Polynomial Time Computability for Set Functions

Sam Buss
Univ. of California, San Diego

Steklov Institute
July 22, 2015

Joint work with A. Beckmann, S.D. Friedman, M. Müller, N. Thapen

Goal: Give definitions of feasible complexity classes that are

- Analogous to complexity classes on bit strings,
- Natural and intrinsic to sets
- Reduce to standard complexity classes on hereditarily finite sets with suitable encodings

Two approaches:

- Safe/normal recursion:
  [Beckmann, B., Sy Friedman' ??]; [Arai'15]
- Cobham recursion:
  [Beckmann, B., Sy Friedman, Müller, Thapen], this talk.

Other approaches:

- Bounded set theory [Sazonov, '97]
- Infinite time Turing machines
  [Hamkins-Lewis'00; Friedman-Welch'07]

Cobham-style definition of polynomial time (P): [1965]

Inputs and outputs are binary strings in $\{0,1\}^*$

Initial functions including

the empty string ϵ;

the two successor functions $s \mapsto si$ (for $i = 0, 1$);

$\mathrm{cond}(a, c, d) = c$ if $a = \epsilon$ and d otherwise;

$a\#b := a^{|b|}$. ("Smash", concatenate $|b|$ many copies of a)

Closed under:

Composition, and

Limited Iteration on Notation:

$$\begin{aligned} f(\epsilon, \vec{a}) &= g(\vec{a}) \\ f(s0, \vec{a}) &= h_0(s, \vec{a}, f(s, \vec{a})) \\ f(s1, \vec{a}) &= h_1(s, \vec{a}, f(s, \vec{a})). \end{aligned}$$

provided $|f(s, \vec{a})| \leq |t(s, \vec{a})|$ for some $\#$/successor term t.

or equivalently, $|f(s, \vec{a})| \leq p(|s|, |\vec{a}|)$ for a polynomial p.

In limited iteration on notation, the computation of $f(s,a)$ uses only $|s|$ rounds of iteration: this is needed to have polynomial time computability.

But in addition, the polynomial bound on $|f(s,\vec{a})|$ is important to maintain polynomial time computability.

For example:

$$\begin{aligned} f(\epsilon,a) &= a \\ f(si,\vec{a}) &= f(s,a)\#00 = f(s,a)f(s,a), \end{aligned}$$

gives $|f(s,a)| = 2^{|s|}|a|$. This is exponential growth rate, and hence is not polynomial time computable.

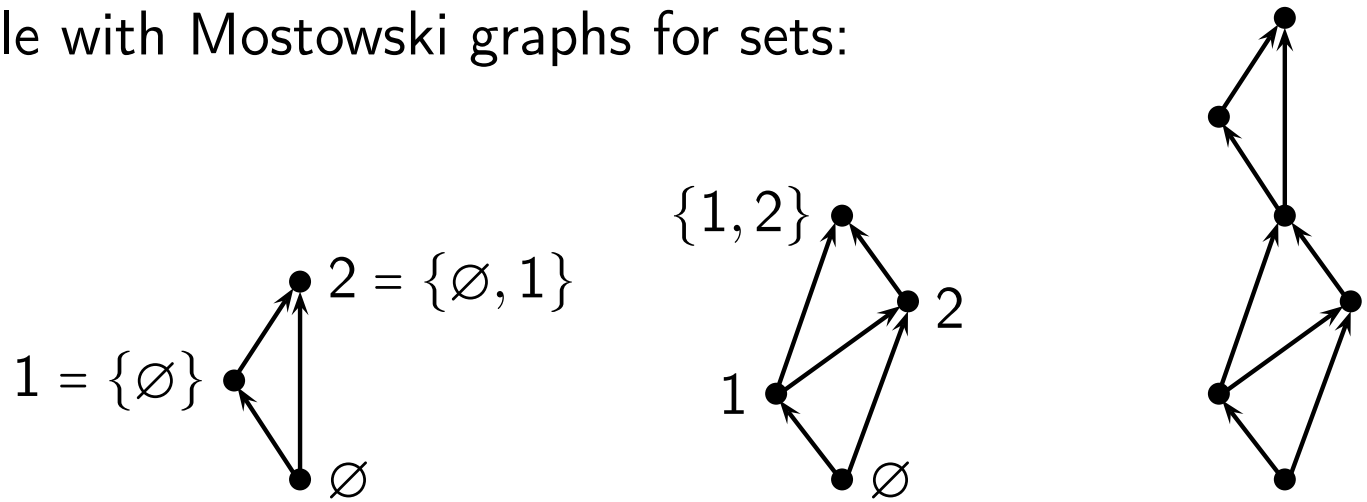
The natural set-theoretic replacement for Cobham's Limited Iteration on Notation is $\in$-recursion. However, first we need a new kind of smash function ($\#$) that operates on sets.

