Levitz class and punctually non-standard models of natural numbers

Ivan Georgiev[1]

Sofia University "St. Kliment Ohridski",
Faculty of Mathematics and Informatics

Logic conference on the occasion of the 70th anniversary of Prof. Tinko Tinchev

Gyolechitsa, 5-8 December 2025

---

[1] This work is supported by NextGenerationEU, through the NRRP of Bulgaria, project no. BG-RRP-2.004-0008-C01 - 70/123/195

Some memories

13 years ago

Work team

This is a joint project with:

- Stefan Vatev;
- Nikolay Bazhenov (Sobolev Institute, Novosibirsk);
- Dariusz Kalociński (Polish Academy of Sciences);
- Michał Wrocławski (University of Warsaw).

Plan for the talk

1. introduce Skolem's class of arithmetic functions;

Plan for the talk

1. introduce Skolem's class of arithmetic functions;
2. consider the subclass of Levitz, in which unique normal forms exist;

Plan for the talk

1. introduce Skolem's class of arithmetic functions;
2. consider the subclass of Levitz, in which unique normal forms exist;
3. present an ad-hoc construction for punctual copies of $(\mathbb{N}, S)$ and discuss its limitations;

Plan for the talk

1. introduce Skolem's class of arithmetic functions;
2. consider the subclass of Levitz, in which unique normal forms exist;
3. present an ad-hoc construction for punctual copies of $(\mathbb{N}, S)$ and discuss its limitations;
4. introduce the islands and archipelago technique for overcoming these limitations.

Skolem's class

Let $\mathbb{N}^+$ be the set of positive integers.

Skolem's class

Let $\mathbb{N}^+$ be the set of positive integers.

Definition

Skolem's class $\mathfrak{A}$ of arithmetic functions $\mathbb{N}^+ \to \mathbb{N}^+$ is the least class, which contains the constant $\lambda x.1$ and the identity $\lambda x.x$ and is closed under sum, product and exponentiation:

$$\begin{aligned} f, g \in \mathfrak{A} \ \Rightarrow\ & \lambda x.[f(x) + g(x)] \in \mathfrak{A}, \\ & \lambda x.[f(x) \cdot g(x)] \in \mathfrak{A}, \\ & \lambda x.[f(x)^{g(x)}] \in \mathfrak{A}. \end{aligned}$$

Skolem's class

Let $\mathbb{N}^+$ be the set of positive integers.

Definition

Skolem's class $\mathfrak{A}$ of arithmetic functions $\mathbb{N}^+ \to \mathbb{N}^+$ is the least class, which contains the constant $\lambda x.1$ and the identity $\lambda x.x$ and is closed under sum, product and exponentiation:

$$\begin{aligned} f, g \in \mathfrak{A} \ \Rightarrow\ & \lambda x.[f(x) + g(x)] \in \mathfrak{A}, \\ & \lambda x.[f(x) \cdot g(x)] \in \mathfrak{A}, \\ & \lambda x.[f(x)^{g(x)}] \in \mathfrak{A}. \end{aligned}$$

The set $\mathbb{N}^+[x]$ is a proper subset of $\mathfrak{A}$.

Skolem's class

Let $\mathbb{N}^+$ be the set of positive integers.

Definition

Skolem's class $\mathfrak{A}$ of arithmetic functions $\mathbb{N}^+ \to \mathbb{N}^+$ is the least class, which contains the constant $\lambda x.1$ and the identity $\lambda x.x$ and is closed under sum, product and exponentiation:

$$\begin{aligned} f, g \in \mathfrak{A} \ \Rightarrow\ & \lambda x.[f(x) + g(x)] \in \mathfrak{A}, \\ & \lambda x.[f(x) \cdot g(x)] \in \mathfrak{A}, \\ & \lambda x.[f(x)^{g(x)}] \in \mathfrak{A}. \end{aligned}$$

The set $\mathbb{N}^+[x]$ is a proper subset of $\mathfrak{A}$.
Different terms may represent the same function, for example:

$$x \cdot (x + x) \ = \ x \cdot x + x \cdot x \ = \ (1 + 1) \cdot (x \cdot x).$$

Equality and domination

We say that g *eventually dominates* f, denoted $f \preccurlyeq g$, if there exists n_0, such that

$$\forall x \geq n_0 \, [f(x) \leq g(x)].$$

Equality and domination

We say that g *eventually dominates* f, denoted $f \preccurlyeq g$, if there exists n_0, such that

$$\forall x \geq n_0 \, [f(x) \leq g(x)].$$

The number n_0 is called *a* (f, g)*-domination witness*.

Equality and domination

We say that g *eventually dominates* f, denoted $f \preccurlyeq g$, if there exists n_0, such that

$$\forall x \geq n_0 \, [f(x) \leq g(x)].$$

The number n_0 is called *a* (f, g)*-domination witness*.
If in addition

$$\forall x \geq n_0 \, [f(x) < g(x)],$$

n_0 is called a *strict* (f, g)-domination witness.

Equality and domination

We say that g *eventually dominates* f, denoted $f \preccurlyeq g$, if there exists n_0, such that

$$\forall x \geq n_0 \, [f(x) \leq g(x)].$$

The number n_0 is called *a* (f, g)*-domination witness*.
If in addition

$$\forall x \geq n_0 \, [f(x) < g(x)],$$

n_0 is called a *strict* (f, g)-domination witness.

As usual, we denote $f \prec g$ if $f \preccurlyeq g$ & $g \not\preccurlyeq f$.

Equality and domination

We say that g *eventually dominates* f, denoted $f \preccurlyeq g$, if there exists n_0, such that

$$\forall x \geq n_0 \; [f(x) \leq g(x)].$$

The number n_0 is called *a* (f, g)*-domination witness*.
If in addition

$$\forall x \geq n_0 \; [f(x) < g(x)],$$

n_0 is called a *strict* (f, g)-domination witness.

As usual, we denote $f \prec g$ if $f \preccurlyeq g$ & $g \not\preccurlyeq f$.

Note that $f \preccurlyeq g$ & $g \preccurlyeq f$ iff f and g are almost equal (they differ only on a finite set).

Domination restricted to $\mathfrak{A}$

Theorem (Saks & Tarski, Richardson)

$\preccurlyeq$ *is a linear order on* $\mathfrak{A}$.

Domination restricted to $\mathfrak{A}$

Theorem (Saks & Tarski, Richardson)

$\preccurlyeq$ *is a linear order on* $\mathfrak{A}$*.*

Proof.

There exists an algorithm, which given two terms f and g belonging to $\mathfrak{A}$, computes a bound k on *the number* of the common roots of f and g.

Domination restricted to $\mathfrak{A}$

Theorem (Saks & Tarski, Richardson)

$\preccurlyeq$ *is a linear order on* $\mathfrak{A}$*.*

Proof.

There exists an algorithm, which given two terms f and g belonging to $\mathfrak{A}$, computes a bound k on *the number* of the common roots of f and g.
In particular, there exists n_0, such that $f(x) \neq g(x)$ for all $x \geq n_0$.
By continuity, we have $f(x) < g(x)$ for all $x \geq n_0$ $(f \prec g)$ or $f(x) > g(x)$ for all $x \geq n_0$ $(g \prec f)$.

Domination restricted to $\mathfrak{A}$

Theorem (Saks & Tarski, Richardson)

$\preccurlyeq$ *is a linear order on* $\mathfrak{A}$*.*

Proof.

