemma 26 *We have for all $\mathcal{L}_1$ sentences A:*

$$\Pi^1_0\text{-}\mathsf{CA} \vdash A \leftrightarrow (A^{\mathsf{N}})^\star.$$

Proof This claim is verified by induction on the build-up of A. □

We are now ready to state the embedding of KLE_{pr} into Π^1_0-CA, which will yield that KLE_{pr} and Π^1_0-CA prove the same $\mathcal{L}_1$ sentences.

Theorem 27 *We have for all* L *formulas A:*

$$\mathsf{KLE}_{pr} \vdash A \quad \Longrightarrow \quad \Pi^1_0\text{-}\mathsf{CA} \vdash A^\star.$$

Proof This claim is proved by induction on the length of a derivation of A in KLE_{pr}. If A is a non-logical axiom of KLE_{pr} different from induction, one proceeds by directly working with the formalized model construction of $\mathcal{K}_{pr}$ in Π^1_0-CA to show that $A^\star$ holds. If A is an instance of (L-$\mathsf{I_N}$), then $A^\star$ is (equivalent to) an instance of ($\mathcal{L}_2$-$\mathsf{I_N}$) and, hence, is derivable in Π^1_0-CA. □

Together with Theorem 6 and Lemma 26 we immediately obtain the following corollary.

Corollary 28 *We have that* KLE_{pr} *and* Π^1_0-CA *prove the same $\mathcal{L}_1$ sentences.*

5.2 Embedding SUS_{pr} into Π^1_1-CA

For our embedding of SUS_{pr} in Π^1_1-CA we will follow the same pattern as in the previous paragraph; accordingly, we will now formalize the expression $[e](\vec{b}) = c$ in Π^1_1-CA for any n-ary index e of *SusPrim*. The construction, however, will be more involved due to the presence of E_1. Nevertheless, we can define a hierarchy of finite levels in Π^1_1-CA which is sufficient for modelling SUS_{pr}. Hence, the situation heavily differs from the corresponding treatment of E_1 on the basis of BON, cf. Jäger and Strahm [13].

In the following we prepare the construction of our new hierarchy. For this purpose, we can reuse the formula $\mathcal{A}$ from the last paragraph in order to deal with triples with indices of level 0. For the other triples, we introduce a new formula $\mathcal{C}$ which extends the formula $\mathcal{B}$ from above.

Definition 29 For describing the triples s with indices of level less than or equal to $l{+}1$ we define the formula $\mathcal{C}(U, s, l)$ as follows:

$$\begin{aligned}
\mathcal{C} &:= s \in U \lor [\mathcal{B}_0 \land (s)_0 \in \mathit{SusPrim} \land (\mathcal{B}_1 \lor \mathcal{B}_2 \lor \mathcal{B}_3 \lor \mathcal{C}_0 \lor \mathcal{C}_1)] \\
\mathcal{C}_0 &:= (s)_{0,0} = 6 \land (s)_2 = 0 \land \\
&\quad\ (\exists G)(\forall a)(\langle (s)_{0,2}, (s)_1 * \langle G(a'), G(a)\rangle, 0\rangle \in U) \\
\mathcal{C}_1 &:= (s)_{0,0} = 6 \land (s)_2 = 1 \land \\
&\quad\ (\forall G)(\exists a)(\langle (s)_{0,2}, (s)_1 * \langle G(a'), G(a)\rangle, 0\rangle \notin U)
\end{aligned}$$

The formulas $\mathcal{B}_0$, $\mathcal{B}_1$, $\mathcal{B}_2$, and $\mathcal{B}_3$ are the same as in Definition 18.

The hierarchy formula $\mathcal{H}_1$ is now defined in an analogous manner to $\mathcal{H}_0$, with $\mathcal{B}$ replaced by $\mathcal{C}$.

Definition 30 $\mathcal{H}_1(W, u)$ is the formula specifying that the slices $(W)_l$ contain the triples with indices of level l for all $l \leq u$:

$$\begin{aligned}
\mathcal{H}_1 &:= (\forall x)[(x \in (W)_0 \leftrightarrow \mathcal{A}(x)) \land \\
&\qquad (\forall l < u)(x \in (W)_{l+1} \leftrightarrow \mathcal{C}((W)_l, x, l))]
\end{aligned}$$

The next lemma is the analogue of Lemma 20 in the context of SUS_{pr} and Π^1_1-CA. Thereby, the stronger form of comprehension makes it possible to deal with the formula $\mathcal{C}$.

