Cobham Recursive Set Functions

Submitted to Annals of Pure and Applied Logic

Arnold Beckmann*[1]
Department of Computer Science
Swansea University
Swansea SA2 8PP, UK
a.beckmann@swansea.ac.uk

Sam Buss[†1].
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093-0112, USA
sbuss@ucsd.edu

Sy-David Friedman[‡1]
Kurt Gödel Research Center
University of Vienna
A-1090 Vienna, Austria
sdf@logic.univie.ac.at

Moritz Müller[1]
Kurt Gödel Research Center
University of Vienna
A-1090 Vienna, Austria
moritz.mueller@univie.ac.at

Neil Thapen[§1]
Institute of Mathematics
Academy of Sciences of the Czech Republic
Prague, Czech Republic
thapen@math.cas.cz

August 8, 2015

[1]The initial results of this work were obtained at the Kurt Gödel Institute at the University of Vienna, with funding provided by Short Visit Grant 4932 from the Humanities Research Networking Programmes of the European Science Foundation (ESF).

[*]Also supported by the Simons Foundation (grant 208717 to S. Buss)

[†]Supported in part by NSF grants DMS-1101228 and CCR-1213151, and by the Simons Foundation, award 306202

[‡]Supported in part by Austrian Science Fund (FWF), Project Number I-1238 on "Definability and Computability"

[§]Supported in part by grant P202/12/G061 of GA ČR and RVO: 67985840

Abstract

This paper introduces the Cobham Recursive Set Functions (CRSF) as a version of polynomial time computable functions on general sets, based on a limited (bounded) form of $\in$-recursion. This is inspired by Cobham's classic definition of polynomial time functions based on limited recursion on notation. We introduce a new set composition function, and a new smash function for sets which allows polynomial increases in the ranks and in the cardinalities of transitive closures. We bootstrap CRSF, prove closure under (unbounded) replacement, and prove that any CRSF function is embeddable into a smash term. When restricted to natural encodings of binary strings as hereditarily finite sets, the CRSF functions define precisely the polynomial time computable functions on binary strings. Prior work of Beckmann, Buss and Friedman and of Arai introduced set functions based on safe-normal recursion in the sense of Bellantoni-Cook. We prove an equivalence between our class CRSF and a variant of Arai's predicatively computable set functions.

1 Introduction

This paper presents a definition of "Cobham Recursive Set Functions" which is designed to be a version of polynomial time computability based on computation on sets. This represents an alternate (or, a competing) approach to the recent work of Beckmann, Buss and S. Friedman [2], who defined the Safe Recursive Set Functions (SRSF), and to the work of Arai [1], who introduced the Predicatively Computable Set Functions (PCSF). SRSF and PCSF were based on Bellantoni-Cook style safe-normal recursion, but using $\in$-recursion for computation on sets in place of recursion on strings. Both [2] and [1] were motivated by the desire to find analogues of polynomial time native to sets. For hereditarily finite sets, the class SRSF turned out to correspond to functions computable by Turing machines which use alternating exponential time with polynomially many alternations. For infinite sets, SRSF corresponds to definability at a polynomial level in the relativized L-hierarchy. For infinite binary strings of length ω, it corresponds to computation by infinite polynomial time Turing machines, which use time less than ω^n for some $n > 0$. The class PCSF, on the other hand, does correspond to polynomial time functions when restricted to appropriate encodings of strings by hereditarily finite sets. No characterization of PCSF for non-hereditarily finite sets is known.

In this paper, we give a different approach to polynomial time computability on sets, using an analogue of Cobham limited recursion on notation, inspired by one of the original definitions of polynomial time computable functions [7]. The class P (sometimes denoted FP) of polynomial time computable functions on binary strings can be defined as the smallest class of functions that (a) contains as initial functions the constant empty string ϵ, the two successor functions $s \mapsto s0$ and $s \mapsto s1$ and the projection functions, and (b) is closed under composition and limited recursion on notation. If g, h_0 and h_1 are functions, and p is a polynomial, then the following function f is said to be defined by *limited recursion on notation*:

$$\begin{aligned} f(\vec{a}, \epsilon) &= g(\vec{a}) \\ f(\vec{a}, s0) &= h_0(\vec{a}, f(\vec{a}, s), s) \\ f(\vec{a}, s1) &= h_1(\vec{a}, f(\vec{a}, s), s) \end{aligned} \tag{1}$$

provided that

$$|f(a_1, \dots, a_n, s)| \ \le \ p(|a_1|, \dots, |a_n|, |s|) \tag{2}$$

always holds. Here $\vec{a}$ and s are (vectors of) binary strings; and $|a|$ denotes the length of the binary string a.

A slightly different version of limited recursion uses the smash (#) function instead (c.f., [10] and [6]). For this, the smash function is defined as $a\#b = 0^{|a|\cdot|b|}$ so that $|a\#b|$ is the product of the lengths of the binary strings a and b. The smash function can be included in the small set of initial functions, and then the bound (2) can be replaced by the condition that

$$|f(\vec{a}, s)| \le |k(\vec{a}, s)| \tag{3}$$

where k is a function already known to be in P. In this version, f is said to be defined by *limited recursion on notation from g, h_0, h_1 and k.*

Section 3 defines the Cobham Recursion Set Functions (CRSF) via an analogue of the definition of polynomial time functions with limited recursion. CRSF uses $\in$-recursion instead of recursion on notation. In $\in$-recursion, the value of $f(x)$, for x a set, is defined in terms of the set of values $f(y)$ for all $y \in x$. This means that the recursive computation of $f(x)$ requires computing $f(y)$ for all y in the transitive closure, $\mathrm{tc}(x)$, of x. The depth of the recursion is bounded by the rank, $\mathrm{rank}(x)$, of x. Of course, the cardinality of the transitive closure of x, $|\mathrm{tc}(x)|$, can be substantially larger than the cardinality of the rank of x. The computational complexity of $f(x)$ is thus bounded by both the rank of x and by $|\mathrm{tc}(x)|$; however, the bounds act in different ways. The intuition is that $|\mathrm{tc}(x)|$ polynomially bounds the overall work performed to compute $f(x)$, while $\mathrm{rank}(x)$ polynomially bounds the depth of the recursion in the computation of $f(x)$.

The definition of CRSF requires a set-theoretic analogue of the binary string # function. For this, Section 2 introduces a new *set composition function*, denoted $\odot$, and a new *set smash function*, denoted $\#$. The binary string function # allows defining functions of polynomial growth rate. The set smash function $\#$ is used to bound the sizes of sets introduced by $\in$-recursion. The set function $\#$, which can be viewed as a structured crossproduct, thus plays a similar role to the binary string # function. However, the set smash function has to do double duty by providing polynomial bounds on both the ranks of sets and the cardinalities of the transitive closures of sets. Namely, if $z = x\#y$, then (a) the rank of z is polynomially bounded by the ranks of x and y and (b) $|\mathrm{tc}(z)|$ is polynomially bounded by $|\mathrm{tc}(x)|$ and $|\mathrm{tc}(y)|$. The set function smash does more than just bound the ranks and cardinalities; it also bounds the internal structure of sets. For this reason, the bounding condition (3) needs to be replaced by a more complicated condition called $\preccurlyeq$-embeddability. Section 2 defines "τ $\preccurlyeq$-embeds x into y", denoted $\tau : x \preccurlyeq y$, in a way that faithfully captures the notion that x is structurally "no more complex" than y. For technical reasons, the function τ is a one-to-many mapping. The condition "$\tau : f(\vec{a}, s) \preccurlyeq k(\vec{a}, s)$" is then the analogue of (3) which works for Cobham recursion on sets.

The outline of the paper is as follows. Section 2 defines the set composition and smash functions; these are defined first using $\in$-recursion and then in terms of Mostowski graphs. Section 3.1 defines various operations on set functions, and the class CRSF of Cobham Recursive Set Functions. Section 3.2 does simple bootstrapping of CRSF, and shows the crossproduct and rank functions are in CRSF. Section 3.3 gives a normal form for CRSF functions by showing that a restricted class of "$\#$-terms" can be used as the $\preccurlyeq$-bounds. As a corollary, it is shown that the growth rate of CRSF functions can be polynomially bounded. Sections 3.4 and 3.5 show that CRSF is closed under (unbounded) replacement and under course-of-values recursion. Section 3.6 proves that CRSF is closed under an impredicative version of Cobham recursion, which has a relaxed embedding condition.

Section 4 takes up the question of how CRSF functions correspond to polynomial time computability on binary strings. Following [11, 2, 1], we choose a natural method of encoding binary strings as hereditarily finite sets. We then prove that, relative to these encodings, the CRSF functions are precisely the usual polynomial time computable functions. As mentioned earlier, similar results were obtained by Arai for the PCSF functions. Sazonov [11] also defined a class of polynomial time set functions. Sazonov's polynomial time functions are the same as CRSF functions when operating on hereditarily finite sets suitably encoding binary strings, but are rather different for inputs which are general sets. In particular, Sazonov's functions when taking general hereditarily finite sets as inputs can be characterized as functions which operate in polynomial time on the (finite) Mostowski graphs of the inputs. In contrast, our CRSF functions have recursion depth bounded by a polynomial of the rank of its inputs. As a result, CRSF is a more restricted computational model of polynomial computation for general hereditarily finite sets. We feel it is natural and desirable that the computational power of CRSF depends on the ranks and the hereditary structure of its inputs.

Section 5 discusses a relationship between CRSF and PCSF. Instead of using the class PCSF identified by Arai, we work with a (conjecturally) larger class of functions which we call PCSF^+. Theorems 35 and 36 and Corollary 37 state that CRSF and PCSF^+ have equivalent power over all sets (taking inputs as normal inputs in the case of PCSF^+).

The present paper is part of a cycle of three papers in preparation about CRSF functions. Another paper [3] discusses circuit computation models for set functions based on an alternative formulation of CRSF. A third paper [4] discusses set theoretic axioms and proof theory for CRSF.

Throughout the paper, we work in theory ZFC of Zermelo-Fraenkel set theory with choice. The axiom of choice is used only when we discuss cardinalities, and is not needed for anything else.

2 The set smash and lex smash functions

This section defines the "smash" function # for sets. We define a set composition operation $\odot$ and then the set smash function. We then present intuitive conceptual definitions of these functions in terms of the Mostowski graphs of sets.

Definition 1. *The* set composition *function is the function $a \odot b$ defined by $\in$-recursion as*

$$\begin{aligned} \emptyset \odot b &= b \\ a \odot b &= \{x \odot b : x \in a\}, \quad \textit{for } a \neq \emptyset. \end{aligned}$$

We use $\mathrm{rank}(a)$ and $\mathrm{tc}(a)$ to denote the rank and the transitive closure of a. We write $\mathrm{tc}^+(a)$ for $\mathrm{tc}(a) \cup \{a\}$, and $\mathrm{rank}^+(a)$ for $\mathrm{rank}(a) + 1$. As usual, $|a|$ denotes the cardinality of a.

Lemma 2. *The set composition function $\odot$ satisfies the following:*

1. $a\odot\emptyset = a$.
2. $\mathrm{rank}(a\odot b) = \mathrm{rank}(b) + \mathrm{rank}(a)$.
3. *If $a \neq a'$, then $a\odot b \neq a'\odot b$.*
4. $\mathrm{tc}(a\odot b) = \mathrm{tc}(b) \cup \{a'\odot b : a' \in \mathrm{tc}(a)\}$.
5. $|\mathrm{tc}(a\odot b)| = |\mathrm{tc}(a)| + |\mathrm{tc}(b)|$.
6. *$\odot$ is associative: $a\odot(b\odot c) = (a\odot b)\odot c$..*

Proof. Parts 1., 2., 4., and 6. are easily proved by $\in$-induction on a. Part 3. is proved using extensionality and induction on the ranks of a and a'. Part 5. is an immediate consequence of parts 3. and 4. and the observation that $b \in \mathrm{tc}^+(a'\odot b)$, so the right hand side of part 4. is a disjoint union. □

Definition 3. *The* set smash *function is the function $a\#b$ defined by $\in$-recursion on a as*

$$a\#b \;=\; b\odot\{x\#b : x \in a\}. \tag{4}$$

Lemma 4. *The set smash function $\#$ satisfies the following:*

1. $\emptyset\#b = b$
2. $a\#\emptyset = a$
3. $\mathrm{rank}(a\#b)+1 = (\mathrm{rank}(b)+1)(\mathrm{rank}(a)+1)$. *Equivalently,* $\mathrm{rank}^+(a\#b) = \mathrm{rank}^+(b) \cdot \mathrm{rank}^+(a)$.
4. $|\mathrm{tc}(a\#b)| + 1 = (|\mathrm{tc}(a)| + 1)(|\mathrm{tc}(b)| + 1)$. *Equivalently,* $|\mathrm{tc}^+(a\#b)| = |\mathrm{tc}^+(a)| \cdot |\mathrm{tc}^+(b)|$.
5. *$\#$ is associative.*

Proof. Part 1. is immediate from the definitions. Parts 2. and 3. are readily proved by $\in$-induction on a. We postpone the proof of part 4. until after discussing the Mostowski graph next. For part 5. we can first prove the following kind of distributive law by $\in$-induction on a:

$$(a\odot b)\#c \;=\; (a\#c)\odot\{y\#c : y \in b\}.$$

Using this one easily proves $a\#(b\#c) = (a\#b)\#c$ by $\in$-induction on a. □

Observe that we do not have a general distributive law of the form

$$(a\odot b)\#c \;=\; (a\#c)\odot(b\#c),$$

as $\mathrm{rank}((1\odot 1)\#1) = 5$ but $\mathrm{rank}((1\#1)\odot(1\#1)) = 6$.

An intuitive understanding of the $\odot$ and $\#$ functions can be obtained by considering the Mostowski graph of a set.

Definition 5. *Let A be a set. The* Mostowski graph *of A is the directed graph with vertex set $V = \mathrm{tc}^+(A)$, and edge relation E defined by $\langle v_1, v_2\rangle \in E$ iff $v_1 \in v_2$. More generally, any directed graph isomorphic to the Mostowski graph of A is called a Mostowski graph of A.*

The Mostowski graph of A is *well-founded* (i.e., any subset of V has an E-minimal element) and is *extensional* (i.e., any distinct v_1, v_2 in V have different sets of E-predecessors). Furthermore, a Mostowski graph must be "pointed": (V, E) is *pointed* provided there is a $v \in V$ such that for all $v' \in V$, $v' E^* v$ holds, where E^* is the reflexive, transitive closure of E. This v is the unique sink node of (V, E); in fact, v corresponds to the vertex A. Conversely, it is an elementary fact that any well-founded, extensional, pointed, directed graph is a Mostowski graph for a unique set.

As usual, the integers are coded as von Neumann integers, so $0 = \emptyset$, $1 = \{0\}$, $2 = \{0, 1\}$, etc. The Mostowski graphs of $A = 2$ and $B = \{1, 2\}$ are shown in Figure 1.

We now define $\odot$ and $\#$ in terms of Mostowski graphs. First, note that extensionality and wellfoundedness imply that a Mostowski graph has a unique source node, and that pointedness and wellfoundedness imply that it has a unique sink node. Let $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$ be Mostowski graphs for the sets A and B. Then, assuming $V_A \cap V_B = \emptyset$, the Mostowski graph (V, E) for $A\odot B$ can be obtained by identifying the sink vertex of G_B and the source vertex of G_A. In other words, the sink node of B is replaced by a copy of G_A; equivalently, the source node of A is replaced by a copy of G_B. (See Figure 1.)

More formally, a Mostowski graph for $A\odot B$ can be defined letting the nodes be $V := \{\langle 1, a\rangle : a \in \mathrm{tc}^+(A)\} \cup \{\langle 0, b\rangle : b \in \mathrm{tc}(B)\}$, and letting the edges be the following:

- $\langle\langle 0, b'\rangle, \langle 0, b\rangle\rangle$ for all $b' \in b \in \mathrm{tc}(B)$,
- $\langle\langle 0, b\rangle, \langle 1, \emptyset\rangle\rangle$ for all $b \in B$, and
- $\langle\langle 1, a'\rangle, \langle 1, a\rangle\rangle$ for all $a' \in a \in \mathrm{tc}^+(A)$.

Note that the nodes $\langle 0, b\rangle$ correspond to the sets b, and the nodes $\langle 1, a\rangle$ correspond to the sets $a\odot B$.

The Mostowski graph of $A\#B$ is obtained by replacing every vertex of G_A with a copy of the graph G_B. This is pictured in Figure 1. Formally, we can define a Mostowski graph $G = (V, E)$ for $A\#B$ by letting the graph have vertex set $V = \{\langle a, b\rangle : a \in \mathrm{tc}^+(A), b \in \mathrm{tc}^+(B)\}$, and letting the edge set E contain:

- $\langle\langle a, b'\rangle, \langle a, b\rangle\rangle$ for $b' \in b \in \mathrm{tc}^+(B)$ and
- $\langle\langle a', B\rangle, \langle a, \emptyset\rangle$ for $a' \in a \in \mathrm{tc}^+(A)$.

The intent is that $\langle a, b\rangle$ corresponds to the set

$$\sigma_{A,B}(a, b) \;:=\; b\odot\{a'\#B : a' \in a\}. \tag{5}$$

It is easy to check, by a double $\in$-recursion, that the nodes $\langle a, b\rangle$ of G actually correspond to these sets: For $b \neq \emptyset$, the members of $\sigma_{A,B}(a, b)$ are the sets $\sigma_{A,B}(a, b')$ for $b' \in b$. From (4) and (5), $\sigma_{A,B}(a, B) = a\#B$. Therefore,

$$\sigma_{A,B}(a, \emptyset) \;=\; \{a'\#B : a' \in a\} \;=\; \{\sigma_{A,B}(a', B) : a' \in a\}.$$

From these facts, it follows readily that G is extentional and is a correct Mostowski graph of $a\#b$. Part 4. of Lemma 4 follows immediately.

Clearly, (5) defines a bijection between $\mathrm{tc}^+(A)\times\mathrm{tc}^+(B)$ and $\mathrm{tc}^+(A\#B)$. This lets $\#$ serve as a replacement for the crossproduct functions. The analogous projection functions $\pi_{1,A,B}$ and $\pi_{2,A,B}$ are defined so that, for $u = \sigma_{A,B}(a, b)$ with $a \in \mathrm{tc}^+(A)$ and $b \in \mathrm{tc}^+(B)$, we have $\pi_{1,A,B}(u) = a$ and $\pi_{2,A,B}(u) = b$.

As a side remark, the functions $\sigma_{A,B}$, $\pi_{1,A,B}$ and $\pi_{2,A,B}$ do not depend on A at all. However, in our applications, the set A is always known, and it seems less confusing to include A in the subscript than to omit it.

