THE PATTERN-OF-CALLS EXPANSION IS THE CANONICAL FIXPOINT FOR RECURSIVE DEFINITIONS*

Michael A. Arbib
Computer and Information Science
and
Ernest G. Manes
Mathematics and Statistics
University of Massachusetts
Amherst, MA 01003

(Revised May 1980)

Abstract

In partially-additive semantics, a program is interpreted as a 'sum' of the paths taken during execution. Part I motivates partial-addition by looking at partial functions; axiomatizes the general notion of partially-additive monoid; and discusses examples. The pattern-of-calls expansion is an 'infinite sum' decomposition which provides a new expression for the interpretive semantics of a recursively defined program. Part II defines this expansion, provides associated rules for proofs of correctness and demonstrates program transformation by algebraic manipulation. Part III introduces the general notion of a canonical fixpoint for a family of recursive definitions as an assignment of a fixpoint to each definition in a fashion preserved by a given class of homomorphisms. As a corollary of the 'canonical fixpoint theorem' it is shown that the least fixpoint is the unique canonical fixpoint for order semantics. We offer a general axiomatic treatment of the pattern-of-calls expansion for abstract recursion schemes, and show that our pattern-of-calls expansion is the unique canonical fixpoint for abstract recursion schemes. We close with a comparison of order semantics and partially-additive semantics, showing that in the usual recursive calls defined over partial functions, the least fixpoint and the pattern-of-calls expansion return the same interpretation (but via different formulas).

---

* The research reported in this paper was supported in part by the National Science Foundation under grant MCS76-84477. This paper is a substantially revised and improved version of "Abstract Theory of Recursive Calls", COINS Technical Report 78-18, Department of Computer and Information Science, University of Massachusetts at Amherst (August 1978). Accepted December 1980 for publication in the Journal of the ACM.

There have been a number of different attempts to define a mathematical setting in which to provide a general formal theory of the semantics of computer programs. Perhaps the approach most widely accepted at present emphasizes the use of partial orders with appropriate additional structure (see [6, 18, 19, 26, 30, 33, 34, 35, 36, 38, 39, 41] and the references cited there), but other authors have urged such structures as algebraic theories [14, 16, 21, 40] and metric spaces [6, 32].

In a recent paper [1] we introduced an axiomatization of a partially-defined addition. In the present paper, which is self-contained and is written for computer scientists, we extend that theory to obtain a new approach to the semantics of recursive calls. Specialized to the arena of partial function interpretations, our "pattern-of-calls expansion" provides a new expansion formula for the least fix-point semantics which is a candidate for the desideratum of [4] of allowing properties of a program to be obtained by simple algebraic manipulation. (See the analysis of the program of 3.5.)

It is our intention to offer partially-additive semantics as an insightful auxiliary rather than a supplanting competitor to order semantics. The tree induction rule (5.1, Theorem 9.9) is an example of how the two theories overlap. Whereas the order approach makes contact with the well-established mathematical discipline of lattice theory, partially-additive semantics relies on a new structure, a partially-additive monoid (Section 2).

An important principle of order semantics much emphasized by Dana Scott is the idea that each element of a semantic domain should be 'canonically approximated by its finite pieces'. This idea is also a major principle of partially-additive semantics, even at the most abstract level (the limit axiom of 2.1; the abstract characterization of partially-additive categories [1, 5.3] wherein a countable copower is the inverse limit of its finite projections).

In Part I, we introduce partially-additive semantics. Section 1 motivates our approach by looking at the interpretation of simple flow diagrams in terms of partial functions -- interpreting the program as the 'sum' of the interpretation of its paths. This addition is only partially defined, and in Section 2 we provide a general axiomatization of partial addition, and give examples.

In Part II we work with partial-function semantics to associate the pattern-of-calls expansion with an arbitrary recursive call. The expansion has one term for each distinct pattern of calls, where the elementary calls correspond to choosing those paths in the definition which involve n variables for a single integer $n \geq 0$. In the motivating case, each term

in the expansion is a partial function which is disjoint from all other terms, and the semantics of the recursive call is simply the 'sum' = union of these terms. Section 5 then offers techniques for proving program properties with the help of this expansion.

Part III extends our analysis of the pattern-of-calls expansion to a far more general setting than that of partial functions. A noteworthy feature of the pattern-of-calls expansion is that it bases its semantics on an abstract syntax, i.e., the syntax does not depend on the particular syntax (and thus set of operator and predicate symbols) of any one programming language. Moreover, Part III makes a contribution that is purely foundational. We ask what makes the least fixpoint play so special a role in order semantics. We do this by looking at recursive definitions which are related by appropriate homomorphisms to introduce the notion of a canonical fixpoint as an assignment of a fixpoint to each definition in a manner consistent with the homomorphic structure. The impact of the canonical fixpoint theorem 6.5 and its corollary 6.6 is that the Kleene formula* for the least fixpoint $\bigvee h^n(\perp)$ of h can be stated in a different language which requires no mention of ordered sets. This both paves the way for the development of alternate foundations (partially-additive semantics being one such) and provides a tool for proving that two semantical theories agree on their overlap. Section 9 offers a comparison with order semantics.

---

* The Knaster-Tarski theorem asserts that if L is a complete lattice, then any monotone and continuous mapping from L to itself has a least fixed point. The usual citation for this result is [37], but a special case appears in [24] -- in 1927, Knaster and Tarski jointly proved a set-theoretical fixpoint theorem by which every function $2^S \longrightarrow 2^S$ which is increasing with respect to set inclusion has at least one fixpoint. The formula $\bigcup_{n\geq 0} h^n(\emptyset)$ for the least fixed point of a continuous $h : 2^S \longrightarrow 2^S$ occurs as part of the proof of Kleene's recursion theorem ([31, p. 193]; see [23] for the theorem).

(For a comparison of partially-additive semantics and iterative algebraic theories see [1, Section 7].)

We thank the editor and referees for thoughtful criticism of earlier drafts of this paper.

Part I: Partially-Additive Structure

1. Partial-Function Semantics of Flow Diagrams

In following the execution of a program, we note that control will follow different paths depending on the values of the data. In a deterministic program, each such path may be thought of as defining a partial function; distinct paths define disjoint partial functions (i.e. no initial data can cause execution of more than one path); and the semantics of the program may be thought of as the 'sum' of all these paths. To put all this on a more definite basis, let us fix a set D which will include the input and output values of our computations, so that we may (for the moment) restrict attention to the set $\underline{\underline{Pfn}}(D,D)$ of partial functions from the set D to itself. Partially-additive semantics [1] provides the formal basis for the above notion of sum-of-paths as follows. Define a partial addition for families $(f_i : i \in J)$ in $\underline{\underline{Pfn}}(D,D)$ by saying that Σf_i is defined only if the domains of the f_i are disjoint, and we then have

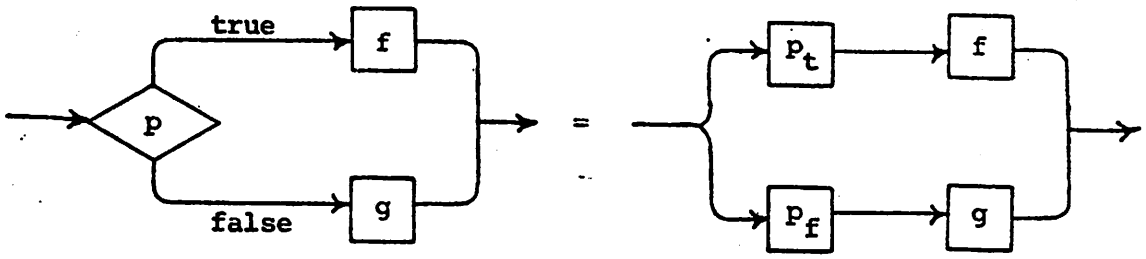
1

$$(\Sigma f_i)(x) = \begin{cases} f_j(x) & \text{if } x \in \text{dom}(f_j) \\ \text{undefined} & \text{if no such } j \text{ exists.} \end{cases}$$

With this partial-addition, we may explicitly express the semantics of a flow diagram as a sum of paths. We show this for conditionals and then for iteration.

Given a test $p: D \longrightarrow \{\text{true, false}\}$, we may define partial functions in $\underline{\underline{Pfn}}(D,D)$, $p_t(d) = d$ with domain $\{d : p(d) = \text{true}\}$, and $p_f(d) = d$ with domain $\{d : p(d) = \text{false}\}$. In this framework, the conditional $\underline{\text{if}}$ p $\underline{\text{then}}$ f $\underline{\text{else}}$ g is interpreted as $f \cdot p_t + g \cdot p_f$, as diagrammed in 2. This sum puts 'the different possible paths in a test' in the abstract language. Conversely, any sum can be thought of as a generalized conditional.

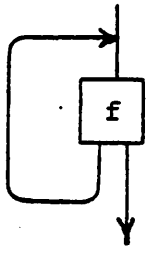
2



As another example of sum-of-paths semantics, consider iteration as shown in the simple flow diagram (3) where at any defined stage of the computation, the state lies in some fixed data set D.

3

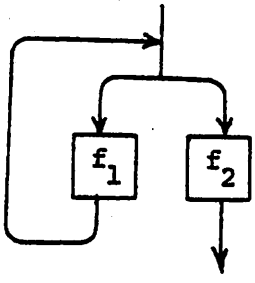
$f^\dagger$ =



Thus f is to be interpreted as a map $D \longrightarrow D + D$ (where $D + D$ is the disjoint union $\{(d,1) : d \in D\} \cup \{(d,2) : d \in D\}$ of two copies of D), while the iterate of f is to be interpreted as a partial function (the computation may get 'stuck in the loop') $f^\dagger : D \longrightarrow D$.

We may rewrite 3 as

4



where we have broken $f: D \longrightarrow D + D$ into disjoint pieces f_j, $j = 1, 2$, where

$$f_j(d) = \begin{cases} d' & \text{if } f(d) = (d',j) \\ \text{undefined} & \text{if no such } d' \text{ exists.} \end{cases}$$

We may then observe that for each d in D, either $f^\dagger(d)$ is undefined, or

there exists exactly one path involving $n \geq 0$ traversals of the loop of 4 which defines the value of $f^\dagger(d)$ as $f_2 f_1^n$. Hence

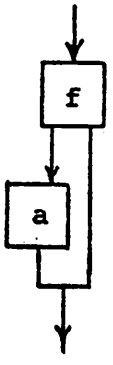
5 $$f^\dagger = \sum_{n \geq 0} f_2 f_1^n$$

expresses $f^\dagger$ as a sum of disjoint partial functions, one for each distinct path from the entry to the exit of 4.

Many authors view the iterate $f^\dagger$ as being defined as a fixed point of h, where

6

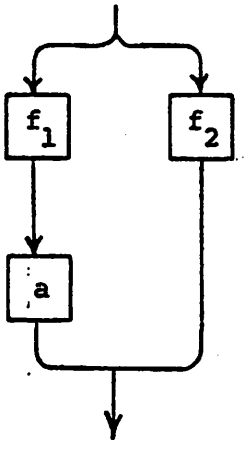
h(a) =



which may be rewritten as

7

h(a) =



$= a \cdot f_1 + f_2$.