There exists an algorithm, which given two terms f and g belonging to $\mathfrak{A}$, computes a bound k on *the number* of the common roots of f and g.
In particular, there exists n_0, such that $f(x) \neq g(x)$ for all $x \geq n_0$.
By continuity, we have $f(x) < g(x)$ for all $x \geq n_0$ $(f \prec g)$ or $f(x) > g(x)$ for all $x \geq n_0$ $(g \prec f)$.

□

Note that the algorithm implies that f and g are almost equal iff $f = g$, so that $\preccurlyeq$ is antisymmetric on $\mathfrak{A}$.

Domination restricted to $\mathfrak{A}$

Theorem (Saks & Tarski, Richardson)

$\preccurlyeq$ *is a linear order on* $\mathfrak{A}$*.*

Proof.

There exists an algorithm, which given two terms f and g belonging to $\mathfrak{A}$, computes a bound k on *the number* of the common roots of f and g.
In particular, there exists n_0, such that $f(x) \neq g(x)$ for all $x \geq n_0$.
By continuity, we have $f(x) < g(x)$ for all $x \geq n_0$ $(f \prec g)$ or $f(x) > g(x)$ for all $x \geq n_0$ $(g \prec f)$.

□

Note that the algorithm implies that f and g are almost equal iff $f = g$, so that $\preccurlyeq$ is antisymmetric on $\mathfrak{A}$.

For an alternative proof, one can also use Wilkie's theorem on o-minimality of $\mathrm{Th}(\mathbb{R}^{exp})$.

Computability of identity and domination

Note that the identity problem $f = g$? is decidable, because given the upper bound k on the number of common roots,
$f = g$ if and only if $f(i) = g(i)$ for all $1 \leq i \leq k + 1$.

Computability of identity and domination

Note that the identity problem $f = g$? is decidable, because given the upper bound k on the number of common roots,
$f = g$ if and only if $f(i) = g(i)$ for all $1 \leq i \leq k + 1$.

But the domination problem $f \preccurlyeq g$? is much more complex and it is not known to be decidable.

Computability of identity and domination

Note that the identity problem $f = g$? is decidable, because given the upper bound k on the number of common roots,
$f = g$ if and only if $f(i) = g(i)$ for all $1 \leq i \leq k + 1$.

But the domination problem $f \preccurlyeq g$? is much more complex and it is not known to be decidable.
We do not have an algorithm to compute the strict domination witness n_0.

Computability of identity and domination

Note that the identity problem $f = g$? is decidable, because given the upper bound k on the number of common roots,
$f = g$ if and only if $f(i) = g(i)$ for all $1 \leq i \leq k + 1$.

But the domination problem $f \preccurlyeq g$? is much more complex and it is not known to be decidable.
We do not have an algorithm to compute the strict domination witness n_0.
This question is clearly connected with the decidability of Th($\mathbb{R}^{exp}$), which is a major open problem.

Order type of $\mathfrak{A}, \preccurlyeq$

In fact, $\mathfrak{A}$ is well-ordered by $\preccurlyeq$.

Order type of $\mathfrak{A}, \preccurlyeq$

In fact, $\mathfrak{A}$ is well-ordered by $\preccurlyeq$. This was proved by Ehrenfeucht. The proof involves an application of Kruskal's tree theorem.

Order type of $\mathfrak{A}, \preccurlyeq$

In fact, $\mathfrak{A}$ is well-ordered by $\preccurlyeq$. This was proved by Ehrenfeucht. The proof involves an application of Kruskal's tree theorem.

The ordinal of this well-ordering is not known.

Order type of $\mathfrak{A}, \preccurlyeq$

In fact, $\mathfrak{A}$ is well-ordered by $\preccurlyeq$. This was proved by Ehrenfeucht. The proof involves an application of Kruskal's tree theorem.

The ordinal of this well-ordering is not known.

As a simple example: the order type of $\mathbb{N}^+[x]$ is ω^ω.

Levitz class

Definition

The Levitz class $\mathfrak{L}$ of arithmetic functions $\mathbb{N} \to \mathbb{N}$ is the least class, which contains the constants $\lambda x.0$ and $\lambda x.1$ and is closed under sum, product and exponentiation with base x and base $n \geq 2$:

$$\begin{aligned} f, g \in \mathfrak{L} \;\Rightarrow\; & \lambda x.[f(x) + g(x)] \in \mathfrak{L}, \\ & \lambda x.[f(x) \cdot g(x)] \in \mathfrak{L}, \\ & \lambda x.[x^{f(x)}] \in \mathfrak{L}, \\ & \lambda x.[n^{f(x)}] \in \mathfrak{L}. \end{aligned}$$

Levitz class

Definition

The Levitz class $\mathfrak{L}$ of arithmetic functions $\mathbb{N} \to \mathbb{N}$ is the least class, which contains the constants $\lambda x.0$ and $\lambda x.1$ and is closed under sum, product and exponentiation with base x and base $n \geq 2$:

$$\begin{aligned} f, g \in \mathfrak{L} \ \Rightarrow \ & \lambda x.[f(x) + g(x)] \in \mathfrak{L}, \\ & \lambda x.[f(x) \cdot g(x)] \in \mathfrak{L}, \\ & \lambda x.[x^{f(x)}] \in \mathfrak{L}, \\ & \lambda x.[n^{f(x)}] \in \mathfrak{L}. \end{aligned}$$

It is immaterial that we extend the functions from $\mathbb{N}^+$ to $\mathbb{N}$.

Additive and multiplicative primes in $\mathfrak{L}$

A function $f \in \mathfrak{L}, f \neq 0$ is called *an additive prime* if $f = g + h$ implies $g = 0$ or $h = 0$.

Additive and multiplicative primes in $\mathfrak{L}$

A function $f \in \mathfrak{L}, f \neq 0$ is called *an additive prime* if $f = g + h$ implies $g = 0$ or $h = 0$.

A function $f \in \mathfrak{L}, f \neq 0$ is called *a multiplicative prime* if $f = g \cdot h$ implies $g = 1$ or $h = 1$.

Additive and multiplicative primes in $\mathfrak{L}$

A function $f \in \mathfrak{L}, f \neq 0$ is called *an additive prime* if $f = g + h$ implies $g = 0$ or $h = 0$.

A function $f \in \mathfrak{L}, f \neq 0$ is called *a multiplicative prime* if $f = g \cdot h$ implies $g = 1$ or $h = 1$.

Every additive prime $f \neq 1$ has a unique *mutiplicative normal form* $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$, where:

Additive and multiplicative primes in $\mathfrak{L}$

A function $f \in \mathfrak{L}, f \neq 0$ is called *an additive prime* if $f = g + h$ implies $g = 0$ or $h = 0$.

A function $f \in \mathfrak{L}, f \neq 0$ is called *a multiplicative prime* if $f = g \cdot h$ implies $g = 1$ or $h = 1$.

Every additive prime $f \neq 1$ has a unique *mutiplicative normal form* $f = u_1^{f_1} u_2^{f_2} \ldots u_k^{f_k}$, where:

1. each f_i is additive prime;

Additive and multiplicative primes in $\mathfrak{L}$

A function $f \in \mathfrak{L}, f \neq 0$ is called *an additive prime* if $f = g + h$ implies $g = 0$ or $h = 0$.

A function $f \in \mathfrak{L}, f \neq 0$ is called *a multiplicative prime* if $f = g \cdot h$ implies $g = 1$ or $h = 1$.