For purposes of illustration, we conclude this section by mentioning a variant of the set smash function, called the "lex smash". Like the set smash, lex smash uses the crossproduct $A \times B$ as the vertex set of its Mostowski graph; it is the set whose Mostowski graph is the lexicographic product of the Mostowski graphs of A and B.

Definition 6. *The* lex smash function *maps a pair of sets a and b to the set $a\#^{\mathrm{lex}}b$ which is the set with Mostowski graph (V, E) defined by $V = \mathrm{tc}^+(a) \times \mathrm{tc}^+(b)$, and*

$$E(\langle a_1, b_1\rangle, \langle a_2, b_2\rangle) \;\Leftrightarrow\; a_1 \in a_2 \vee (a_1 = a_2 \wedge b_1 \in b_2). \tag{6}$$

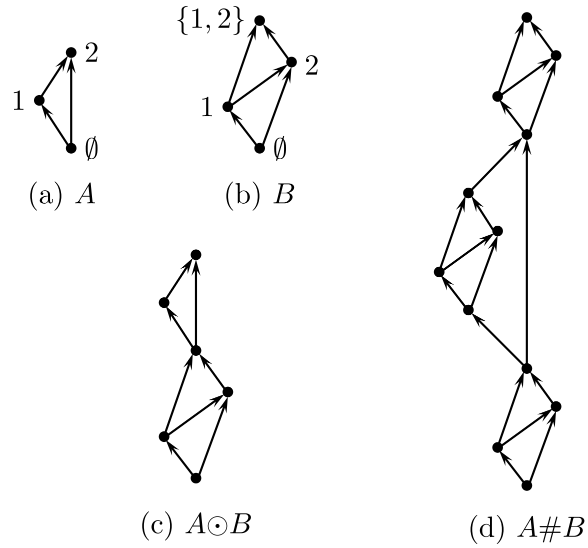

Figure 1: The Mostowski graphs for (a) $A = 2$, (b) $B = \{1, 2\}$, (c) $A \odot B$, and (d) $A\#B$.

The lex smash $a\#^{\text{lex}}b$ has structure similar to $a\#b$ but with more edges in its Mostowski graph. We expect that using $\#^{\text{lex}}$ instead of $\#$ would give us the same class of functions CRSF, but we prefer $\#$ because it has a simple recursive definition.

Theorem 7. *If* (6) *holds, so* $\langle a_1, b_1\rangle$ *precedes* $\langle a_2, b_2\rangle$, *then* $\sigma_{A,B}(a_1, b_1) \in \mathrm{tc}(\sigma(a_2, b_2))$.

The proof of Theorem 7 is obvious from the Mostowski graph representation of $A\#B$.

3 Cobham recursive set functions

This section defines the Cobham recursive set functions (CRSF) and proves a variety of closure properties.

3.1 Definition of CRSF

CRSF will be defined as an algebra of functions which take sets as inputs and produce sets as outputs. The following are the initial functions for CRSF.

(Projection) For $1 \leq j \leq n$,

$$\pi^n_j(a_1, \ldots, a_n) \;=\; a_j$$

(Pair)

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

(Null)

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

(Union)

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

(Conditional$_\in$)

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

CRSF also enjoys a variety of closure properties. Some of these hold by definition, and others will be derived.

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

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

(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:

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

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

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

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

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

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

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

Note that the function h serves as a size bound for the values of f. This size bound is rather crude however, since it requires $f(x, \vec{y})$ to be a subset of $h(x, \vec{y})$. Definition 9 gives a more general notion of size bound by requiring only that (the transitive closure of) $f(x, \vec{y})$ be "embedded" into (the transitive closure of) $h(x, \vec{y})$. We first define a simplified notion of "single-valued" embedding.

Definition 8. *A set A is* single-valued $\preccurlyeq$-embeddable *in a set B if the following holds: There is an injective function $\tau : \mathrm{tc}(A) \to \mathrm{tc}(B)$ such that for all $x \in y \in \mathrm{tc}(A)$, we have $\tau(x) \in \mathrm{tc}(\tau(y))$. We call τ a* single-valued embedding *of A into B.*

The idea for embeddings is that $\mathrm{tc}(A)$ and $\mathrm{tc}(B)$ are identified with the Mostowski graphs of A and B. The relation "$\tau(x) \in \mathrm{tc}(\tau(y))$" means that $\tau(x)$ precedes $\tau(y)$ in the sense that there is a non-trivial path in the Mostowski graph from $\tau(x)$ to $\tau(y)$. The function τ shows that a copy of A is contained inside B, so A is structurally "no more complex" than B.

The actual definition of embedding uses multi-valued embeddings; namely $\tau(x)$ will be a subset of $\mathrm{tc}(B)$, and in effect, is mapping x to each member of $\tau(x)$. Let $\mathcal{P}(\cdots)$ denote the power set.

Definition 9. *A set A is* $\preccurlyeq$-embeddable *in a set B, denoted $A \preccurlyeq B$, if the following holds: There is a function $\tau : \mathrm{tc}(A) \to \mathcal{P}(\mathrm{tc}(B))$ such that for all x, $\tau(x) \neq \emptyset$ and for all $x \neq y$, $\tau(x) \cap \tau(y) = \emptyset$, and such that for all $x \in y$ in $\mathrm{tc}(A)$ and every $u \in \tau(y)$, there is some $v \in \tau(x) \cap \mathrm{tc}(u)$. We call τ an* embedding *of A into B. The condition $\tau(x) \cap \tau(y) = \emptyset$ for $x \neq y$ is called the* injectivity *condition.*

As we shall see, $A \preccurlyeq B$ is a more general way to capture the intuition that A is structurally "no more complex" than B. A single-valued embedding τ can easily be converted into a (multi-valued) embedding, namely via $x \mapsto \{\tau(x)\}$. The multi-valued embedding $x \mapsto \{x\}$ is called the *identity* embedding.

The next proposition gives bounds on the rank of A and the cardinality of $\mathrm{tc}(A)$. The proof is simple and left to the reader.

Proposition 10. *Suppose $A \preccurlyeq B$. Then* $\mathrm{rank}(A) \leq \mathrm{rank}(B)$ *and* $|\mathrm{tc}(A)| \leq |\mathrm{tc}(B)|$.

An example of an embedding is given by Theorem 7. Here, the map that sends (the set corresponding to) $\langle x, y \rangle$ to $\sigma_{A,B}(x, y)$ is a single-valued $\preccurlyeq$-embedding of $A \#^{\mathrm{lex}} B$ into $A \# B$.

(Cobham Recursion$_\preccurlyeq$) 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$_\preccurlyeq$)** gives the n-ary function f:

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

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

The last condition means that the function $x \mapsto \tau(x, \vec{a}, c)$ is an embedding $f(\vec{a}, c) \preccurlyeq h(\vec{a}, c)$. Later, in Section 3.6, we will use a more general, impredicative notion of embedding which allows $f(\vec{a}, c)$ to also be an input to τ.

There is also an embedded version of replacement:

(Embedded Replacement) 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 **(Embedded Replacement)** gives the n-ary function f:

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

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

Cobham recursion can also be defined using a course-of-values ("CofV") recursion. If $f(\vec{a}, c)$ is a function, let $f_{\restriction c}(\vec{a}, -)$ denote the set of ordered pairs $\langle c', f(\vec{a}, c')\rangle$ such that $c' \in c$. As usual, an ordered pair $\langle x, y\rangle$ is equal to $\{\{x\}, \{x, y\}\}$.

(Cobham Recursion$_\preccurlyeq^{\mathrm{CofV}}$) 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$_\preccurlyeq^{\mathrm{CofV}}$)** gives the n-ary function f:

$$f(\vec{a}, c) \;=\; g(\vec{a}, c, f_{\restriction \mathrm{tc}(c)}(\vec{a}, -)), \tag{7}$$

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

Definition 11. *Recall that integers are represented by the von Neumann integers. The* characteristic function χ_R *of a relation* R *is defined by* $\chi_R(\vec{a}) = 1$ *if* $R(\vec{a})$ *and* $\chi_R(\vec{a}) = 0$ *if* $\neg R(\vec{a})$.

We now give the formal definition of CRSF.

Definition 12. *The* Cobham Recursive Set Functions, CRSF, *are the functions that are obtained by starting with the initial functions* **(Projection)**, **(Pair)**, **(Null)**, **(Union)**, **(Conditional$_\in$)**, *and the set smash function* $\#$, *and taking the closure 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.

The next theorem shows that CRSF is also closed under **(Bounded Replacement)** and **(Embedded Replacement)**, as well as Δ_0-separation. It follows that CRSF contains all the rudimentary relations [8]. Later on, Theorems 23 and 29 will show closure under **(Replacement)** and **(Cobham Recursion$^{\mathrm{CofV}}_{\preccurlyeq}$)**. Theorem 30 will prove closure under impredicative versions of Cobham recursion.

3.2 Simple closure properties for CRSF

Theorem 13 establishes some basic properties of CRSF. After that, the crossproduct and rank functions are shown to be in CRSF; however, the proof for crossproduct will be finished only after CRSF is shown to be closed under **(Replacement)**.

It is useful to note that parts 1.-11. of Theorem 13 do not require the use of smash, and part 1. does not use recursion. Furthermore, its proof requires only single-valued embeddings. (Subsequent to Theorem 13 we will need almost exclusively to consider multi-valued embeddings.)

Theorem 13. *The following hold for* CRSF.

1. CRSF *contains the functions* $a \mapsto \{a\}$ *and*

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

2. CRSF *is closed under* **(Embedded Replacement)**.
3. CRSF *is closed under* **(Separation)**.
4. CRSF *contains the binary functions* $a \setminus b$ *and* $a \cap b$.
5. CRSF *is closed under* **(Cobham Recursion$_{\subseteq}$)**.
6. CRSF *is closed under* **(Bounded Replacement)**.
7. *The* CRSF *relations are closed under Boolean operations.*
8. *The* CRSF *relations are closed under bounded* (Δ_0) *quantification.*
9. *The function* $a \mapsto \bigcap a$ *is in* CRSF.

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

11. *The function* $a, b \mapsto \langle a, b\rangle := \{\{a\}, \{a, b\}\}$ *is in* CRSF. *In addition,* CRSF *contains projection functions satisfying* $\pi_1(\langle a, b\rangle) = a$ *and* $\pi_2(\langle a, b\rangle) = b$, *and contains the relation* $\mathrm{ispair}(x)$ *that tests whether* x *is an ordered pair* $\langle a, b\rangle$.

12. *The binary functions* $\odot$ *and* $\odot^{-1}$ *are in* CRSF, *where*

$$a \odot^{-1} b \ = \ \begin{cases} z & \textit{such that } a = z \odot b \\ \emptyset & \textit{if no such } z \textit{ exists.} \end{cases}$$

13. *The three functions* $a, b, a', b' \mapsto \sigma_{a,b}(a', b')$ *and* $a, b, x \mapsto \pi_{1,a,b}(x)$ *and* $a, b, x \mapsto \pi_{2,a,b}(x)$ *are in* CRSF.

Proof. 1. As usual, $\{a\} = \{a, a\}$. Then $\mathrm{cond}_=$ can be defined as $\mathrm{cond}_\in(a, b, c, \{d\})$.

2. To define f from g, h and τ as in **(Embedded Replacement)**, first define k by

$$k(\vec{a}, b, c) \ = \ \begin{cases} g(\vec{a}, b, c) & \text{if } b \in c \\ \{k(\vec{a}, b, c) : b \in c\} & \text{if } b = c \\ \emptyset & \text{otherwise} \end{cases}$$

with the aid of $\mathrm{cond}_=$ and $\mathrm{cond}_\in$, and using **(Cobham Recursion$_\preccurlyeq$)** with the bounding function $h'(\vec{a}, b, c) = h(\vec{a}, c)$ and the embedding function $\tau'(x, \vec{a}, b, c) = \tau(x, \vec{a}, c)$. Then $f(\vec{a}, c) = k(\vec{a}, c, c)$.

3. To define f from g as in **(Separation)**, first define $k(\vec{a}, b, c)$, again with the aid of $\mathrm{cond}_=$ and $\mathrm{cond}_\in$, as

$$k(\vec{a}, b, c) \ = \ \begin{cases} \{b\} & \text{if } b \in c \text{ and } g(\vec{a}, b) \neq 0 \\ \emptyset & \text{otherwise.} \end{cases}$$

Then define $f(\vec{a}, c) = \bigcup\{k(\vec{a}, b, c) : b \in c\}$ by **(Embedded Replacement)** using the bounding function $h(\vec{a}, b, c) = c$, and the single-valued embedding function $\tau(x, \vec{a}, b, c) = x$.

4. The set difference function can be defined using **(Separation)** by

$$a \setminus b \ = \ \{x \in a : \mathrm{cond}_\in(\emptyset, 1, x, b) \neq \emptyset\}.$$

Intersection is defined by $a \cap b = ((a \cup b) \setminus (a \setminus b)) \setminus (b \setminus a)$.

5. The fact that **(Cobham Recursion$_\subseteq$)** can be simulated by **(Cobham Recursion$_\preccurlyeq$)** is immediate from the facts that binary intersection ($\cap$) is in CRSF and that $a \subseteq b$ implies $a \preccurlyeq b$ using the identity function as a single-valued embedding.

6. Suppose f is defined from g and h as in **(Bounded Replacement)**. Then define

$$g'(\vec{a}, b, c) \;=\; \begin{cases} g(\vec{a}, b, c) & \text{if } g(\vec{a}, b, c) \in h(\vec{a}, c) \\ h(\vec{a}, c) & \text{otherwise.} \end{cases}$$

Define $f'(\vec{a}, c) = \{g'(\vec{a}, b, c) : b \in c\}$ by **(Embedded Replacement)** using the bounding function $h'(\vec{a}, c) = \{h(\vec{a}, c)\} = 1 \odot h(\vec{a}, c)$ and the single-valued embedding function $\tau(x, \vec{a}, b, c) = x$. Finally, $f(\vec{a}, c) = f'(\vec{a}, c) \setminus \{h(\vec{a}, c)\}$, so $f \in \mathrm{CRSF}$.

7. Define the function $f_\neg$ and $f_\vee$ by

$$f_\neg(a) \;=\; 1 \setminus a \qquad \text{and} \qquad f_\vee(a, b) \;=\; a \cup b.$$

These functions implement negation and disjunction, and therefore, by composition, the CRSF relations are closed under Boolean operations.

8. To show closure under Δ_0 quantification, it now suffices to prove that if R is a CRSF relation, then so is $S(\vec{a}, c) \Leftrightarrow \exists b{\in}c\, R(\vec{a}, c)$. For this, define

$$\chi_S(\vec{a}, c) \;=\; \bigcup \{\chi_R(\vec{a}, b) : b \in c\}.$$

This is a valid use of **(Bounded Replacement)** and **(Union)** since $\chi_S(a, \vec{c}) \subseteq 1$.

9. $\bigcap a$ can be defined as $\{x \in \bigcup a : \forall y{\in}a\, (x \in y)\}$.

10. The transitive closure $\mathrm{tc}(a)$ can be defined using **(Cobham Recursion$_{\preccurlyeq}$)** as

$$\mathrm{tc}(a) \;=\; a \cup \bigcup \{\mathrm{tc}(x) : x \in a\}$$

with the bounding function $h(a) = a$ since $\mathrm{tc}(a) \preccurlyeq a$ using the identity function as the single-valued embedding.

11. The ordered pair function $\langle a, b\rangle$ is in CRSF as it is defined with three uses of pair. To define the projection functions note that, for all a, b,

$$\{a, b\} \;=\; \bigcup \langle a, b\rangle,$$

$$a \;=\; \bigcup \{z \in \{a, b\} : \{z\} \in \langle a, b\rangle\},$$

$$b \;=\; \begin{cases} a & \text{if } \langle a, b\rangle = \langle a, a\rangle \\ \bigcup(\{a, b\} \setminus \{a\}) & \text{otherwise.} \end{cases}$$

These facts immediately allow π_1 and π_2 to be expressed as CRSF functions. Finally,

$$\mathrm{ispair}(z) \;=\; \mathrm{cond}_=(1, \emptyset, z, \langle \pi_1(z), \pi_2(z)\rangle).$$

12. The function $a \odot b$ is defined as in Definition 1 by **(Cobham Recursion$_{\preccurlyeq}$)** by letting

$$a \odot b \;=\; \begin{cases} b & \text{if } a = \emptyset \\ \{a' \odot b : a' \in a\} & \text{otherwise.} \end{cases}$$

For the bounding function, let $h(a, b) = a \# b$.[1] The single-valued embedding function τ can be defined as

$$\tau(x, a, b) \;=\; \begin{cases} x & \text{if } x \in \mathrm{tc}^+(b) \\ x \# b & \text{otherwise.} \end{cases}$$

To define $\odot^{-1}$, observe that there is a z such that $a = z \odot b$ exactly when

$$b \in \mathrm{tc}^+(a) \wedge (\forall c \in \mathrm{tc}^+(a))(c \in \mathrm{tc}^+(b) \vee (\forall d \in c)(b \in \mathrm{tc}^+(d))). \qquad (8)$$

So $\odot^{-1}$ is defined by **(Cobham Recursion$_{\preccurlyeq}$)** as

$$a \odot^{-1} b \;=\; \begin{cases} \{a' \odot^{-1} b : a' \in a\} & \text{if } a \neq b \text{ and (8) holds} \\ \emptyset & \text{otherwise.} \end{cases}$$

For the single-valued embedding, let $h(a, b) = a \odot b$ and $\tau(x, a, b) = x \odot b$.

13. The function $a, b \mapsto \{a' \# b : a' \in a\}$ is defined by **(Bounded Replacement)**, since $a' \# b \in \mathrm{tc}(a \# b)$ for $a' \in a$. Therefore, (5) gives a CRSF definition of $\sigma_{a,b}(a', b')$.

The function $\pi_{1,a,b}$ can be defined using **(Separation)** and **(Union)** as[2]

$$\pi_{1,a,b}(u) \;=\; \bigcup \{a' \in \mathrm{tc}^+(a) : \exists b' \in \mathrm{tc}^+(b) \text{ s.t. } u = \sigma_{a,b}(a', b')\}.$$

Note that the union is taken over a set of size at most one. The function $\pi_{2,a,b}$ is defined similarly. □

The next theorem states that crossproduct is a CRSF function; for this, # is needed. This is not surprising as # is itself a kind of crossproduct; however, the proof is somewhat difficult and will be completed in Section 3.4.

Theorem 14. *The crossproduct function $a \times b$ is in* CRSF.

The proof of Theorem 14 defines crossproduct as

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

[1] It is overkill to use the # function to bound the $\odot$ function. The alternative would be to include $\odot$ in the base functions in the definition of CRSF.