Our problem, then, is to associate a 'sum of disjoint paths' with any recursive definition in the way that we associated 5 with the (right linear) recursive definition 7 of iteration.

2. Partially-Additive Monoids

In this section we offer the general definition of partially-additive monoid which generalizes the properties of $\underline{Pfn}(D,D)$ discussed in Section 1.

We saw that given any family of partial functions $(f_i : D \longrightarrow D : i \in I)$, with one term f_i for each i in I, which were mutually disjoint we could define their sum $\Sigma_{i\in I} f_i : D \longrightarrow D$ by the rule

$$\Sigma_{i\in I} f_i(d) = \begin{cases} f_j(d) & \text{for } d \in \text{dom}(f_j) \\ \text{undefined} & \text{if no such } j \text{ exists} \end{cases}$$

The reader should easily see that this definition satisfies the partition-associativity, limit, and unary sum axioms listed in the definition below, so that $(\underline{Pfn}(D,D),\Sigma)$ is indeed an example of our general concept of a partially-additive monoid.

1 Definition [1]: A partially-additive monoid is a pair (A,Σ) where A is a set and Σ is a partial operation on countable (i.e. finite or denumerable) sequences in A subject to the following axioms:

Partition-associativity axiom: If the countable set I is partitioned into $(I_j : j \in J)$ then for each family $(a_i : i \in I)$ in A,

$$\Sigma(a_i : i \in I) = \Sigma(\Sigma(a_i : i \in I_j) : j \in J)$$

in the sense that the left side is defined iff the right side is defined and then the values are equal.

Limit axiom: If $(a_i : i \in I)$ is a countable family in A and if $\Sigma(a_i : i \in F)$ is defined for every finite subsequence F of I, then $(a_i : i \in I)$ is defined.

Unary sum axiom: For one-element families, Σa is defined, and $\Sigma a = a$.

We use the notations $f_1 + f_2 + \ldots + f_n$ for $\Sigma(f_i : i \in \{1,2,\ldots,n\})$. Note that the unary sum axiom permits the notation f_1 for $\Sigma(f_i : i \in \{1\})$. In the partition-associativity axiom, I_j may be empty. (We emphasize that for the purpose of our definitions, a partition of a countable set I is any countable family $(I_j : j \in J)$ of pairwise disjoint sets I_j whose union is I. In particular the set of j for which I_j is empty is not otherwise restricted.) Since the unary-sum axiom ensures that some sums exist, it follows that the empty sum is defined and provides an additive zero which we denote 0 or $\perp$. Moreover, since $(I_j : j \in \underline{N})$ is a partition of the empty set if each I_j is empty, we infer that $\perp$ is even a denumerable zero, i.e. $\Sigma(\perp,\perp,\ldots) = \perp$ whether the sequence is finite or denumerable. Finally, for any permutation $\sigma : I \to I$, $\Sigma(a_i : i \in I) = \Sigma(a_{\sigma i} : i \in I)$.

2 Example: Let M be the set $\underline{N} \cup \{\infty\}$, with Σ defined as follows: If $(a_1,\ldots,a_n)$ is a finite sequence of elements of $\underline{N}$, then $\Sigma(a_1,\ldots,a_n) = a_1 + \ldots + a_n$, the usual addition in $\underline{N}$. For all other sequences x, $\Sigma x = \infty$. Then M satisfies all the axioms for a partially-additive monoid with the sole exception that the partition-associativity axiom fails when more than a finite number of I_j are empty, since $\Sigma(0,0,\ldots) = \infty$. (We thank a referee for this example.)

---

3 Multi-functions: Zeiger [42] has offered a setting in which partial functions are embedded in a space of multi-functions. Here, we associate a map $f: D \to (D \to \underline{N})$ to a relation g by $f(d)(d') = m$ in $\underline{N}$ if g(d) takes the value d' some m times. A partial function then corresponds to the case in which for each d in D there is at most one d' for which $f(d)(d') \neq 0$, and it may then only equal 1. He defines pairwise addition

by $(f+g)(d)(d') = f(d)(d') + g(d)(d')$. If f and g represent partial functions, then f+g is always defined, but will represent a partial function iff f and g represent disjoint functions. We would further point out that countable sums Σf_i will not be defined unless $(f_i : i \in I)$ satisfies the condition that for all d, d' in D, $\{i \mid f_i(d)(d') \neq 0\}$ is finite. We then set $(\Sigma f_i)(d)(d') = \Sigma f_i(d)(d')$. But even with this notion of summability, $([D \to [D \to \underline{N}], \Sigma)$ is not a partially-additive monoid since the limit axiom fails. Zeiger's method was to explicitly check when sums were well-defined and yielded the representative of a partial function.

4 Nondeterministic Semantics: Note that the axioms are also satisfied by Rel(D,D), the set of all relations from D to D, with Σ being the union:

$$(d,d') \in \sum_{i \in I} R_i \iff (d,d') \in R_j \quad \text{for some } j \in I$$

and so is defined for any family of relations from D to D. Thus the limit axiom is redundant in this case, while the partition-associativity and unary sum axioms assert the general associativity and commutativity properties of the union. The use of relations under union provides the setting for program semantics given by [10,11] where (in another notation) the formula $f \cdot p_t + g \cdot p_f$ is given for the conditional -- going back, apparently, to [22].

Some colleagues have suggested that it is unnecessary to deal with partially-additive monoids since the union operation deals correctly with the issues involved with partial functions and allows Σ to be always-defined, and since 'at the end' one can verify that "partial functions lead to partial functions". But since partial functions provide the major concrete example of program semantics, we argue the interest of an axiomatic setting that

applies directly to partial functions. Lattices were well-established ten years before Kleene [23] introduced the least fixpoint approach to recursive functionals. Some general facts about partially-additive monoids first appear in [1]. These structures are new and much research remains to be done.

We offer a comparison with the order semantics that may be seen as an outgrowth of Kleene's approach in Part IV.

5 Dijkstra's Guarded Commands: Dijkstra [12] has introduced the alternate construct $IF = \underline{\text{if}}\ B_1 \to S_1 \ [] \ \ldots \ [] \ B_k \to S_k\ \underline{\text{fi}}$ with the interpretation "pick any j for which the guard B_j is true, then execute S_j for that j". If B_j is interpreted as the relation $p^j : D \to \{t,f\}$, and S_j is interpreted as the relation $g_j : D \to D$, then we form $p^j_t : D \to D$ by $(d,d) \in p^j_t$ just in case $(d,t) \in p^j$ and obtain the partially-additive semantics of the alternate construct as

$$\underline{6} \qquad g_1 \cdot p^1_t + \ldots + g_k \cdot p^k_t .$$

Dijkstra's repetitive construct $\underline{\text{do}}\ B_1 \to S_1 \ [] \ \ldots \ [] \ B_k \to S_k\ \underline{\text{od}}$ is semantically equivalent to the iteration $\underline{\text{while}}\ B_1 \vee \ldots \vee B_k\ \underline{\text{do}}\ IF$, and so is given a partially-additive semantics in terms of 6 by applying the appropriate version of 1.5 in the partially-additive monoid $\underline{\text{Rel}}(D,D)$.

7 Formal Languages: The theory of formal languages over an alphabet X lives in the partially-additive monoid $L_X = (2^{X^*}, \cup)$ of subsets of X^* (i.e., languages over X) under union as addition. In 7.11, we relate the theory of context-free languages to our general theory of recursive calls.

8 Formal Power Series over a Semiring: The set $\mathcal{B} = \{f,t\}$ of truth values satisfies the axioms for a semiring with $+$ interpreted as exclusive or, and $\cdot$ interpreted as and.

A language $L \subset X^*$ may be identified with its characteristic function $f : X^* \to \mathcal{B}$, and this in turn may be identified with the formal power series $\sum_{w \in X^*} f(x)w$ with coefficients in $\mathcal{B}$. Schützenberger [32] extended language theory by generalizing from $\mathcal{B}$ to an arbitrary semiring A. Regular sets then correspond to rational power series, and context-free languages correspond to algebraic power series. In general, the semiring of formal power series over A has no appropriate order structure though the classic cases with A equal to B or $\underline{\underline{N}}$ certainly do. To prove limit theorems, Schützenberger used the metric $d(f,g) = 1/2^n$ where n is the length of the shortest w in X^* such that $f(w) \neq g(w)$.

A <u>partially-additive semiring</u> is a partially-additive monoid additionally equipped with a multiplicative monoid structure (totally-defined associative $x,y \mapsto xy$ with unit 1, $x1 = x = 1x$) with the property that for each x the maps $y \mapsto xy$, $z \mapsto zx$ are additive (as defined in 7.2). These were introduced in [1]; cf. the complete semirings of [13] in which the sum is totally-defined but for arbitrary (not necessarily countable) sequences. The Schützenberger theory may be adapted to formal power series over an arbitrary partially-additive semiring, save that sum replaces metric structure to provide limits.

Part II: The Pattern-of-Calls Expansion: The Motivating Case

3. Motivating Examples

As a first step toward the general definition of the pattern-of-calls expansion, we re-examine the recursive definition of the iterate given in 1.7. We distinguish two situations. If the left-hand path is taken, we have a one-substitution path which returns the partial function

1 $$H_1(a) = a \cdot f_1$$

given the partial function a substituted for the single call along that path. If the right-hand path is taken, the partial function returned is

2 $$H_0 = f_2$$

which takes no arguments since we have a 0-substitution path.

We may then rewrite 1.7 as

3 $$h(a) = H_0 + H_1(a)$$

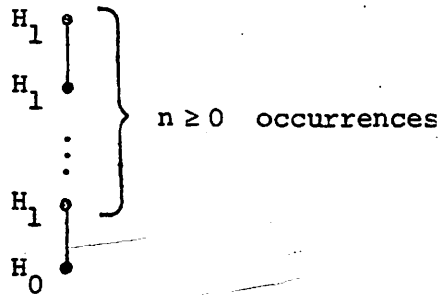
Let H denote the sequence (H_0, H_1) of functions.

We associate with this a set of partial functions PC(H) -- for pattern-of-calls of H -- defined inductively as follows:

4 $$H_0 \in PC(H)$$

If $a \in PC(H)$, then $H_1(a) \in PC(H)$.

In the inductive step, think of $H_1(a)$ as the 1-substitution path, where a computation with interpretation a is substituted for the single call. This clearly implies that the elements of PC(H) are precisely those partial functions which can be diagrammed as

$s_n =$ 

which yields the partial function $H_1^n(H_0) = f_2 \cdot f_1^n$, corresponding to n calls of the 1-substitution path in 1.7 followed by a final call of the 0-substitution path. The semantics

$$f^\dagger = \sum_{n \geq 0} f_2 \cdot f_1^n = \text{the sum of all partial functions in PC(H)}$$

for the iterate may then indeed be termed the pattern-of-calls expansion for $h(a) = f_2 + a \cdot f_1$. To prepare for the general definition of this expansion we now consider the 'nonlinear' recursive definition ([5, p. 629], [27, p. 497]) of a partial function $f: \underline{\mathbb{N}} \longrightarrow \underline{\mathbb{N}}$ by

5 $\qquad f(x) := \underline{\text{if}}\ \underline{\text{not}}\ p(x)\ \underline{\text{then}}\ f(f(g(x)))\ \underline{\text{else}}\ x.$

As in Section 1 following (1), we replace the predicate $p: \underline{\mathbb{N}} \longrightarrow \{\text{true, false}\}$ by the partial functions $p_t, p_f: \underline{\mathbb{N}} \longrightarrow \underline{\mathbb{N}}$ so that 5 corresponds to the equation $h(a) = a$ for

6 $\qquad h(a) = p_t + a^2 g p_f.$

(Here squaring refers to composition of partial functions.)

