RECURSION SCHEMES AND GENERALIZED INTERPRETATIONS*

(Extended Abstract)

Jean H. Gallier
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104 U.S.A.

Abstract

This paper investigates some of the underlying axioms allowing the fixpoint-semantics approach to hold for tree-like recursion schemes. The notions of scheme and interpretation are generalized. The axioms satisfied by "algebraic theories" are shown to be adequate for the definition of the notion of an interpretation. It is also shown that in order to provide the semantics of arbitrary finite recursion schemes, rational algebraic theories are insufficient and it is necessary to introduce a new class of "recursion-closed" algebraic theories. Finally, free recursion-closed algebraic theories are shown to exist.

1. Introduction

The object of this paper is to study the fixpoint-semantics of recursion schemes in the sense of Courcelle and Nivat [8] (without a reserved symbol if then else) under a generalized notion of interpretation (algebraic theories [10,15,26,27]). The main contribution of this paper is to pinpoint some of the properties (axioms) that an interpretation should satisfy in order for the fixpoint-semantics approach to go through. Two such axioms (satisfied by functions) are:

(1) The operations assigned by an interpretation can be composed;

(2) they can be grouped to form vectors of operations (tupling).

We show that the axioms satisfied by an (ordered) algebraic theory are adequate.

An algebraic theory is a domain of objects (arrows) which can be thought of as functions, together with two operations, a composition operation and a "tupling" operation and satisfying some simple axioms (associativity, existence of identities, etc.).

Our investigations proceed in three steps.

(1) By extending slightly the definition of a recursion scheme, we define an operation of substitution of schemes which confers an interesting structure on the class of schemes.

(2) Exploiting a suggestion made in Goguen, Thatcher, Wagner and Wright [15], we define an extended interpretation I as a function $I : \Sigma \to T$ from the alphabet Σ from which the schemes are constructed to an ordered algebraic theory T. Then, with every scheme α is associated a functional α_I which is shown to be monotonic and the mapping which assigns the functional α_I to the scheme α is a homomorphism of algebraic theories, substitution of schemes corresponding to the composition of functionals.

(3) We investigate the minimal requirements on an interpretation I for the functional α_I associated with a scheme α to have a least fixpoint. We show that the "rational algebraic theories" of [15] are insufficient for that purpose and we define a new class of ordered algebraic theories called "recursion-closed" algebraic theories which satisfy the desired condition. It is shown that every "recursion-closed" algebraic theory is rational in the sense of [15], and we prove that for every ranked alphabet Σ there is a free "recursion-closed" algebraic theory RCT_Σ generated by Σ, generalizing results of [15]. The structure of the free "recursion-closed" algebraic theory RCT_Σ generated by Σ can be described explicitly. Indeed, its elements are n-tuples of (usually infinite) trees having the property that a suitable encoding of their set of branches is a deterministic context-free language. This result is similar to a result of Courcelle [6].

One of the features of this paper is that we generalize the notion of an interpretation, taking the notion of an algebraic theory as a key concept. Conventionally, an interpretation is a mapping assigning functions to the symbols of the base alphabet, and since functions can obviously be composed, the role played by composition is obscured. Our more general notion of an interpretation (which includes the standard notion) clarifies the role played by composition and the nature of the axioms that an interpretation should satisfy for the fixpoint approach to hold.

This paper supports the view that "algebraic theories" constitute a unifying framework for studying the semantics of recursive program schemes. Elgot [10,11] and Wagner [28,29] first recognized the importance of algebraic theories for semantics. Following Elgot [10,11,12], Ginali [13], Burstall and Thatcher [5], Goguen, Thatcher, Wagner and Wright [14,15], and Thatcher, Wagner and Wright [26,27] have used algebraic theories in semantic studies. In particular, the semantics of flowchart programs and of monadic recursion schemes is very nicely treated in [27] using the "introduction of variables construction". A brief sketch of the "introduction of variables construction", which is very closely related to our treatment, is also given in [26] for monadic recursion schemes. Related studies of schemes are those of Nivat [20], Courcelle [6,7], Courcelle and Nivat [8] and Guessarian [17]. Recent work of Arnold [1] and Arnold and Nivat [4] attacks the difficult problem of tackling nondeterminism. Nivat and Arnold [21] is noteworthy since it bases its foundations on the concept of complete metric spaces instead of partial orders. We finally point out that there seems to be very close connections between algebraic theories and the "magmoides" of Arnold and Dauchet [2].