[2] Here we take advantage of the fact that a is available, but with a little more work it is possible to define $\pi_{1,a,b}(u)$ without using a.

where

$$\{z\} \times b \;:=\; \{\langle z, b' \rangle : b' \in b\},$$

and uses two applications of **(Replacement)** and **(Union)**. However, the closure of CRSF under **(Replacement)** will not be proved until Theorem 23. We thus postpone completing the proof of Theorem 14 pending the proof of Theorem 23.

We now prove that the rank function is in CRSF. The proof uses **(Cobham Recursion$_{\preccurlyeq}$)**, but establishing the embedding condition is unexpectedly difficult and uses a multi-valued embedding. We do not know any way to use a single-valued embedding instead.

Theorem 15. *The function* $a \mapsto \mathrm{rank}(a)$ *is in* CRSF*.*

Proof. Since $\mathrm{rank}(a) = \bigcup \mathrm{rank}^+(a)$, it suffices to show rank^+ is in CRSF. The latter can be defined using **(Cobham Recursion$_{\preccurlyeq}$)** since

$$\mathrm{rank}^+(a) \;=\; \mathrm{Succ}\Big(\bigcup\{\mathrm{rank}^+(x) : x \in a\}\Big), \tag{9}$$

where $\mathrm{Succ}(S) = S \cup \{S\}$. For the bounding function, take $h(a) = \{a\}$. We define the (multi-valued) embedding τ by letting $\tau(\alpha)$ equal the members of $\mathrm{tc}^+(a)$ of rank α. This τ is defined with the aid of a function $\mathrm{RksLE}(a, b)$ (for "ranks less than or equal to") which is equal to the set of $a' \in \mathrm{tc}^+(a)$ which have rank $\leq \mathrm{rank}(b)$. RksLE is defined using **(Cobham Recursion$_{\subseteq}$)** and **(Separation)** by

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

We have $\mathrm{rank}(a) \leq \mathrm{rank}(b)$ iff $a \in \mathrm{RksLE}(a, b)$. Then τ is defined using **(Separation)** as

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

3.3 #-terms as bounding functions

The section states and proves a crucial technical result which states that CRSF functions can be embedded into sets constructed from terms, called "#-terms", involving $\odot$ and #. Corollary 22 shows that this immediately implies polynomial bounds on the growth rates of CRSF functions. This is also the key tool needed in Section 3.4 for the proof that CRSF is closed under **(Replacement)**.

Definition 16. *A* #-term *is a term built up from variables, the constant symbol* 1*, and the function symbols* $\odot$ *and* #*.*

Any #-term $t(a_1, \ldots, a_k)$ represents a CRSF function. The next theorem shows that CRSF functions have bounded growth rate in the sense that their values are embeddable in a #-term.

Theorem 17. *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* $\tau : f(a_1, \ldots, a_k) \preccurlyeq t(a_1, \ldots, a_k)$*.*

For the embedding τ of Theorem 17, the inputs a_i serve as parameters, or "side variables": it is the mapping $x \mapsto \tau(x, \vec{a})$ that satisfies the properties of Definition 9. Before proving Theorem 17, we establish some simple lemmas showing how #-terms act like monotone functions w.r.t. embeddings.

Lemma 18. $\preccurlyeq$ *is transitive: If* $A \preccurlyeq B$ *and* $B \preccurlyeq C$*, then* $A \preccurlyeq C$*. Furthermore, if* $\tau_1 : A \preccurlyeq B$ *and* $\tau_2 : B \preccurlyeq C$ *are valid, where* A*,* B*,* C*,* τ_1 *and* τ_2 *are given by* CRSF *functions, then there is a* CRSF *function* τ *such that* $\tau : A \preccurlyeq C$ *is valid.*

The hypothesis of the second half of Lemma 18 means that there are parameters $\vec{a}$ so that $A = A(\vec{a})$, $B = B(\vec{a})$ and $C = C(\vec{a})$ are functions of $\vec{a}$, and the embeddings τ_i depend on the parameters and have the forms $x \mapsto \tau_i(x, \vec{a})$. In this, case, the function $x \mapsto \tau(x, \vec{a})$ gives an embedding $\tau : A(\vec{a}) \preccurlyeq C(\vec{a})$.

Proof. Let $\tau_1 : A \preccurlyeq B$ and $\tau_2 : B \preccurlyeq C$. Thus, $\tau_1 : \mathrm{tc}(A) \to \mathcal{P}(\mathrm{tc}(B))$ and $\tau_2 : \mathrm{tc}(B) \to \mathcal{P}(\mathrm{tc}(C))$. Define $\tau : \mathrm{tc}(A) \to \mathcal{P}(\mathrm{tc}(C))$ by letting

$$\tau(x) \;=\; \bigcup\{\tau_2(z) : z \in \tau_1(x)\}. \tag{10}$$

We claim $\tau : A \preccurlyeq C$. It is clear that if $x \neq y$, then $\tau(x) \cap \tau(y) = \emptyset$. Suppose $x \in y \in \mathrm{tc}(A)$ and $u \in \tau(y)$. We have $u \in \tau_2(z)$ for some $z \in \tau_1(y)$. Since τ_1 is a $\preccurlyeq$-embedding, there is a $w \in \tau_1(x) \cap \mathrm{tc}(z)$. By the definition of transitive closure, there is a finite sequence $w_0{=}w, w_1, \ldots, w_\ell{=}z$ such that $w_i \in w_{i+1}$ for all i. Since τ_2 is also a $\preccurlyeq$-embedding, there are $v_0, \ldots, v_\ell{=}u$ such that each $v_i \in \tau_2(w_i) \cap \mathrm{tc}(v_{i+1})$. Thus $v = v_0$ is in $\tau_2(w) \cap \mathrm{tc}(u)$ and hence in $\tau(x) \cap \mathrm{tc}(u)$.

Since $\tau(x) \subseteq \mathrm{tc}(C)$, τ is defined by **(Bounded Replacement)** and **(Union)**, or alternately by **(Separation)**. Thus by Theorem 13, τ is in CRSF if A, B, C, τ_1 and τ_2 are. ☐

Lemma 19. *If $A \preccurlyeq B$ and $C \preccurlyeq D$, then $A \odot C \preccurlyeq B \odot D$ and $A \# C \preccurlyeq B \# D$. Furthermore, if $\tau_1 : A \preccurlyeq B$ and $\tau_2 : C \preccurlyeq D$ are valid, for A, B, C, D, τ_1 and τ_2 given by* CRSF *functions, then there are* CRSF *functions τ and τ' such that $\tau : A \odot C \preccurlyeq B \odot D$ and $\tau' : A \# C \preccurlyeq B \# D$ are valid.*

Proof. Let $\tau_1 : A \preccurlyeq B$ and $\tau_2 : C \preccurlyeq D$. Define $\tau : \mathrm{tc}(A \odot C) \to \mathcal{P}(\mathrm{tc}(B \odot D))$ by setting $\tau(x) = \tau_2(x)$ for $x \in \mathrm{tc}(C)$, and setting $\tau(x \odot C) = \tau_1(x) \odot D$ for $x \in \mathrm{tc}(A)$. More formally,

$$\tau(x) \;=\; \begin{cases} \tau_2(x) & \text{if } x \in \mathrm{tc}(C) \\ \{y \odot D : y \in \tau_1(x \odot^{-1} C)\} & \text{otherwise.} \end{cases} \tag{11}$$

Since $\tau(x) \subseteq \mathrm{tc}(B \odot D)$, closure under **(Separation)** implies that if A, B, C, D, τ_1 and τ_2 are given by CRSF functions, then so is τ.

We claim that $\tau : A \odot C \preccurlyeq B \odot D$. The fact that $\tau(x)$ and $\tau(y)$ are disjoint for $x \neq y$ follows from the properties of τ_1 and τ_2 and part 3. of Lemma 2. So, suppose $x \in y \in \mathrm{tc}(A \odot C)$ and $u \in \tau(y)$. We need to prove there is a $v \in \tau(x) \cap \mathrm{tc}(u)$. There are three cases to consider. The first case is where $x \in y \in \mathrm{tc}(C)$: there must be a $v \in \tau_2(x) \cap \mathrm{tc}(u)$, and this v works for τ as well. The second case is when $x = x' \odot C$ and $y = y' \odot C$ for some $x' \in y' \in \mathrm{tc}(A)$. We also have $u = u' \odot D$ and $u' \in \tau_1(y')$; thus there is a $v' \in \tau_1(x') \cap \mathrm{tc}(u')$. Then $v = v' \odot D \in \tau(x) \cap \mathrm{tc}(u)$. The third case is where $x \in C$ and $y = C$. Then, $\tau(x) \subseteq \mathrm{tc}(D)$, and $\tau(y) = \tau_1(\emptyset) \odot D$. Any $u \in \tau(y)$ has the form $u = u' \odot D$, and since $\tau(x)$ and $\tau_1(\emptyset)$ are both non-empty, the desired v exists. Thus $\tau : A \odot C \preccurlyeq B \odot D$.

For the second assertion, we now define $\tau' : \mathrm{tc}(A \# C) \to \mathcal{P}(\mathrm{tc}(B \# D))$. For $x \in \mathrm{tc}(A \# C)$, set $y = \pi_{1,A,C}(x)$ and $z = \pi_{2,A,C}(x)$ so that $y \in \mathrm{tc}^+(A)$, $z \in \mathrm{tc}^+(C)$, and $x = \sigma_{A,C}(y, z) = z \odot \{y' \# C : y' \in y\}$, and define

$$\tau'(x) \;=\; \{\sigma_{B,D}(y', z') : y' \in \tau_1^+(y) \text{ and } z' \in \tau_2^+(z)\}, \tag{12}$$

where τ_2^+ is the same as τ_2 except extended to map C to $\{D\}$. It is easy to verify that if A, B, C, D, τ_1 and τ_2 are given by CRSF functions, then so is τ'.

The proof that $\tau' : A \# C \preccurlyeq B \# D$ is similar in spirit to the above argument and is left for the reader. □

Lemma 20. *Suppose $\tau_i : A_i \preccurlyeq B_i$ is valid for $i = 1, \ldots, n$, where A_i, B_i and τ_i are given by* CRSF *functions. Let $t(x_1, \ldots, x_n)$ be a $\#$-term. Then there is a* CRSF *function τ so that $\tau : t(\vec{A}) \preccurlyeq t(\vec{B})$ is valid.*

Proof. This follows readily from Lemmas 18 and 19, and induction on the complexity of t. □

Proof of Theorem 17. We use induction based on the definition of CRSF functions. For the projection function π^n_j, it is trivial; just take the bounding term $t(a_1, \ldots, a_n) = a_j$ and let the single-valued embedding function τ be the identity function. For $\mathrm{cond}_\in$, use $t(a, b, c, d) = a \odot b$ and define a single-valued embedding τ by letting $\tau(x)$ equal x for $x \in \mathrm{tc}(b)$ and equal $x \odot b$ for $x \in \mathrm{tc}(a) \setminus \mathrm{tc}(b)$. For $\mathrm{pair}(a, b)$, let $t(a, b) = 1 \odot a \odot 1 \odot b$ and define a single-valued embedding $\tau(x)$ to equal x for $x \in \mathrm{tc}^+(b)$ and to equal $x \odot 1 \odot b$ for $x \in \mathrm{tc}^+(a) \setminus \mathrm{tc}^+(b)$. For union, the identity function is a single-valued embedding of $\bigcup a$ into a. Of course, the set smash function $a\#b$ is single-valued $\preccurlyeq$-embedded into itself by the identity function.

Suppose $f(\vec{a})$ is defined by **(Composition)** from $g(u_1, \ldots, u_n)$ and $h_i(\vec{a})$ for $i = 1, \ldots, n$. By the induction hypothesis, there are #-terms s and t_i and CRSF functions $\tau(x, \vec{u})$ and $\tau_i(x, \vec{a})$ so that $\tau : g(\vec{u}) \preccurlyeq s(\vec{u})$ and $\tau_i : h_i(\vec{a}) \preccurlyeq t_i(\vec{a})$ for $i = 1, \ldots, n$. In particular, $\tau(x, \vec{h}(\vec{a})) : g(\vec{h}(\vec{a})) \preccurlyeq s(\vec{h}(\vec{a}))$. Lemma 20 gives a CRSF-function $\sigma(x, \vec{a})$ so that $\sigma : s(\vec{h}(\vec{a})) \preccurlyeq s(\vec{t}(\vec{a}))$. Then Lemma 18 gives $\rho(x, \vec{a})$ such that $\rho : f(\vec{a}) = g(\vec{h}(\vec{a})) \preccurlyeq s(\vec{t}(\vec{a}))$ as desired.

Finally, if f is defined by **(Cobham Recursion$_\preccurlyeq$)**, then $\tau : f(a, \vec{c}) \preccurlyeq h(a, \vec{c})$ for some CRSF functions h and τ. The induction hypothesis gives a #-term $t(a, \vec{c})$ and a CRSF function τ' such that $\tau' : h(a, \vec{c}) \preccurlyeq t(a, \vec{c})$. Lemma 18 immediately gives a CRSF function τ'' so that $\tau'' : f(a, \vec{c}) \preccurlyeq t(a, \vec{c})$. □

The proof of Theorem 17 actually gives a stronger result. Examination of its proof and the proofs of Lemmas 18 and 19 shows that the embedding functions are created using only the closure properties of CRSF established in Theorem 13. Indeed, they are created from the functions $\odot$, #, $\odot^{-1}$, $\pi_{1,A,B}$, $\pi_{2,A,B}$ and functions already shown to be in CRSF using composition and Theorem 13. Furthermore, the proof of Theorem 13 shows that the same closure properties still apply when only #-terms are allowed as bounding functions. This establishes:

Theorem 21. *The class* CRSF *would be unchanged if the definition of* **(Cobham Recursion$_\preccurlyeq$)** *were changed to require the bounding function* $h(\vec{a}, c)$ *to be a #-term.*

Corollary 22. *Let* $f(\vec{a})$ *be a* CRSF *function. Then there are polynomials* p *and* q *so that* $\mathrm{rank}(f(\vec{a})) \le p(\max_i\{\mathrm{rank}(a_i)\})$ *and* $|\mathrm{tc}(f(\vec{a}))| \le q(\max_i\{|\mathrm{tc}(a_i)|\})$.

The corollary follows immediately from Lemma 4, Proposition 10, and Theorem 17. Namely, #-terms have polynomially bounded increase in rank, and polynomially bounded increase in cardinality of their transitive closure, and these bounds are preserved by embeddings.

3.4 Closure of CRSF under replacement

We can now show closure under (unbounded) replacement.

Theorem 23. CRSF *is closed under* **(Replacement)**.

Proof. Suppose $g(\vec{a}, b, c)$ is in CRSF and $f(\vec{a}, c)$ is defined from g by **(Replacement)** as $f(\vec{a}, c) = \{g(\vec{a}, b, c) : b \in c\}$. We must show f is also in CRSF. By Theorem 17, there is a #-term $t_g(\vec{a}, b, c)$ and a CRSF function $\tau_g(x, \vec{a}, b, c)$ such that $\tau_g : g(\vec{a}, b, c) \preccurlyeq t_g(\vec{a}, b, c)$. Since f depends on $\vec{a}$ and c but not b, it is inconvenient to have t_g and τ_g depend on b. Accordingly, we let $t'_g(\vec{a}, c) = t_g(\vec{a}, c, c)$. By Lemmas 18-20, there is a CRSF function $\tau'_g(x, \vec{a}, b, c)$ so that for all $\vec{a}$ and c and all $b \in \mathrm{tc}(c)$, we have $\tau'_g : g(\vec{a}, b, c) \preccurlyeq t'_g(\vec{a}, c)$.

A slight modification of τ'_g gives a CRSF function τ''_g such that $\tau''_g : \{g(\vec{a}, b, c)\} \preccurlyeq t''_g(\vec{a}, c)$, where $t''_g(\vec{a}, c)$ is the #-term $1 \odot t'_g(\vec{a}, c)$

Let $t(\vec{a}, c)$ be the #-term $c \# t'_g(\vec{a}, c)$. The intuition is that $f(\vec{a}, c)$, which is the set of $g(\vec{a}, b, c)$'s for $b \in c$, can be embedded into $t(\vec{a}, c)$ by an embedding τ that sends (the transitive closure of) each $\{g(\vec{a}, b, c)\}$ into the "b-th copy" of $t'_g(\vec{a}, c)$. Formally, we let T abbreviate $t'_g(\vec{a}, c)$, and define

$$\begin{aligned} \tau(x, \vec{a}, c) \ = \ \{z \in c \# T : \pi_{1,c,T}(z) \in c \ \wedge \ x \in \mathrm{tc}^{+}(g(\vec{a}, \pi_{1,c,T}(z), c)) \\ \wedge \ \pi_{2,c,T}(z) \in \tau''_g(x, \vec{a}, \pi_{1,c,T}(z), c)\}. \end{aligned}$$

This is a definition by **(Separation)**, and thus τ is a CRSF function. To understand τ, note that $\tau(x, \vec{a}, c)$ is the set of values $z = \sigma_{c,T}(b, u)$ for the values of b and u such that $b \in c$, $x \in \mathrm{tc}^{+}\{g(\vec{a}, b, c)\}$, and $u \in \tau''_g(x, \vec{a}, b, c)$. From this, it is clear that $\tau : f(\vec{a}, c) \preccurlyeq c \# T = t(\vec{a}, c)$. This means that the definition of $f(\vec{a}, c)$ is actually a definition by **(Embedded Replacement)**, so f is in CRSF. □

This also establishes Theorem 14 about forming crossproducts since, as discussed earlier, it follows from the closure of CRSF under **(Replacement)**.

3.5 Course-of-values encodings

The *graph* of a function f is the class of tuples $\langle \vec{a}, b \rangle$ such that $f(\vec{a}) = b$. When $f = f(\vec{a}, c)$ has a distinguished input c, we will also define the "course-of-values function of f" to be the function f^* such that $f^*(\vec{a}, c)$ gives simultaneously all tuples $\langle c', f(\vec{a}, c')\rangle$ such that $c' \in \mathrm{tc}^+(c)$. The conventional way to encode these tuples would be as a set of ordered pairs; e.g., to define $f^*(\vec{a}, c)$ to be the same as $f_{\restriction \mathrm{tc}^+(c)}(\vec{a}, c)$. However, we shall use an alternate specialized encoding instead. Specifically, we define

$$f^*(\vec{a}, c) \;=\; \{\emptyset, \langle c, f(\vec{a}, c)\rangle\} \odot \{f^*(\vec{a}, c') : c' \in c\}. \tag{13}$$

The intuition is that the tuple $\langle c, f(\vec{a}, c)\rangle$ sits "on top of" all the tuples $\langle c', f(\vec{a}, c')\rangle$ for $c' \in \mathrm{tc}(c)$. This will be helpful for defining $\preccurlyeq$-embeddings, as it can give the embedding function access to the values of $f(\vec{a}, c')$ for $c' \in \mathrm{tc}(c)$.