Definition (Set composition $\odot$)

The *set composition* function $a \odot b$ is defined by $\in$-recursion as

$$\begin{aligned} \varnothing \odot b &= b \\ a \odot b &= \{x \odot b : x \in a\}, \quad \text{for } a \neq \varnothing \end{aligned}$$

Example with Mostowski graphs for sets:



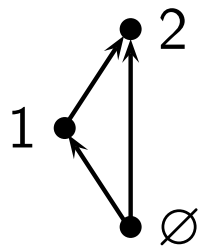
(a) $A = \{\varnothing, 1\}$ (b) $B = \{1, 2\}$ (c) $A \odot B$

Definition (Set smash #)

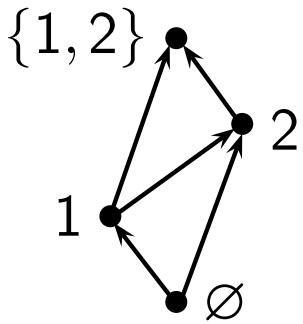
The *set smash* function is the function $a\#b$ defined by $\in$-recursion on a as

$$a\#b \;=\; b\odot\{x\#b : x \in a\}.$$

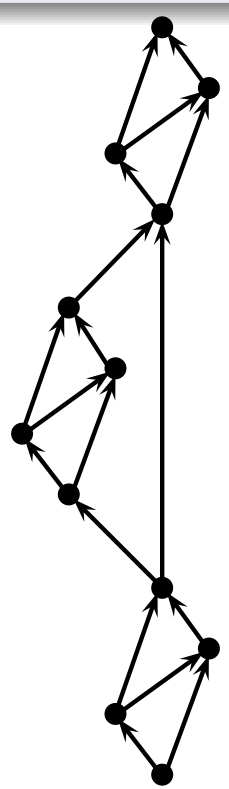
Mostowski graphs example:



(a) A



(b) B



(d) $A\#B$

Theorem

The set smash function # satisfies the following:

1. $\mathrm{rank}(a\#b) + 1 = (\mathrm{rank}(b) + 1)(\mathrm{rank}(a) + 1)$.
2. $|\mathrm{tc}(a\#b)| + 1 = (|\mathrm{tc}(a)| + 1)(|\mathrm{tc}(b)| + 1)$.
   Equivalently, $|\mathrm{tc}^+(a\#b)| = |\mathrm{tc}^+(a)| \cdot |\mathrm{tc}^+(b)|$,
   where $\mathrm{tc}^+(a) := \{a\} \cup \mathrm{tc}(a)$.

Thus, the # function gives polynomial growth rate both for rank, and for cardinality of the transitive closure (tc).

A first attempt at generalizing Cobham limited iteration to sets is:

- Use $\in$-recursion instead of recursion on notation.
- Requiring $f(a, \vec{c})$ to be a subset of an already constructed function $h(a, \vec{c})$:

Definition ((Cobham Recursion$_\subseteq$))

If g is an $(n+1)$-ary function and h is an n-ary function, then **(Cobham Recursion$_\subseteq$)** gives the n-ary function f:

$$f(a, \vec{c}) \;=\; g(\{f(b, \vec{c}) : b \in a\}, a, \vec{c}) \cap h(a, \vec{c}).$$

A more sophisticated size bound can be obtained by using embeddings ....

Definition ($\preccurlyeq$ embedding)

A set A is $\preccurlyeq$-*embedded* into a set B, denoted $A \preccurlyeq B$, provided either

(a) There is injective $\tau : \mathrm{tc}(A) \to \mathrm{tc}(B)$ s.t. for all $x \in y \in \mathrm{tc}(A)$, we have $\tau(x) \in \mathrm{tc}(\tau(y))$, or

(b) There is $\tau : \mathrm{tc}(A) \to \mathcal{P}(\mathrm{tc}(B))$, s.t.
(i) if $x \neq y$, then $\tau(x) \neq \emptyset$ and $\tau(x) \cap \tau(y) = \emptyset$;
(ii) if $x \in y \in \mathrm{tc}(A)$ and $u \in \tau(y)$, then $\tau(x) \cap \mathrm{tc}(u) \neq \emptyset$.