Corresponding to 3, we may rewrite this h as

7 $\qquad h(a) = H_0 + H_2(a,a)$

where $H_0 = p_t$ corresponds to the 0-substitution path, while $H_2(a_1, a_2) = a_2 a_1 g p_f$ corresponds to the 2-substitution path, with a_1 being the partial function substituted for the first call along the path, while a_2 is the partial function substituted for the second call along the path.

It is only when the same substitution is made at both places (as in checking the fixpoint equation 7) that we force the two arguments of H_2 to be equal.

We now define the set PC(H) of patterns-of-calls for the H of 7 to be the set of partial functions $\underline{\underline{N}} \longrightarrow \underline{\underline{N}}$ given by the inductive definition

8 $H_0 \in PC(H)$

If a_1 and a_2 are in PC(H), then $H_2(a_1;a_2) \in PC(H)$.

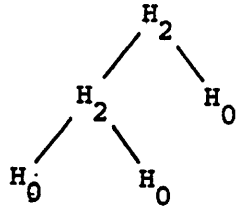
Then the general theory to be developed in later sections assures us that

9 PC(H) is a disjoint family of partial functions whose union

$$\underline{\underline{e}}^H = \Sigma(a : a \in PC(H)),$$

known as the pattern-of-calls expansion for h, is the semantics of 5.

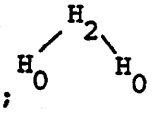
The idea is that PC(H) exhausts all possible patterns of calls. For example, consider the pattern of calls represented by the tree

$t_2 =$ 

which evaluates to

$$p_t(p_t p_t g p_f) g p_f = p_t (g p_f)^2$$

on noting that $p_t^2 = p_t$.

Here H_0 corresponds to the 0-substitution path; $t_1 =$  corresponds to the 2-substitution path with each call being to H_0; while the overall pattern t_2 corresponds to the 2-substitution path with the first call being to the pattern t_1 while the second call is to H_0.

The user of order semantics may still find the partially-additive representation of a recursive definition useful. For example, if

$$h(a) = H_0 + H_2(a,a)$$

where H_2 is additive in each argument, the Kleene sequence takes the form

$$h^0(\perp) = \perp$$

$$h^1(\perp) = H_0$$

$$h^2(\perp) = H_0 + \begin{array}{ccc} & H_2 & \\ H_0 & & H_0 \end{array}$$

$$h^3(\perp) = H_0 + \begin{array}{ccc} & H_2 & \\ H_0 & & H_0 \end{array} + \begin{array}{ccc} & H_2 & \\ H_0 & & H_2 \\ & H_0 & H_0 \end{array} + \begin{array}{ccc} & H_2 & \\ H_2 & & H_0 \\ H_0 & H_0 & \end{array} + \begin{array}{ccc} & H_2 & \\ H_2 & & H_2 \\ H_0 & H_0 & H_0 \quad H_0 \end{array}$$

The least upper bound $\bigcup_{n\geq 0} h^n(\perp)$ of this sequence is precisely the pattern-of-calls expansion PC(H). In Section 9 we shall indeed prove that for such recursive definitions over partial functions, the least fixpoint and pattern-of-calls expansion define the same partial function. In some cases, however, the particular form of the pattern-of-calls expansion may offer certain opportunities for algebraic simplification.

As an example of this utility of the pattern-of-calls expansion we use it to prove that the f of 5 in fact equals the interpretation of **while not** p **do** g. To analyze $\underline{e}^H$ we recall that

$H_0 = p_t$, while

$H_2(a_1,a_2) = a_2 a_1 g p_f$.

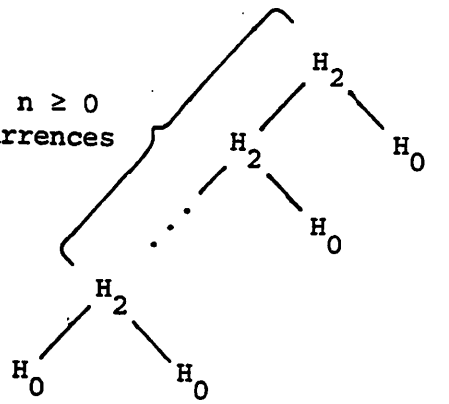
It is thus clear that the leftmost term in every $a \in PC(H)$ is p_t, while the rightmost term is p_f unless $a = H_0$. Thus

$$H_2(a_1,a_2) = (\ldots p_f)(p_t \ldots) g p_f \quad \text{unless} \quad a_2 = H_0$$

$$= 0 \quad \text{unless} \quad a_2 = H_0, \text{ since } p_f p_t = 0, \text{ the nowhere-defined partial function.}$$

Thus the only patterns-of-calls which can make a non-zero contribution to $\underline{e}^H$ are of the form

$t_n =$ $n \geq 0$ occurrences



Now $t_0 = H_0 = p_t$, while $t_{n+1} = H_2(t_n,p_t) = p_t t_n g p_f$ and we see by induction that $t_n = p_t(gp_f)^n$ since $p_t p_t = p_t$. Thus

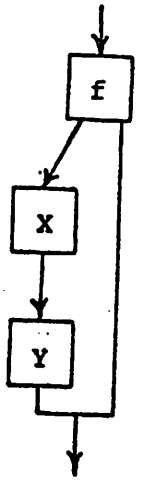
$$\underline{e}^H = \sum_{n\geq 0} t^n = \sum_{n\geq 0} p_t (gp_f)^n$$

which we recognize as the iterative fixpoint 1.5 corresponding to $f_1 = gp_f$ and $f_2 = p_t$, so that $\underline{e}^H$ does indeed equal the interpretation of <u>while not</u> p <u>do</u> g.

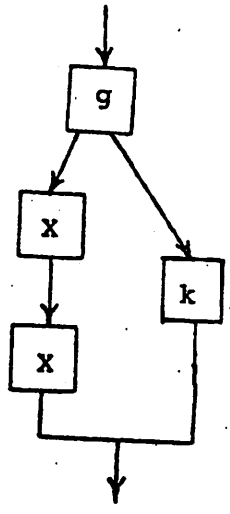
For a 'more generic' example of a recursive definition consider one which involves two procedure calls.

10

X =



Y =



Here we look for a solution to a suitable fixpoint equation $h(a) = a$ where a lies in the set $A = \underline{\underline{Pfn}}(D,D) \times \underline{\underline{Pfn}}(D,D)$ of pairs of partial functions. This A inherits a partial addition from $\underline{\underline{Pfn}}(D,D)$ under componentwise partial-addition. Decomposing f into f_1 and f_2 and g into g_1 and g_2 as before, we then rewrite 10 as

$$X = Y \cdot X \cdot f_1 + f_2, \qquad Y = X \cdot X \cdot g_1 + k \cdot g_2 .$$

We are thus led to define $H_0 \in A$ and $H_2 : A^2 \longrightarrow A$ by

11 $$H_0 = (f_2,\ k \cdot g_2)$$

$$H_2((a_{x1},a_{y1}),(a_{x2},a_{y2})) = (a_{y2} \cdot a_{x1} \cdot f_1,\ a_{x2} \cdot a_{x1} \cdot g_1).$$

Let us make clear the significance of the arguments of H_2. The first argument (a_{x1},a_{y1}) contains the interpretations -- whether for X or Y -- to be substituted for the first call along a 2-substitution path; while the second argument (a_{x2},a_{y2}) of H_2 specifies the interpretations to be substituted for the second call along a 2-substitution path. The reader should

check that this is indeed the way we have derived the formula for H_2 from the diagrams of 10. In this example, we build patterns-of-calls using H_0 and H_2 just as in 8 and 9, save that now the H's are evaluated according to 11. To check this, the reader should consider any finite pattern of calls that will lead to a value for some particular element of the domain, and then construct the tree for that pattern to note that it yields a partial function which returns the proper value.

The disjointness of the paths in 10 ensures that for each $a = (a_x, a_y) \in A$,

$$h(a) = H_0 + H_2(a,a) = (a_y \cdot a_x \cdot f_1 + f_2,\ a_x \cdot a_x \cdot g_1 + k \cdot g_2)$$

is well-defined, and that a fixed-point of this h is indeed what we seek in providing a semantics for the recursive procedure calls of 10.

Our final example shows that our method does not preclude the analysis of recursive programs which involve a changing number of variables.

12 Example: Ackermann's function may be recursively defined by

a(m,n) := if m=0 then n+1 else if n=0 then a(m-1,1) else a(m-1,a(m,n-1)).

With $(id \times a)(m,n,p) = (m, a(n,p))$, $p_1 = (m=0)_t$, $p_2 = (m=0)_f$, $q_1 = (n=0)_t$, $q_2 = (n=0)_f$, $f(m,n) = n+1$, $g(m,n) = (m-1,1)$ and $k(m,n) = (m-1,m,n-1)$, we seek solutions in $A = \underline{Pfn}(\underline{N}^2, \underline{N})$ of the equation

$$ha = H_0 + H_1 a + H_2(a,a)$$

where

$$H_0 = fp_1$$

$$H_1 a = agq_1 p_2$$

$$H_2(a_1,a_2) = a_2 \cdot (id \times a_1) \cdot kq_2 p_2 .$$

The general situation is thus clear. Given a simultaneous definition of m procedures, the A of the corresponding fixpoint equation has the form of an m-fold Cartesian product with each component being an appropriate space of partial functions. The j^{th} component of the corresponding H_n then represents the sum of the semantics of all those paths in the flow diagram defining the j^{th} procedure which involve exactly n occurrences of the to-be-defined procedures.

13 Example: Consider $H_0 + H_2(f,f)$ with $H_2(a_1,a_2) = a_2a_1\alpha + a_2a_1\beta$, where α and β are disjoint partial functions.

At first, the reader might be confused because the same variables a_1, a_2 appear in the independent paths $\alpha \rightarrow a_1 \rightarrow a_2$, $\beta \rightarrow a_1 \rightarrow a_2$, thinking that this forces the same pattern of calls "first a_1; then a_2" in the two different paths. However, the pattern-of-calls expansion exercises all patterns of calls -- for each term $H_2(a_1,a_2)$ in the expansion, there is a term $H_2(a_2,a_1)$, so that both paths are exercised both with "first a_1; then a_2" and "first a_2; then a_1". Thus lumping the 2-variable paths into H_2 does not lose any of the possible patterns of calls, but does let us use an abstract syntax (cf. 7.3) which does not require the generation of path-expressions for every distinct path using the program-dependent vocabulary of, in this case, α and β.

4. The Pattern-of-Calls Expansion

In this brief section we crystallize the concepts motivated in the previous section.

