

# Cobham recursive set functions

Sam Buss  
Univ. of California, San Diego

Workshop on Sets and Computations  
IMS, National University Singapore April 13, 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]

- Cobham recursion:

[Beckmann, B., Sy Friedman, Müller, Thapen], this talk.

## Cobham-style definition of polynomial time (P):

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

*Initial functions* including

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

$\text{cond}(a, c, d) = c$  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:

$$f(\epsilon, \vec{a}) = g(\vec{a})$$

$$f(s0, \vec{a}) = h_0(s, \vec{a}, f(s, \vec{a}))$$

$$f(s1, \vec{a}) = h_1(s, \vec{a}, f(s, \vec{a})).$$

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  $p$ .

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.

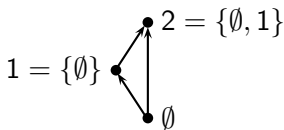
### Definition (Set composition $\odot$ )

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

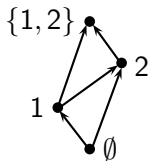
$$\emptyset \odot b = b$$

$$a \odot b = \{x \odot b : x \in a\}, \quad \text{for } a \neq \emptyset$$

Example with Mostowski graphs for sets:



(a)  $A$



(b)  $B$



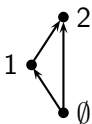
(c)  $A \odot B$

## Definition (Set smash #)

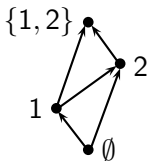
The *set smash* function is the function  $a\#b$  defined by  $\in$ -recursion on  $a$  as

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

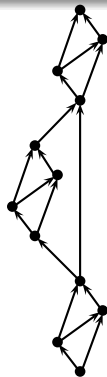
Mostowski graphs example:



(a)  $A$



(b)  $B$



(d)  $A\#B$

## Theorem

*The set smash function  $\#$  satisfies the following:*

1.  $\text{rank}(a\#b) + 1 = (\text{rank}(b) + 1)(\text{rank}(a) + 1)$ .
2.  $|\text{tc}(a\#b)| + 1 = (|\text{tc}(a)| + 1)(|\text{tc}(b)| + 1)$ .

*Equivalently,  $|\text{tc}^+(a\#b)| = |\text{tc}^+(a)| \cdot |\text{tc}^+(b)|$ ,  
where  $\text{tc}^+(a) := \{a\} \cup \text{tc}(a)$ .*

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

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 ( $\preceq$ embedding)

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

- (a) There is injective  $\tau : \text{tc}(A) \rightarrow \text{tc}(B)$  s.t. for all  $x \in y \in \text{tc}(A)$ , we have  $\tau(x) \in \text{tc}(\tau(y))$ , or
- (b) There is  $\tau : \text{tc}(A) \rightarrow \mathcal{P}(\text{tc}(B))$ , s.t.
  - (i) if  $x \neq y$ , then  $\tau(x) \neq \emptyset$  and  $\tau(x) \cap \tau(y) = \emptyset$ ;
  - (ii) if  $x \in y \in \text{tc}(A)$  and  $u \in \tau(y)$ , then  $\tau(x) \cap \text{tc}(u) \neq \emptyset$ .

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

The relation  $A \preceq B$  faithfully captures the intuition that  $A$  is structurally “no more complex” than  $B$ .

## Theorem

Suppose  $A \preceq B$ . Then  $\text{rank}(A) \leq \text{rank}(B)$  and  $|\text{tc}(A)| \leq |\text{tc}(B)|$ .



# Cobham Set Recursion

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

## Definition ((Cobham Recursion $\preceq$ ))

If  $g$  is an  $(n+1)$ -ary function,  $h$  is an  $n$ -ary function and  $\tau$  is a  $n$ -ary function, then **(Cobham Recursion $\preceq$ )** 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  $\tau(x, a, \vec{c}) : f(a, \vec{c}) \preceq h(a, \vec{c})$ .

Remark on “predicativity”: Note that  $\tau$  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):

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

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

$$\text{null}() = \emptyset$$

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

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

**(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 <sub>$\preceq$</sub> )**. 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  $\Delta_0$  (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  $f$ :

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

**(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  $\tau$  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  $\tau(x, \vec{a}, b) : f(\vec{a}, b) \preceq h(\vec{a}, b)$ .