We record some simple but useful properties of $f^*(\vec{a}, c)$. First, we have that $f^*(\vec{a}, c) = \{z, \langle c \odot z, f(\vec{a}, c) \odot z\rangle\}$ where we write z for the set $\{f^*(\vec{a}, c') : c' \in c\}$. So $f^*(\vec{a}, c)$ has exactly two elements, of different ranks; the lower rank element is z and the higher rank element is an ordered pair. Second, $f^*(\vec{a}, c)$ is not an ordered pair. If $z = \emptyset$, this is direct. Otherwise, an ordered pair is either a singleton or has one element a subset of the other, and neither is possible here. Third, z is not an ordered pair, since an ordered pair must contain a singleton; and, by the above, z does not contain any ordered pairs.

We need a variety of utility CRSF functions to decode structures of the form (13).

Definition 24. *We define*

$$\begin{aligned}
\mathrm{MnR}'(F) &= \textstyle\bigcup\{u \in F : \forall u' \in F,\, \mathrm{rank}(u) \le \mathrm{rank}(u')\} \\
\mathrm{MxR}'(F) &= \textstyle\bigcup\{u \in F : \forall u' \in F,\, \mathrm{rank}(u) \ge \mathrm{rank}(u')\} \\
\mathrm{MxR}(u) &= \mathrm{MxR}'(u) \odot^{-1} \mathrm{MnR}'(u) \\
\mathrm{MxR}_1(u) &= \pi_1(\mathrm{MxR}(u)) \\
\mathrm{MxR}_2(u) &= \pi_2(\mathrm{MxR}(u)).
\end{aligned}$$

Here "MnR" and "MxR" stand for "minimum/maximum rank". If $u = \{\emptyset, \langle c, v\rangle\} \odot z$, then $\mathrm{MxR}'(u) = \langle c, v\rangle \odot z$ and $\mathrm{MnR}'(u) = \emptyset \odot z = z$. Thus $\mathrm{MxR}(u) = \langle c, v\rangle$, $\mathrm{MxR}_1(u) = c$ and $\mathrm{MxR}_2(u) = v$. In particular this gives $\mathrm{MnR}'(f^*(\vec{a}, c)) = \{f^*(\vec{a}, c') : c' \in c\}$, $\mathrm{MxR}_1(f^*(\vec{a}, c)) = c$ and $\mathrm{MxR}_2(f^*(\vec{a}, c)) = f(\vec{a}, c)$. Hence

Proposition 25. *If $f^* \in \mathrm{CRSF}$, then $f \in \mathrm{CRSF}$.*

Lemma 26. *There is a* CRSF *function* AllValues *such that, for any function f and sets $\vec{a}, c$ we have* $\mathrm{AllValues}(f^*(\vec{a}, c)) = f_{\restriction \mathrm{tc}(c)}(\vec{a}, -)$.

Proof. We define an auxiliary function Stars recursively by

$$\text{Stars}(F) = \begin{cases} \emptyset & \text{if } F \text{ is an ordered pair} \\ \bigcup\{\text{Stars}(F') : F' \in F\} & \text{if } F \text{ is not an ordered pair, but} \\ & \quad \text{contains an ordered pair} \\ F \cup \bigcup\{\text{Stars}(F') : F' \in F\} & \text{otherwise.} \end{cases}$$

By the earlier remarks about the structure of f^*, writing z for the set $\{f^*(\vec{a}, c') : c' \in c\}$ we have

$$\begin{aligned} \text{Stars}(f^*(\vec{a}, c)) &= \text{Stars}(\{z, \langle c \odot z, f(\vec{a}, c) \odot z \rangle\}) \\ &= \text{Stars}(z) \cup \text{Stars}(\langle c \odot z, f(\vec{a}, c) \odot z \rangle) \\ &= \text{Stars}(z) \\ &= \{f^*(\vec{a}, c') : c' \in c\} \cup \bigcup_{c' \in c} \text{Stars}(f^*(\vec{a}, c')). \end{aligned}$$

Hence $\text{Stars}(f^*(\vec{a}, c)) = \{f^*(\vec{a}, c') : c' \in \text{tc}(c)\}$. Each value $\text{Stars}(F)$ is a subset of $\text{tc}(F)$, so this is an instance of **(Cobham Recursion$_\subseteq$)** and thus Stars is in CRSF. We define $\text{AllValues}(F)$ by **(Replacement)** as $\{\text{MxR}(u) : u \in \text{Stars}(F)\}$. ☐

Finally, we introduce two predicates IsCofVTop_g and IsCofVSet_g to help us find our place inside the internal structure of sets $f^*(\vec{a}, c)$. These will be used when constructing embeddings from such sets into smash terms.

Definition 27. *Let f be defined by (possibly unbounded) course-of-values recursion from a function g so that*

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

$\text{IsCofVTop}_g(F, \vec{a})$ *expresses that F is a set of the form $f^*(\vec{a}, c)$ for some c.* $\text{IsCofVSet}_g(F, \vec{a})$ *expresses that F is a set of such sets.*

Lemma 28. *If g is in* CRSF*, then so are* IsCofVTop_g *and* IsCofVSet_g*.*

Proof. Combining the recursive definitions of f^* in terms of f and of f in terms of g, we can write down a definition of IsCofVTop_g and IsCofVSet_g by simultaneous recursion. We will do this slightly indirectly. Let IsCofV_g be the function

$$\text{IsCofV}_g(F, \vec{a}) \;=\; \begin{cases} \emptyset & \text{if } \text{IsCofVTop}_g(F, \vec{a}) \\ 1 & \text{if } \text{IsCofVSet}_g(F, \vec{a}) \\ 2 & \text{otherwise.} \end{cases}$$

Since this has range $\{0,1,2\}$, we can write the simultaneous recursion as a definition of IsCofV_g by **(Cobham Recursion$_\subseteq$)**:

$$\mathrm{IsCofV}_g(F,\vec{a}) \;=\; \begin{cases} \emptyset & \text{if } F = \{\emptyset, \langle \mathrm{MxR}_1(F), \mathrm{MxR}_2(F)\rangle\} \odot \mathrm{MnR}'(F), \\ & \qquad \mathrm{IsCofV}_g(\mathrm{MnR}'(F),\vec{a}) = 1, \\ & \qquad \mathrm{MxR}_1(F) = \{\mathrm{MxR}_1(F') : F' \in \mathrm{MnR}'(F)\} \text{ and} \\ & \qquad \mathrm{MxR}_2(F) = g(\vec{a}, \mathrm{MxR}_1(F), \mathrm{AllValues}(F)) \\ 1 & \text{if } \{\mathrm{IsCofV}_g(F',\vec{a}) : F' \in F\} \subseteq \{\emptyset\} \\ 2 & \text{otherwise.} \end{cases}$$

This is not quite an instance of **(Cobham Recursion$_\subseteq$)** as written, but becomes one if we replace the second line of the $\emptyset$ case with $1 \in \{\mathrm{IsCofV}_g(F',\vec{a}) : F' \in F\}$. This is equivalent, since F consists of MnR' and an ordered pair which cannot satisfy $\mathrm{IsCofVSet}_g$. Hence IsCofV_g is in CRSF, so the two predicates are as well. □

Theorem 29. *The* CRSF *functions are closed under* **(Cobham Recursion$^{\mathrm{CofV}}_{\preccurlyeq}$)**.

Proof. Suppose CRSF functions g and τ_1 and a #-term h are used to define a function f_1 by **(Cobham Recursion$^{\mathrm{CofV}}_{\preccurlyeq}$)**

$$f_1(\vec{a},c) \;=\; g(\vec{a},c,f_{1\restriction \mathrm{tc}(c)}(\vec{a},-)),$$

where $\tau_1(x,\vec{a},c) : f_1(\vec{a},c) \preccurlyeq h(\vec{a},c)$. We want to show $f_1 \in \mathrm{CRSF}$. It will be helpful to have c available as an extra side parameter, so we define a new function f by **(Cobham Recursion$^{\mathrm{CofV}}_{\preccurlyeq}$)** as

$$f(\vec{a},c,c') \;=\; \begin{cases} g(\vec{a},c',f_{\restriction \mathrm{tc}(c')}(\vec{a},c,-)) & \text{if } c' \in \mathrm{tc}^+(c) \\ \emptyset & \text{otherwise.} \end{cases} \tag{14}$$

Since $f_1(\vec{a},c) = f(\vec{a},c,c)$, it suffices to prove that $f(\vec{a},c,c')$ is in CRSF. Letting $\tau(x,\vec{a},c,c') = \tau_1(z,\vec{a},c')$, we have $\tau(x,\vec{a},c,c') : f(\vec{a},c,c') \preccurlyeq h(\vec{a},c')$. We henceforth implicitly assume that $c' \in \mathrm{tc}^+(c)$.

Let f^* be the course-of-values function for f:

$$f^*(\vec{a},c,c') \;=\; \{\emptyset, \langle c', f(\vec{a},c,c')\rangle\} \odot \{f^*(\vec{a},c,c'') : c'' \in c'\}. \tag{15}$$

By Proposition 25, it suffices to show f^* is in CRSF. We will use **(Cobham Recursion$_{\preccurlyeq}$)**, by giving a recursive definition of f^*, a bounding term $h^*(\vec{a},c)$ and a CRSF embedding function $\tau^*(x,\vec{a},c)$.[3]

[3]It would be permitted to have c' be a parameter to τ^* and h^*, but we do not need it.

For the recursive definition of f^*, observe that

$$f_{\restriction \mathrm{tc}(c')}(\vec{a}, c, -) = \bigcup_{c'' \in c'} f_{\restriction \mathrm{tc}^+(c'')}(\vec{a}, c, -)$$
$$= \bigcup_{c'' \in c'} \left[\{\mathrm{MxR}(f^*(\vec{a}, c, c''))\} \cup \mathrm{AllValues}(f^*(\vec{a}, c, c''))\right].$$

So from $\{f^*(\vec{a}, c, c'') : c'' \in c'\}$ we can construct $f_{\restriction \mathrm{tc}(c')}(\vec{a}, c, -)$, then use g to construct $f(\vec{a}, c, c')$, then use (15) to construct $f^*(\vec{a}, c, c')$, all in CRSF.

The main difficulty in defining the embedding τ^* is that it has to analyze the meaning of its input x. Here x comes from the course-of-values, but will not in general be a course-of-values set itself, but rather will be a member of $\mathrm{tc}(f^*(\vec{a}, c, c'))$. By construction, for every such x there is $c'' \in \mathrm{tc}(c')$ and $y \in \mathrm{tc}^+(\{\emptyset, \langle c'', f(\vec{a}, c, c'')\rangle\})$ such that

$$x \;=\; y \odot \{f^*(\vec{a}, c, c''') : c''' \in c''\}. \tag{16}$$

Define

$$\mathrm{TopCofVSet}_g(x, \vec{a}) = \bigcup \{F \in \mathrm{tc}^+(x) : \mathrm{IsCofVSet}_g(F, \vec{a}) \wedge$$
$$\neg(\exists F' {\in} \mathrm{tc}^+(x))(F \in \mathrm{tc}(F') \wedge \mathrm{IsCofVSet}_g(F', \vec{a}))\}.$$

We claim that $\mathrm{TopCofVSet}_g(x, \vec{a}) = \{f^*(\vec{a}, c, c''') : c''' \in c''\}$. To see this, let $G = \{f^*(\vec{a}, c, c''') : c''' \in c''\}$ and suppose there is an $F = y' \odot G \neq G$ satisfying $\mathrm{IsCofVSet}_g(F, \vec{a})$ with $y' \in \mathrm{tc}(y)$. Take F and y' to be of minimal rank satisfying these conditions. We have $y' \neq \emptyset$; furthermore, any $y'' \in y'$ satisfies $\mathrm{IsCofVTop}_g(y'' \odot G)$. Thus $y'' \neq \emptyset$, and $y''' = \mathrm{MnR}(y'') \in y''$ satisfies $\mathrm{IsCofVSet}_g(y''' \odot G)$. By the minimality of y', we have $y''' \odot G = G$. It follows that $y'' = \{\emptyset, \langle c'', f(\vec{a}, c, c'')\rangle\}$. This contradicts $y'' \in \mathrm{tc}(y)$ and the choice of y.

Therefore, we can recover c'' from x by

$$\mathrm{cValue}_g(x, \vec{a}) \;=\; \{\mathrm{MxR}_1(F) : F \in \mathrm{TopCofVSet}_g(x, \vec{a})\}.$$

We are now ready to define τ^* and h^*. By Lemma 20, from τ we can construct a CRSF function τ' such that $\tau'(x, \vec{a}, c, c'') : f(\vec{a}, c, c'') \preccurlyeq h(\vec{a}, c)$, as long as $c'' \in \mathrm{tc}^+(c)$ since in this case $c'' \preccurlyeq c$ by the identity embedding. From this, it follows readily that there is a #-term $s(\vec{a}, c)$ and a CRSF function τ'' such that

$$\tau''(x, \vec{a}, c, c'') : \{\{\emptyset, \langle c'', f(\vec{a}, c, c'')\rangle\}\} \preccurlyeq s(\vec{a}, c) \tag{17}$$

whenever $c'' \in \text{tc}^+(c)$. Let $h^*(\vec{a}, c)$ equal $c\#s(\vec{a}, c)$. Finally define $\tau^*(x, \vec{a}, c)$ to equal

$$\{\sigma_{c,s(\vec{a},c)}(c'', u) : u \in \tau''(x\odot^{-1}\text{TopCofVSet}_g(x, \vec{a}), \vec{a}, c, c'')\}$$

where $c'' = \text{cValue}(x, \vec{a})$.

It is straightforward to verify that τ^* is a CRSF function and is a (multi-valued) embedding $f^*(\vec{a}, c) \preccurlyeq h^*(\vec{a}, c)$. The intuition is that c'' is such that x is in the "$\{\emptyset, \langle c'', f(\vec{a}, c, c'')\rangle\}$" part of the course-of-values set, and then τ^* is computed taking the values given by τ'' and mapping them to the c''-th copy of $s(\vec{a}, c)$ in $c\#s(\vec{a}, c)$. In particular, suppose that $x_2 \in x_1 \in \text{tc}(f^*(\vec{a}, c, c'))$ and let $c'' = \text{cValue}(x_1, \vec{a})$ and $c''' = \text{cValue}(x_2, \vec{a})$. If $c'' = c'''$ then $\text{TopCofVSet}_g(x_1, \vec{a}) = \text{TopCofVSet}_g(x_2, \vec{a})$ and for every $u \in \tau^*(x_1, \vec{a}, c)$ there is a $v \in \tau^*(x_2, \vec{a}, c) \cap \text{tc}(u)$ by the properties of τ''. The only other possibility is that $x_1 = \text{TopCofVSet}_g(x_1, \vec{a})$ and $x_2 = f^*(\vec{a}, c''')$ with $c''' \in c''$. In this case the embedding property follows from the properties of $\sigma_{c,s(\vec{a},c)}$.

This completes the proof that f^* is in CRSF. $\square$

3.6 Impredicative embeddings

The section proves that CRSF is closed under Cobham recursion even when "impredicative" embeddings are used to bound functions. Recall that **(Cobham Recursion$_{\preccurlyeq}$)** and **(Cobham Recursion$_{\preccurlyeq}^{\text{CofV}}$)** were defined with the condition that for all $\vec{a}, c$ we have $\tau(x, \vec{a}, c) : f(\vec{a}, c) \preccurlyeq h(\vec{a}, c)$. We form impredicative versions of these by allowing τ to have $f(\vec{a}, c)$ as an additional input and requiring instead that, for all $\vec{a}, c$,

$$\tau(x, \vec{a}, c, f(\vec{a}, c)) : f(\vec{a}, c) \preccurlyeq h(\vec{a}, c). \tag{18}$$

Like the earlier bounding condition, this impredicative bounding condition implies that $f(\vec{a}, c)$ has rank bounded by $\text{rank}(h(\vec{a}, c))$ and has $|\text{tc}(f(\vec{a}, c))| \leq |\text{tc}(h(\vec{a}, c))|$. The difference is that with $f(\vec{a}, c)$ as an additional parameter, it is potentially easier for τ to compute a $\preccurlyeq$-embedding. Nonetheless, the next theorem shows that this gives no additional power for defining CRSF functions.

Theorem 30. CRSF *is closed under the impredicative versions of* **(Cobham Recursion$_{\preccurlyeq}$)** *and* **(Cobham Recursion$_{\preccurlyeq}^{\text{CofV}}$)**.

As a corollary, CRSF is also closed under the impredicative version of **(Embedded Replacement)**, as the proof of part 2. of Theorem 13 still applies.

Proof. The proof uses the techniques of the proof of Theorem 29 from the previous section. We prove only the **(Cobham Recursion$_{\preccurlyeq}^{\mathrm{CofV}}$)** case. The case of **(Cobham Recursion$_{\preccurlyeq}$)** follows as a corollary, or alternatively can be proved directly by using the "RecValues" function introduced in Section 5.1 below in place of the "AllValues" function.

Similarly to the proof of Theorem 29, assume $f(\vec{a}, c, c')$ is defined from the CRSF functions g and τ and a #-term h by

$$f(\vec{a}, c, c') \;=\; \begin{cases} g(\vec{a}, c', f_{\restriction \mathrm{tc}(c')}(\vec{a}, c, -)) & \text{if } c' \in \mathrm{tc}^+(c) \\ \emptyset & \text{otherwise} \end{cases}$$

but now with only the impredicative embedding condition

$$\tau(x, \vec{a}, c, c', f(\vec{a}, c, c')) : f(\vec{a}, c, c') \preccurlyeq h(\vec{a}, c'). \tag{19}$$

We henceforth implicitly assume whenever necessary that $c' \in \mathrm{tc}^+(c)$. Let $f^*(\vec{a}, c, c')$ be defined by (15). To show that f^*, and thus f, is in CRSF it suffices to give a bounding term h^* and an embedding function $\tau^*(x, \vec{a}, c) : f^*(\vec{a}, c, c') \preccurlyeq h^*(\vec{a}, c)$ in CRSF. By (19) and similarly to (17) there is a CRSF function $\tau'(x, \vec{a}, c, c', u)$ and a #-term $s(\vec{a}, c)$ so that

$$\tau'(x, \vec{a}, c, c', f(\vec{a}, c, c')) : \{\emptyset, \langle c', f(\vec{a}, c, c')\rangle\} \preccurlyeq s(\vec{a}, c)$$

whenever $c' \in \mathrm{tc}^+(c)$. Again let $h^*(\vec{a}, c)$ equal the #-term $c \# s(\vec{a}, c)$.