Given a recursive definition, we seek a solution in some suitable space A on which a partial-addition is defined. In the motivating examples of Section 3, A has always been of the form Pfn(D,E) for 'input' and 'output' sets D and E, or has been the cartesian product of sets of such partial functions -- with Σ being defined componentwise, so that, e.g., $(f_1,f_1') + (f_2, f_2')$ is defined just in case f_1 is disjoint from f_2 and f_1' is disjoint from f_2' and then equals $(f_1+f_2, f_1'+f_2')$. We shall see in Part III that the theory extends to the partially-additive monoids of Section 2, but in this section and the next, we explore the definition and use of the pattern-of-calls expansion in the motivating setting of partial functions.

The partially-additive semantics of Section 1 allows us to replace any recursive definition by a sum of paths, and we then gather into a term H_n all those paths which contain n procedure calls. The semantics of the recursive definition is then to be sought as a solution in A of the equation $h(a) = a$ for

1 $$h(a) = H_0 + \ldots + H_n(a,\ldots,a) + \ldots$$

where we have a term H_n for each n such that at least one n-substitution path occurs in the original recursive definition. Here

2 $$H_n : A^n \longrightarrow A$$

is defined by setting $H_n(a_1,\ldots,a_n)$ to be the sum of the values of all paths in the recursive definition which involve n procedure calls, when (the suitable component of) a_j is substituted for the j^{th} procedure call along each path. Thus a recursive definition is given by a triple (A,Σ,H), where H is the family of H_n's. Denote the set of integers n for which H_n is defined by N(H). Then our treatment of a recursive definition is as follows:

Given a recursive definition (A,Σ,H) with a term H_n for each n in some set N(H), we set

$$h(a) = \Sigma(H_n(a,\ldots,a) : n \in N(H)).$$

(In each case, we can check that h(a) is a well-defined element of A for each a in A.) We then define PC(H), the set of <u>patterns-of-calls</u> of H, inductively by

3 $$H_0 \in PC(H)$$

If $n \in N(H)$, and $a_1,\ldots,a_n \in PC(H)$, then $H_n(a_1,\ldots,a_n) \in PC(H)$.

Our main result is then the following:

4 The sum $\underline{e}^H = \Sigma(a : a \in PC(H))$ of all patterns-of-calls of h is a well-defined element of A, and satisfies the equation

$$h(\underline{e}^H) = \underline{e}^H$$

thus providing a semantics for the recursive definition underlying h. We call $\underline{e}^H$ the pattern-of-calls expansion for H.

We defer the formal proof of **4** to Part III where we not only prove the result in the general setting of partially-additive monoids, but also show that e^H is not simply a fixpoint but has algebraic properties which make the assignment $(A,\Sigma,H) \mapsto \underline{e}^H$ canonical in a sense to be defined in Section 6.

For the reader uninterested in this general theory, we provide the following 'informal proof' of **4** in the setting Pfn(D,D) of partial functions which easily extends to cover all recursive definitions of the kind given in Section 3.

First, the claim that the sum $\Sigma(a : a \in PC(H))$ is well-defined says that any two different patterns of calls yield disjoint partial functions. To see this, note that to say a_1 and a_2 are different patterns of calls is just to say that though they may coincide initially, there comes a place where a_1 chose an m-substitution path where a_2 chose an n-substitution path for $m \neq n$. In other words, the paths in a_1 and a_2 must branch apart at some stage, and will thus be interpreted as disjoint partial functions. The second claim, that $\underline{e}^H$ provides the semantics for h, simply corresponds to the observation that if any value d in D for the initial data finally returns a result d' from the recursive definition corresponding to h, then some pattern of calls must have been invoked in computing d', and so $a(d) = d'$ for the corresponding a in PC(H); and, conversely, that if $a(d) = d'$ for some a in PC(H), then the recursive definition must return d' from initial data d.

We close by stressing that nothing in our theory requires that the recursive definitions be given in terms of finite flow diagrams. For an example of a recursive definition in which N(H) is infinite, see the discussion [26] of an APL program which evaluates the determinant of a square matrix by cofactor expansion along the first row.

5. Proofs of Correctness

We now state three correctness principles, and illustrate them by proving McCarthy's 91 function. We restrict attention to the recursive definition of partial functions $f: D \to D$ by the pattern-of-calls expansion $\underline{e}^H$ of 4.4. Σ is defined on $\underline{Pfn}(D,D)$ as in 1.1, and $f \sqsubseteq g$ just in case whenever $f(x)$ is defined then $g(x)$ is also defined with $g(x) = f(x)$.

Our first rule interrelates partially-additive and order semantics:

1 Tree Induction Rule (Partial Correctness): Let $g \in \underline{Pfn}(D,D)$. To prove $\underline{e}^H \sqsubseteq g$ it is necessary and sufficient to prove that for all n in $N(H)$ and a_i in $PC(H)$ with $a_i \sqsubseteq g$ for $i = 1,\ldots,n$ we have $H_n(a_1,\ldots,a_n) \sqsubseteq g$.

Proof: If $\underline{e}^H \sqsubseteq g$ then $H_n(a_1,\ldots,a_n) \leq \underline{e}^H \sqsubseteq g$. Conversely, the hypothesis implies $H_0 \sqsubseteq g$ and so by induction $a \sqsubseteq g$ for all $a \in PC(H)$. □

2 Disjointness Lemma: If $d \in D$, then $a(d)$ is defined for at most one a in $PC(H)$. □

3 Termination Lemma: If for each d in D there exists a in PC(H) with a(d) defined then $\underline{e}^H$ is total. □

4 McCarthy's 91 Function, Partial Correctness by Tree Induction:

Here f(x) := if x>100 then x-10 else f(f(x+11)),

g(x) := if x>100 then x-10 else 91.

Setting A = Pfn(N,N), the partially-additive fixpoint equation is $f = h(f) = H_0 + H_2(f,f)$ where H_0 = if x>100 then x-10 else undefined, $H_2(s,t)$ = stu where u = if x≤100 then x+11 else undefined. Hence N(H) = {0,2}.

We prove that $f \sqsubseteq g$ by tree induction. For n = 0, $H_0 \sqsubseteq g$, clearly. For n = 2, we assume s, t in PC(H) satisfy $s,t \sqsubseteq g$ and show that $stu \sqsubseteq g$, i.e. we show that if stu(x) is defined, then g(x) = stu(x). Now, if stu(x) is defined then both u(x) and tu(x) are defined and we have:

u(x) = x+11 and x≤100;

and, because $t \sqsubseteq s$

$$tu(x) = \begin{cases} (x+11)-10 = x+1 & \text{if } x+11>100 \\ 91 \text{ if } 90 \le x \le 100 \end{cases}$$

Because $s \sqsubseteq g$

stu(x) = (x+1)-10 and x+1>100, or stu(x)=91,

i.e. stu(x)=x-9 and x=100, or stu(x)=91,

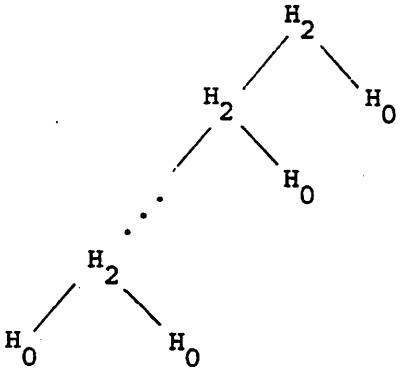
i.e. stu(x)=91.

Thus $stu \sqsubseteq g$, as was to be shown, and hence $f \sqsubseteq g$. □

5 McCarthy's 91 Function, Total Correctness by Exhaustion.

Consider g, h, H_0, H_2, u as in 4. We will show directly that $g = \Sigma(s : s \in PC(H))$. (The actual choice of trees which exhaust D as in 3 requires prior analysis which is not repeated here.) For $90 \le n \le 100$ let

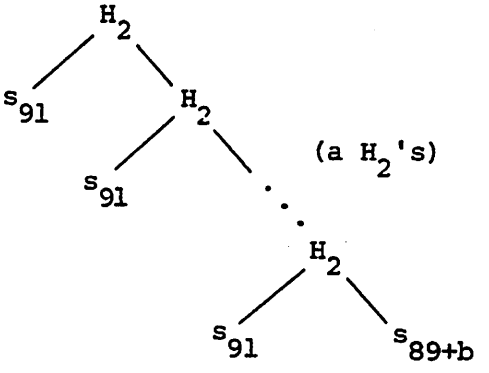
$s_n =$



(101-n H_2's)

so that $s_n = H_0(H_0u)^{101-n}$ = *if* x=n *then* 91. For $0 \le a \le 9$, $1 \le b \le 11$ let

$t_{ab} =$



(a H_2's)

Then $t_{ab} = H_0(H_0u)^{10})^a H_0(H_0u)^b$

$= \underline{\text{if}}\ x=n\ \underline{\text{then}}\ 91$

where $n+11a = 101-b$. But given $0 \le x \le 100$, such a, b exist (uniquely). □

Part III. Abstract Recursion Schemes and Canonical Fixpoints

We first analyze fixpoints of recursion schemes in a very general setting which emphasizes homomorphisms between schemes. In such a setting, a canonical fixpoint is an assignment of a fixpoint to each scheme in a fashion compatible with the homomorphic structure. The 'canonical fixpoint theorem' asserts that if there exists an initial scheme with unique fixpoint, then there exists a unique canonical fixpoint.

We then extend partially-additive semantics to introduce abstract recursion schemes, for which an abstract pattern-of-calls semantics provides the unique canonical fixpoint. In the special case of the recursively-defined programs of Part II, the abstract pattern-of-calls expansion is the same as the one already discussed, and defines the same partial function as the least fixpoint semantics.

6. General Theory of Canonical Fixpoints

Many workers in algebraic semantics have taken a fixpoint approach to recursive definitions. The question is: "Which fixpoint?". Elgot (see, e.g. [14]) replaced partial functions by a setting of 'iterative algebraic theories' in which the recursive fixpoint equation for iteration is required to have a unique solution. The approach in order semantics, due essentially to [23], is to use the least fixpoint of a suitable continuous functional h as given by the Kleene formula $\sup(h^n\perp : n \geq 0)$.

Thus, the Elgot approach requires uniqueness, and the order approach requires a partial ordering. More generally, we shall consider a class $\mathcal{A}$ of 'structures' A, A', ... and a notion of 'homomorphism' $\phi: A \longrightarrow A'$ which is closed under composition.

1 Definition: An $\mathcal{A}$-recursion scheme is a pair (A,H) where A is a structure and the 'recursion information' H is an additional element of structure whose only formal requirement is to induce a corresponding map $h: A \longrightarrow A$ in a specified way. A fixpoint of (A,H) is an a in A with $ha = a$. We fix a set Φ of homomorphisms which is closed under composition, and is such that $h'\phi = \phi h$ for each $\phi: (A,H) \longrightarrow (A',H')$ in Φ.

The reader may note that we may consider Φ to be the morphisms of a suitable category; and that Definition 4 is then standard category theory. The proof of Theorem 5 is then a straightforward exercise in initiality. However, since only a few lines would be saved by their omission, we supply full details of the definition and proof.

In the context of order semantics, A is a partially-ordered set with minimal element $\perp$ and suprema for ascending (denumerable) chains, h = H, ϕ and h both preserve suprema of ascending chains, while homomorphisms ϕ preserve $\perp$ as well. Note that h is not a homomorphism, since it need not preserve $\perp$. Another example will be provided by the abstract recursion schemes introduced in the next section.