(b) is the "multi-valued" version of (a), and generalizes (a).

The relation $A \preccurlyeq B$ faithfully captures the intuition that A is structurally "no more complex" than B.

Theorem

Suppose $A \preccurlyeq B$. *Then* $\mathrm{rank}(A) \leq \mathrm{rank}(B)$ *and* $|\mathrm{tc}(A)| \leq |\mathrm{tc}(B)|$.

Cobham Set Recursion

Now we can state the full analogue of Cobham limited iteration on notation for sets:

Definition ((Cobham Recursion$_{\preccurlyeq}$))

If g is an $(n{+}1)$-ary function, h is an n-ary function and τ is a n-ary function, then **(Cobham Recursion$_{\preccurlyeq}$)** gives the n-ary function f:

$$f(a,\vec{c}) \;=\; g(\{f(b,\vec{c}) : b \in a\}, a, \vec{c}),$$

provided that, for all $a, \vec{c}$, we have $\tau(x, a, \vec{c}) : f(a,\vec{c}) \preccurlyeq h(a,\vec{c})$.

Remark on "predicativity": Note that τ does not have $f(a,\vec{c})$ as an input. This condition can be relaxed.

The initial functions for the Cobham Recursive Set Functions (CRSF):

$$\pi_j^n(a_1, \ldots, a_n) = a_j$$

$$\mathrm{pair}(a, b) = \{a, b\}$$

$$\mathrm{null}(\,) = \varnothing$$

$$\mathrm{union}(a) = \bigcup a$$

$$\mathrm{cond}_\in(a, b, c, d) = \begin{cases} a & \text{if } c \in d \\ b & \text{otherwise} \end{cases}$$

(Composition) If g is an n-ary function and $\vec{h}$ is a vector of n many m-ary functions, then **(Composition)** gives the m-ary function f:

$$f(\vec{a}) \;=\; g(\vec{h}(\vec{a})).$$

Definition (CRSF)

The *Cobham Recursive Set Functions*, CRSF, are the set functions obtained from the initial functions and the set smash function $\#$ by closing under **(Composition)** and **(Cobham Recursion$_{\preccurlyeq}$)**.
A relation $R(\vec{a})$ is in CRSF iff its characteristic function $\chi_R(\vec{a})$ is in CRSF.

Theorem (Bootstrapping CRSF)

- *The* CRSF *functions include functions such as* $a \setminus b$ *and* $a \cap b$ *and* $\langle a, b \rangle$ *and* $\bigcap a$ *and* $a \odot b$.
- *The* CRSF *relations are closed under Boolean operations and* Δ_0 *(bounded) quantification.*
- CRSF *is closed under* **(Cobham Recursion$_\subseteq$)**, **(Separation)**, **(Bounded Replacement)**, *and* **(Embedded Replacement)**.

(Separation) If g is an n-ary function, then **(Separation)** gives the n-ary function f:

$$f(\vec{a}, c) \;=\; \{b \in c : g(\vec{a}, b) \neq \varnothing\}.$$

(Bounded Replacement) If g is an n-ary function with $n \geq 3$, then **(Bounded Replacement)** gives the $(n-1)$-ary function f:

$$f(\vec{a}, b, c) \;=\; c \cap \bigcup \{ g(\vec{a}, x, b, c) : x \in b \}.$$

(Embedded Replacement) If g is an $(n+1)$-ary function, h is an n-ary function, and τ is an $(n+1)$-ary function, then **(Embedded Replacement)** gives the n-ary function f:

$$f(\vec{a}, b) \;=\; \{ g(\vec{a}, x, b) : x \in b \}$$

provided that, for all $\vec{a}, b$, we have $\tau(x, \vec{a}, b) : f(\vec{a}, b) \preccurlyeq h(\vec{a}, b)$.

Theorem

The function $a \mapsto \mathrm{rank}(a)$ *is in* CRSF.

Proof sketch: Using **(Cobham Recursion$_{\preccurlyeq}$)** and **(Embedded Replacement)**, define

$$\mathrm{rank}^+(a) \;=\; \mathrm{Succ}(\bigcup\{\mathrm{rank}^+(x) : x \in a\}),$$

where $\mathrm{Succ}(y) = y \cup \{y\}$. The bounding function is $h(a) = a$. The (multivalued!) embedding τ is defined by

$$\tau(x, a) \;=\; \{a' \in \mathrm{tc}(a) : \mathrm{rank}(a') = \mathrm{rank}(x)\}.$$

This is a $\preccurlyeq$ embedding, but we need to show that τ is in CRSF.