Now define $\tau^*(x, \vec{a}, c)$ to equal

$$\{\sigma_{c, s(\vec{a}, c)}(c'', u) : u \in \tau'(x \odot^{-1} F, \vec{a}, c, c'', g(\vec{a}, c'', \mathrm{AllValues}(F)))\}$$

where $F = \mathrm{TopCofVSet}_g(x, \vec{a})$ and $c'' = \mathrm{cValue}_g(x, \vec{a})$. Clearly, τ^* is in CRSF. The input x is in the "$\{\emptyset, \langle c'', f(\vec{a}, c, c'')\rangle\}$" part of the course-of-values set, so $x \odot^{-1} F$ is a member of $\mathrm{tc}^+(\{\emptyset, \langle c'', f(\vec{a}, c, c'')\rangle\})$. The embedding τ^* takes the values given by τ' and maps them to the c''-th copy of $s(\vec{a}, c)$ in $c \# s(\vec{a}, c)$. For this, τ' needs to have $f(\vec{a}, c, c'')$ as an input: this is computed by applying g to the course-of-values set $\mathrm{AllValues}(F)$ obtained from the earlier values of f encoded in F. □

4 Polynomial time on binary strings

This section proves that polynomial time functions, and only polynomial time functions, can be defined in CRSF under a canonical encoding of (finite) binary strings as hereditarily finite sets.

There are many good ways to encode binary strings $s \in \{0,1\}^*$ as hereditarily finite sets. These include the "list" or "map" methods of [2] and the sequence-based encoding used by Arai [1]. We shall use instead the simpler encoding defined below. All these methods have the property that an encoding $\nu(s)$ of a binary string s has its rank and the cardinality of its transitive closure polynomially bounded (even linearly bounded) by the length $|s|$ of s, and in addition has rank $\geq |s|$. Furthermore, all these methods are "natural" and, although we omit the proofs, it is not hard to show that these methods are equivalent in that there are CRSF functions which translate between these encodings. Thus, for the purpose of defining CRSF functions on binary strings, it does not matter which of these encodings we use.

There are encodings such as the "tree" or "Ackermann" encodings of [2] which are not suitable for our purposes; for these encodings, the rank of $\nu(s)$ is too small and does not permit sufficiently long $\in$-recursion. See Sazonov [11] for more discussion of how to select encodings.

Definition 31. *Let $s = s_0 s_1 \cdots s_{|s|-1}$ be a binary string in $\Sigma = \{0,1\}^*$. The* encoding *$\nu(s)$ of s is the set defined by*

$$\nu(s) \;=\; \{|s|\} \cup \{i < |s| : s_i = 1\}.$$

For example, $\nu(11010) = \{0,1,3,5\}$. The empty string is denoted ϵ, and $\nu(\epsilon) = \{0\} = 1$. We use the notation $\nu(\vec{a})$ for $\nu(a_1), \ldots, \nu(a_n)$.

Definition 32. *A function $f : \Sigma^n \to \Sigma$ 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 $a_1, \ldots, a_n \in \Sigma$. When this holds, we write $f = F^\nu$.

The next two theorems state that the CRSF functions represent exactly the polynomial time functions.

Theorem 33. *If f is a polynomial time function, then $f = F^\nu$ for some F in* CRSF*.*

Theorem 34. *Every function $f = F^\nu$ for F in* CRSF *is in polynomial time.*

To define some simple CRSF functions that operate on encodings of strings, note that if $s \in \{0,1\}^*$ and $S = \nu(s)$, and $n \geq 0$, then

$$\begin{aligned}
|s| &= \bigcup S \\
\nu(s0) &= (S \setminus \{|s|\}) \cup \{\mathrm{Succ}(|s|)\} \\
\nu(s1) &= S \cup \{\mathrm{Succ}(|s|)\} \\
s{\restriction}n &= (S \cap n) \cup \{n\}
\end{aligned}$$

where $\mathrm{Succ}(x) = x \cup \{x\}$, and where $s{\restriction}n$ is the string consisting of the first n bits of s when $n \le |s|$.

The notation $|s|$ should not be confused with the use of $|\cdot|$ for set cardinality; it should always be clear from the context which is intended. For an integer $i > 0$, its predecessor $i - 1$ is denoted $\mathrm{Pred}(i)$ and it also equals $\bigcup i$. Thus Pred is a CRSF function.

For $S = \nu(s)$, the value s_i is computable by the CRSF function

$$\mathrm{Bit}(i, S) \;=\; \begin{cases} 1 & \text{if } i \in S \text{ and } i < \bigcup S \\ 0 & \text{otherwise.} \end{cases}$$

Proof of Theorem 33. As discussed in the introduction, Cobham's characterization of P states that the class of polynomial time functions is the smallest class containing the constant function ϵ and the two successor functions $s \mapsto s0$ and $s \mapsto s1$ and closed under composition and limited recursion on notation. The constant $\nu(\epsilon)$ is clearly represented by a CRSF function. As just shown above, $s \mapsto s0$ and $s \mapsto s1$ are represented by CRSF functions. Also, CRSF is closed under composition. So it suffices to establish closure under Cobham limited recursion. For this, suppose that the functions $g(\vec{a})$, $h_0(\vec{a}, b, s)$, and $h_1(\vec{a}, b, s)$ are represented by CRSF functions $G(\vec{A})$, $H_0(\vec{A}, B, S)$, and $H_1(\vec{A}, B, S)$, and that p is a polynomial, and let $f(\vec{a}, s)$ be defined by limited recursion, with

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

and satisfying $|f(a_1, \ldots, a_n, s)| \le p(|a_1|, \ldots, |a_n|, |s|)$. We need to show that a function F that represents f is also in CRSF.

It suffices to prove that there is a CRSF function $F'(N, \vec{A}, S)$ so that for all strings $\vec{a}, s$ and finite ordinals N,

$$F'(N, \nu(\vec{a}), \nu(s)) \;=\; \nu(f(\vec{a}, s{\restriction}N)),$$

since then $F(\vec{A}, S) = F'(|S|, \vec{A}, S)$ is a CRSF function which represents f.

Using Lemma 4, we can define an ordinal-valued CRSF function $P(N, \vec{A}, S)$ where for all such $N, \vec{a}, s$,

$$P(N, \nu(\vec{a}), \nu(s)) \ge p(|a_1|, \ldots, |a_n|, |s{\restriction}N|) + 1,$$

with the consequence that $\nu(f(\vec{a}, s{\restriction}N)) \subseteq P(N, \nu(\vec{a}), \nu(s))$. We then define $F'(N, \vec{A}, S)$ by **(Cobham Recursion$^{\mathrm{CofV}}_{\preccurlyeq}$)** so that

$$F'(N, \vec{A}, S) \;=\; \begin{cases} G(\vec{A}) \cap P(N, \vec{A}, S) & \text{if } N = 0 \\ H_0(\vec{A}, F'(\mathrm{Pred}(N), \vec{A}, S), S{\restriction}\mathrm{Pred}(N)) \cap P(N, \vec{A}, S) & \\ \qquad\qquad \text{if } N \neq 0 \text{ and } \mathrm{Bit}(\mathrm{Pred}(N), S) = 0 & \\ H_1(\vec{A}, F'(\mathrm{Pred}(N), \vec{A}, S), S{\restriction}\mathrm{Pred}(N)) \cap P(N, \vec{A}, S) & \\ \qquad\qquad \text{if } N \neq 0 \text{ and } \mathrm{Bit}(\mathrm{Pred}(N), S) = 1. & \end{cases}$$

The value of $F'(\mathrm{Pred}(N), \vec{A}, S)$ can be computed by a CRSF function from $F'_{\restriction \mathrm{tc}(N)}(-, \vec{A}, S)$. The intersection with $P(N, \vec{A}, S)$ has no effect when $N \in \omega$ and $\vec{A}, S$ are encodings of binary strings; however, it ensures that for all inputs there is a trivial embedding $F'(N, \vec{A}, S) \preccurlyeq P(N, \vec{A}, S)$. Hence F' is a CRSF function. □

Proof of Theorem 34. Since the Mostowski graph of a set a is a directed graph on the set of nodes $\mathrm{tc}(a)$, the Mostowski graph of a hereditarily finite set a can be described by a binary string of length $O(|\mathrm{tc}(a)|^2)$.

Theorem 34 follows from the observation that if $f(x_1, \ldots, x_n)$ is a CRSF function, then there is an n-ary polynomial time function g such that if g is given (binary strings describing the) Mostowski graphs of hereditarily finite sets $a_1, \ldots, a_n$, then g outputs (a binary string describing) the Mostowski graph for the hereditarily finite set $f(\vec{a})$. This fact is proved by induction on the definition of CRSF functions.

For instance, for the base function $\mathrm{cond}_{\in}$, the condition $c \in d$ can be tested by checking whether $c = x$ for each $x \in d$. This is polynomial time since equality of two sets given by Mostowski graphs is readily calculated by determining an isomorphism between all members of their transitive closures, traversing the graphs in rank-order.

The main case to consider is a CRSF function $f(\vec{a}, c)$ defined by **(Cobham Recursion$_{\preccurlyeq}$)** using recursion on g with respect to c. For this, the embedding condition ensures that all intermediate values $f(\vec{a}, c')$ for $c' \in \mathrm{tc}^+(c)$ are sets that have polynomial size Mostowski graphs. Therefore, by the induction hypothesis applied to g, all these values $f(\vec{a}, c')$ can be computed in polynomial time.

To finish the proof of Theorem 34, note that there is a polynomial time function mapping a binary string s to a description of the Mostowski graph of $\nu(s)$, and vice-versa. □

It is worth remarking that the converse to the *proof* of Theorem 34 does not hold; namely, there are polynomial time functions that operate on Mostowski graphs of sets, and which do not calculate a function in CRSF. For instance, there is a polynomial time function, which given a Mostowski graph for a set a, produces a Mostowski graph for the von Neumann integer $|\mathrm{tc}(a)|$. However, the function $a \mapsto |\mathrm{tc}(a)|$ is *not* in CRSF. To prove this, note that on the one hand, $|a|$ may be superexponentially larger than $\mathrm{rank}(a)$, but on the other hand, any CRSF function f has $\mathrm{rank}(f(a))$ polynomially bounded by $\mathrm{rank}(a)$ by Corollary 22.

5 An equivalence of CRSF and PCSF$^+$

In this section we prove an equivalence between the power of CRSF and an extension PCSF$^+$ of the class PCSF of predicatively computable set functions introduced by Arai [1]. For functions on binary strings (equivalently, on integers), the notion of safe/normal functions was introduced by Bellantoni and Cook [5], extending related constructions of Leivant [9]. The notion of safe/normal recursion for *set functions* was introduced by [2], who defined a class of Safe Recursive Set Functions (SRSF) and showed that, using hereditarily finite sets with suitable encodings, SRSF can define precisely the functions of binary strings which can be computed by alternating Turing machines that use exponential time and polynomially many alternations. Arai modified the definition of SRSF in [2], and defined a class of safe/normal set functions called the Predicatively Computable Set Functions (PCSF) which, on hereditarily finite sets, captures exactly the functions on binary strings which are in polynomial time.

We give here a quick definition of the classes PCSF and PCSF$^+$; the reader should refer to [1, 2] for more details.

In the safe/normal setting, functions take two types of parameters, "normal" and "safe". The notation $f(\vec{x}\,/\,\vec{y})$ indicates that the parameters $\vec{x}$ are normal, whereas the parameters $\vec{y}$ are safe. A function is called m,n-ary if it has m normal parameters and n safe parameters. The class PCSF of *Predicatively Computable Set Functions* is the smallest class of functions containing the following five initial functions and three closure operations.

(Projection$^{\text{SN}}$) For $m, n \geq 0$ and $1 \leq j \leq n+m$,

$$\pi_j^{n,m}(a_1,\ldots,a_n\,/\,a_{n+1},\ldots,a_{n+m}) \;=\; a_j.$$

(Null$^{\text{SN}}$)

$$\mathrm{null}(\,/\,) \;=\; \emptyset.$$

(Pair$^{\mathrm{SN}}$)

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

(Union$^{\mathrm{SN}}$)

$$\mathrm{union}(/a) \;=\; \bigcup a.$$

(Conditional$^{\mathrm{SN}}_{\in}$)

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

(Composition$^{\mathrm{SN}}$) If g is a m, n-ary function, $\vec{h}$ is a vector of m many $k, 0$-ary functions, and $\vec{r}$ is a vector of n many k, ℓ-ary functions, then safe composition gives the k, ℓ-ary function f:

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

(Safe Separation$^{\mathrm{SN}}$) If g is a $0, n$-ary function with $n \geq 1$, then safe separation gives the $0, n$-ary function f:

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

(Predicative Set Recursion$^{\mathrm{SN}}$) If g is m, n-ary with $m, n \geq 1$, then predicative set recursion gives the $m, n{-}1$-ary function f:

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

Arai [1] proves a variety of closure properties for PCSF, including under the following recursion that takes values of f on c as a set of ordered pairs:

(Predicative Function Recursion$^{\mathrm{SN}}$) If g is m, n-ary with $m, n \geq 1$, then predicative function recursion gives the $m, n{-}1$-ary function f:

$$f(\vec{a}, c/\vec{d}) \;=\; g(\vec{a}, c/\vec{d}, f_{\restriction c}(\vec{a}, -/\vec{d})).$$

Arai [1] also mentions a form of separation which allows normal parameters:

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

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

We define the class PCSF^{+} similarly to PCSF, but using **(Normal Separation$^{\mathrm{SN}}$)** in place of **(Safe Separation$^{\mathrm{SN}}$)**. Arai conjectures that PCSF^{+} strictly contains PCSF, but this remains an open question.

PCSF^+ enjoys all the closure properties that Arai [1] established for PCSF. In addition, it follows easily from **(Normal Separation$^{\mathrm{SN}}$)** that the PCSF^+ relations are closed under set bounded quantification. That is, if $R(\vec{a}/\vec{b}, x)$ is a PCSF^+ relation, then so is $S(\vec{a}/\vec{b}, c) \Leftrightarrow (\forall x \in c)R(\vec{a}/\vec{b}, x)$.

5.1 CRSF includes PCSF^+

We show that every PCSF^+ function can be expressed as a CRSF function.

Theorem 35. *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 a #-term $t(\vec{a})$ such that, for all $\vec{a}, \vec{b}$,*

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

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

c. *τ is the identity on $\mathrm{tc}(\{\vec{b}\})$. Namely, if $x \in \mathrm{tc}(\{\vec{b}\})$, then $\tau(x, \vec{a}, \vec{b}) = \{x\}$. And, if $\tau(x, \vec{a}, \vec{b}) \cap \mathrm{tc}(\{\vec{b}\}) \neq \emptyset$, then $x \in \mathrm{tc}(\{\vec{b}\})$.*

The notation $\{\vec{b}\}$ denotes $\{b_1, \ldots, b_m\}$, namely the set of safe parameters. Part b. of Theorem 35 puts sharp bounds on how the safe parameters $\vec{b}$ can affect the value of $f(\vec{a}/\vec{b})$. A similar bound is given by Theorem 5.1 of [1] in terms of the cardinality of the transitive closure of $f(\vec{a}/\vec{b})$ when $\vec{a}$ and $\vec{b}$ are hereditarily finite. Theorem 35(b) sharpens this, and is applicable to all sets, not just hereditarily finite sets.

Proof. The proof is by induction on the formation of the PCSF^+ function $f(\vec{a}/\vec{b})$. For f one of the initial functions null, pair, union, $\mathrm{cond}_\in$ or the projection function $\pi_j^{n,m}$ with $j > n$, the theorem is obviously true with $t(\vec{a}) = 1$. (Even $t(\vec{a}) = \emptyset$ would work, but $\emptyset$ is not a permitted #-term.) For the projection function $\pi_i^{n,m}(\vec{a}/\vec{b})$ with $i \leq n$, set $t(\vec{a}) = a_i$ (the i-th normal input to f), and set the embedding function equal to

$$\tau(x, \vec{a}, \vec{b}) \;=\; \begin{cases} \{x\} & \text{if } x \in \mathrm{tc}(\{\vec{b}\}) \\ \{x \odot \{\vec{b}\}\} & \text{otherwise.} \end{cases}$$

For f defined by **(Normal Separation$^{\mathrm{SN}}$)**,

$$f(\vec{d}/\vec{a}, c) \;=\; \{b \in c : f_1(\vec{d}/\vec{a}, b) \neq \emptyset\},$$

the induction hypothesis for f_1 gives a CRSF function $g_1(\vec{d}, \vec{a}, b)$ equal to $f_1(\vec{d}/\vec{a}, b)$. By (**Separation**) using g_1, the function $g(\vec{d}, \vec{a}, c) = f(\vec{d}/\vec{a}, c)$ is in CRSF. Since $g(\vec{d}, \vec{a}, c) \subseteq c$, setting $t = 1$ (again, even $t = \emptyset$ would work) and τ the identity, $\tau : x \mapsto \{x\}$, proves the theorem for f.