2 Observation: Any homomorphism in Φ preserves fixpoints: If $ha = a$, then $h'\phi a = \phi ha = \phi a$. □

The philosophy embodied by other workers has been to provide the A's with enough structure so that any (A,H) has a distinguished fixpoint. To place this in a more general perspective, we introduce the idea of canonical fixpoint which is explicitly

based solely on the homomorphic structure of $\mathcal{A}$.

3 Definition: Relative to Φ, a canonical fixpoint α is an assignment of a fixpoint $\alpha_H = h\alpha_H$ to each $\mathcal{A}$-recursion scheme (A,H) in such a way that for every $\phi: (A,H) \longrightarrow (A',H')$ in Φ we have

$$\phi\alpha_H = \alpha_{H'} .$$

This general theory makes interesting contact with the initial algebra semantics of [19] as follows.

4 Definition: We say that the $\mathcal{A}$-recursion scheme (A_0,H_0) is initial relative to the class Φ if for each $\mathcal{A}$-recursion scheme (A,H) there is one and only one ψ_H in Φ from (A_0,H_0) to (A,H).

5 Canonical Fixpoint Theorem: Let there be an initial recursion scheme (A_0,H_0) relative to the class Φ. If (A_0,H_0) has a unique fixed point $a_0 = h(a_0)$, then $\mathcal{A}$-recursion schemes have a unique canonical fixpoint relative to Φ given by the assignment

$$(A,H) \mapsto \psi_H(a_0).$$

Proof: First note that if $\mathcal{A}$-recursion schemes have any canonical fixpoint α, it must satisfy

$$\alpha_H = \psi_H\alpha_{H_0} = \psi_H(a_0)$$

by 3, since ψ_H is in Φ, and a_0 is the only possible choice for a fixpoint of (A_0,H_0). We have thus established uniqueness. Now each $\psi_H(a_0)$ certainly is a fixpoint of (A,H) by 2. To see that the assignment is canonical, let $\phi: (A,H) \longrightarrow (A',H')$ be in Φ. Then $\phi\psi_H = \psi_{H'}$, by the definition of initiality, and so

$$\psi_{H'}(a_0) = \phi\psi_H(a_0)$$

and hence the assignment $(A,H) \mapsto \psi_H(a_0)$ is indeed canonical. □

We now apply this result to show that the Kleene formula is canonical.

6 Theorem: In the context of ordered semantics as defined above, the least fixpoint $\bigvee(h^n\perp : n \geq 0)$ is the unique canonical fixpoint.

Proof: Let N be the partially-ordered set of natural numbers with adjoined greatest element ∞, and let $s(n) = n+1$, $s(\infty) = \infty$. Then (N,s) has unique fixpoint ∞. Moreover, (N,s) is initial with the unique homomorphism ψ_h: (N,s) $\longrightarrow$ (A,h) defined by

$$\psi_h(n) = \psi_h(s^n 0) = h^n \psi_h(\perp) = h^n \perp$$

while

$$\psi_h(\infty) = \psi_h(\bigvee_{n\geq 0} n) = \bigvee_{n\geq 0} (\psi_h s^n(0)) = \bigvee_{n\geq 0} (h^n \perp : n \geq 0).$$

Hence these recursion schemes have unique canonical fixpoint given by

$$(A,h) \mapsto \psi_h(\infty) = \bigvee_{n\geq 0} (h^n \perp : n \geq 0). \qquad \square$$

7 Metric Spaces: A number of workers [3, 7, 32] have used the Banach contraction theorem instead of the Kleene formula for problems of 'fixpoint semantics' (compare 2.7). This theorem also provides an instance of the canonical fixpoint theorem. Let $\mathcal{A}$ have objects (A,d,h) where (A,d) is a nonempty metric space and h: (A,d) $\longrightarrow$ (A',d') satisfies $d(hx,hy) < Kd(x,y)$ for all x, y and some $K < 1$. Φ can comprise any collection, closed under composition, of maps ϕ: A $\longrightarrow$ A' satisfying $h'\phi = \phi h$. The one element set with its unique metric and endomorphism is the initial object of $\mathcal{A}$ and possesses its only element as unique fixpoint. (The proof uses the well-known fact that each object of $\mathcal{A}$ has a unique fixpoint.)

Eilenberg [personal communication, May 1979] informs us that he had independently noted the canonical properties of the least fixed point for ordered sets and the unique fixpoint for metric spaces under contractions.

7. Abstract Recursion Schemes

We now give the general setting for the examples of pattern-of-calls semantics provided in Section 3. We seek to interpret recursive definitions in some partially-additive monoid, A. The recursive definition is specified by a family of suitable maps $H_n : A^n \rightarrow A$. This generalizes the motivating examples where $H_n(a_1,\ldots,a_n)$ was defined as follows: For each distinct n-substitution path in a recursive call, replace the j^{th} occurrence of a variable by a_j $(1 \leq j \leq n)$, and compose the partial functions along the path; then sum the partial functions for each of the n-substitution paths

(they will be disjoint). Then $h(a)$ is just the sum of the $H_n(a,\ldots,a)$ for $n \geq 0$ (and will thus have only finitely many terms if h is induced by finite flow diagrams). Our general task is to provide a method for providing fixpoints for abstract recursion schemes as made precise by definitions 1 and 4.

---

1 Definition: Let (A,Σ), (A',Σ) be partially-additive monoids. Say that $H: A \longrightarrow A'$ is additive if whenever Σa_n is defined then $\Sigma H(a_n)$ is defined and $\Sigma H(a_n) = H(\Sigma a_n)$.

More generally, say that $H: A^n \longrightarrow A'$ is n-additive if when all but one of the n variables are fixed with arbitrary elements of A, the resulting map $A \longrightarrow A'$ is additive; e.g., $H_n(\Sigma_j a_{1j}, a_2, \ldots, a_n) = \Sigma_j H(a_{1j}, a_2, \ldots, a_n)$. (We consider each constant in A' to be a 0-additive map.) Say that $h: A \longrightarrow A'$ is a power series map if there exist n-additive $H_n: A^n \longrightarrow A'$ such that

$$h(a) = \sum_{n=0}^{\infty} H_n(a,\ldots,a) \qquad (a \in A). \tag{2}$$

(Part of the assertion, of course, is that the sums in (2) are always defined.)

3 Example: In the recursive definition of iteration (1.7), $h(a) = H_0 + H_1(a)$ where $H_1(a) = a \cdot f_1$ is 1-additive = additive, and $H_0 = f_2$ is 0-additive = constant. The additivity of H_1 is based on the more general principle that composing on either side with a fixed partial function preserves both disjointness of families and their unions -- an important aspect of the partially-additive structure of sets and partial functions as a category [1]. Taking $H_n \equiv \perp$ $(n > 1)$ in (2), h is a power series map.

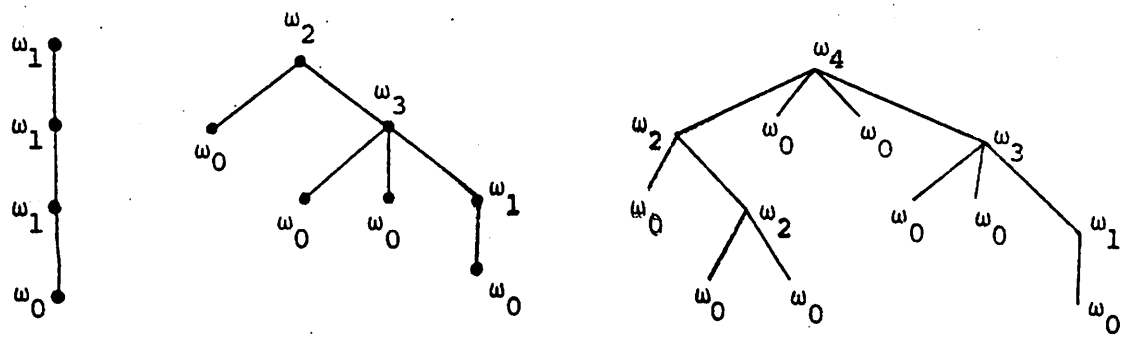
Power series maps generalize 'polynomials', which is the case $H_n = 0$ for $n > N$ in (3). (The reader might conjecture that all power series maps arising from recursive calls are polynomials, but this is not so even for a recursive program for determinant by cofactor expansion; see [26].)

4 Definition: A abstract recursion scheme is (A,Σ,H) where (A,Σ) is a partially-additive monoid and H is a collection of n-additive maps $(H_n: A^n \longrightarrow A \mid n \geq 0)$ subject to the condition that $h(a) = \Sigma_n H_n(a,\ldots,a)$ is defined for all $a \in A$. A fixpoint of (A,Σ,H) is a fixpoint of the power series map h, i.e. an a in A satisfying $h(a) = a$.

5 Remarks: It is clear that, with the disjoint-sum addition of Pfn(D,D), the composition map $(\underline{Pfn}(D,D))^n \longrightarrow \underline{Pfn}(D,D)$, $(f_1,\ldots,f_n) \mapsto f_n \circ \ldots \circ f_1$ is n-additive; it follows quickly that the H_n's of the examples of Section 3 are n-additive. As supported by Section 3, it is our thesis that fixpoint equations in recursive programs always involve power series maps.

We have dropped the "N(H)" notation following 4.2 since unwanted H_n may be set equal to 0.

Now let Ω be the operator domain [9] with exactly one operator symbol, ω_n, for each integer $n \geq 0$. An Ω-tree is then any rooted ordered tree such that a node of outdegree n is labelled by ω_n. Typical such trees are:



Each such tree can then be thought of as the abstract specification of a pattern of calls. Given an abstract recursion scheme, we interpret such a tree by evaluating from leaf to root, replacing ω_n by H_n as we go.

7 Definition: The abstract syntax is the set $\underline{e}$ of all Ω-trees over $\Omega = \{\omega_n : n \geq 0\}$ with each ω_n having arity n; together with the operations $\hat{\omega}_n$ which combine n Ω-trees via a root labelled ω_n:

$$\hat{\omega}_0 = \overset{\bullet}{\omega_0} \ ;$$

$$\hat{\omega}_n[t_1,\ldots,t_n] = \overset{\omega_n}{\underset{t_1 \quad t_n}{/\ldots\backslash}} \ .$$

8 Definition: Given an abstract recursion scheme (A,Σ,H), the interpretation s^H of an Ω-tree s is the element of A obtained by 'running H on s':

Basis Step: $(\overset{\bullet}{\omega_0})^H = H_0$

Induction Step: $(\overset{\omega_n}{\underset{t_1 \quad t_n}{/\ldots\backslash}})^H = H_n(t_1^H,\ldots,t_n^H)$.

$\underline{e}$ is the abstract expression of the result of all possible iterated substitutions or patterns of calls, starting with ω_0 which represents the result of 'replacing all occurrences of variables by $\perp$' thus nulling out all paths save those containing no variables. (Our approach differs from the 'set of Ω-paths' of [27 p. 380]. Successive terms in Manna's expansion need not be disjoint since they result from global substitution in a recursive definition rather than substitution in delimited path

components as specified by the ω_n.) But note that we have done this at a level of abstraction which relieves us of keeping track of any specific path structure.