Every additive prime $f \neq 1$ has a unique *mutiplicative normal form* $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$, where:

1. each f_i is additive prime;
2. each u_i belongs to $\mathbb{N} \setminus \{0, 1\} \cup \{x\}$;

Additive and multiplicative primes in $\mathfrak{L}$

A function $f \in \mathfrak{L}, f \neq 0$ is called *an additive prime* if $f = g + h$ implies $g = 0$ or $h = 0$.

A function $f \in \mathfrak{L}, f \neq 0$ is called *a multiplicative prime* if $f = g \cdot h$ implies $g = 1$ or $h = 1$.

Every additive prime $f \neq 1$ has a unique *mutiplicative normal form* $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$, where:

1. each f_i is additive prime;
2. each u_i belongs to $\mathbb{N} \setminus \{0, 1\} \cup \{x\}$;
3. if $i \neq j$ and $u_i, u_j \in \mathbb{N}$, then $f_i \neq f_j$;

Additive and multiplicative primes in $\mathfrak{L}$

A function $f \in \mathfrak{L}, f \neq 0$ is called *an additive prime* if $f = g + h$ implies $g = 0$ or $h = 0$.

A function $f \in \mathfrak{L}, f \neq 0$ is called *a multiplicative prime* if $f = g \cdot h$ implies $g = 1$ or $h = 1$.

Every additive prime $f \neq 1$ has a unique *mutiplicative normal form* $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$, where:

1. each f_i is additive prime;
2. each u_i belongs to $\mathbb{N} \setminus \{0, 1\} \cup \{x\}$;
3. if $i \neq j$ and $u_i, u_j \in \mathbb{N}$, then $f_i \neq f_j$;
4. if $u_i \in \mathbb{N}$, then $f_i \neq 1$;

Additive and multiplicative primes in $\mathfrak{L}$

A function $f \in \mathfrak{L}, f \neq 0$ is called *an additive prime* if $f = g + h$ implies $g = 0$ or $h = 0$.

A function $f \in \mathfrak{L}, f \neq 0$ is called *a multiplicative prime* if $f = g \cdot h$ implies $g = 1$ or $h = 1$.

Every additive prime $f \neq 1$ has a unique *mutiplicative normal form* $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$, where:

1. each f_i is additive prime;
2. each u_i belongs to $\mathbb{N} \setminus \{0, 1\} \cup \{x\}$;
3. if $i \neq j$ and $u_i, u_j \in \mathbb{N}$, then $f_i \neq f_j$;
4. if $u_i \in \mathbb{N}$, then $f_i \neq 1$;
5. $u_k^{f_k} \preccurlyeq \dots \preccurlyeq u_2^{f_2} \preccurlyeq u_1^{f_1}$.

Comparing two additive primes in $\mathfrak{L}$ with respect to $\preccurlyeq$

In order to compare two additive primes $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$ and $g = v_1^{g_1} v_2^{g_2} \dots v_\ell^{g_\ell}$ in multiplicative normal form:

Comparing two additive primes in $\mathfrak{L}$ with respect to $\preccurlyeq$

In order to compare two additive primes $f = u_1^{f_1} u_2^{f_2} \ldots u_k^{f_k}$ and $g = v_1^{g_1} v_2^{g_2} \ldots v_\ell^{g_\ell}$ in multiplicative normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;

Comparing two additive primes in $\mathfrak{L}$ with respect to $\preccurlyeq$

In order to compare two additive primes $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$ and $g = v_1^{g_1} v_2^{g_2} \dots v_\ell^{g_\ell}$ in multiplicative normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;

Comparing two additive primes in $\mathfrak{L}$ with respect to $\preccurlyeq$

In order to compare two additive primes $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$ and $g = v_1^{g_1} v_2^{g_2} \dots v_\ell^{g_\ell}$ in multiplicative normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $u_i^{f_i} \neq v_i^{g_i}$;

Comparing two additive primes in $\mathfrak{L}$ with respect to $\preccurlyeq$

In order to compare two additive primes $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$ and $g = v_1^{g_1} v_2^{g_2} \dots v_\ell^{g_\ell}$ in multiplicative normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $u_i^{f_i} \neq v_i^{g_i}$;
4. if $f_i \prec g_i$, then $f \prec g$;

Comparing two additive primes in $\mathfrak{L}$ with respect to $\preccurlyeq$

In order to compare two additive primes $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$ and $g = v_1^{g_1} v_2^{g_2} \dots v_\ell^{g_\ell}$ in multiplicative normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $u_i^{f_i} \neq v_i^{g_i}$;
4. if $f_i \prec g_i$, then $f \prec g$;
5. if $g_i \prec f_i$, then $g \prec f$;

Comparing two additive primes in $\mathfrak{L}$ with respect to $\preccurlyeq$

In order to compare two additive primes $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$ and $g = v_1^{g_1} v_2^{g_2} \dots v_\ell^{g_\ell}$ in multiplicative normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $u_i^{f_i} \neq v_i^{g_i}$;
4. if $f_i \prec g_i$, then $f \prec g$;
5. if $g_i \prec f_i$, then $g \prec f$;
6. if $f_i = g_i$ & $u_i \prec v_i$, then $f \prec g$;

Comparing two additive primes in $\mathfrak{L}$ with respect to $\preccurlyeq$

In order to compare two additive primes $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$ and $g = v_1^{g_1} v_2^{g_2} \dots v_\ell^{g_\ell}$ in multiplicative normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $u_i^{f_i} \neq v_i^{g_i}$;
4. if $f_i \prec g_i$, then $f \prec g$;
5. if $g_i \prec f_i$, then $g \prec f$;
6. if $f_i = g_i$ & $u_i \prec v_i$, then $f \prec g$;
7. if $f_i = g_i$ & $v_i \prec u_i$, then $g \prec f$.

Comparing two additive primes in $\mathfrak{L}$ with respect to $\preccurlyeq$

In order to compare two additive primes $f = u_1^{f_1} u_2^{f_2} \dots u_k^{f_k}$ and $g = v_1^{g_1} v_2^{g_2} \dots v_\ell^{g_\ell}$ in multiplicative normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $u_i^{f_i} \neq v_i^{g_i}$;
4. if $f_i \prec g_i$, then $f \prec g$;
5. if $g_i \prec f_i$, then $g \prec f$;
6. if $f_i = g_i$ & $u_i \prec v_i$, then $f \prec g$;
7. if $f_i = g_i$ & $v_i \prec u_i$, then $g \prec f$.

Additive normal forms in $\mathfrak{L}$

Every $f \neq 0$ has a unique *additive normal form* $f = p_1 + p_2 + \ldots + p_k$, where each p_i is additive prime and $p_k \preccurlyeq \ldots \preccurlyeq p_2 \preccurlyeq p_1$.

Additive normal forms in $\mathfrak{L}$

Every $f \neq 0$ has a unique *additive normal form* $f = p_1 + p_2 + \ldots + p_k$, where each p_i is additive prime and $p_k \preccurlyeq \ldots \preccurlyeq p_2 \preccurlyeq p_1$.

In order to compare $f = p_1 + p_2 + \ldots + p_k$ and $g = q_1 + q_2 + \ldots + q_\ell$ in additive normal form:

Additive normal forms in $\mathfrak{L}$

Every $f \neq 0$ has a unique *additive normal form* $f = p_1 + p_2 + \ldots + p_k$, where each p_i is additive prime and $p_k \preccurlyeq \ldots \preccurlyeq p_2 \preccurlyeq p_1$.