Next suppose f is defined by **(Composition$^{\mathbf{SN}}$)** as

$$f(\vec{a}/\vec{b}) \ = \ f_1(\vec{f_2}(\vec{a}/)/\vec{f_3}(\vec{a}/\vec{b})).$$

The normal parameters $\vec{f_2}$ (resp., safe parameters $\vec{f_3}$) are a list of functions $f_{2,j}$ for $1 \leq j \leq \ell_2$ (resp., functions $f_{3,j}$ for $1 \leq j \leq \ell_3$). The induction hypothesis for the PCSF$^+$ function $f_1(\vec{c}/\vec{d})$ gives CRSF functions $g_1(\vec{c}, \vec{d})$ and $\tau_1(x, \vec{c}, \vec{d})$, and a #-term $t_1(\vec{c})$. The induction hypotheses for the $f_{2,j}$'s and $f_{3,j}$'s give CRSF functions $g_{2,j}(\vec{a})$, $\tau_{2,j}(x, \vec{a})$, $g_{3,j}(\vec{a}, \vec{b})$, and $\tau_{3,j}(x, \vec{a}, \vec{b})$, and #-terms $t_{2,j}(\vec{a})$ and $t_{3,j}(\vec{a})$. We must define $g(\vec{a}, \vec{b})$, $\tau(x, \vec{a}, \vec{b})$, and $t(\vec{a})$ for f. The function $g(\vec{a}, \vec{b}) = f(\vec{a}/\vec{b})$ is immediately seen to be CRSF by **(Composition)**:

$$g(\vec{a}, \vec{b}) \ = \ g_1(\vec{g}_2(\vec{a}), \vec{g}_3(\vec{a}, \vec{b})).$$

By the induction hypothesis, $x \mapsto \tau_1(x, \vec{g}_2(\vec{a}), \vec{g}_3(\vec{a}, \vec{b}))$ is a $\preccurlyeq$-embedding of

$$g_1(\vec{g}_2(\vec{a}), \vec{g}_3(\vec{a}, \vec{b})) \preccurlyeq t_1(\vec{g}_2(\vec{a})) \odot \{\vec{g}_3(\vec{a}, \vec{b})\}, \tag{20}$$

and for $j = 1, \ldots, \ell_2$,

$$\tau_{2,j} : g_{2,j}(\vec{a}) \preccurlyeq t_{2,j}(\vec{a}) \odot \emptyset = t_{2,j}(\vec{a}).$$

By composition and Lemma 20, $\tau_{2,1}, \ldots, \tau_{2,\ell_2}$ give a CRSF function $\tau'_1(x, \vec{a})$ such that

$$\tau'_1 : t_1(\vec{g}_2(\vec{a})) \preccurlyeq t'_1(\vec{a}) \tag{21}$$

where $t'_1(\vec{a})$ is the #-term

$$t_1(t_{2,1}(\vec{a}), \ldots, t_{2,\ell_2}(\vec{a})).$$

The induction hypothesis also gives, for $1 \leq j \leq \ell_3$,

$$\tau_{3,j} : g_{3,j}(\vec{a}, \vec{b}) \preccurlyeq t_{3,j}(\vec{a}) \odot \{\vec{b}\}.$$

Letting $t'_{3,j}(\vec{a}) = 1 \odot t_{3,j}(\vec{a})$, we readily get a CRSF function $\tau'_{3,j}(x, \vec{a}, b)$ so that

$$\tau'_{3,j} : \{g_{3,j}(\vec{a}, \vec{b})\} \preccurlyeq t'_{3,j}(\vec{a}) \odot \{\vec{b}\}.$$

Further letting $t'_3(\vec{a})$ be $t'_{3,1}(\vec{a}) \odot \cdots \odot t'_{3,\ell_3}(\vec{a})$, we can define a CRSF function $\tau'_3(x, \vec{a}, \vec{b})$ so that

$$\tau'_3 : \{\vec{g}_3(\vec{a}, \vec{b})\} \preccurlyeq t'_3(\vec{a}) \odot \{\vec{b}\}; \tag{22}$$

namely, letting $\tau'_3(x, \vec{a}, \vec{b}) = \{x\}$ for $x \in \mathrm{tc}(\{\vec{b}\})$, and for all other x, letting $\tau'_3(x, \vec{a}, \vec{b})$ equal

$$\{u \odot t'_{3,k+1}(\vec{a}) \odot \cdots \odot t'_{3,\ell_3}(\vec{a}) \odot \{\vec{b}\} : u \odot \{\vec{b}\} \in \tau'_{3,k}(x, \vec{a}, \vec{b}),\ 1 \le k \le \ell_3\}.$$

With (20), (21) and (22), it is straightforward to combine τ_1, τ'_1 and τ'_3 to form a CRSF function $\tau(x, \vec{a}, \vec{b})$ so that

$$\tau : g(\vec{a}, \vec{b}) \preccurlyeq t'_1(\vec{a}) \odot t'_3(\vec{a}) \odot \{\vec{b}\}.$$

All of $\tau_{3,j}$, $\tau'_{3,j}$, τ_3 and τ are the identity on $\{\vec{b}\}$. Letting $t(\vec{a})$ be the #-term $t'_1(\vec{a}) \odot t'_3(\vec{a})$, this completes the proof of Lemma 35 for PCSF$^+$ functions defined using composition.

Finally, suppose $f(\vec{a}/\vec{b})$ is defined by **(Predicative Set Recursion$^{\mathbf{SN}}$)**,

$$f(\vec{a}, c/\vec{b}) \;=\; f_1(\vec{a}, c/\vec{b}, \{f(\vec{a}, c'/\vec{b}) : c' \in c\}).$$

The induction hypothesis for $f_1(\vec{a}, c/\vec{b}, F)$ gives CRSF functions $g_1(\vec{a}, c, \vec{b}, F)$ and $\tau_1(x, \vec{a}, c, \vec{b}, F)$ and a #-term $t_1(\vec{a}, c)$. We must find suitable $g(\vec{a}, c, \vec{b})$, $\tau(x, \vec{a}, c, \vec{b})$, and $t(\vec{a}, c)$ for f.

It is straightforward to write a recursive definition of g, but unlike in previous cases where we showed that a function is in CRSF, this time there is no readily available bound on the complexity of f which we could use to construct an embedding that bounds g. Hence the main work in the proof is to construct such an embedding. For this, it is crucial to use the assumption that the embeddings given by the induction hypothesis are the identity on safe arguments; in particular, the fact that τ_1 is the identity on the argument F of g_1 which holds the previous recursive values.

The construction of g and τ is based on the proof of Theorem 29. In order to use c as a side parameter, define

$$g'_1(\vec{a}, c, c', \vec{b}, F) \;=\; \begin{cases} g_1(\vec{a}, c', \vec{b}, F) & \text{if } c' \in \mathrm{tc}^+(c) \\ \emptyset & \text{otherwise.} \end{cases}$$

Define a function g^* by

$$g^*(\vec{a}, c, c', \vec{b}) \;=\; \{\emptyset, \langle c', g'_1(\vec{a}, c, c', \vec{b}, \mathrm{RecValues}(G))\rangle\} \odot G, \tag{23}$$

where $G = G(\vec{a}, c, c', \vec{b}) = \{g^*(\vec{a}, c, c'', \vec{b}) : c'' \in c'\}$ and

$$\mathrm{RecValues}(G) \;=\; \{\mathrm{MxR}_2(G') : G' \in G\}.$$

Thus, g^* is the course-of-values set obtained by iterating g'_1. We define $g(\vec{a}, c, \vec{b}) = \mathrm{MxR}_2(g^*(\vec{a}, c, c, \vec{b}))$. Hence $g(\vec{a}, c, \vec{b}) = f(\vec{a}, c/\vec{b})$, and to show that g is in CRSF it suffices to show that g^* is.

Analogously to the earlier definition of IsCofV_g, define IsCofV'_{g_1} by

$$\mathrm{IsCofV}'_{g_1}(F, \vec{a}, c, \vec{b}) = \begin{cases} \emptyset & \text{if } F = \{\emptyset, \langle \mathrm{MxR}_1(F), \mathrm{MxR}_2(F)\rangle\} \odot \mathrm{MnR}'(F), \\ & \quad \mathrm{IsCofV}'_{g_1}(\mathrm{MnR}'(F), \vec{a}, c, \vec{b}) = 1, \\ & \quad \mathrm{MxR}_1(F) = \{\mathrm{MxR}_1(F') : F' \in \mathrm{MnR}'(F)\}, \text{ and} \\ & \quad \mathrm{MxR}_2(F) = g'_1(\vec{a}, c, \mathrm{MxR}_1(F), \vec{b}, \mathrm{RecValues}(\mathrm{MnR}'(F))) \\ 1 & \text{if } \{\mathrm{IsCofV}'_{g_1}(F', \vec{a}, c, \vec{b}) : F' \in F\} \subseteq \{\emptyset\} \\ 2 & \text{otherwise.} \end{cases}$$

This is similar to the definition of IsCofV_g except that "RecValues" replaces "AllValues" since we are now using **(Cobham Recursion$_{\preccurlyeq}$)** instead of course-of-values recursion. Further, define $\mathrm{TopCofVSet}'_{g_1}(x, \vec{a}, c, \vec{b})$ and $\mathrm{cValue}'_{g_1}(x, \vec{a}, c, \vec{b})$ similarly to $\mathrm{TopCofVSet}_g$ and cValue_g but using IsCofV'_{g_1} instead of IsCofV_g.

We want to define an embedding function $\tau^* \in$ CRSF and a CRSF function h^* so that

$$\tau^*(x, \vec{a}, c, \vec{b}) : g^*(\vec{a}, c, c', \vec{b}) \preccurlyeq h^*(\vec{a}, c), \tag{24}$$

showing that g^* is in CRSF. (We will use τ^* and h^* to construct suitable functions τ and t bounding g.) Since (24) has to hold for all $c' \in \mathrm{tc}^+(c)$, it is equivalent to

$$\tau^*(x, \vec{a}, c, \vec{b}) : g^*(\vec{a}, c, c, \vec{b}) \preccurlyeq h^*(\vec{a}, c)$$

and this is what we will show.

By the induction hypothesis, $\tau_1 : g_1(\vec{a}, c, \vec{b}, F) \preccurlyeq t_1(\vec{a}, c) \odot \{\vec{b}, F\}$. From this, it is easy to see there is a CRSF function $\tau'_1(x, \vec{a}, c, c', \vec{b}, F)$ and a #-term $s(\vec{a}, c)$ so that

$$\tau'_1 : \{\{\emptyset, \langle c', g'_1(\vec{a}, c, c', \vec{b}, F)\rangle\}\} \preccurlyeq s(\vec{a}, c) \odot \{\vec{b}, F\} \tag{25}$$

whenever $c' \in \mathrm{tc}^+(c)$. Furthermore, τ_1 and τ'_1 are the identity on $\mathrm{tc}(\{\vec{b}, F\})$. We shall construct a CRSF function $\tau_2(x, \vec{a}, c, c', \vec{b}, G)$ such that

$$\tau_2 : \{\{\emptyset, \langle c', g(\vec{a}, c', \vec{b})\rangle\}\} \preccurlyeq (c\#(s(\vec{a}, c) \odot 1)) \odot \{\vec{b}\} \tag{26}$$

whenever $\mathrm{IsCofVSet}_{g'_1}(G, \vec{a}, c, \vec{b})$ and $c' \in \mathrm{tc}^+(\mathrm{cValue}_{g'_1}(G, \vec{a}, c, \vec{b}))$, and such that τ_2 is the identity on $\mathrm{tc}(\{\vec{b}\})$. We henceforth write S for $s(\vec{a}, c) \odot 1$.

It is easy to define τ^* once we have τ_2. Let $h^*(\vec{a}, c, \vec{b})$ be $c\#((c\#S)\odot\{\vec{b}\})$. To define $\tau^*(x, \vec{a}, c, \vec{b})$, suppose $x \in \mathrm{tc}(g^*(\vec{a}, c, c, \vec{b}))$. Let $c' = \mathrm{cValue}'(x, \vec{a}, c, \vec{b})$, let $G = \mathrm{TopCofVSet}'_{g_1}(x, \vec{a}, c, \vec{b})$, and let $y = x\odot^{-1}G$. Then $y \in \mathrm{tc}^+(\{\emptyset, \langle c', g(\vec{a}, c', \vec{b})\rangle\})$ and we set $\tau^*(x, \vec{a}, c, \vec{b})$ equal to

$$\{\sigma_{c,(c\#S)\odot\{\vec{b}\}}(c', w) : w \in \tau_2(y, \vec{a}, c, c', \vec{b}, G)\}.$$

This shows that g^* and g are in CRSF.

Given τ_2, we can now define the embedding function τ and the #-term h as needed for the theorem. Define $\tau(x, \vec{a}, c, \vec{b})$ to equal $\tau_2(x, \vec{a}, c, c, \vec{b}, G)$ where G is the course-of-values set $g^*(\vec{a}, c, c, \vec{b})$. From (26),

$$\tau : \{\{\emptyset, \langle c, g(\vec{a}, c, \vec{b})\rangle\}\} \preccurlyeq (c\#S)\odot\{\vec{b}\}$$

and is the identity on $\{\vec{b}\}$. Set t equal to the #-term $c\#S$. It follows that τ is also an embedding of $g(\vec{a}, c, \vec{b})$ into $t(\vec{a}, c, \vec{b})\odot\{\vec{b}\}$ and satisfies conditions b. and c. of the theorem.

It remains to define $\tau_2(x, \vec{a}, c, c', \vec{b}, G)$. We use **(Cobham Recursion$_{\preccurlyeq}^{\mathrm{CofV}}$)** on c'. We first obtain the set $F = \{g(\vec{a}, c'', \vec{b}) : c'' \in c'\}$ as a CRSF function of G by $F = \mathrm{rng}(\mathrm{AllValues}(G)\restriction c')$, where rng is the range function. Note that $g'_1(\vec{a}, c, c', \vec{b}, F) = g(\vec{a}, c', \vec{b})$, so the domains of τ'_1 and τ_2, as shown in (25) and (26), are the same. Then there are four cases.

i. If $x \notin \mathrm{tc}(\{\vec{b}, F\})$, then τ_2 maps x to

$$\{\sigma_{c,S}(c', w\odot 1)\odot\{\vec{b}\} : w\odot\{\vec{b}, F\} \in \tau'_1(x, \vec{a}, c, c', \vec{b}, F)\}.$$

Because τ'_1 is the identity on $\mathrm{tc}(\{\vec{b}, F\})$, τ'_1 maps x to a subset of the "$s(\vec{a}, c)$ part" of the righthand side of (25). Thus the value of τ_2 gives the corresponding subset of the c'-th copy of $s(\vec{a}, c)$ in (26).

ii. If $x \in \mathrm{tc}(\{\vec{b}\})$ then τ_2 is the identity, mapping x to $\{x\}$.

iii. If $x = F$ and $F \notin \mathrm{tc}(\{\vec{b}\})$, then τ_2 maps x to $\{\sigma_{c,S}(c', \emptyset)\odot\{\vec{b}\}\}$.

iv. If none of i.-iii. hold, then $x \in \mathrm{tc}(F)$. By choice of F, we have $x \in \mathrm{tc}^+(g(\vec{a}, c'', \vec{b}))$ for one or more values of $c'' \in c'$. Then τ_2 uses course-of-values recursion, mapping x to

$$\{w : c'' \in \mathrm{tc}(c'),\ w \in \tau_2(x, \vec{a}, c, c'', \vec{b}, G),\ x \in \mathrm{tc}^+(g(\vec{a}, c'', \vec{b})),\ x \notin \mathrm{tc}(F_{c''})\} \quad (27)$$

where $F_{c''} = \{g(\vec{a}, c''', \vec{b}) : c''' \in c''\}$. Both $F_{c''}$ and $g(\vec{a}, c'', \vec{b})$ can be computed from $\mathrm{AllValues}(G)$. The values c'' satisfying the conditions above are exactly the minimal values $c'' \in \mathrm{tc}^+(c')$ for which $x \in \mathrm{tc}^+(g(\vec{a}, c'', \vec{b}))$, so there is at least one such c''. The condition $x \in \mathrm{tc}^+(g(\vec{a}, c'', \vec{b}))$ implies that x is in the domain of $\tau_2(x, \vec{a}, c, c'', \vec{b}, G)$, and the condition $x \notin \mathrm{tc}(F_{c''})$ implies that it falls under case i. or iii. there. Hence the part of (27) corresponding to c'' is a subset of the c''-th copy of $s(\vec{a}, c)\odot 1$ in (26).

To prove that $\tau_2(x, \vec{a}, c, c', \vec{b}, G)$ is an $\preccurlyeq$-embedding, we inductively assume that $\tau_2(x, \vec{a}, c, c'', \vec{b}, G)$ is an embedding for $c'' \in \mathrm{tc}(c')$. Then for c', restricted to each case τ_2 is a total injective multifunction, and the cases have disjoint ranges. The embedding property is clear by inspection.

This completes the proof of Theorem 35. □

5.2 PCSF$^+$ includes CRSF

We show that every CRSF function can be expressed as a PCSF$^+$ function.

Theorem 36. *If $f(\vec{a})$ is in* CRSF, *then $g(\vec{a}/) = f(\vec{a})$ is in* PCSF$^+$.

Corollary 37. *Suppose $g(\vec{a}/) = f(\vec{a})$. Then $f(\vec{a}) \in$* CRSF *if and only if $g(\vec{a}/) \in$* PCSF$^+$.

Before proving Theorem 36, we need to bootstrap some PCSF functions. The safe transitive closure function $f(/a) = \mathrm{tc}(a)$ is not in PCSF$^+$, since $f(/a)$ has no normal parameters and thus **(Predicative Set Recursion$^{\mathbf{SN}}$)** cannot be used. However, we can define $\mathrm{tc}(a)$ for a safe parameter a, provided we are given a normal input c of sufficiently large rank. Define, as a PCSF function,

$$\mathrm{tc}'(c/a) \ = \ a \cup \bigcup\bigcup\{\mathrm{tc}'(c'/a) : c' \in c\}.$$

It is easy to verify that $\mathrm{tc}'(c/a) = \mathrm{tc}(a)$ whenever either $\mathrm{rank}(c) \geq \mathrm{rank}(a)$ or $\mathrm{rank}(c) \geq \omega$. When proving Theorem 36, we will always have a $\#$-term t involving only normal inputs $\vec{A}$ so that $c = t$ has sufficiently large rank. To reduce clutter, we often abuse notation by writing just $\mathrm{tc}(a)$ instead of $\mathrm{tc}'(t/a)$.

Second, the function $f_\odot(a/b) = a\odot b$ is in PCSF since

$$f_\odot(a/b) \ = \ \begin{cases} b & \text{if } a = \emptyset \\ \{f_\odot(a'/b) : a' \in a\} & \text{otherwise.} \end{cases}$$

Likewise, $f_{\#}(a, b/) = a\#b$ is in PCSF by (4):

$$f_{\#}(a, b/) \;=\; f_{\odot}(b/\{f_{\#}(a', b/) : a' \in a\}).$$

These constructions are not good enough for our purposes however, as we will need to compute $a{\odot}b$ and $a\#b$ even when a and b are safe. The next two lemmas give replacement constructions:

Lemma 38. *There is a* PCSF$^+$ *function* $f'_{\odot}(A/a, b)$ *such that, whenever* $a \in \mathrm{tc}^+(A)$*, we have* $f'_{\odot}(A/a, b) = a{\odot}b$*.*

Lemmas 38 and 39 also hold for PCSF instead of PCSF$^+$, but we give only the proof for PCSF$^+$ as it better motivates our constructions.