In the remainder of this section we establish that our axiomatic characterization of partially-additive monoids and abstract recursion schemes does indeed guarantee that the sum $\underline{e}^H = \Sigma(s^H : s \in \underline{e})$ of interpretations of the abstract syntax is well-defined (this is the summability lemma). We then prove, in Section 8, that the assignment $(A,\Sigma,H) \to \underline{e}^H$ is indeed a canonical fixpoint for abstract recursion schemes. We record the following obvious consequence of n-additivity and partition associativity.

9 Observation: If H_n is n-additive and the n sums $\Sigma_k a^n_k$ are defined then

$$H_n(\Sigma_{k_1} a^1_{k_1}, \ldots, \Sigma_{k_n} a^n_{k_n}) = \Sigma_{k_1,\ldots,k_n} H_n(a^1_{k_1},\ldots,a^n_{k_n})$$

(including the assertion that the right-hand sum is defined). □

10 The Summability Lemma: The pattern-of-calls expansion

$$\underline{e}^H = \Sigma(s^H : s \in \underline{e})$$

is well-defined as an element of A.

Proof: By the limit axiom for partially-additive monoids, we may deduce that the sum $\Sigma(s^H : s \in \underline{e})$ is well-defined if we can show that every finite subfamily is summable. Since any subfamily of a summable family is summable by the partition-associativity axiom, this is equivalent to showing that there exists an ascending sequence of finite subsets of $\underline{e}$

$$S_0 \subset S_1 \subset S_2 \subset \ldots$$

such that $\bigcup_{k\geq 0} S_k = \underline{e}$, and each $\Sigma(s^H : s \in S_k)$ is defined. But if we take $S_0 = \{\hat{\omega}_0\}$ and $S_{k+1} = \{\hat{\omega}_n[t_1,\ldots,t_n] : n \geq 0, \text{ each } t_i \text{ is in } S_k\}$ then certainly $S_k \subset S_{k+1}$ for all $k \geq 0$, and $\underline{e} = \bigcup_{k\geq 0} S_k$. It only remains to show that $\Sigma(s^H : s \in S_k)$ exists in A. For $k = 0$, use the unary sum axiom. Given that $\Sigma(s^H : s \in S_k)$ exists, we may deduce that

$$\Sigma(s^H : s \in S_{k+1}) = \Sigma(H_n(t_1^H,\ldots,t_n^H) : n \geq 0, \text{ each } t_i \in S_k)$$

exists by Observation 9. □

The next example places partially-additive monoids in a categorical perspective.

11 Partially-Additive Categories: The axiomatization of partially-additive monoids, together with the notion of abstract recursion scheme will suffice to yield our abstract theory of recursive calls. However, a complete partially-additive semantics must include a notion of composition -- just as we used composition of partial functions in the above examples. The interplay of the partially-additive structure of each $\underline{Pfn}(A,B)$ with the compositions $\underline{Pfn}(A,B) \times \underline{Pfn}(B,C) \longrightarrow \underline{Pfn}(A,C)$, $(f,g) \mapsto g \circ f$ is captured in our axiomatization of partially-additive categories in [1], where we show, inter alia, that this axiomatization is rich enough to support the

semantics of sequential composition, generalized conditionals, and iteration for multi-entry, multi-exit programs. It is proved in Theorem 3.7 of that paper that a given category can be partially-additive in at most one way. In particular, while there are several ways to think of $\underline{\underline{Pfn}}(A,B)$ as a partially-additive monoid, 'mutually-disjoint sums' is the only way consistent with axioms on 'partial functions' as a partially-additive category. We refer the reader to the cited paper for examples of partially-additive categories (and hence abstract recursion schemes) which are not based on partial functions.

12 Language Theory: Our abstract theory of recursive calls includes (a generalization of) the theory of context-free languages as a special case. As stated in 2.6, the theory of formal languages over an alphabet X lives in the partially-additive monoid $L_X = (2^{X^*}, \cup)$ of subsets of X^*. L_X is further enriched by the operation of concatenation $A \cdot B = (w \cdot w' : w \in A, w' \in B)$ which (like composition for partial functions) distributes over addition:

$$A \cdot (\Sigma B_i) = \Sigma (A \cdot B_i); \quad (\Sigma B_i) \cdot A = \Sigma (B_i \cdot A).$$

We then see that a 'sum-of-productions' grammar [8,17] such as

$$S = S + aSb$$

does indeed yield the corresponding formal language $\{a^n b^n : n \geq 0\} \subset \{a,b\}^*$ by sum-of-calls semantics. For an example of a nonlinear grammar, consider the following additive form for a grammar which generates arithmetic expressions with operations * and - over the natural numbers in prefix form.

$$E = *EE + -E + N$$

$$N = D + ND$$

$$D = 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9.$$

The language defined by this nonlinear recursive definition is indeed the fixpoint of the abstract recursion scheme over subsets of $\{*,-,0,1,2,3,4,5,6,7,8,9\}^*$.

8. The Canonical Fixpoint for Abstract Recursion Schemes

In this section we define a special class Φ of homomorphisms of abstract recursion schemes, and construct an abstract recursion scheme (A_0,Σ_0,H_0) initial relative to this class. We shall then see that (A_0,Σ_0,H_0) has a unique fixpoint, and then apply Theorem 6.5 to prove that the pattern-of-calls expansion provides the unique canonical fixpoint for abstract recursion schemes.

1 Definition: An additive map $\phi: (A,\Sigma) \longrightarrow (A',\Sigma')$ is a scheme homomorphism $(A,\Sigma,H) \longrightarrow (A',\Sigma',H')$ if it is, further, an Ω-algebra homomorphism, i.e., $H'_n(\phi a_1,\ldots,\phi a_n) = \phi H_n(a_1,\ldots,a_n)$ for each $n \geq 0$ and each $(a_1,\ldots,a_n) \in A^n$.

To see that such $\phi: (A,\Sigma,H) \longrightarrow (A',\Sigma',H')$ meets the condition on Φ in 6.1, observe that

$$\begin{aligned} \phi ha &= \phi\Sigma H_n(a,\ldots,a) \\ &= \Sigma'\phi H_n(a,\ldots,a) && \text{since } \phi \text{ is a partially-additive morphism} \\ &= \Sigma' H'_n(\phi a,\ldots,\phi a) && \text{since } \phi \text{ is an } \Omega\text{-algebra homomorphism} \\ &= h'\phi a. \end{aligned}$$

Define (A_0,Σ_0,H_0) as follows:

A_0 = the set of all subsets of the abstract syntax $\underline{e}$ of 7.7.

$\Sigma_0(S_i : i \in I)$ is defined when the sets S_i are disjoint -- $i \neq j$ implies $S_i \cap S_j = \emptyset$ -- and is then $\bigcup(S_i : i \in I)$

$H_{0n}: A_0^n \longrightarrow A_0$, $(S_1,\ldots,S_n) \mapsto \{\omega_n[t_1,\ldots,t_n] : t_i \in S_i\}$.

For example,

$$H_{02}(\{\omega_0, \omega_2[\omega_0,\omega_0]\}, \{\omega_1[\omega_0]\}) =$$

$$\{\omega_2[\omega_0,\omega_1[\omega_0]], \omega_2[\omega_2[\omega_0,\omega_0], \omega_1[\omega_0]]\}.$$

We now prove several results about (A_0,Σ_0,H_0):

2 Lemma: (A_0,Σ_0,H_0) is an abstract recursion scheme.

Proof: It is obvious that (A_0,Σ_0) is a partially-additive monoid. We must show H_{0n} is additive in each variable. For notational convenience we show additivity in the first variable. Thus let $S_2,\ldots,S_n \in A_0$ be fixed and consider a summable family $(T_i : i \in I)$. Then an element of $H_{0n}(T_i,S_2,\ldots,S_n)$ has form $\omega[t,s_2,\ldots,s_n]$ with $t \in T_i$, $s_j \in S_j$. Since $T_i \cap T_j = \emptyset$ if $i \neq j$, $H_{0n}(T_i,S_2,\ldots,S_n) \cap H_{0n}(T_j,S_2,\ldots,S_n) = \emptyset$. It is then clear that $H_{0n}(\Sigma_i T_i,S_2,\ldots,S_n) = \Sigma_i H_{0n}(T_i,S_2,\ldots,S_n)$.

Finally we must show that for each $S \in A_0$, $h_0(S) = \Sigma_{n\geq 0} H_{0n}(S,\ldots,S)$ is defined, that is, that $H_{0n}(S,\ldots,S) \cap H_{0m}(S,\ldots,S) = \emptyset$ if $m \neq n$. This is clear since each tree in $H_{0n}(S,\ldots,S)$ has root ω_n. □

The reader may wish to compare the role of free, or Herbrand interpretations in the theory of program schemes [4, 15, 25]. There, the syntax is "concrete" in that it depends on the choice of predicate symbols, etc.

3 Lemma: (A_0,Σ_0,H_0) has a unique fixpoint, namely $\underline{e}$.

Proof: The fixpoint equation is

$$S = h_0(S) = \bigcup_n \{ \overset{\omega_n}{\underset{s_1 \quad \ldots \quad s_n}{/\ \ldots\ \backslash}} : s_i \in S\}$$

and is certainly satisfied by $\underline{e}$. To see that there is no other fixpoint, note that the equation emplies that $\{\omega_0\} \subset S$, and it then follows inductively that any ω-tree is in S, so $S = \underline{e}$. □

4 Lemma: (A_0,Σ_0,H_0) is initial relative to the class of scheme homomorphisms. For each abstract recursion scheme (A,Σ,H) the unique scheme homomorphism $\psi_H: (A_0,\Sigma_0,H_0) \longrightarrow (A,\Sigma,H)$ is defined by

$$\psi_H(S) = \Sigma(s^H : s \in S)$$

for each subset S of $\underline{e}$ (i.e., each S in A_0). We call ψ_H the canonical homomorphism to (A,Σ,H).

Proof: By the summability lemma, 7.5, the sum $\underline{e}^H = \Sigma(s^H : s \in \underline{e})$ is well-defined as an element of A, and so by the partition-associativity axiom of Definition 2.1, the sum $\Sigma(s^H : s \in S)$ is defined for each subset S of $\underline{e}$. Thus $\psi_H(S)$ is well-defined. To show that ψ_H is a scheme homomorphism we must verify that it is both an Ω-algebra homomorphism and a partially-additive morphism.

(i) Let $S_1,\ldots,S_n$ be n subsets of $\underline{e}$. Then

$$\begin{aligned}\psi_H H_{On}(S_1,\ldots,S_n) &= \Sigma((\omega_n[t_1,\ldots,t_n])^H : t_i \in S_i)\\ &= \Sigma(H_n(t_1^H,\ldots,t_n^H) : t_i \in S_i)\\ &= H_n(\Sigma(t_1^H : t_i \in S_1),\ldots,\Sigma(t_n^H : t_n \in S_n))\\ &\quad \text{by n-additivity of } H_n \text{ and 3.13}\\ &= H_n(\psi_H(S_1),\ldots,\psi_H(S_n))\end{aligned}$$

so that ψ_H is an Ω-algebra homomorphism.