Lemma 31 Π^1_1-CA *proves* $(\forall z)(\exists X)\mathcal{H}_1(X, z)$.

PROOF We reason informally in Π^1_1-CA and prove the claim by induction on z. The case $z = 0$ is immediate by arithmetic comprehension using the formula $\mathcal{A}$. For the induction step, assume that we are given a set Y satisfying $\mathcal{H}_1(Y, z)$. Now observe that the formula $\mathcal{C}$ is a boolean combination of Σ^1_1 and Π^1_1 formulas and, hence, comprehension is available in Π^1_1-CA with respect to $\mathcal{C}$. Consequently, the required set X satisfying $\mathcal{H}_1(X, z{+}1)$ exists in Π^1_1-CA, and, hence, we are done. □

As before, our hierarchy with respect to $\mathcal{H}_1$ is unique and increasing.

We are now ready to define the interpretation of the application operation for the embedding of SUS_{pr} into Π^1_1-CA. The following definition is the precise formalization of *SusPrimEv* in Π^1_1-CA, making use of the new hierarchy formula $\mathcal{H}_1$.

Definition 32 We define $\mathsf{SusApp}(u, v, w)$ to be the following formula:

$$\begin{aligned}
\mathsf{SusApp} &:= [u \in \mathit{SusPrim} \land (u)_1 = 1 \land \\
&\qquad (\exists X)(\mathcal{H}_1(X, \mathit{lev}(u)) \land \langle u, \langle v\rangle, w\rangle \in (X)_{\mathit{lev}(u)})] \lor \\
&\quad\ [(u \notin \mathit{SusPrim} \lor (u)_1 \neq 1) \land w = 0]
\end{aligned}$$

As expected, by construction, SusApp is functional in its third argument.

Lemma 33 Π^1_1-CA *proves*

$$\mathsf{SusApp}(u, v, w_0) \wedge \mathsf{SusApp}(u, v, w_1) \to w_0 = w_1$$

In a next step we have to verify that the so-defined application relation indeed satisfies the axioms about E_1. In particular, we have to deal with the impredicative phenomenon which we have already discussed at the end of Section 4. We will need a so-called *inside-outside property* which is analogous to the treatment of E_1 in [13] (Theorem 16). We start with some preparatory definitions:

$$\begin{array}{rcl} [e](b) = c & := & \mathsf{SusApp}(e, b, c) \\ \mathsf{CDC}(c, e, a) & := & (\forall x)([e](\langle\langle a, [c](x')\rangle, [c](x)\rangle) = 0) \\ \mathsf{FDC}(F, e, a) & := & (\forall x)([e](\langle\langle a, F(x')\rangle, F(x)\rangle) = 0) \end{array}$$

$\mathsf{CDC}(c, e, a)$ signifies that c codes an infinite descending sequence in the sense of SusApp with respect to the relation given by e and parameter a. The meaning of $\mathsf{FDC}(F, e, a)$ is analogous but now the infinite descending chain is an *arbitrary* function F.

We are now ready to state the crucial property which is needed to verify that $\mathcal{S}_{pr}$ is a model of SUS_{pr}. In the following equivalence, the direction from left to right is immediate by Lemma 31 and Lemma 33. In order to establish the direction from right to left one can essentially follow the argument in [13] and make use of a standard leftmost branch construction to define an infinite branch by primitive recursion (in E_1). The details in the given setting are presented in [16].

Theorem 34 (Inside-outside property) Π^1_1-CA *proves*

$$(\exists c)\mathsf{CDC}(c, e, a) \leftrightarrow (\exists F)\mathsf{FDC}(F, e, a)$$

In order to embed SUS_{pr} into Π^1_1-CA we can employ the same translation $\star$ as before, but of course using SusApp instead of $\mu\mathsf{App}$. For reasons of notational simplicity, we name the translation $\star$ as before.