Proof. The idea is to define $f'_{\odot}$ by recursion on A instead of a:

$$f'_{\odot}(A/a, b) \;=\; \begin{cases} b & \text{if } A = \emptyset \\ \{f'_{\odot}(A'/a, b) : A' \in A\} & \text{if } a \notin \mathrm{tc}(A) \text{ and } A \neq \emptyset \\ f'_{\odot}(a/a, b) & \text{if } a \in \mathrm{tc}(A) \end{cases}$$

To see that this works, observe that, arguing by induction on A, $f'_{\odot}(A/a, b) = A{\odot}b$ when $a \notin \mathrm{tc}(A)$, and is equal to $a{\odot}b$ when $a \in \mathrm{tc}(A)$. This however does not give $f'_{\odot}$ as a PCSF$^+$ function since the third case uses a as a normal parameter. Instead, we let F abbreviate $f'_{\odot \restriction A} = \{\langle A', f'_{\odot}(A'/a, b)\rangle : A' \in A\}$ and use **(Predicative Function Recursion$^{\mathbf{SN}}$)**:

$$f'_{\odot}(A/a, b) \;=\; \begin{cases} b & \text{if } A = \emptyset \\ \mathrm{rng}(/F) & \text{if } a \notin \mathrm{tc}(a) \text{ and } A \neq 0 \\ \bigcup\{z \in \mathrm{rng}(/F) : \exists A'{\in}A, \\ \qquad a \in \mathrm{tc}^+(A') \wedge \langle A', z\rangle{\in}F\} & \text{if } a \in \mathrm{tc}(A) \end{cases}$$

$\mathrm{rng}(/F)$ is equal to $\{z \in \bigcup\bigcup F : \exists y{\in}\bigcup\bigcup F, \langle y, z\rangle \in F\}$. The third case uses **(Normal Separation$^{\mathbf{SN}}$)**, so this shows $f'_{\odot}$ is in PCSF$^+$. □

Lemma 39. *There is a* PCSF$^+$ *function* $f'_{\#}(A, B/a, b)$ *such that, whenever* $a \in \mathrm{tc}^+(A)$ *and* $b \in \mathrm{tc}^+(B)$*, we have* $f'_{\#}(A, B/a, b) = a\#b$*.*

Proof. The idea is to define $f'_{\#}$ by

$$f'_{\#}(A, B/a, b) \;=\; \begin{cases} f'_{\odot}(B/b, \{f'_{\#}(A', B/a, b) : A' \in A\}) & \text{if } a \notin \mathrm{tc}(A) \\ f'_{\#}(a, B/a, b) & \text{if } a \in \mathrm{tc}(A) \end{cases}$$

The details of the proof are similar to the definition $f'_{\odot}$ given above. Note that the normal parameter B is needed in order to invoke $f'_{\odot}$. □

The proofs of Lemmas 38 and 39 showed how recursion on a safe parameter a can be simulated using recursion on a normal parameter A, as long as $a \in \mathrm{tc}^+(A)$. This suggests that CRSF functions can be simulated by PCSF$^+$ functions in the sense that any CRSF function $f(a)$ should be computable as a PCSF$^+$ function $F(A/a)$ with $F(A/a) = f(a)$ provided that $a \in \mathrm{tc}^+(A)$. However, we need an even more general construction for the proof of Theorem 36; namely, instead of assuming $a \in \mathrm{tc}^+(A)$ we assume only that a is $\preccurlyeq$-embeddable in A. The assumption $a \preccurlyeq A$ means that a is no more complex than A; and this allows recursion on a to be simulated by recursion on A. The assumption "$a \preccurlyeq A$" needs to be expressed in a rather strong way, with the embedding variable x a safe input to the embedding function.

Definition 40. *Let u and v be sets. A safe embedding $u \preccurlyeq v$ is given by a function $\tau(/x)$ such that the mapping $x \mapsto \tau(/x)$ is an embedding $u \preccurlyeq v$.*

As usual, $\vec{a}$ denotes $a_1, \ldots, a_k$; and similarly for $\vec{A}$. We write $\vec{\sigma} : \vec{a} \preccurlyeq \vec{A}$ to mean that $\vec{\sigma}$ is a vector of functions such that each σ_i is a safe embedding $\sigma_i : a_i \preccurlyeq A_i$. It is implicit in the notation that the a_i's and A_i's may be given by functions of other variables, and the σ_i's may have these variables as additional inputs. The next definition uses α_i's as metavariables for safe embeddings.

Definition 41. *Let $\alpha_1, \ldots, \alpha_k$ be a vector of function symbols, with α_i a $0,1$-ary symbol, that is, with no normal inputs and one safe input, so $\alpha_i = \alpha_i(/x)$. A* PCSF$^+(\alpha_1, \ldots, \alpha_k)$ term *T is a term built from the functions α_i, the initial functions of* PCSF$^+$*, and the operations of* **(Composition$^{\mathbf{SN}}$)***,* **(Normal Separation$^{\mathbf{SN}}$)***, and* **(Predicative Set Recursion$^{\mathbf{SN}}$)***. In other words,* PCSF$^+(\vec{\alpha})$ *terms are specifications of* PCSF$^+$ *functions, but allowing the α_i's as $0,1$-ary metavariables for additional initial functions. If $\sigma_1, \ldots, \sigma_k$ are* PCSF$^+$ *functions, then $T[\vec{\sigma}]$ denotes the* PCSF$^+$ *function which is the result of substituting the σ_i's for the α_i's.*

In the sequel, variables α (generally with subscripts) will always be $0,1$-ary function symbols and serve as metavariables for safe embeddings. We allow a very general notion of substitution when substituting the σ_i's for the α_i's. Namely, each σ_i has arity $m_i, n_i{+}1$: one of the safe inputs of σ_i is the distinguished embedding variable x. The remaining $m_i + n_i$ inputs are side parameters. Note that σ_i has as safe input x plus n_i other safe inputs; thus α_i's can be used only in contexts where safe parameters are permitted.

The next lemma shows how safe embeddings are sufficient for defining PCSF$^+$ analogues of $\odot$, $\#$, and their inverses. Its proof method will be helpful for the main induction case of Theorem 44 below. In addition, Corollary 43 will be important for the proof of Theorem 44 and thus Theorem 36.

Lemma 42. *There are* $\mathrm{PCSF}^+(\alpha_a,\alpha_b)$ *terms* $G_\odot(A,B\,/\,a,b)$, $G_\#(A,B\,/\,a,b)$, $G_{\odot^{-1}}(A,B\,/\,a,b)$, $G_\sigma(A,B\,/\,a,b,a',b')$, $G_{\pi_1}(A,B\,/\,a,b,u)$ *and* $G_{\pi_2}(A,B\,/\,a,b,u)$, *such that whenever* $\sigma_a : a \preccurlyeq A$ *and* $\sigma_b : b \preccurlyeq B$ *are safe embeddings, then*

1. $G_\odot[\sigma_a,\sigma_b](A,B\,/\,a,b) = a \odot b$.
2. $G_\#[\sigma_a,\sigma_b](A,B\,/\,a,b) = a\#b$.
3. $G_{\odot^{-1}}[\sigma_a,\sigma_b](A,B\,/\,a,b) = a\odot^{-1}b$.
4. $G_\sigma(A,B\,/\,a,b,a',b') = \sigma_{a,b}(a',b')$ *for* $a' \in \mathrm{tc}^+(a)$ *and* $b' \in \mathrm{tc}^+(b)$.
5. $G_{\pi_1}[\sigma_a,\sigma_b](A,B\,/\,a,b,u) = \pi_{1,a,b}(u)$.
6. $G_{\pi_2}[\sigma_a,\sigma_b](A,B\,/\,a,b,u) = \pi_{2,a,b}(u)$.

Proof. The idea for the proof of part 1. is to somewhat mimic the construction of Lemma 38, but exploit the embedding $\alpha_a : a \preccurlyeq A$ instead of using $a \in \mathrm{tc}^+(A)$. We again recurse on $A' \in \mathrm{tc}^+(A)$ instead of on $a' \in \mathrm{tc}^+(a)$. For the purposes of recursing on $A' \in \mathrm{tc}^+(A)$, a set A' in $\mathrm{tc}(A)$ will correspond to the unique $a' \in \mathrm{tc}(a)$ such that $A' \in \alpha_a(/a')$ (if there is any such a'). And, for $A' = A \in \mathrm{tc}^+(A)$, A corresponds to the whole set a. We write just "$A' \overset{\alpha_a}{\approx} a'$" to succinctly denote the condition that $A' \in \mathrm{tc}^+(A)$ corresponds to $a' \in \mathrm{tc}^+(a)$. Formally, "$A' \overset{\alpha_a}{\approx} a'$" means

$$A' \in \alpha_a(/a') \vee (A' = A \wedge a' = a).$$

This relation is, of course, given by a $\mathrm{PCSF}(\alpha_a)$ term with normal inputs A and A' and safe inputs a and a'.

In the proof below, the function tc is applied to safe parameters, e.g., for $\mathrm{tc}^+(a) = \mathrm{tc}(\{a\})$. These can always be replaced with uses of tc'. Indeed, every set constructed in the proof will have rank less than the rank of $4\odot A \odot B$, and this set can serve as the normal parameter for computing transitive closures with tc'. (The parameter B is usually suppressed in the notation.) Similar considerations apply also to future proofs.

We will define $G'_\odot(A,A'/a,b)$ as a $\mathrm{PCSF}^+(\alpha_a)$ term that computes the "course-of-values" set (sets of this type are denoted with the variable e):

$$\{\langle A'', a''\odot b\rangle : A''\in\mathrm{tc}^+(A') \wedge a''\in\mathrm{tc}^+(a) \wedge A'' \overset{\alpha_a}{\approx} a''\}. \tag{28}$$

To define $G'_{\odot}$ by **(Predicative Set Recursion$^{\mathbf{SN}}$)**, we need to extract from e the set of values $a''\odot b$ such that $a'' \in a'$. This is done with:

$$G''_{\odot}(A, A'/a, e) \;=\; \{u \in \mathrm{tc}(e) : \exists A''{\in}\mathrm{tc}(A')\, \exists a'{\in}\mathrm{tc}^{+}(a)\, \exists a''{\in}a' \text{ s.t. } \langle A'', u\rangle{\in}e \wedge A' \overset{\alpha_a}{\approx} a' \wedge A'' \overset{\alpha_a}{\approx} a''\}.$$

This definition uses **(Normal Separation$^{\mathbf{SN}}$)** and the fact that Δ_0 predicates are in PCSF; therefore, $G''_{\odot}$ is a $\mathrm{PCSF}^{+}(\alpha_a)$ term. Now, the course-of-values function can be defined using **(Predicative Set Recursion$^{\mathbf{SN}}$)** by

$$G'_{\odot}(A, A'/a, b) \;=\; \begin{cases} e \cup \{\langle A', G''_{\odot}(A, A'/a, e)\rangle\} & \text{if } (\exists a' \in \mathrm{tc}^{+}(a)\backslash\{\emptyset\}) A' \overset{\alpha_a}{\approx} a' \\ e \cup \{\langle A', b\rangle\} & \text{otherwise} \end{cases}$$

where e is $\bigcup\{G'_{\odot}(A, A''/a, b) : A'' \in A'\}$. Finally, $G_{\odot}$ can be defined as a $\mathrm{PCSF}^{+}(\alpha_a)$ term by

$$G_{\odot}(A, B/a, b) \;=\; \bigcup\{u \in \mathrm{tc}(G'_{\odot}(A, A/a, b)) : \langle A, u\rangle \in G'_{\odot}(A, A/a, b)\}.$$

Here we again used **(Normal Separation$^{\mathbf{SN}}$)**. That completes the proof of part 1. The proofs of parts 2. and 3. are similar, and left to the reader.

Parts 1. and 2. imply that the function satisfying

$$G_{\sigma}(A, B/a, b, a', b') \;=\; \sigma_{a,b}(a', b')$$

for $a' \in \mathrm{tc}^{+}(a)$ and $b' \in \mathrm{tc}^{+}(b)$ is given by a $\mathrm{PCSF}^{+}(\alpha_a, \alpha_b)$ term (since the embeddings α_a and α_b also are embeddings $a' \preccurlyeq A$ and $b' \preccurlyeq B$ respectively). Therefore the definitions for $\pi_{1,a,b}$ and $\pi_{2,a,b}$ given for the proof of part 13. of Theorem 13 immediately yield $\mathrm{PCSF}^{+}(\alpha_a, \alpha_b)$ definitions for G_{π_1} and G_{π_2}. □

Corollary 43. *Let $h(\vec{a})$ be a #-term. Then there is a $\mathrm{PCSF}^{+}(\vec{\alpha})$ term $T(\vec{A}/\vec{a}, x)$ so that the following holds: If $\vec{\sigma}$ is a vector of PCSF^{+} functions such that $\vec{\sigma} : \vec{a} \preccurlyeq \vec{A}$ are safe embeddings and if $\tau = T[\vec{\sigma}]$, then $x \mapsto \tau(\vec{A}/\vec{a}, x)$ is a safe embedding $h(\vec{a}) \preccurlyeq h(\vec{A})$.*

Proof. This is a consequence of Lemma 42, the proof of Lemma 19, and Proposition 3.2 of Arai [1], using induction on the complexity of h. □

We can now establish the main technical result for this section.

Theorem 44. *Suppose $f(\vec{a})$ is in CRSF. Then there are $\mathrm{PCSF}^{+}(\vec{\alpha})$ terms $G(\vec{A}/\vec{a})$ and $T(\vec{A}/\vec{a}, x)$, and a #-term $t(\vec{A})$ so that the following holds: If $\vec{\sigma}$ is a vector of PCSF^{+} functions such that $\vec{\sigma} : \vec{a} \preccurlyeq \vec{A}$ are safe embeddings, and if $g = G[\vec{\sigma}]$ and $\tau = T[\vec{\sigma}]$, then*

1. $g(\vec{A}/\vec{a}) = f(\vec{a})$,

2. $\tau(\vec{A}/\vec{a}, x)$ *is a safe embedding,* $\tau : g(\vec{A}/\vec{a}) \preccurlyeq t(\vec{A})$.

Theorem 36 is an immediate consequence of Theorem 44 since we may let $\vec{A}$ equal $\vec{a}$ and let $\vec{\sigma}$ be the identity (multi-valued) embeddings $x \mapsto \{x\}$.

Proof. Theorem 44 is proved by induction on the formation of CRSF functions. The initial function **(Null)** is trivial. The **(Projection)** function π^n_i is also trivial: $g(\vec{A}/\vec{a}) = a_i$ is an initial function of PCSF, and we let $t(\vec{A})$ be A_i, and $T(\vec{A}/\vec{a}, x)$ be equal to $\alpha_i(/x)$.

For f equal to $\mathrm{pair}(a_1, a_2)$, let $g(A_1, A_2/a_1, a_2)$ equal $\mathrm{pair}(/a_1, a_2)$, and let t equal $1 \odot A_1 \odot 1 \odot A_2$. The (multivalued) safe embedding T is defined by

$$T(A_1, A_2/a_1, a_2, x) \;=\; \begin{cases} \alpha_2(/x) & \text{if } x \in \mathrm{tc}(a_2) \\ \{A_2\} & \text{if } x = a_2 \\ \{y \odot 1 \odot A_2 : y \in \alpha_1(/x)\} & \text{if } x \in \mathrm{tc}(a_1) \setminus \mathrm{tc}^+(a_2) \\ \{A_1 \odot 1 \odot A_2\} & \text{otherwise.} \end{cases} \tag{29}$$

Here we are using the convention that $\mathrm{tc}(a_2)$ means $\mathrm{tc}'(A_2/a_2)$. The set $\{y \odot 1 \odot A_2 : y \in \alpha_1(/x)\}$ in the third case is equal to

$$\{z \in \mathrm{tc}(A_1 \odot 1 \odot A_2) : \exists y {\in} \mathrm{tc}(A_1),\ z{=}y \odot 1 \odot A_2 \wedge y {\in} \alpha_1(/x)\}.$$

Although y is a safe parameter, $y \odot 1 \odot A_2$ can be computed by the PCSF$^+$ function $f'_\odot(A_1/y, 1 \odot A_2)$. Thus T is a PCSF$^+$ function.

The case of f equal to $\mathrm{cond}_\in$ is handled similarly to pair. The **(Union)** function, $a_1 \mapsto \bigcup a_1$, is easily handled by letting $G(A_1/a_1) = \bigcup a_1$, $t(A_1) = A_1$, and $T(A_1/a_1, x) = \alpha_1(/x)$.

For f the smash function, $f(a_1, a_2) = a_1 \# a_2$, the function $g(A_1, A_2/a_1, a_2) = a_1 \# a_2$ is definable with a PCSF$^+(\alpha_1, \alpha_2)$ term by Lemma 42. Define the $\#$-term $t(A_1, A_2)$ to equal $A_1 \# A_2$. Then Corollary 43 gives a PCSF$^+(\alpha_1, \alpha_2)$ term $T(A_1, A_2/a_1, a_2, x)$ which gives the desired safe embeddings.

Now suppose $f(\vec{a})$ is defined by **(Composition)** as

$$f(\vec{a}) \;=\; f_0(f_1(\vec{a}), \ldots, f_\ell(\vec{a})).$$

The induction hypotheses for the f_i's, for $i > 0$, give PCSF$^+(\vec{\alpha})$ terms $G_i(\vec{A}/\vec{a})$ and $T_i(\vec{A}/\vec{a}, x)$ and $\#$-terms $t_i(\vec{A})$. For appropriate embeddings $\vec{\sigma}$, let $g_i = G_i[\vec{\sigma}]$ and $\tau_i = T_i[\vec{\sigma}]$. The induction hypothesis also gives that $g_i(\vec{A}/\vec{a}) = f_i(\vec{a})$ and

$$\tau_i(\vec{A}/\vec{a}, x) : g_i(\vec{A}/\vec{a}) \preccurlyeq t_i(\vec{A}) \tag{30}$$

for each $i > 0$. The induction hypothesis for $f_0(b_1, \ldots, b_\ell)$ gives PCSF$^+(\vec{\beta})$ terms $G_0(\vec{B}/\vec{b})$ and $T_0(\vec{B}/\vec{b}, x)$ and a #-term $t_0(\vec{B})$. Let $g_0 = G_0[\vec{\tau}]$ and $\tau_0 = T_0[\vec{\tau}]$. Furthermore, let $B_i = t_i(\vec{A})$ and $b_i = f_i(\vec{a})$. Then we have, again by induction hypothesis for $f_0(b_1, \ldots, b_\ell)$ and using (30), that $g_0(\vec{B}/\vec{b}) = f_0(\vec{b})$ and

$$\tau_0(\vec{B}/\vec{b}, x) : f_0(\vec{b}) \preccurlyeq t_0(\vec{B}) \ .$$

Let G be the PCSF$^+(\vec{\alpha})$ term

$$G(\vec{A}/\vec{a}) \ = \ (G_0[T_1, \ldots, T_\ell])(t_1(\vec{A}), \ldots, t_\ell(\vec{A}) / G_1(\vec{A}/\vec{a}), \ldots, G_\ell(\vec{A}/\vec{a})),$$

T be the PCSF$^+(\vec{\alpha})$ term

$$T(\vec{A}/\vec{a}) \ = \ (T_0[T_1, \ldots, T_\ell])(t_1(\vec{A}), \ldots, t_\ell(\vec{A}) / G_1(\vec{A}/\vec{a}), \ldots, G_\ell(\vec{A}/\vec{a}), x),$$

and t be the #-term $t_0(t_1(\vec{A}), \ldots, t_\ell(\vec{A}))$. Finally, let g be $G[\vec{\sigma}]$ and τ be $T[\vec{\sigma}]$. Unwinding the definitions shows that

$$\begin{aligned} g(\vec{A}/\vec{a}) \ &= \ g_0(t_1(\vec{A}), \ldots, t_\ell(\vec{A}) / g_1(\vec{A}/\vec{a}), \ldots, g_\ell(\vec{A}/\vec{a})) \\ &= \ g_0(B_1, \ldots, B_\ell / b_1, \ldots, b_\ell) \\ &= \ f_0(f_1(\vec{a}), \ldots, f_\ell(\vec{a})) \ = \ f(\vec{a}), \end{aligned}$$

and $\tau(\vec{A}/\vec{a}, x) : g(\vec{A}/\vec{a}) \preccurlyeq t(\vec{A})$.