In order to compare $f = p_1 + p_2 + \ldots + p_k$ and $g = q_1 + q_2 + \ldots + q_\ell$ in additive normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;

Additive normal forms in $\mathfrak{L}$

Every $f \neq 0$ has a unique *additive normal form* $f = p_1 + p_2 + \ldots + p_k$, where each p_i is additive prime and $p_k \preccurlyeq \ldots \preccurlyeq p_2 \preccurlyeq p_1$.

In order to compare $f = p_1 + p_2 + \ldots + p_k$ and $g = q_1 + q_2 + \ldots + q_\ell$ in additive normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;

Additive normal forms in $\mathfrak{L}$

Every $f \neq 0$ has a unique *additive normal form* $f = p_1 + p_2 + \ldots + p_k$, where each p_i is additive prime and $p_k \preccurlyeq \ldots \preccurlyeq p_2 \preccurlyeq p_1$.

In order to compare $f = p_1 + p_2 + \ldots + p_k$ and $g = q_1 + q_2 + \ldots + q_\ell$ in additive normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $p_i \neq q_i$;

Additive normal forms in $\mathfrak{L}$

Every $f \neq 0$ has a unique *additive normal form* $f = p_1 + p_2 + \ldots + p_k$, where each p_i is additive prime and $p_k \preccurlyeq \ldots \preccurlyeq p_2 \preccurlyeq p_1$.

In order to compare $f = p_1 + p_2 + \ldots + p_k$ and $g = q_1 + q_2 + \ldots + q_\ell$ in additive normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $p_i \neq q_i$;
4. if $p_i \prec q_i$, then $f \prec g$;

Additive normal forms in $\mathfrak{L}$

Every $f \neq 0$ has a unique *additive normal form* $f = p_1 + p_2 + \ldots + p_k$, where each p_i is additive prime and $p_k \preccurlyeq \ldots \preccurlyeq p_2 \preccurlyeq p_1$.

In order to compare $f = p_1 + p_2 + \ldots + p_k$ and $g = q_1 + q_2 + \ldots + q_\ell$ in additive normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $p_i \neq q_i$;
4. if $p_i \prec q_i$, then $f \prec g$;
5. if $q_i \prec p_i$, then $g \prec f$.

Additive normal forms in $\mathfrak{L}$

Every $f \neq 0$ has a unique *additive normal form* $f = p_1 + p_2 + \ldots + p_k$, where each p_i is additive prime and $p_k \preccurlyeq \ldots \preccurlyeq p_2 \preccurlyeq p_1$.

In order to compare $f = p_1 + p_2 + \ldots + p_k$ and $g = q_1 + q_2 + \ldots + q_\ell$ in additive normal form:

1. if f is an initial segment of g, then $f \preccurlyeq g$;
2. if g is an initial segment of f, then $g \preccurlyeq f$;
3. otherwise, compute the least i, such that $p_i \neq q_i$;
4. if $p_i \prec q_i$, then $f \prec g$;
5. if $q_i \prec p_i$, then $g \prec f$.

The existence and uniqueness of normal forms implies that the order type of $\mathfrak{L}, \preccurlyeq$ is ϵ_0.

Our contribution

Let us code the terms in $\mathfrak{L}$ in a primitive recursive way.

Our contribution

Let us code the terms in $\mathfrak{L}$ in a primitive recursive way.
There exists a primitive recursive algorithm, such that:

Our contribution

Let us code the terms in $\mathfrak{L}$ in a primitive recursive way.
There exists a primitive recursive algorithm, such that:

1. for an additive prime p, it produces its multiplicative normal form;

Our contribution

Let us code the terms in $\mathfrak{L}$ in a primitive recursive way.
There exists a primitive recursive algorithm, such that:

1. for an additive prime p, it produces its multiplicative normal form;
2. given two additive primes p, q, it resolves $p \prec q, p = q, q \prec p$;

Our contribution

Let us code the terms in $\mathfrak{L}$ in a primitive recursive way.
There exists a primitive recursive algorithm, such that:

1. for an additive prime p, it produces its multiplicative normal form;
2. given two additive primes p, q, it resolves $p \prec q, p = q, q \prec p$;
3. for any $f \in \mathfrak{L}$, it produces its additive normal form;

Our contribution

Let us code the terms in $\mathfrak{L}$ in a primitive recursive way.
There exists a primitive recursive algorithm, such that:

1. for an additive prime p, it produces its multiplicative normal form;
2. given two additive primes p, q, it resolves $p \prec q, p = q, q \prec p$;
3. for any $f \in \mathfrak{L}$, it produces its additive normal form;
4. given two $f, g \in \mathfrak{L}$, it resolves $f \prec g, f = g, g \prec f$;

Our contribution

Let us code the terms in $\mathfrak{L}$ in a primitive recursive way.
There exists a primitive recursive algorithm, such that:

1. for an additive prime p, it produces its multiplicative normal form;
2. given two additive primes p, q, it resolves $p \prec q, p = q, q \prec p$;
3. for any $f \in \mathfrak{L}$, it produces its additive normal form;
4. given two $f, g \in \mathfrak{L}$, it resolves $f \prec g, f = g, g \prec f$;
5. for additive primes p, q with $p \prec q$, it produces a strict (p, q)-domination witness;

Our contribution

Let us code the terms in $\mathfrak{L}$ in a primitive recursive way.
There exists a primitive recursive algorithm, such that:

1. for an additive prime p, it produces its multiplicative normal form;
2. given two additive primes p, q, it resolves $p \prec q, p = q, q \prec p$;
3. for any $f \in \mathfrak{L}$, it produces its additive normal form;
4. given two $f, g \in \mathfrak{L}$, it resolves $f \prec g, f = g, g \prec f$;
5. for additive primes p, q with $p \prec q$, it produces a strict (p, q)-domination witness;
6. given two $f, g \in \mathfrak{L}$ with $f \prec g$, it produces a strict (f, g)-domination witness.

Punctual copies of $(\mathbb{N}, S)$

We consider structures $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ of the following kind:

Punctual copies of $(\mathbb{N}, S)$

We consider structures $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ of the following kind:

- their domain is the set $\mathbb{N}$ of all natural numbers

Punctual copies of $(\mathbb{N}, S)$

We consider structures $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ of the following kind:

- their domain is the set $\mathbb{N}$ of all natural numbers
- they are computably isomorphic with $(\mathbb{N}, S)$

Punctual copies of $(\mathbb{N}, S)$

We consider structures $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ of the following kind:

- their domain is the set $\mathbb{N}$ of all natural numbers
- they are computably isomorphic with $(\mathbb{N}, S)$
- they are *punctual* ($S^{\mathcal{A}}$ is primitive recursive)

Punctual copies of $(\mathbb{N}, S)$

We consider structures $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ of the following kind:

- their domain is the set $\mathbb{N}$ of all natural numbers
- they are computably isomorphic with $(\mathbb{N}, S)$
- they are *punctual* ($S^{\mathcal{A}}$ is primitive recursive)

We call them punctual copies of $(\mathbb{N}, S)$.

Punctual copies of $(\mathbb{N}, S)$

We consider structures $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ of the following kind:

- their domain is the set $\mathbb{N}$ of all natural numbers
- they are computably isomorphic with $(\mathbb{N}, S)$
- they are *punctual* ($S^{\mathcal{A}}$ is primitive recursive)

We call them punctual copies of $(\mathbb{N}, S)$.