## Theorem

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

*Proof sketch:* Using **(Cobham Recursion <sub>$\preceq$</sub> )** and **(Embedded Replacement)**, define

$$\text{rank}(a) = \text{Succ}\left(\bigcup\{\text{rank}^+(x) : x \in a\}\right),$$

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

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

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

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

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

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

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

Now define  $\tau$  as a CRSF function by

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

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

# A normal form for embeddings

## Definition (# term)

Let  $v_1, v_2, v_3, \dots$  be variables (ranging over sets). A  $\#$ -term is a term built up from variables, the constant symbol  $1 = \{\emptyset\}$ , 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  $f(a_1, \dots, a_k)$  be in CRSF. Then there is a  $\#$ -term  $t(a_1, \dots, a_k)$  and a CRSF function  $\tau(x, a_1, \dots, a_k)$  such that  $\tau : f(a_1, \dots, a_k) \preceq t(a_1, \dots, a_k)$ .

(Proof omitted.) In fact CRSF would be equivalently defined if **(Cobham Recursion <sub>$\preceq$</sub> )** 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

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

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

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

**(Cobham Recursion<sub>≲</sub><sup>CofV</sup>)** If  $n \geq 1$ ,  $g$  is an  $(n+1)$ -ary function,  $h$  is an  $n$ -ary function and  $\tau$  is an  $(n+1)$ -ary function, then **(Cobham Recursion<sub>≲</sub><sup>CofV</sup>)** gives the  $n$ -ary function  $f$ :

$$f(\vec{a}, c) = g(\vec{a}, c, f_{\upharpoonright_{\text{tc}(c)}}(\vec{a}, -)),$$

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

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

## Theorem

CRSF is closed under **(Cobham Recursion<sub>≲</sub><sup>CofV</sup>)**.

# 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  $\ell$  should be encoded by a set  $\nu(w)$  of rank  $O(\ell)$  and  $|\text{tc}(\nu(w))| = O(\ell)$  (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  $f : (\{0, 1\}^*)^n \rightarrow \{0, 1\}^*$  is *represented* by the  $n$ -ary set function  $F$  (under the encoding  $\nu$ ) provided

$$F(\nu(a_1), \dots, \nu(a_n)) = \nu(f(a_1, \dots, a_n))$$

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

## Theorem

*Every polynomial time function is represented by a function in CRSF under the encoding  $\nu$ .*

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

## Theorem

Suppose  $F \in \text{CRSF}$ . 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  $F_{\text{Mos}}$  which, given Mostowski graphs for hereditarily finite sets  $\vec{a}$ , computes the Mostowski graph for  $F(\vec{a})$ .* □

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

[Beckmann, B., S.Friedman] introduced a class of “safe recursive” set functions  $\text{SRSF}$  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  $\nu$ , 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  $\nu$ , can compute precisely the polynomial time functions.*

We obtain the class  $\text{PCSF}^+$  by adding closure under:

**(Normal Separation<sup>SN</sup>)** If  $g$  is a  $m, n$ -ary function with  $n \geq 1$ , then normal separation gives the  $m, n$ -ary function  $f$ :

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

The same theorem holds for  $\text{PCSF}^+$  (by the same proof as for PCSF).

## Theorem

Suppose  $f(\vec{a}/\vec{b})$  is a PCSF<sup>+</sup> 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  $\vec{a}, \vec{b}$ ,

- $g(\vec{a}, \vec{b}) = f(\vec{a}/\vec{b})$ , and
- $\tau : f(\vec{a}/\vec{b}) \preceq t(\vec{a}) \odot \{\vec{b}\}$ .
- $\tau$  is the identity on  $\text{tc}(\{\vec{b}\})$ :
  - If  $z \in \text{tc}(\{\vec{b}\})$ , then  $\tau(z, \vec{a}, \vec{b}) = \{z\}$ .
  - If  $\tau(z, \vec{a}, \vec{b}) \cap \text{tc}(\{\vec{b}\}) \neq \emptyset$ , then  $z \in \text{tc}(\{\vec{b}\})$ .

Part a. of the Theorem shows that CRSF includes PCSF<sup>+</sup>.

Parts b. and c. give strong bounds on the dependence of PCSF<sup>+</sup> 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<sup>+</sup> 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<sup>+</sup>.

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

## Corollary

CRSF and PCSF<sup>+</sup> are equivalent: For all set functions  $f$ ,

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

iff

$g(\vec{a}/) = f(\vec{a})$  is a PCSF<sup>+</sup> function.

Remark: Arai conjectures that PCSF<sup>+</sup> 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  $S_2^1$  (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!