The rest of the proof deals with the case where f is defined by **(Cobham Recursion$_\preccurlyeq$)**. We have

$$f(\vec{a}, c) \ = \ f_0(\vec{a}, c, \{f(\vec{a}, c') : c' \in c\}),$$

with $z \mapsto \tau_1(z, \vec{a}, c)$ as the embedding function $\tau_1 : f(\vec{a}, c) \preccurlyeq h(\vec{a}, c)$ where, w.l.o.g. by Theorem 21, h is a #-term. The induction hypothesis for $f_0(\vec{a}, c, d)$ gives PCSF$^+(\vec{\alpha})$ terms $G_0(\vec{A}, C, D/\vec{a}, c, d)$ and $T_0(\vec{A}, C, D/\vec{a}, c, d, x)$ and a #-term $t_0(\vec{A}, C, D)$. The induction hypothesis for τ_1 gives a PCSF$^+(\vec{\alpha})$ term $G_1(Z, \vec{A}, C/z, \vec{a}, c)$. (It also gives $T_1(Z, \vec{A}, C/z, \vec{a}, c, x)$ and #-term $t_1(Z, \vec{A}, C)$, but we will not need to use these, since h is a #-term.)

Let the lists $\vec{a}$ and $\vec{A}$ have length k. We let $\vec{\alpha}_a$ denote a vector of metavariables $\alpha_{a_i}(/x)$ for safe embeddings $a_i \preccurlyeq A_i$ for $i = 1, \ldots k$. We also let $\alpha_c(/x)$, $\alpha_d(/x)$ and $\alpha_z(/x)$ be metavariables for safe embeddings $c \preccurlyeq C$, $d \preccurlyeq D$ and $z \preccurlyeq Z$ respectively. We use $\vec{\sigma}_a$, σ_c, σ_d and σ_z to denote particular safe embeddings (that are substituted for the α's).

The idea for this case is to define an intermediate PCSF$^+(\vec{\alpha}_a, \alpha_c)$ term $G'(\vec{A}, C, C'/\vec{a}, c)$ which represents the "course-of-values" function for f as a set of ordered pairs. There are two difficulties that have to be overcome in order for this work. The first difficulty is that PCSF$^+$ functions cannot recurse on safe inputs: for this reason, G' takes a normal parameter C' and the recursion will be on members C' of $\mathrm{tc}^+(C)$, not on members c' of $\mathrm{tc}^+(c)$. As in the proof of Lemma 42, the embedding α_c will be used to make C' represent a set $c' \in \mathrm{tc}^+(c)$, allowing us to simulate $\in$-recursion on members $c' \in \mathrm{tc}^+(c)$ using $\in$-recursion on members $C' \in \mathrm{tc}^+(C)$. The second difficulty is that G' will work by recursively invoking G_0 to generate the course-of-values, but to use G_0 we need a safe embedding σ_d of the safe parameter d (representing the set of previous values of f) into some #-term D. The natural way to define σ_d would be by a separate recursion, but this seems not to work easily. Instead, G' will compute the graph of such a safe embedding at the same time as it computes the course-of-values of f.

Specifically, we will define G' so that, when $\vec{\sigma}_a, \sigma_c$ are safe embeddings for $\vec{a}, c$ into $\vec{A}, C$ and $g' = G'[\vec{\sigma}_a, \sigma_c]$ and $C' \in \mathrm{tc}^+(C)$, then $g'(\vec{A}, C, C' / \vec{a}, c)$ is equal to a set $e = \langle e_1, e_2 \rangle$ for which the following hold:

(A) The set e_1 gives the course-of-values pairs for f on $\mathrm{tc}^+(C')$. Namely, e_1 is equal to

$$\{\langle C'', f(\vec{a}, c'')\rangle : C'' \in \mathrm{tc}^+(C') \wedge c'' \in \mathrm{tc}^+(c) \wedge C'' \overset{\sigma_c}{\approx} c''\}. \tag{31}$$

(B) For each $C'' \in \mathrm{tc}^+(C')$, the set e_2 explicitly describes an embedding of $f(\vec{a}, c'')$ into $h(\vec{A}, C)$, for c'' corresponding to C''. Formally, if there is a $c'' \in \mathrm{tc}^+(c)$ such that $C'' \overset{\sigma_c}{\approx} c''$, then e_2 contains triples $\langle C'', x, y \rangle$ where $x \in \mathrm{tc}(f(\vec{a}, c''))$ and $y \in h(\vec{A}, C)$ such that the map

$$x \mapsto \{y : \langle C'', x, y \rangle \in e_2\} \tag{32}$$

gives a safe embedding of $f(\vec{a}, c'')$ into $h(\vec{A}, C)$.

Note that in (B), we used "$\mathrm{tc}(f(\vec{a}, c''))$"; this is a permitted use of transitive closure since $\mathrm{rank}(f(\vec{a}, c''))$ is bounded by $\mathrm{rank}(h(\vec{A}, C))$. Similar considerations apply to later uses of the transitive closure function with safe parameters.

We define $G'(\vec{A}, C, C' / \vec{a}, c)$ by **(Predicative Set Recursion$^{\mathbf{SN}}$)**. The set U of previous values from the recursion,

$$U \ := \ \{G'(\vec{A}, C, C'' / \vec{a}, c) : C'' \in C'\},$$

is used as a safe parameter. Each member of U is a pair $\langle e_1^{C''}, e_2^{C''} \rangle$. Forming the unions of these components, we let e^- henceforth denote the expression

$$\left\langle \bigcup \{\pi_1(u) : u \in U\}, \bigcup \{\pi_2(u) : u \in U\} \right\rangle \tag{33}$$

and will call the first and second components of this respectively e_1^- and e_2^-: they are given by PCSF functions of U with U as safe parameter. Suppose inductively that conditions (A) and (B) hold for all pairs in U. Let (A$^-$) and (B$^-$) be (A) and (B) with each occurrence of "$C'' \in \mathrm{tc}^+(C')$" replaced with "$C'' \in \mathrm{tc}(C')$". Then we have that e_1^- satisfies (A$^-$) and e_2^- satisfies (B$^-$). In the case of e_1^- this is automatic. The fact e_2^- satisfies (B$^-$) follows from the fact that our recursive construction of e_2 will make the embedding at each C' uniquely determined by the embeddings on $\mathrm{tc}(C')$; this ensures the embeddings encoded by members of U are consistent with each other.

Suppose that there is no $c' \in \mathrm{tc}^+(c)$ such that $C' \overset{\alpha_c}{\approx} c'$. In this case we let $G'(\vec{A}, C, C' / \vec{a}, c)$ simply be e^-, and (A) and (B) follow from (A$^-$) and (B$^-$). On the other hand, if there is such a c' then it must be unique, and we can compute it from $\vec{A}, C, C' / \vec{a}, c$ using **(Normal Separation$^{\mathrm{SN}}$)**. We then have three tasks. The first is to compute the set

$$d = \{f(\vec{a}, c'') : c'' \in c'\}.$$

The second is to use G_0 to compute $f(\vec{a}, c')$, and add the pair $\langle C', f(\vec{a}, c')\rangle$ to e_1^- to get e_1 satisfying (A). The third is to use G_1 and T_0 to find an embedding $f(\vec{a}, c') \preccurlyeq h(\vec{A}, C)$, so that we can extend e_2^- to e_2 satisfying (B).

The first task is easy, as we can recover d by reading the values $f(\vec{a}, c'')$ out of e_1^-, using **(Normal Separation$^{\mathrm{SN}}$)**. Precisely, d is

$$\{u \in \mathrm{tc}(e^-) : (\exists C'' {\in} \mathrm{tc}(C'))(\exists c'' {\in} c')[C'' \overset{\alpha_c}{\approx} c'' \wedge \langle C'', u\rangle {\in} e_1^-]\}\ . \tag{34}$$

Before we can use G_0 and T_0, we need a safe embedding σ_d of d into some set D, where D can be used as a normal parameter. We let D be given by the #-term $D(\vec{A}, C) = C\#(1{\odot}h(\vec{A}, C))$ and define $\sigma_d(\vec{A}, C / \vec{a}, c, e^-, z)$ to be the PCSF$^+(\vec{\alpha}_a, \alpha_c)$ term equal to

$$\{\sigma_{C,1\odot h(\vec{A},C)}(C'', y) : C'' \in \mathrm{tc}^+(C) \wedge y \in \mathrm{tc}^+(h(\vec{A}, C)) \\ \wedge[\langle C'', z, y\rangle \in e_2^- \vee (\langle C'', z\rangle \in e_1^- \wedge y = h(\vec{A}, C))]\}.$$

By (A$^-$) and (B$^-$), the expression in square brackets describes an embedding of $\mathrm{tc}^+(f(\vec{a}, c''))$ into $1{\odot}h(\vec{A}, C)$, if $C'' \overset{\sigma_c}{\approx} c''$. The function $\sigma_{C,1\odot h(\vec{A},C)}$ is used to combine these into an embedding of d into D. The value $\sigma_{C,1\odot h(\vec{A},C)}$ is computed with the PCSF$^+$ function G_σ of Lemma 42.

For the second task, recall that $G_0(\vec{A}, C, D / \vec{a}, c, d)$ is a PCSF$^+(\vec{\alpha}_a, \alpha_c, \alpha_d)$ term given by the induction hypothesis for the function $f_0(\vec{a}, c, d)$ which computes one step in the recursion defining f. Let $G_0[\sigma_d]$ be the PCSF$^+(\vec{\alpha}_a, \alpha_c)$ term that results from G_0 by substituting $\sigma_d(\vec{A}, C / \vec{a}, c, e^-, x)$ for $\alpha_d(/x)$.

We compute $f(\vec{a},c')$ as $G_0[\sigma_d](\vec{A},C,D(\vec{A},C)\,/\,\vec{a},c',d)$. This requires having embeddings $\vec{a} \preccurlyeq \vec{A}$ and $c' \preccurlyeq C$; since $c' \preccurlyeq c$, we can use σ_c as the latter embedding. The result is that $f(\vec{a},c')$ is expressed as a PCSF$^+(\vec{\alpha}_a,\alpha_c)$ term with arguments $\vec{A},C,C'\,/\,\vec{a},c,U$, since D, c', d and e^- are computed by such terms. We let e_1 be $e_1^- \cup \{\langle C', f(\vec{a},c')\rangle\}$.

For the third task, we want a PCSF$^+(\vec{\alpha}_a,\alpha_c)$ term $K(\vec{A},C,C'\,/\,\vec{a},c,e^-,z)$ which, when we substitute safe embeddings $\vec{\sigma}_a : \vec{a} \preccurlyeq \vec{A}$ and $\sigma_c : c \preccurlyeq C$ for $\vec{\alpha}_a$ and α_c, computes an embedding $f(\vec{a},c') \preccurlyeq h(\vec{A},C)$. Recall the PCSF$^+(\alpha_z,\vec{\alpha}_a,\alpha_c)$ term $G_1(Z,\vec{A},C\,/\,z,\vec{a},c)$ from the induction hypothesis, which gives an embedding $f(\vec{a},c) \preccurlyeq h(\vec{a},c)$, with z being used as the embedding variable. Below we will define a #-term $Z(\vec{A},C)$ and a PCSF$^+(\vec{\alpha}_a,\alpha_c)$ term $\sigma_z(\vec{A},C,C'\,/\,\vec{a},c,e^-,x)$ which defines an embedding $z \preccurlyeq Z(\vec{A},C)$ for all $z \in \mathrm{tc}^+(f(\vec{a},c'))$, with embedding variable x. Then, substituting σ_z for α_z, $G_1[\sigma_z](Z(\vec{A},C),\vec{A},C\,/\,z,\vec{a},c')$ is almost the required PCSF$^+(\vec{\alpha}_a,\alpha_c)$ term, since it computes, for suitable $\vec{\sigma}_a$ and σ_c, an embedding $f(\vec{a},c') \preccurlyeq h(\vec{a},c')$. To get the term K we compose this with a PCSF$^+(\vec{\alpha}_a,\alpha_c)$ term given by Corollary 43, computing an embedding $h(\vec{a},c') \preccurlyeq h(\vec{A},C)$ whenever suitable safe embeddings $\vec{\sigma}_a,\sigma_c$ are substituted for $\vec{\alpha}_a,\alpha_c$. (Throughout we are using, as before, that a safe embedding $c \preccurlyeq C$ is also a safe embedding $c' \preccurlyeq C$.)

To define $Z(\vec{A},C)$, recall from the inductive hypothesis that we have a PCSF$^+(\vec{\alpha}_a,\alpha_c,\alpha_d)$ term $T_0(\vec{A},C,D\,/\,\vec{a},c,d,x)$ and a #-term $t_0(\vec{A},C,D)$ such that, for suitable safe embeddings $\vec{\sigma}_a,\sigma_c,\sigma_d$, the term T_0 gives an embedding $f_0(\vec{a},c,d) \preccurlyeq t_0(\vec{A},C,D)$. We let $\sigma_z(\vec{A},C,C'\,/\,\vec{a},c,e^-,x)$ be the PCSF$^+(\vec{\alpha}_a,\alpha_c)$ term $T_0[\sigma_d](\vec{A},C,D\,/\,\vec{a},c',d,x)$ and let $Z(\vec{A},C)$ be the #-term $t_0(\vec{A},C,D)$, where D, c' and d are computed from $A,C,C'\,/\,\vec{a},c,U$ as above. Then σ_z gives an embedding $f_0(\vec{a},c',d) \preccurlyeq Z(\vec{A},C)$. But $f_0(\vec{a},c',d)$ equals $f(\vec{a},c')$, and it follows that σ_z and $Z(\vec{A},C)$ have exactly the properties needed in the previous paragraph. This completes the construction of the embedding K. We let e_2 be

$$e_2^- \cup \{\langle C',x,y\rangle : x \in \mathrm{tc}(f(\vec{a},c')) \wedge y \in \mathrm{tc}(h(\vec{A},C)) \wedge y \in K(\vec{A},C,C'\,/\,\vec{a},c,e^-,x)\}.$$

We let $e = \langle e_1,e_2\rangle$. We have shown how e can be computed by a PCSF$^+(\vec{\alpha}_a,\alpha_c)$ term from $\vec{A},C,C'\,/\,\vec{a},c,U$. This completes the definition of $G'(\vec{A},C,C'\,/\,\vec{a},c)$ by **(Predicative Set Recursion$^{\mathbf{SN}}$)**.

Now that G' has been defined, it is easy to define the desired $G(\vec{A},C\,/\,\vec{a},c)$ as a PCSF$^+(\vec{\alpha},\alpha_c)$ term. Namely, $G(\vec{A},C\,/\,\vec{a},c)$ is the unique $u \in \mathrm{tc}(G'(\vec{A},C,C\,/\,\vec{a},c))$ such that $\langle C,u\rangle$ is in $\pi_1(G'(\vec{A},C,C\,/\,\vec{a},c))$. (Here we use **(Normal Separation$^{\mathbf{SN}}$)**.) With this, we have $G[\vec{\sigma}_a,\sigma_c](\vec{A},C\,/\,\vec{a},c) = f(\vec{a},c)$ whenever $\vec{\sigma}_a$ and σ_c are appropriate safe embeddings. The desired $\mathrm{PCSF}^+(\vec{\alpha}_a, \alpha_c)$ term T and #-term t for part 2. of Theorem 44 are obtained by letting $t = h(\vec{A}, C)$ and defining $T(\vec{A}, C/\vec{a}, c, x)$ to equal $\{y \in h(\vec{A}, C) : \langle C, x, y\rangle \in \pi_2(G'(\vec{A}, C, C/\vec{a}, c))\}$. This completes the proof of Theorem 44. □

References

[1] T. Arai, *Predicatively computable functions on sets*, Archive for Mathematical Logic, 54 (2015), pp. 471–485.

[2] A. Beckmann, S. R. Buss, and S.-D. Friedman, *Safe recursive set functions*, Journal of Symbolic Logic, 80 (2015), pp. 730–762.

[3] A. Beckmann, S. R. Buss, S.-D. Friedman, M. Müller, and N. Thapen, *Cobham set functions and Boolean circuits, preliminary version*. Manuscript in preparation, July 2014.

[4] ——, *Title to be determined, preliminary version*. Manuscript in preparation, October 2014.

[5] S. Bellantoni and S. Cook, *A new recursion-theoretic characterization of the polytime functions*, Computational Complexity, 2 (1992), pp. 97–110.

[6] S. R. Buss, *Bounded Arithmetic*, Bibliopolis, 1986. Revision of 1985 Princeton University Ph.D. thesis.

[7] A. Cobham, *The intrinsic computational difficulty of functions*, in Logic, Methodology and Philosophy of Science, Proceedings of the Second International Congress, held in Jerusalem, 1964, Y. Bar-Hillel, ed., Amsterdam, 1965, North-Holland.

[8] R. B. Jensen, *The fine structure of the constructible hierarchy*, Annals of Mathematical Logic, 4 (1972), pp. 229–208. Errata, ibid **4** (1972) 443.

[9] D. Leivant, *A foundational delineation of computational feasibility*, in Proceedings Sixth Annual IEEE Symposium on Logic in Computer Science, 1991, pp. 2–11.

[10] E. Nelson, *Predicative Arithmetic*, Princeton University Press, 1986.

[11] V. Y. Sazonov, *On bounded set theory*, in Logic and Scientific Methods, M. L. D. C. et al., ed., Synthese Library Volume 259, Kluwer Academic, 1997, pp. 85–103.