Proof continued: $\tau(x,a) := \{a' \in a : \mathrm{rank}(a') = \mathrm{rank}(x)\}$.

To show $\tau \in \mathrm{CRSF}$, use the CRSF function $\mathrm{RksLE}(a,b)$ which gives the set of $y \in \mathrm{tc}(a)$ with $\mathrm{rank}(y) \leq \mathrm{rank}(b)$, defined by **(Cobham Recursion$_\subseteq$)** and **(Separation)**:

$$\mathrm{RksLE}(a,b) \;=\; \{a' \in \mathrm{tc}(a) : a' \subseteq \bigcup\{\mathrm{RksLE}(a,b') : b' \in b\}\}$$

So $\mathrm{rank}(a) \leq \mathrm{rank}(b)$ iff $a \in \mathrm{RksLE}(\{a\},b)$.

Now define τ as a CRSF function by

$$\tau(x,a) \;=\; \{a' \in \mathrm{tc}(a) : \mathrm{RksLE}(\{x\},a') \wedge \mathrm{RksLE}(\{a'\},x)\}.$$

using **(Cobham Recursion$_\subseteq$)**. □

A normal form for embeddings

Definition (# term)

Let $v_1, v_2, v_3, \ldots$ be variables (ranging over sets). A *#-term* is a term built up from variables, the constant symbol $1 = \{\varnothing\}$, and the function symbols $\odot$ and $\#$.

Any #-term is a CRSF function. The #-terms give arbitrarily large polynomially growth rate of ranks and of sizes of transitive closures.

Theorem (Bounding with #-terms)

Let $f(a_1, \ldots, a_k)$ be in CRSF. Then there is a #-term $t(a_1, \ldots, a_k)$ and a CRSF function $\tau(x, a_1, \ldots, a_k)$ such that $\tau : f(a_1, \ldots, a_k) \preccurlyeq t(a_1, \ldots, a_k)$.

(Proof omitted.) In fact CRSF would be equivalently defined if **(Cobham Recursion$_\preccurlyeq$)** was changed to require the bounding function $h(a, \vec{c})$ to be an #-term.(!)

Corollary (Polynomially bounding CRSF functions)

Let $f(\vec{a})$ be a CRSF function. Then there are polynomials p and q so that

- $\mathrm{rank}(f(\vec{a})) \leq p(\max_i\{\mathrm{rank}(a_i)\})$ *and*
- $|\mathrm{tc}(f(\vec{a}))| \leq q(\max_i(|\mathrm{tc}(a_i)|))$.

This corollary is an indication that CRSF is a correct notion of polynomial time computation for set functions.

Unbounded Replacement

Theorem

CRSF *is closed under* **(Replacement)**.

(Replacement) If g is an n-ary function with $n \geq 2$, then **(Replacement)** gives the $(n-1)$-ary function f:

$$f(\vec{a}, b) \;=\; \{g(\vec{a}, x, b) : x \in b\}.$$

Corollary

$a \times b$ *is in* CRSF.

Proof: Define, with two applications of **(Replacement)**,

$$a \times b \;=\; \bigcup\{\{x\} \times b : x \in a\},$$

where

$$\{x\} \times b \;:=\; \{\langle x, y\rangle : y \in b\}. \quad \square$$

Course of values recursion

(Cobham Recursion$^{\mathrm{CofV}}_{\preccurlyeq}$) If $n \geq 1$, g is an $(n{+}1)$-ary function, h is an n-ary function and τ is an $(n{+}1)$-ary function, then **(Cobham Recursion$^{\mathrm{CofV}}_{\preccurlyeq}$)** gives the n-ary function f:

$$f(\vec{a}, c) \;=\; g(\vec{a}, c, f_{\restriction \mathrm{tc}(c)}(\vec{a}, -)),$$

provided that, for all $a, \vec{c}$, we have $\tau(x, \vec{a}, c) : f(\vec{a}, c) \preccurlyeq h(\vec{a}, c)$.

Defn: $f_{\restriction \mathrm{tc}(c)}(\vec{a}, -)$ equals the set of ordered pairs $\langle c', f(\vec{a}, c')\rangle$ such that $c' \in \mathrm{tc}(c)$.