$0^{\mathcal{A}}$

Punctual copies of $(\mathbb{N}, S)$

We consider structures $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ of the following kind:

- their domain is the set $\mathbb{N}$ of all natural numbers
- they are computably isomorphic with $(\mathbb{N}, S)$
- they are *punctual* ($S^{\mathcal{A}}$ is primitive recursive)

We call them punctual copies of $(\mathbb{N}, S)$.

$$0^{\mathcal{A}} \longrightarrow S^{\mathcal{A}}(0^{\mathcal{A}})$$

Punctual copies of $(\mathbb{N}, S)$

We consider structures $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ of the following kind:

- their domain is the set $\mathbb{N}$ of all natural numbers
- they are computably isomorphic with $(\mathbb{N}, S)$
- they are *punctual* ($S^{\mathcal{A}}$ is primitive recursive)

We call them punctual copies of $(\mathbb{N}, S)$.

$$0^{\mathcal{A}} \longrightarrow S^{\mathcal{A}}(0^{\mathcal{A}}) \longrightarrow S^{\mathcal{A}}(S^{\mathcal{A}}(0^{\mathcal{A}})) \longrightarrow \dots$$

Image of a function in $\mathcal{A}$

Let $\mathcal{A}$ be a punctual copy of $(\mathbb{N}, S)$.

Image of a function in $\mathcal{A}$

Let $\mathcal{A}$ be a punctual copy of $(\mathbb{N}, S)$.
We denote by $c_{\mathcal{A}}$ the unique isomorphism from $(\mathbb{N}, S)$ to $\mathcal{A}$, which acts as $c_{\mathcal{A}}(n) = (S^{\mathcal{A}})^n(0^{\mathcal{A}})$.

Image of a function in $\mathcal{A}$

Let $\mathcal{A}$ be a punctual copy of $(\mathbb{N}, S)$.
We denote by $c_{\mathcal{A}}$ the unique isomorphism from $(\mathbb{N}, S)$ to $\mathcal{A}$, which acts as $c_{\mathcal{A}}(n) = (S^{\mathcal{A}})^n(0^{\mathcal{A}})$.
Clearly, $c_{\mathcal{A}}$ is primitive recursive and $c_{\mathcal{A}}^{-1}$ is computable, but in general $c_{\mathcal{A}}^{-1}$ is not primitive recursive.

Image of a function in $\mathcal{A}$

Let $\mathcal{A}$ be a punctual copy of $(\mathbb{N}, S)$.
We denote by $c_{\mathcal{A}}$ the unique isomorphism from $(\mathbb{N}, S)$ to $\mathcal{A}$, which acts as $c_{\mathcal{A}}(n) = (S^{\mathcal{A}})^n(0^{\mathcal{A}})$.
Clearly, $c_{\mathcal{A}}$ is primitive recursive and $c_{\mathcal{A}}^{-1}$ is computable, but in general $c_{\mathcal{A}}^{-1}$ is not primitive recursive.

For a function $f : \mathbb{N} \to \mathbb{N}$ we will use the standard denotation $f^{\mathcal{A}}$ for *the image of f* in the model $\mathcal{A}$, more precisely
$f^{\mathcal{A}} = c_{\mathcal{A}} \circ f \circ c_{\mathcal{A}}^{-1}$.

Image of a function in $\mathcal{A}$

Let $\mathcal{A}$ be a punctual copy of $(\mathbb{N}, S)$.
We denote by $c_{\mathcal{A}}$ the unique isomorphism from $(\mathbb{N}, S)$ to $\mathcal{A}$, which acts as $c_{\mathcal{A}}(n) = (S^{\mathcal{A}})^n(0^{\mathcal{A}})$.
Clearly, $c_{\mathcal{A}}$ is primitive recursive and $c_{\mathcal{A}}^{-1}$ is computable, but in general $c_{\mathcal{A}}^{-1}$ is not primitive recursive.

For a function $f : \mathbb{N} \to \mathbb{N}$ we will use the standard denotation $f^{\mathcal{A}}$ for *the image of* f in the model $\mathcal{A}$, more precisely $f^{\mathcal{A}} = c_{\mathcal{A}} \circ f \circ c_{\mathcal{A}}^{-1}$.
It is clear that

$$f \text{ is computable if and only if } f^{\mathcal{A}} \text{ is computable.}$$

Image of a function in $\mathcal{A}$

Let $\mathcal{A}$ be a punctual copy of $(\mathbb{N}, S)$.
We denote by $c_{\mathcal{A}}$ the unique isomorphism from $(\mathbb{N}, S)$ to $\mathcal{A}$, which acts as $c_{\mathcal{A}}(n) = (S^{\mathcal{A}})^n(0^{\mathcal{A}})$.
Clearly, $c_{\mathcal{A}}$ is primitive recursive and $c_{\mathcal{A}}^{-1}$ is computable, but in general $c_{\mathcal{A}}^{-1}$ is not primitive recursive.

For a function $f : \mathbb{N} \to \mathbb{N}$ we will use the standard denotation $f^{\mathcal{A}}$ for *the image of f* in the model $\mathcal{A}$, more precisely
$f^{\mathcal{A}} = c_{\mathcal{A}} \circ f \circ c_{\mathcal{A}}^{-1}$.
It is clear that

$$f \text{ is computable if and only if } f^{\mathcal{A}} \text{ is computable.}$$

But the complexity of f and $f^{\mathcal{A}}$ can be very different.

Main question

For a punctual copy $\mathcal{A}$ we are interested in the class of primitive recursive functions, relative to $\mathcal{A}$:

$$Pr(\mathcal{A}) \; = \; \{f^{\mathcal{A}} \mid f \in Pr\},$$

where $Pr = Pr((\mathbb{N}, S))$ is the class of standard primitive recursive functions.

Main question

For a punctual copy $\mathcal{A}$ we are interested in the class of primitive recursive functions, relative to $\mathcal{A}$:

$$Pr(\mathcal{A}) \ = \ \{f^{\mathcal{A}} \mid f \in Pr\},$$

where $Pr = Pr((\mathbb{N}, S))$ is the class of standard primitive recursive functions.

In general, we would like to explore how $Pr(\mathcal{A})$ and Pr are related.

Main question

For a punctual copy $\mathcal{A}$ we are interested in the class of primitive recursive functions, relative to $\mathcal{A}$:

$$Pr(\mathcal{A}) \; = \; \{f^{\mathcal{A}} \mid f \in Pr\},$$

where $Pr = Pr((\mathbb{N}, S))$ is the class of standard primitive recursive functions.

In general, we would like to explore how $Pr(\mathcal{A})$ and Pr are related.

For example, we know that $Pr(\mathcal{A})$ and Pr are incomparable with respect to inclusion, unless $c_{\mathcal{A}}^{-1}$ is primitive recursive (so that $\mathcal{A}$ is primitive recursively isomorphic to $(\mathbb{N}, S)$).

Main question

For a punctual copy $\mathcal{A}$ we are interested in the class of primitive recursive functions, relative to $\mathcal{A}$:

$$Pr(\mathcal{A}) = \{f^{\mathcal{A}} \mid f \in Pr\},$$

where $Pr = Pr((\mathbb{N}, S))$ is the class of standard primitive recursive functions.

In general, we would like to explore how $Pr(\mathcal{A})$ and Pr are related.

For example, we know that $Pr(\mathcal{A})$ and Pr are incomparable with respect to inclusion, unless $c_{\mathcal{A}}^{-1}$ is primitive recursive (so that $\mathcal{A}$ is primitive recursively isomorphic to $(\mathbb{N}, S)$).