We obtain the following property of translations in analogy to Lemma 26.

Lemma 35 *We have for all $\mathcal{L}_1$ sentences A:*

$$\Pi^1_1\text{-}\mathsf{CA} \vdash A \leftrightarrow (A^{\mathsf{N}})^\star.$$

Our embedding is now complete.

Theorem 36 *We have for all* L *formulas A:*

$$\mathsf{SUS}_{pr} \vdash A \quad \Longrightarrow \quad \Pi^1_1\text{-}\mathsf{CA} \vdash A^\star.$$

PROOF The embedding of SUS_{pr} into Π^1_1-CA is analogous to the one of KLE_{pr} into Π^1_0-CA except for the additional treatment of E_1. The fact that the $\star$ translations of the E_1 axioms is derivable in Π^1_1-CA is due to the *inside-outside property* (Theorem 34). □

Together with Theorem 7 and Lemma 35 we obtain the following corollary.

Corollary 37 *We have that* SUS_{pr} *and* Π^1_1-CA *prove the same* $\mathcal{L}_1$ *sentences.*

6 Concluding remarks

In previous research on higher type functionals in applicative theories, the so-called principle of *set induction* has played a prominent role, cf. [9, 11, 12]. By set induction (S-I_N) on the natural numbers N we mean the principle

$$f \in \mathcal{P}(\mathsf{N}) \land f0 = 0 \land (\forall x \in \mathsf{N})(fx = 0 \to f(x') = 0) \;\to\; (\forall x \in \mathsf{N})(fx = 0),$$

i.e., complete induction is available for (total characteristic functions of) sets of natural numbers. What is the proof-theoretic strength of KLE_{pr} and SUS_{pr} with induction on the natural numbers restricted to sets? In order to make this a sensible question, we have to assume that the recursion operator r of PRON transforms total operations on N into total operations on N, a property which we can show with (L-I_N), but not by means of (S-I_N).

Indeed, observe that this totality assertion of r is built into the definition of BON, and in the sequel we let PRON^t denote PRON augmented by this totality property of r (cf. [16] for details). Indeed, the applicative systems based on PRON^t plus set induction are directly contained in the corresponding theories axiomatized by means of BON, cf. [16]. Hence, we get the following proof-theoretic equivalences by making use of the previously known upper bound results for $\mathsf{BON}(\mu)$ and $\mathsf{BON}(\mu, \mathsf{E}_1)$ with set induction, cf. [9, 13]:

$$\begin{array}{rcccl} \mathsf{BON}(\mu) + (\mathsf{S\text{-}I_N}) & \equiv & \Pi^1_0\text{-}\mathsf{CA}_0 & \equiv & \mathsf{PRON^t}(\mu) + (\mathsf{S\text{-}I_N}) \\ \mathsf{BON}(\mu, \mathsf{E}_1) + (\mathsf{S\text{-}I_N}) & \equiv & \Pi^1_1\text{-}\mathsf{CA}_0 & \equiv & \mathsf{PRON^t}(\mu, \mathsf{E}_1) + (\mathsf{S\text{-}I_N}) \end{array}$$

Summing up, the decrease of proof-theoretic strength of μ or E_1 on the basis of PRON instead of BON is a phenomenon which only occurs in the presence of induction principles stronger than (S-I_N), such as full formula induction (L-I_N).

In fact, the increase of proof-theoretic strength in the presence of (L-$\mathsf{I_N}$) (and our type two functionals) in going from PRON to BON is solely due to the combinator s, which is the crucial difference between PRON and BON:

$$(\mathsf{S}) \qquad \mathsf{s}xy\downarrow \wedge\ \mathsf{s}xyz \simeq xz(yz).$$

A combinator which also embodies the full strength of s is the universal application combinator u, given by its defining equation

$$(\mathsf{Univ}) \qquad \mathsf{u}{<}x, y{>} \simeq xy.$$

Hence, adding (Univ) to $\mathsf{PRON}(\mu)$ and $\mathsf{PRON}(\mu, \mathsf{E}_1)$ causes a jump in proof-strength from Π^1_0-CA to Δ^1_1-CA and Π^1_1-CA to Δ^1_2-CA, respectively:

$$\begin{aligned} \mathsf{PRON}(\mu) + (\mathsf{Univ}) + (\text{L-}\mathsf{I_N}) &\equiv \Delta^1_1\text{-}\mathsf{CA} \\ \mathsf{PRON}(\mu, \mathsf{E}_1) + (\mathsf{Univ}) + (\text{L-}\mathsf{I_N}) &\equiv \Delta^1_2\text{-}\mathsf{CA} \end{aligned}$$