(ii) Let the S_i be disjoint subsets of $\underline{e}$ so that $\Sigma_i S_i$ is defined. Then

$$\begin{aligned}\psi_H(\Sigma_i S_i) = \psi_H(\bigcup_i S_i) &= \Sigma(s^H : s \in \bigcup_i S_i)\\ &= \Sigma_i\, \Sigma(s^H : s \in S_i) \quad \text{by partition-associativity, since the } S_i \text{ are disjoint}\\ &= \Sigma_i\, \psi_H(S_i)\end{aligned}$$

so that ψ_H is a partially-additive homomorphism.

It only remains to show that ψ_H is unique. But suppose that ϕ is any scheme homomorphism $(A_0,\Sigma_0,H_0) \longrightarrow (A,\Sigma,H)$. Then S is the sum of its 1-element subsets and

$$\phi(S) = \Sigma(\phi(\{s\}) : s \in S)$$

for each $S \subset \underline{e}$, by partial-additivity, while the fact that ϕ is an Ω-algebra homomorphism tells us that

$$\phi(\{\omega_0\}) = H_0$$

and $\phi(\{\omega_n[t_1,\ldots,t_n]\}) = H_n(\phi(\{t_1\},\ldots,\{t_n\}))$ for any n Ω-trees t_i. But these last two equations together imply that $\phi(\{s\}) = s^H$ for each s in $\underline{e}$, and so $\phi(S) = \Sigma(s^H : s \in S) = \psi_H(S)$. Hence ψ_H is unique. □

We then have our main theorem on abstract recursion schemes:

5 The Canonical Expansion Theorem: The assignment

$$(A,\Sigma,H) \mapsto \underline{e}^H = \Sigma(s^H : s \in \underline{e})$$

of the pattern-of-calls expansion to each abstract recursion scheme provides the unique canonical fixpoint for abstract recursion schemes relative to the class of scheme homomorphisms.

Proof: Theorem 6.5 tells us that if there is an initial abstract recursion scheme (A_0,Σ_0,H_0) -- with ψ_H the unique scheme homomorphism to (A,Σ,H) -- and if this initial scheme has a unique fixpoint a_0, then there is a unique canonical fixpoint, namely that given by $(A,\Sigma,H) \mapsto \psi_H(a_0)$. Applying this to the present circumstances, we have that $a_0 = \underline{e}$ by Lemma 3, while $\psi_H(S) = \Sigma(s^H : s \in S)$ by Lemma 4. Hence

$$(A,\Sigma,H) \mapsto \underline{e}_H = \psi_H(\underline{e}) = \Sigma(s^H : s \in \underline{e})$$

is the unique canonical fixpoint for abstract recursion schemes relative to scheme homomorphisms. □

9. Comparison with Order Semantics

Since the relationship of any new approach to order semantics is of prime importance to many workers, we provide a careful comparison, followed by the demonstration that our pattern-of-calls expansion agrees with the least fixpoint semantics for recursive definitions over partial functions. However, the pattern-of-calls expansion uses a different formula which lends itself to algebraic manipulation, and is defined even when an ordered structure is not available.

The set $\underline{\underline{Pfn}}(D,E)$ of partial functions from the set D to the set E is a partially ordered set under the approximation ordering $f \sqsubseteq g$ if g extends f, that is, for all $x \in D$, if f(x) is defined then g(x) is defined and $g(x) = f(x)$. The following axioms, meaningful for any partially-ordered set, hold in $\underline{\underline{Pfn}}(D,E)$:

A There is a least element $\perp$. (In $\underline{\underline{Pfn}}(D,E)$, $\perp$ is the 'totally undefined function' with empty domain.)

B Each countable ascending chain $f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \ldots$ has a least upper bound $\bigsqcup f_n$. (In $\underline{\underline{Pfn}}(D,E)$, if $f = \bigsqcup f_n$, $dom(f) = \bigcup dom(f_n)$, $f(x) = f_n(x)$ for any n with $x \in dom(f_n)$.)

C Each two elements f, g have a greatest lower bound $f \wedge g$. (In $\underline{\underline{Pfn}}(D,E)$, $dom(f \wedge g) = \{x \mid f(x), g(x)$ defined and $f(x) = g(x)\}$, and $(f \wedge g)(x) = f(x) = g(x)$, on this domain.)

In $\underline{\underline{Pfn}}(D,E)$, the order and partially-additive structures are interrelated in the following ways:

D $f \sqsubseteq g$ if and only if $g = f+h$ for some h.

E A family (f_n) is disjoint if and only if $f_n \wedge f_m = \perp$ whenever $n \neq m$. A disjoint family has a least upper bound Σf_n.

Thus (D) defines a binary relation in any partially-additive monoid whereas (E) (in conjunction with (A) and (C)) is an axiom on any partially-ordered set which attempts to define a partial sum. Note that axioms (A) and (B) define the notion of ω-cpo which has become popular in the order semantics literature, but that we must add the axiom (C) if we are to have a notion of disjointness that allow us to easily compare the order and partially-additive structure.

1 Example: In general, if (A,Σ) is a partially-additive monoid and $\sqsubseteq$ is defined as in (D) then $\sqsubseteq$ (which is obviously reflexive and transitive) may fail to be antisymmetric. To see this, let S be any commutative semigroup, let $A = S \cup \{\perp,\infty\}$ (where $\perp \notin S$, $\infty \notin S$) and define Σ on A by

$$\Sigma a_n = \begin{cases} \infty & \text{if some } a_n = \infty \text{ or if } a_n \neq \perp \text{ infinitely often} \\ s & \text{if no } a_n = \infty,\ \{n : a_n \in S\} \text{ is finite and non-empty,} \\ & \quad s = \Sigma(a_n : a_n \in S) \text{ in } S \\ \perp & \text{if all } a_n = \perp. \end{cases}$$

Then (A,Σ) is a partially-additive monoid (indeed, Σ is totally defined) with additive zero $\perp$. If S is the two-element group $\{0,1\}$ we have $0+1 = 1$, $1+1 = 0$ so that $0 \sqsubseteq 1$, $1 \not\sqsubseteq 0$.

2 Example: In general, a partially-ordered set possessing a summation operation as in (E) need not be a partially-additive monoid under this operation. Indeed if $A = \{\perp,a,b,c,\top\}$ with $\perp \sqsubseteq a,b,c \sqsubseteq \top$ defining $\sqsubseteq$, (a,b,c) is disjoint and $a+b+c = \top$ but $(a+b)+c = \top+c$ is not defined since $\top \wedge c = c \neq \perp$.

Our viewpoint, then, is that partially-ordered sets and partially-additive monoids constitute different types of structure which, however, are closely related by (D) and (E) when specialized to Pfn(D,E). To further this comparison, we now introduce 'power series maps' for partially-additive monoids and show that on Pfn(D,E) 'power series maps are continuous'.

3 Definition: Let $(P,\sqsubseteq)$, $(P',\sqsubseteq)$ be two partially-ordered sets satisfying axioms (A,B). (Following conventional usage, we employ the same symbol $\sqsubseteq$ in the different partially-ordered sets.) A function $h: P \longrightarrow P'$ is continuous if (i) whenever $p_1 \sqsubseteq p_2$, $h(p_1) \sqsubseteq h(p_2)$ and (ii) whenever $p_0 \sqsubseteq p_1 \sqsubseteq p_2 \sqsubseteq \ldots$ is a denumerable ascending chain in P, $h(\bigsqcup p_n) = \bigsqcup h(p_n)$ (noting that by (i), $h(p_0) \sqsubseteq h(p_1) \sqsubseteq h(p_2) \sqsubseteq \ldots$ is necessarily a denumerable ascending chain).

We will prove that on Pfn(D,E) the pattern-of-calls expansion produces the same partial function as the least fixpoint expansion (for arbitrary recursive calls). An important part of the proof is the observation that power series maps are continuous. We now develop this idea.

4 Definition: Let (A,Σ) be a partially-additive monoid and let (a_k) be a sequence in A, $a \in A$. Say that $a \in \lim(a_k)$ if there exist $x_k \in A$ $(k \geq 1)$ such that

5
$$a_k = a_{k-1} + x_k \quad (k \geq 1); \text{ and}$$
$$a = a_0 + \Sigma(x_k : k \geq 1).$$

Say that $h: (A,\Sigma) \longrightarrow (A',\Sigma)$ is continuous if whenever $a \in \lim(a_k)$, $h(a) \in \lim(h(a_k))$. The following result is obvious:

6 Proposition: In Pfn(D,E), $a \in \lim(a_k)$ if and only if (a_k) is an ascending chain with least upper bound a. In particular, the definition of 'continuous map' is not ambiguous in this case. □

7 Proposition: Let $h_n: (A,\Sigma) \longrightarrow (A',\Sigma)$ be continuous and assume that $h(a) = \Sigma h_n(a)$ is defined for all $a \in A$. Then $h: (A,\Sigma) \longrightarrow (A',\Sigma)$ is continuous.

Proof. Let $a \in \lim(a_k)$ in A. For all n, $h_n(a) \in \lim_k h_n(a_k)$ so that there exist $x_{k,n} \in A'$ with

$$h_n(a_k) = h_n(a_{k-1}) + x_{k,n} \qquad (k \geq 1)$$

$$h_n(a) = h_n(a_0) + \sum_{k\geq 1} x_{k,n} .$$

Then
$$\begin{aligned} h(a_k) &= \sum_n h_n(a_k) \\ &= \sum_n (h_n(a_{k-1}) + x_{k,n}) \\ &= \sum_n h_n(a_{k-1}) + (\sum_n x_{k,n}) \qquad \text{(partition-associativity)} \\ &= h(a_{k-1}) + x_k \end{aligned}$$

where $x_k = \sum_n x_{k,n} \qquad (k \geq 1)$.

Further,
$$\begin{aligned} h(a) &= \sum_n h_n(a) \\ &= \sum_n (h_n(a_0) + \sum_{k\geq 1} x_{k,n}) \\ &= (\sum_n h_n(a_0)) + (\sum_n \sum_{k\geq 1} x_{k,n}) \\ &= h(a_0) + \sum_{k\geq 1} \sum_n x_{k,n} \\ &= h(a_0) + \sum_{k\geq 1} x_k \end{aligned}$$

so that $h(a) \in \lim(h_n(a))$ as desired. □

8 Theorem: A power series map $h: (A,\Sigma) \longrightarrow (A',\Sigma)$ is continuous.

Proof. In view of proposition 7 we need only prove that if (A,Σ), (A',Σ) are partially-additive monoids and $H_n: A^n \longrightarrow A'$ is n-additive, then $g(a) \mapsto H_n(a,\ldots,a)$ is continuous. If $n = 0$ this amounts to observing that if $a_k = a$ for all k then $a \in \lim(a_k)$; for set $x_k = 0$ in (5). Now assume $n \geq 1$.

Set $g(a) = H_n(a,\ldots,a)$ and let

$$a_k = a_{k-1} + x_k \qquad (k \geq 1)$$

$$a = a_0 + \sum_{k=1}^{\infty} x_k .$$

We must find y_k with

$$g(a_k) = g(a_{k-1}) + y_k$$

$$g(a) = g(a_0) + \sum_{k=1}^{\infty} y_k .$$

Write $x_0 = a_0$. For $k \geq 1$ define

$$I_k = \{(i_1,\ldots,i_n) : 0 \leq i_j \leq k, \text{ at least one } i_j = k\}$$

$$y_k = \Sigma(H_n(x_{i_1},\ldots,x_{i_n}) : (i_1,\ldots,i_n) \in I_k).$$

Since $a_k = \sum_{n=0}^{k} x_k$ is immediate the existence of y_k follows from observation 6.9 (noting that partition-associativity implies that any subfamily of a summable family is summable).