Our first line of work was to build specific $\mathcal{A}$, in which some concrete functions in $Pr(\mathcal{A})$ are not primitive recursive.

First Model

First goal: build a punctual copy $\mathcal{A}$, in which *pred*$^{\mathcal{A}}$ is not primitive recursive.

First Model

First goal: build a punctual copy $\mathcal{A}$, in which $pred^{\mathcal{A}}$ is not primitive recursive.

Let $a : \mathbb{N} \to \mathbb{N}$ be a computable function with primitive recursive graph, which grows faster than any primitive recursive function.

First Model

First goal: build a punctual copy $\mathcal{A}$, in which $pred^{\mathcal{A}}$ is not primitive recursive.

Let $a : \mathbb{N} \to \mathbb{N}$ be a computable function with primitive recursive graph, which grows faster than any primitive recursive function. We would like to include arrows of the form $a_n \longrightarrow f(n)$ in our model, where f is primitive recursive.

First Model

First goal: build a punctual copy $\mathcal{A}$, in which $pred^{\mathcal{A}}$ is not primitive recursive.

Let $a : \mathbb{N} \to \mathbb{N}$ be a computable function with primitive recursive graph, which grows faster than any primitive recursive function. We would like to include arrows of the form $a_n \longrightarrow f(n)$ in our model, where f is primitive recursive. Let h be the primitive recursive increasing enumeration of $\mathbb{N} \setminus Ran(a)$.

First Model

First goal: build a punctual copy $\mathcal{A}$, in which $pred^{\mathcal{A}}$ is not primitive recursive.

Let $a : \mathbb{N} \to \mathbb{N}$ be a computable function with primitive recursive graph, which grows faster than any primitive recursive function. We would like to include arrows of the form $a_n \longrightarrow f(n)$ in our model, where f is primitive recursive.
Let h be the primitive recursive increasing enumeration of $\mathbb{N} \setminus Ran(a)$.
Our model $\mathcal{A}$ will be constructed using the following chains:

$$a_n \longrightarrow h(\langle n, 0\rangle) \longrightarrow h(\langle n, 1\rangle) \ldots \longrightarrow h(\langle n, a_{n+1}\rangle) \longrightarrow a_{n+1}$$

First Model

First goal: build a punctual copy $\mathcal{A}$, in which $pred^{\mathcal{A}}$ is not primitive recursive.

Let $a : \mathbb{N} \to \mathbb{N}$ be a computable function with primitive recursive graph, which grows faster than any primitive recursive function. We would like to include arrows of the form $a_n \longrightarrow f(n)$ in our model, where f is primitive recursive.
Let h be the primitive recursive increasing enumeration of $\mathbb{N} \setminus Ran(a)$.
Our model $\mathcal{A}$ will be constructed using the following chains:

$$a_n \longrightarrow h(\langle n, 0\rangle) \longrightarrow h(\langle n, 1\rangle) \ldots \longrightarrow h(\langle n, a_{n+1}\rangle) \longrightarrow a_{n+1}$$

The remaining elements are in the set $\{h(\langle n, i\rangle) \mid i > a_{n+1}\}$ whose strictly increasing enumeration *free* is primitive recursive.

First Model

First goal: build a punctual copy $\mathcal{A}$, in which $pred^{\mathcal{A}}$ is not primitive recursive.

Let $a : \mathbb{N} \to \mathbb{N}$ be a computable function with primitive recursive graph, which grows faster than any primitive recursive function.
We would like to include arrows of the form $a_n \longrightarrow f(n)$ in our model, where f is primitive recursive.
Let h be the primitive recursive increasing enumeration of $\mathbb{N} \setminus Ran(a)$.
Our model $\mathcal{A}$ will be constructed using the following chains:

$$a_n \longrightarrow h(\langle n, 0\rangle) \longrightarrow h(\langle n, 1\rangle) \ldots \longrightarrow h(\langle n, a_{n+1}\rangle) \longrightarrow a_{n+1}$$

The remaining elements are in the set $\{h(\langle n, i\rangle) \mid i > a_{n+1}\}$ whose strictly increasing enumeration *free* is primitive recursive.
We complete the model $\mathcal{A}$ in the following way:

$$a_n \longrightarrow free(n) \longrightarrow h(\langle n, 0\rangle)$$

Images of functions in the model $\mathcal{A}$

Proposition

Let $f : \mathbb{N} \to \mathbb{N}$ *be primitive recursive and* $f(x) \geq 2x$. *Then* $f^{\mathcal{A}}$ *is not primitive recursive.*

Images of functions in the model $\mathcal{A}$

Proposition

Let $f : \mathbb{N} \to \mathbb{N}$ be primitive recursive and $f(x) \geq 2x$. Then $f^{\mathcal{A}}$ is not primitive recursive.

$$\begin{tikzcd}
a_n \arrow[r] & free(n) \arrow[r] \arrow[rrr, bend right, "f^{\mathcal{A}}"'] & h(\langle n, 0\rangle) \arrow[r] & h(\langle n, 1\rangle) & \ldots \arrow[r] & h(\langle n, a_{n+1}\rangle) \arrow[r] & a_{n+1}
\end{tikzcd}$$

Images of functions in the model $\mathcal{A}$

Proposition

Let $f : \mathbb{N} \to \mathbb{N}$ be primitive recursive and $f(x) \geq 2x$. Then $f^{\mathcal{A}}$ is not primitive recursive.

$$\begin{tikzcd}
a_n \arrow[r] & \mathit{free}(n) \arrow[r] \arrow[rrr, bend right, "f^{\mathcal{A}}"'] & h(\langle n, 0 \rangle) \arrow[r] & h(\langle n, 1 \rangle) & \dots \arrow[r] & h(\langle n, a_{n+1} \rangle) \arrow[r] & a_{n+1}
\end{tikzcd}$$

Indeed, $f^{\mathcal{A}}(\mathit{free}(n)) = h(\langle n, i \rangle)$ and i is greater than the position of $\mathit{free}(n)$.

Enforcing primitive recursive sum

Let $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ be a punctual copy.

Enforcing primitive recursive sum

Let $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ be a punctual copy.
Let c be the unique isomorphism from $(\mathbb{N}, S)$ to $\mathcal{A}$.

Enforcing primitive recursive sum

Let $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ be a punctual copy.
Let c be the unique isomorphism from $(\mathbb{N}, S)$ to $\mathcal{A}$.

We construct a new punctual copy $\mathcal{B}$ in the following way:

$$\widetilde{c}(0) = 0, \quad \widetilde{c}(2^{i_k} + 2^{i_{k-1}} + \ldots + 2^{i_0}) = 2^{c(i_k)} + 2^{c(i_{k-1})} + \ldots + 2^{c(i_0)},$$

so that the individuals of $\mathcal{B}$ have the form $\widetilde{c}(n)$ and
$S^{\mathcal{B}}(\widetilde{c}(n)) = \widetilde{c}(n+1)$.

Enforcing primitive recursive sum

Let $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ be a punctual copy.
Let c be the unique isomorphism from $(\mathbb{N}, S)$ to $\mathcal{A}$.