Finally, let us mention that *without* any type two functionals, BON and PRON have the same proof-theoretic strength, no matter which form of complete induction on the natural numbers is assumed. Namely, we get systems of strength PRA and PA in the presence of (S-$\mathsf{I_N}$) and (L-$\mathsf{I_N}$), respectively.

References

[1] Beeson, M. J. *Foundations of Constructive Mathematics: Metamathematical Studies.* Springer, Berlin, 1985.

[2] Cantini, A. Polytime, combinatory logic and positive safe induction. *Archive for Mathematical Logic 41*, 2 (2002), 169–189.

[3] Cantini, A. Choice and uniformity in weak applicative theories. In *Logic Colloquium '01*, M. Baaz, S. Friedman, and J. Krajíček, Eds., vol. 20 of *Lecture Notes in Logic.* Association for Symbolic Logic, 2005.

[4] Feferman, S. A language and axioms for explicit mathematics. In *Algebra and Logic*, J. Crossley, Ed., vol. 450 of *Lecture Notes in Mathematics.* Springer, Berlin, 1975, pp. 87–139.

[5] Feferman, S. A theory of variable types. *Revista Colombiana de Matemáticas 19* (1975), 95–105.

[6] Feferman, S. Recursion theory and set theory: a marriage of convenience. In *Generalized recursion theory II, Oslo 1977*, J. E. Fenstad, R. O. Gandy, and G. E. Sacks, Eds., vol. 94 of *Stud. Logic Found. Math.* North Holland, Amsterdam, 1978, pp. 55–98.

[7] Feferman, S. Constructive theories of functions and classes. In *Logic Colloquium '78*, M. Boffa, D. van Dalen, and K. McAloon, Eds. North Holland, Amsterdam, 1979, pp. 159–224.

[8] Feferman, S. Definedness. *Erkenntnis 43* (1995), 295–320.

[9] Feferman, S., and Jäger, G. Systems of explicit mathematics with non-constructive μ-operator. Part I. *Annals of Pure and Applied Logic 65*, 3 (1993), 243–263.

[10] Glass, T., and Strahm, T. Systems of explicit mathematics with non-constructive μ-operator and join. *Annals of Pure and Applied Logic 82* (1996), 193–219.

[11] Jäger, G., and Strahm, T. Totality in applicative theories. *Annals of Pure and Applied Logic 74*, 2 (1995), 105–120.

[12] Jäger, G., and Strahm, T. Some theories with positive induction of ordinal strength $\varphi\omega 0$. *Journal of Symbolic Logic 61*, 3 (1996), 818–842.

[13] Jäger, G., and Strahm, T. The proof-theoretic strength of the Suslin operator in applicative theories. In *Reflections on the Foundations of Mathematics: Essays in Honor of Solomon Feferman*, W. Sieg, R. Sommer, and C. Talcott, Eds., vol. 15 of *Lecture Notes in Logic*. Association for Symbolic Logic, 2002, pp. 270–292.

[14] Schlüter, A. A theory of rules for enumerated classes of functions. *Archive for Mathematical Logic 34* (1995), 47–63.

[15] Simpson, S. G. *Subsystems of Second Order Arithmetic*. Perspectives in Mathematical Logic. Springer-Verlag, 1998.

[16] Steiner, D. Proof-theoretic strength of PRON with various extensions. Master's thesis, Institut für Informatik und angewandte Mathematik, Universität Bern, 2001. Available at `http://www.iam.unibe.ch/til/publications`.

[17] Strahm, T. Theories with self-application and computational complexity. *Information and Computation 185* (2003), 263–297.

[18] Strahm, T. A proof-theoretic characterization of the basic feasible functionals. *Theoretical Computer Science 329* (2004), 159–176.

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