Theorem

CRSF *is closed under* **(Cobham Recursion$^{\mathrm{CofV}}_{\preccurlyeq}$)**.

CRSF equals polynomial time on HF

There are several encodings of binary strings as hereditary sets, c.f. [Sazonov'97], also [Beckmann, B., Sy Friedman; Arai].
A binary string w of length ℓ should be encoded by a set $\nu(w)$ of rank $O(\ell)$ and $|\mathrm{tc}(\nu(w))| = O(\ell)$ (alternately, $\ell^{O(1)}$.)

Definition

If $w = w_0 \cdots w_{\ell-1} \in \{0,1\}^*$, then

$$\nu(w) \;=\; \{i : w_i = 1\} \cup \{\ell\}.$$

Definition

A function $f : (\{0,1\}^*)^n \to \{0,1\}^*$ is *represented by* the n-ary set function F (under the encoding ν) provided

$$F(\nu(a_1), \ldots, \nu(a_n)) \;=\; \nu(f(a_1, \ldots, a_n))$$

for all $a_1, \ldots, a_n \in \{0,1\}^*$. When this holds, we write $f = F^\nu$.

Theorem

Every polynomial time function is represented by a function in CRSF *under the encoding* ν*.*

Proof idea: Show that Cobham limited iteration on notation can be simulated by CRSF functions. □

A little more detail on the proof:
Use induction on the definition of polynomial time functions with composition and limited iteration.
The base cases, and closure under composition, are easy.
A few other easy-to-define set functions:

$$\begin{aligned} \mathrm{Pred}(N) &:= N \dot{-} 1 \qquad \text{for } N \text{ an integer} \\ S{\restriction}X &:= \text{Restriction of } S \text{ to domain } X \\ \mathrm{Bit}(i,S) &:= 1 \text{ if } i \in S, \text{ 0 otherwise} \end{aligned}$$

(Think of S as both a set of bit positions and a coding of a binary string. Thus $S{\restriction}N := (S \cap N) \cup \{N\}$ is the first N bits of S)

For f defined by limited iteration on notation:

$$\begin{aligned} f(\epsilon, \vec{a}) &= g(\vec{a}) \\ f(si, \vec{a}) &= h_i(s, \vec{a}, f(s, \vec{a})) \quad \text{for } i = 0, 1. \end{aligned}$$

Define F' so that, for N an integer,

$$F'(N, \nu(a_1), \ldots, \nu(a_\ell), \nu(s)) = \nu(f(a_1, \ldots, a_\ell, s \restriction N)).$$

Let G, H_0, H_1 define g, h_0, h_1.
Use **(Cobham Recursion$^{\mathrm{CofV}}_{\preccurlyeq}$)** to define:

$$F'(N, \vec{A}, S) = \begin{cases} G(\vec{A}) & \text{if } N = 0 \text{ or } N \notin \omega \\ H_i(\vec{A}, F'(\mathrm{Pred}(N), \vec{A}, S), S \restriction \mathrm{Pred}(N)) & \\ & \text{if } N \neq 0 \text{ and } \mathrm{Bit}(\mathrm{Pred}(N), S) = i \\ & \text{for } i = 0, 1. \end{cases}$$

□

Theorem

Suppose $F \in \mathrm{CRSF}$. Then $f = F^{\nu}$ is polynomial time.

Proof sketch: Use induction on the definition of CRSF functions to prove the following:

> *Suppose that F is a CRSF function. Then, there is a polynomial time function F_{Mos} which, given Mostowski graphs for hereditarily finite sets $\vec{a}$, computes the Mostowski graph for $F(\vec{a})$.*
> *In addition, there are polynomial time methods to convert between binary strings and their Mostowski graphs.* □

Remark: Binary strings are encoded by finite sets $\nu(w)$ with cardinality $|\mathrm{tc}(\nu(w)|$ of the transitive closure polynomially bounded by $\mathrm{rank}(\nu(w))$.
Thus the proof only needs F_{Mos} for Mostowski graphs with cardinality polynomially bounded by rank.

Converse to the above *proof* fails: There are polynomial time functions acting on general finite Mostowski graphs that do not give CRSF functions.

Example: The function $x \mapsto |\mathrm{tc}(x)|$,

sending x to the von Neumann integer $|\mathrm{tc}(x)|$.

This is contrast to the notion of polynomial time for bounded set theory [Sazonov'97].

Relation to PCSF^+

[Beckmann, B., S.Friedman] introduced a class of "safe recusive" set functions SRSF based on a generalization, to set functions, of Bellantoni and Cook's safe/normal characterization of polynomial time computable functions. [BC'92].