We construct a new punctual copy $\mathcal{B}$ in the following way:

$$\widetilde{c}(0) = 0, \quad \widetilde{c}(2^{i_k} + 2^{i_{k-1}} + \ldots + 2^{i_0}) = 2^{c(i_k)} + 2^{c(i_{k-1})} + \ldots + 2^{c(i_0)},$$

so that the individuals of $\mathcal{B}$ have the form $\widetilde{c}(n)$ and
$S^{\mathcal{B}}(\widetilde{c}(n)) = \widetilde{c}(n+1)$.

Proposition

The functions $S^{\mathcal{B}}$ and $+^{\mathcal{B}}$ are primitive recursive.

Enforcing primitive recursive sum

Let $\mathcal{A} = (\mathbb{N}, S^{\mathcal{A}})$ be a punctual copy.
Let c be the unique isomorphism from $(\mathbb{N}, S)$ to $\mathcal{A}$.

We construct a new punctual copy $\mathcal{B}$ in the following way:

$$\widetilde{c}(0) = 0, \quad \widetilde{c}(2^{i_k} + 2^{i_{k-1}} + \ldots + 2^{i_0}) = 2^{c(i_k)} + 2^{c(i_{k-1})} + \ldots + 2^{c(i_0)},$$

so that the individuals of $\mathcal{B}$ have the form $\widetilde{c}(n)$ and
$S^{\mathcal{B}}(\widetilde{c}(n)) = \widetilde{c}(n+1)$.

Proposition

The functions $S^{\mathcal{B}}$ and $+^{\mathcal{B}}$ are primitive recursive.

Idea: we can simulate binary addition by looking ahead with the successor $S^{\mathcal{A}}$ in the original model $\mathcal{A}$.

Enforcing primitive recursive product

Let again $\mathcal{A}$ be a punctual copy and let us build $\mathcal{B}$ in the same way.

Enforcing primitive recursive product

Let again $\mathcal{A}$ be a punctual copy and let us build $\mathcal{B}$ in the same way.

Proposition

If the function $+^{\mathcal{A}}$ *is primitive recursive, then the function* $.^{\mathcal{B}}$ *is primitive recursive.*

Enforcing primitive recursive product

Let again $\mathcal{A}$ be a punctual copy and let us build $\mathcal{B}$ in the same way.

Proposition

If the function $+^{\mathcal{A}}$ is primitive recursive, then the function $.^{\mathcal{B}}$ is primitive recursive.

Idea: binary multiplication is reduced to sum in the exponents.

Enforcing primitive recursive product

Let again $\mathcal{A}$ be a punctual copy and let us build $\mathcal{B}$ in the same way.

Proposition

If the function $+^{\mathcal{A}}$ is primitive recursive, then the function $.^{\mathcal{B}}$ is primitive recursive.

Idea: binary multiplication is reduced to sum in the exponents.

We apply this construction twice to the first constructed model $\mathcal{A}$.

Enforcing primitive recursive product

Let again $\mathcal{A}$ be a punctual copy and let us build $\mathcal{B}$ in the same way.

Proposition

If the function $+^{\mathcal{A}}$ *is primitive recursive, then the function* $.^{\mathcal{B}}$ *is primitive recursive.*

Idea: binary multiplication is reduced to sum in the exponents.

We apply this construction twice to the first constructed model $\mathcal{A}$. We obtain a model $\mathcal{C}$, such that $+^{\mathcal{C}}$ and $.^{\mathcal{C}}$ are primitive recursive,

Enforcing primitive recursive product

Let again $\mathcal{A}$ be a punctual copy and let us build $\mathcal{B}$ in the same way.

Proposition

If the function $+^{\mathcal{A}}$ is primitive recursive, then the function $.^{\mathcal{B}}$ is primitive recursive.

Idea: binary multiplication is reduced to sum in the exponents.

We apply this construction twice to the first constructed model $\mathcal{A}$. We obtain a model $\mathcal{C}$, such that $+^{\mathcal{C}}$ and $.^{\mathcal{C}}$ are primitive recursive, but for any primitive recursive f with $f(x) \geq x^{\log_2 x}$,
$f^{\mathcal{C}}$ is not primitive recursive.

Triple iteration

Unfortunately this idea cannot be lifted to preserve the exponential function.

Triple iteration

Unfortunately this idea cannot be lifted to preserve the exponential function.
Let again $\mathcal{B}$ be the copy produced from $\mathcal{A}$ with the powers of 2 construction.

Triple iteration

Unfortunately this idea cannot be lifted to preserve the exponential function.
Let again $\mathcal{B}$ be the copy produced from $\mathcal{A}$ with the powers of 2 construction.

Proposition

Let $+^{\mathcal{A}}$ *be primitive recursive in the model* $\mathcal{A}$*. Let* $p(x) = 2^x$*. Then* $p^{\mathcal{B}}$ *is primitive recursive if and only if* $p^{\mathcal{A}}$ *is primitive recursive.*

Method of islands and archipelago

Our goal again is to build a punctual copy $\mathcal{A}$, such that $\mathit{pred}^{\mathcal{A}}$ is not primitive recursive.

Method of islands and archipelago

Our goal again is to build a punctual copy $\mathcal{A}$, such that $\mathit{pred}^{\mathcal{A}}$ is not primitive recursive.

Let $p_0, p_1, \ldots, p_e, \ldots$ be an effective enumeration of the unary primitive recursive functions.

Method of islands and archipelago

Our goal again is to build a punctual copy $\mathcal{A}$, such that $pred^{\mathcal{A}}$ is not primitive recursive.

Let $p_0, p_1, \ldots, p_e, \ldots$ be an effective enumeration of the unary primitive recursive functions.
At stage e we have the following picture:

mainland $\quad 0 \rightarrow a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \ldots \rightarrow a_k$

and

archipelago $\quad w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots$

Method of islands and archipelago

Our goal again is to build a punctual copy $\mathcal{A}$, such that $pred^{\mathcal{A}}$ is not primitive recursive.

Let $p_0, p_1, \ldots, p_e, \ldots$ be an effective enumeration of the unary primitive recursive functions.

At stage e we have the following picture:

mainland $\quad 0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k$

and

archipelago $\quad w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots$

Each b_i is associated with a term $\tau_i(x)$ over a signature $\mathcal{F}$ of functions that we want to make primitive recursive in $\mathcal{A}$.

Method of islands and archipelago

Our goal again is to build a punctual copy $\mathcal{A}$, such that $pred^{\mathcal{A}}$ is not primitive recursive.

Let $p_0, p_1, \ldots, p_e, \ldots$ be an effective enumeration of the unary primitive recursive functions.
At stage e we have the following picture:

mainland $\quad 0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k$

and

archipelago $\quad w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots$

Each b_i is associated with a term $\tau_i(x)$ over a signature $\mathcal{F}$ of functions that we want to make primitive recursive in $\mathcal{A}$.
The element w is associated with x.

Method of islands and archipelago

Our goal again is to build a punctual copy $\mathcal{A}$, such that $pred^{\mathcal{A}}$ is not primitive recursive.

Let $p_0, p_1, \ldots, p_e, \ldots$ be an effective enumeration of the unary primitive recursive functions.
At stage e we have the following picture:

mainland $\quad 0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k$

and

archipelago $\quad w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots$

Each b_i is associated with a term $\tau_i(x)$ over a signature $\mathcal{F}$ of functions that we want to make primitive recursive in $\mathcal{A}$.
The element w is associated with x.
In this stage we wait for $p_e(w)$ to give value v for s steps.