To see $g(a_k) = g(a_{k-1}) + y_k$, use induction on k. We use observation 6.9 frequently in the balance of the proof.

$$g(a_1) = H_n(x_0+x_1, \ldots, x_0+x_1)$$

$$= H_n(x_0,\ldots,x_0) + \Sigma(H_n(x_{i_1},\ldots,x_{i_n}) : (i_1,\ldots,i_n) \in I_1)$$

$$= g(a_0) + y_1.$$

Now assume $g(a_m) = g(a_{m-1}) + y_m$ for $1 \leq m \leq k$. Then $g(a_k) = g(a_0) + \sum_{n=1}^{k} y_k$. Hence

$$g(a_{k+1}) = H_n\left(\sum_{m=0}^{k+1} x_m, \ldots, \sum_{m=0}^{k+1} x_m\right)$$

$$= g(a_0) + \Sigma(H_n(x_{i_1},\ldots,x_{i_n}) : 0 \leq i_j \leq k+1, \text{ not all } i_j = 0)$$

$$= g(a_0) + \Sigma(H_n(x_{i_1},\ldots,x_{i_n}) : 0 \leq i_j \leq k, \text{ not all } i_j = 0)$$

$$+ \Sigma(H_n(x_{i_1},\ldots,x_{i_n}) : (i_1,\ldots,i_n) \in I_{k+1})$$

$$= (g(a_0) + \sum_{m=0}^{k} y_k) + y_{k+1}$$

$$= g(a_k) + y_{k+1} .$$

Finally, noting $a = \sum_{k=0}^{\infty} x_k$,

$$g(a) = H_n(\sum_{k=0}^{\infty} x_k, \ldots, \sum_{k=0}^{\infty} x_k)$$

$$= g(a_0) + \Sigma(H_n(x_{i_1},\ldots,x_{i_n}) : i_j \geq 0, \text{ not all } i_j = 0)$$

$$= g(a_0) + \sum_{k=1}^{\infty} y_k . \qquad \square$$

Although we will not do so in this paper, limits may be used to classify and study partially-additive monoids. For example, limits are unique in $\underline{\underline{\text{Pfn}}}(D,E)$, but not in example 1 with S the two-element group (the constant sequence $a_k = 1$ has $1, \infty \in \lim(a_k)$ where, respectively, x_k is constantly $\perp$, 0).

We now show that abstract recursion schemes over partial functions admit a least fixpoint interpretation, and that this returns the same partial function as the pattern-of-calls expansion. It is, of course, a desirable feature that the two definitions coincide in the motivating example. However, we have seen that the pattern-of-calls expansion uses a different formula which lends itself to algebraic manipulation, and is defined even when an ordered structure is not available. We showed in [2] that the partially-additive semantics of a partial-function recursive definition coincides with its interpretive semantics, which is known to equal the least fixpoint semantics. The present more general theorem is proved using the canonical fixpoint theorem.

9 Theorem: Let us be given any abstract recursion scheme $(\underline{Pfn}(A,B),\Sigma,H)$ over the partially-additive monoid of partial functions from set A to set B (under union of disjoint functions). Then $h(m) = \Sigma H_n(m,\ldots,m)$ preserves suprema of countable ascending chains. Moreover, the partially-additive and least fixpoint semantics coincide:

$$\Sigma(s^H : S \in \underline{e}) = \mathrm{Sup}(h^n(\perp) : n \geq 0).$$

Proof: The first assertion is immediate from 6 and 8. As in the general setting of Definition 6.1, we must equip the collection $\mathcal{A}$ of abstract recursion schemes of the form $(\underline{Pfn}(A,B),H)$ -- with corresponding $h(m) = \Sigma H_n(m,\ldots,m)$ for each $m \in \underline{Pfn}(A,B)$ -- with a suitable family Φ of morphisms. In fact, we take Φ to be the collection of $\phi: (\underline{Pfn}(A,B),H) \longrightarrow (\underline{Pfn}(A',B'),H')$ which are scheme homomorphisms in the sense of Definition 6.1. Now we re-assess the (A_0,Σ_0,H_0) of Lemma 8.4 above by noting that

(A_0, Σ_0), the subsets of the set $\underline{e}$ of abstract syntactic trees under disjoint union, is isomorphic to $\underline{Pfn}(\underline{e},1)$ (where 1 is a one-element set) in a Σ-preserving way. Equipping $\underline{Pfn}(\underline{e},1)$, then, with the H_0 of Lemma 8.4, we see that $(\underline{Pfn}(\underline{e},1), \Sigma, H_0)$ is initial relative to the present Φ, and that $\Sigma(s^H : s \in \underline{e})$ is still the unique canonical fixpoint. We must show that $Sup(h^n(\perp) : n \geq 0)$ is also a canonical fixpoint. But since additive maps are continuous by 6 and 8, this is immediate from 6.6. □

References

1. Arbib, M.A. and Manes, E. G. Partially-additive categories and flow-diagram semantics. J. Algebra, 62 (1980), 203-227.

2. Arbib, M. A. and Manes, E. G. Partially-additive monoids, graph-growing and the algebraic semantics of recursive calls. Proc. Intl. Workshop on Graph Grammars and their Applications to Computer Science and Biology. Springer-Verlag, Lecture Notes in Computer Science 73 (1979), 127-138

3. Arnold, A. and Nivat, M. The metric space of infinite trees: algebraic and topological properties. Report 323, IRIA Laboria, 78150 Le Chesnay, France (1978).

4. Ashcroft, E., Manna, Z. and Pnueli, A. Decidable properties of monadic functional schemes. J.ACM 20 (1973), 489-499.

5. Backus, J. W. Can programming be liberated from the Von Neumann style? A functional style and its algebra of programs. Comm. ACM 21 (1978), 613-641.

6. Bloom, S. L. Varieties of ordered algebras. J. Comptr. Syst. Sci. 13 (1976), 203-212.

7. Bloom, S. L. Iterative and metric algebraic theories. Preprint, Dept. of Pure and Applied Mathematics, Stevens Institute of Technology, Hoboken, N.J. 07030, USA (1977).

8. Chomsky, N. and Schützenberger, M. The algebraic theory of context-free languages. In Computer Programming and Formal Systems, P. Brafford and D. Hirschberg, Eds., North-Holland (1963), 118-161.

9. Cohn, P. M. Universal Algebra. Harper and Row, 1965.

10. De Bakker, J. W. and Meertens, L. G. On the completeness of the inductive assertion method. J. Comptr. Syst. Sci. 11 (1975), 323-257.

11. De Roever, W. P. Jr. Recursive program schemes: semantics and proof theory. Mathematical Centre Tracts 70, Mathematisch Centrum, Amsterdam, 1976.

12. Dijkstra, E. W. Guarded commands, nondeterminacy and formal derivation of programs. Comm. ACM 18 (1975), 453-457.

13. Eilenberg, S. Automata, Languages and Machines, Vol. A, Academic Press, 1974.

14. Elgot, C. C. Monadic computation and iterative algebraic theories. In Logic Colloquium '73, Studies in Logic 80 (1975), 175-230.

15. Engelfriet, J. Simple Program Schemes and Formal Languages. Springer Lecture Notes in Computer Science 20 (1974).

16. Ginali, S. Iterative algebraic theories, infinite trees, and program schemata. Dissertation, University of Chicago, 1976.

17. Ginsburg, S. and Rice, H. G. Two families of languages related to ALGOL. J. ACM 9 (1962), 350-371.

18. Goguen, J. A., Thatcher, J. W., Wagner, E. G. and Wright, J. B. Some fundamentals of order-algebraic semantics. Lecture Notes in Computer Science 45 (1976), 153-168.

19. Goguen, J. A., Thatcher, J. W., Wagner, E. G. and Wright, J. B. Initial algebra semantics and continuous algebras. J. ACM 24 (1977), 68-95.

20. Kalman, R. E., Falb, P. L. and Arbib, M. A. Topics in Mathematical System Theory. McGraw-Hill, 1969.

21. Kamin, S. Rationalizing many-sorted algebraic theories. IBM Research Report RC7574, Box 218, Yorktown Heights, N.Y. 10598, USA, 1979.

22. Karp, R. M. Some applications of logical syntax to digital computer programming. Dissertation, Harvard University, 1959.

23. Kleene, S. C. Introduction to Metamathematics. D. van Nostrand Co., 1952.

24. Knaster, B. Un théorème sure les fonctions des ensembles. Ann. Soc. Polon Math 6 (1928), 133-134.

25. Luckham, D. C., Park, D.M.R., and Paterson, M. S. On formalized computer programs. J. Comp. Sys. Sci. 4 (1970), 220-249.

26. Manes, E. G. Partially-additive semantics: A progress report. In Fundamentals of Computation Theory, FCT'79 (L. Budach, Ed.), Akademie-Verlag, Berlin (1979), 279-290.

27. Manna, Z. Mathematical Theory of Computation. McGraw-Hill, 1974.

28. Manna, Z., Ness, S. and Vuillemin, J. Inductive methods for proving properties of programs. Comm. ACM 16 (1973), 491-502.

29. Manna, Z. and Shamir, A. The convergence of functions to fixedpoints of recursive definitions. Theoretical Computer Science 6 (1978), 109-141.

30. Milne, R. and Strachey, C. A Theory of Programming Language Semantics. London: Chapman and Hall, 2 volumes, 1976.

31. Rogers, H., Jr. Theory of Recursive Functions and Effective Computability. New York: McGraw-Hill, 1967.

32. Schützenberger, M. P. On a theorem of R. Jungen. Proc. Amer. Math. Soc., 1962.

33. Scott, D. Outline of a mathematical theory of computation. Technical Monograph PRG-2, Oxford University Computing Laboratory, 1970.

34. Scott, D. The lattice of flow diagrams. In Symposium on Semantics of Algorithmic Languages, E. Engeler, Ed., Lecture Notes in Mathematics 188 (1971), 311-366.

35. Scott, D. Data types as lattices. SIAM J. Computing 5 (1976), 523-587.

36. Stoy, J. E. Denotational Semantics. Cambridge, Mass.: MIT Press, 1977.

37. Tarski, A. Lattice-theoretical fixpoint theorem and its applications. Proc. J. Math. 5 (1955), 285-309.

38. Tennent, R. D. The denotational semantics of programming languages. Comm. ACM 19 (1976), 437-453.

39. Tiuryn, J. Fixed-points and algebras with infinitely long expressions, II. Lecture Notes in Computer Science 56, Springer-Verlag (1977), 332-339.

40. Wagner, E. G., Wright, J. B. and Thatcher, J. W. Many-sorted and ordered algebraic theories. IBM Research Report RC7595, Box 218, Yorktown Heights, N.Y. 10598, USA, 1979.

41. Wand, M. Fixed-point constructions in order-enriched categories. Computer Science Department Technical Report 23, Indiana University, Bloomington, Indiana 47401, USA, 1977.

42. Zeiger, H. P. Formal models for some features of programming languages. 1st ACM Symposium on Theory of Computing (1969), 211-215.