In the safe/normal approach, functions $f(\vec{a}/\vec{b})$ have two kinds of inputs:

- the inputs $\vec{a}$ are *normal* inputs, and
- the inputs $\vec{b}$ are *safe* inputs.

The idea is that normal inputs can be used for recursion, whereas safe inputs cannot.

Bellantoni and Cook's motivation was to avoid the use of the smash function to bound the values obtained by recursion.

Theorem (Beckman, B, S. Friedman)

The SRSF *functions, using the encoding* ν*, can compute precisely the functions which are computable with alternating Turing machines which use exponential time and polynomially many alternations.*

Arai modified these definitions to obtain a class of predicatively computable set functions, PCSF, which exactly captures polynomial time.

Theorem (Arai)

The PCSF *functions, using the encoding* ν*, can compute precisely the polynomial time functions.*

We obtain the class PCSF^+ by adding closure under:

(Normal Separation$^{\mathsf{SN}}$) If g is a m, n-ary function with $n \geq 1$, then normal separation gives the m, n-ary function f:

$$f(\vec{d}/\vec{a}, c) \;=\; \{b \in c : g(\vec{d}/\vec{a}, b) \neq \varnothing\}.$$

The same theorem holds for PCSF^+ (by the same proof as for PCSF).

An equivalence of CRSF and PCSF^+

Theorem

Suppose $f(\vec{a}/\vec{b})$ is a PCSF^+ function. Then there are CRSF *functions $g(\vec{a},\vec{b})$ and $\tau(x,\vec{a},\vec{b})$, and an # term $t(\vec{a})$ such that, for all $\vec{a},\vec{b}$,*

a. $g(\vec{a},\vec{b}) = f(\vec{a}/\vec{b})$, *and*

b. $\tau : f(\vec{a}/\vec{b}) \preccurlyeq t(\vec{a}) \odot \{\vec{b}\}$.

c. τ *is the identity on* $\mathrm{tc}(\{\vec{b}\})$*:*

- *If* $z \in \mathrm{tc}(\{\vec{b}\})$, *then* $\tau(z,\vec{a},\vec{b}) = \{z\}$.
- *If* $\tau(z,\vec{a},\vec{b}) \cap \mathrm{tc}(\{\vec{b}\}) \neq \emptyset$, *then* $z \in \mathrm{tc}(\{\vec{b}\})$.

Part a. of the Theorem shows that CRSF includes PCSF^+.

Parts b. and c. give strong bounds on the dependence of PCSF^+ functions on their normal and safe inputs.
(Sharpens bounds of [Arai].)

Second direction of the equivalence:

Theorem

If $f(\vec{a})$ is a CRSF function, then $f(\vec{a}/)$ is a PCSF^+ function.

Proof uses induction on the closure properties of CRSF functions, and a very delicate analysis of how CRSF functions can be defined in PCSF^+.

Essential difficulty: CRSF is able to recurse on values obtained by recursion; PCSF^+ is not. We have to replace $\in$-recursion on a set x with $\in$-recursion on a set u such that $x \preccurlyeq u$.

Corollary

CRSF *and* PCSF^+ *are equivalent: For all set functions* f*,*

$$f(\vec{a}) \text{ is a } \mathrm{CRSF} \text{ function}$$

$$\text{iff}$$

$$g(\vec{a}/) = f(\vec{a}) \text{ is a } \mathrm{PCSF}^+ \text{ function.}$$

Remark: Arai conjectures that PCSF^+ is distinct from PCSF; however, this is still open.

Future directions.

- Understand better how CRSF functions differ from the SRSF functions. SRSF has characterizations in terms of the relativized Gödel L and Jensen S hierarchies, and in terms of polynomial time computation on infinite Turing machines. [Beckmann-B-Sy Friedman]. E.g., CRSF cannot compute the set of finite subsets of an infinite set; SRSF can.
  Which class is more natural: SRSF or CRSF?
- Are there interesting feasible complexity classes that arise when CRSF functions are restricted to other types of hereditarily finite sets. For instance, space bounded complexity, alternating time, or other kinds of parallelism?
- (Work in progress.) Develop a proof theory for CRSF, along the lines of S^1_2 (bounded arithmetic) or Rathjen's theory for primitive recursive computation on sets.
- (Work in progress.) Develop a circuit model of computation on sets.

Thank you!

Спасибо!