Method of islands and archipelago (2)

While waiting we must keep extending the mainland with new elements and also the archipelago with new islands.

mainland $0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k$

archipelago $w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots$

Method of islands and archipelago (2)

While waiting we must keep extending the mainland with new elements and also the archipelago with new islands.

mainland $0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k \to a_{k+1}$

archipelago $w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots$

Method of islands and archipelago (2)

While waiting we must keep extending the mainland with new elements and also the archipelago with new islands.

mainland $0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k \to a_{k+1}$

archipelago $w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots \quad b_m^{f(\tau_1)}$

Method of islands and archipelago (2)

While waiting we must keep extending the mainland with new elements and also the archipelago with new islands.

mainland $0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k \to a_{k+1} \to a_{k+2}$

archipelago $w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots \quad b_m^{f(\tau_1)}$

Method of islands and archipelago (2)

While waiting we must keep extending the mainland with new elements and also the archipelago with new islands.

mainland	$0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k \to a_{k+1} \to a_{k+2}$
archipelago	$w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots \quad b_m^{f(\tau_1)} \quad b_{m+1}^{g(\tau_2)}$

Method of islands and archipelago (2)

While waiting we must keep extending the mainland with new elements and also the archipelago with new islands.

mainland $0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k \to a_{k+1} \to a_{k+2}$

archipelago $w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots \quad b_m^{f(\tau_1)} \quad b_{m+1}^{g(\tau_2)} \quad b_{m+2}^{f(\tau_2)}$

Method of islands and archipelago (2)

While waiting we must keep extending the mainland with new elements and also the archipelago with new islands.

mainland $0 \to a_1 \to a_2 \to a_3 \to \ldots \to a_k \to a_{k+1} \to a_{k+2}$

archipelago $w^x \quad b_1^{\tau_1} \quad b_2^{\tau_2} \quad \ldots \quad b_m^{f(\tau_1)} \quad b_{m+1}^{g(\tau_2)} \quad b_{m+2}^{f(\tau_2)}$

When we obtain the result $p_e(w) = v$ we must connect the mainland with the archipelago.

How to connect?

We must choose carefully a position q for w, because this choice will fix the positions of all islands.

How to connect?

We must choose carefully a position q for w, because this choice will fix the positions of all islands.
The island b_i associated with $\tau_i(x)$ must obtain position, which coincides with the value of τ_i at q.

How to connect?

We must choose carefully a position q for w, because this choice will fix the positions of all islands.
The island b_i associated with $\tau_i(x)$ must obtain position, which coincides with the value of τ_i at q.
This guarantees that for any $f \in \mathcal{F}$, we have $f^{\mathcal{A}}(b_i) = b_k$, where the label of b_i if τ and the label of b_k is $f(\tau)$.

How to connect?

We must choose carefully a position q for w, because this choice will fix the positions of all islands.
The island b_i associated with $\tau_i(x)$ must obtain position, which coincides with the value of τ_i at q.
This guarantees that for any $f \in \mathcal{F}$, we have $f^{\mathcal{A}}(b_i) = b_k$, where the label of b_i if τ and the label of b_k is $f(\tau)$.

A suitable choice for q must give different positions for all islands (otherwise we lose injectivity).

How to connect?

We must choose carefully a position q for w, because this choice will fix the positions of all islands.
The island b_i associated with $\tau_i(x)$ must obtain position, which coincides with the value of τ_i at q.
This guarantees that for any $f \in \mathcal{F}$, we have $f^{\mathcal{A}}(b_i) = b_k$, where the label of b_i if τ and the label of b_k is $f(\tau)$.

A suitable choice for q must give different positions for all islands (otherwise we lose injectivity).
Therefore, given the terms $\tau_1, \ldots, \tau_m$ of all islands we want q to be a strict domination witness for all pairs $(\tau_i(q), \tau_j(q))$.

How to connect?

We must choose carefully a position q for w, because this choice will fix the positions of all islands.
The island b_i associated with $\tau_i(x)$ must obtain position, which coincides with the value of τ_i at q.
This guarantees that for any $f \in \mathcal{F}$, we have $f^{\mathcal{A}}(b_i) = b_k$, where the label of b_i if τ and the label of b_k is $f(\tau)$.

A suitable choice for q must give different positions for all islands (otherwise we lose injectivity).
Therefore, given the terms $\tau_1, \ldots, \tau_m$ of all islands we want q to be a strict domination witness for all pairs $(\tau_i(q), \tau_j(q))$.

We know how to compute these for terms in the Levitz class.

How to connect?

We must choose carefully a position q for w, because this choice will fix the positions of all islands.
The island b_i associated with $\tau_i(x)$ must obtain position, which coincides with the value of τ_i at q.
This guarantees that for any $f \in \mathcal{F}$, we have $f^{\mathcal{A}}(b_i) = b_k$, where the label of b_i if τ and the label of b_k is $f(\tau)$.

A suitable choice for q must give different positions for all islands (otherwise we lose injectivity).
Therefore, given the terms $\tau_1, \ldots, \tau_m$ of all islands we want q to be a strict domination witness for all pairs $(\tau_i(q), \tau_j(q))$.

We know how to compute these for terms in the Levitz class.
But since the terms over $\mathcal{F}$ should be closed under substitution, we omit the base-x case from the definition.

How to connect? (2)

After choosing q, every island b_i obtains the corresponding position, which is its label τ_i evaluated at q.

How to connect? (2)

After choosing q, every island b_i obtains the corresponding position, which is its label τ_i evaluated at q.
We also add other auxiliary elements, so that the mainland and the islands become one successor chain:

$$0 \to a_1 \to \ldots \to a_s \to \ldots \to w \to \ldots \to b_1 \ldots \to \ldots \to b_t$$

How to connect? (2)

After choosing q, every island b_i obtains the corresponding position, which is its label τ_i evaluated at q.
We also add other auxiliary elements, so that the mainland and the islands become one successor chain:

$$0 \to a_1 \to \ldots \to a_s \to \ldots \to w \to \ldots \to b_1 \ldots \to \ldots \to b_t$$

And finally, if it happens that $S^{\mathcal{A}}(v) = w$ we must insert a new element a between them, because we want to ensure that $pred^{\mathcal{A}}(w) \neq v$ (so that $p_e \neq pred^{\mathcal{A}}$).

How to connect? (2)

After choosing q, every island b_i obtains the corresponding position, which is its label τ_i evaluated at q.
We also add other auxiliary elements, so that the mainland and the islands become one successor chain:

$$0 \to a_1 \to \ldots \to a_s \to \ldots \to w \to \ldots \to b_1 \ldots \to \ldots \to b_t$$

And finally, if it happens that $S^{\mathcal{A}}(v) = w$ we must insert a new element a between them, because we want to ensure that $pred^{\mathcal{A}}(w) \neq v$ (so that $p_e \neq pred^{\mathcal{A}}$).

We pick a new element w', which is the start of a new archipelago and we proceed to stage $e + 1$.

Bibliography

- Kalimullin, I., Melnikov, A., Ng, K.M.: Algebraic structures computable without delay. Theor. Comput. Sci. **674**, 73–98 (2017)
- Bazhenov, N., Downey, R., Kalimullin, I., Melnikov, A.: Foundations of online structure theory. Bull. Symb. Log. **25**(2), 141–181 (2019)
- Kalimullin, I., Melnikov, A., Montalban, A.: Punctual definability on structures. Ann. Pure Appl. Logic **172**, 102987 (2021)

Thanks for your attention!

HAPPY 70TH ANNIVERSARY,

PROFESSOR TINCHEV!