2. Preliminaries: Labeled Trees, Algebraic Theories

In order to minimize the preliminaries, we will follow as much as possible the definitions and notations found in Thatcher, Wagner and Wright [26,27] and Goguen, Thatcher, Wagner and Wright [14,15]. We have summarized the key definitions used in this paper in an Appendix. We also warn the reader that our definition of an

algebraic theory is the dual of that of [26,27]. This has the advantage of eliminating a number of confusing reversals.

3. Extended Recursion Schemes

An extended recursion scheme can be described as a system of mutually recursive definitions where the left-hand sides are distinct "undefined function symbols" F_i, and the right-hand sides are Σ-trees possibly labeled with the function symbols F_j constituting the left-hand sides. What is new in this definition is that we allow "many-sorted" trees and that these trees can be infinite.

Example 1:

F(x)<=if tips(x) then 1 else G (F(left(x)) ,F(right(x)))

G(m,n)<=if eq(n,0) then m else succ(G(m,pred(n)))

We have three "sorts": binary trees, nonnegative integers and boolean. The sorts of the functions and predicates involved are as follows: F: tree → int, left, right: tree → tree, tips : tree → bool, G: int.int → int, succ, pred : int → int, eq : int.int → bool, if then else : bool.int.int → int.

Given an S-sorted alphabet Σ, let D(S) be the new alphabet consisting of all pairs (u,s) in S* x S. For every string $\bar{u}$ in $D(S)^+$ with $\bar{u} = (u_1,s_1)\ldots(u_N,s_N)$, let $\Phi_{\bar{u}}$ be the set of "undefined function symbols" $\Phi_{\bar{u}} = \{F_1^{\bar{u}}, \ldots, F_N^{\bar{u}}\}$. Each $F_i^{\bar{u}}$ is a symbol of sort s_i and of arity u_i (corresponding to the i-th symbol (u_i,s_i) in $\bar{u}$).

Definition 3.1

A recursion scheme α using undefined function symbols in the set $\Phi_{\bar{u}}$ (where $\bar{u} = (u_1,s_1) \ldots (u_N,s_N)$) is a mapping assigning a tree α_i of sort s_i to each undefined function symbol $F_i^{\bar{u}}$, and such that the only symbols besides those in Σ which may occur in the tree α_i are the function variables $F_i^{\bar{u}}$ in $\Phi_{\bar{u}}$ and the variables in the set X_{u_i} associated with the arity u_i of $F_i^{\bar{u}}$. Formally, a recursion scheme is defined as a function $\alpha : \Phi_{\bar{u}} \to \underline{\underline{CT}}_{\Sigma \cup \Phi_{\bar{u}}}$, such that each tree $\alpha_i = \alpha(F_i^{\bar{u}})$ belongs to $\underline{\underline{CT}}_{\Sigma \cup \Phi_{\bar{u}}}(u_i,s_i)$.

A scheme using undefined function symbols in $\Phi_{\bar{u}}$ is a scheme of type $\bar{u}$.

Given a scheme α, we obtain a program by assigning an interpretation I to the symbols in Σ. Then, following Scott's approach [22,23,24], we associate a functional α_I to the pair (α,I) and, provided that the functional α_I has a least fixpoint, we define the meaning of the program (α,I) as the least fixpoint of α_I.

4. Generalized Interpretations and Functionals

Definition 4.1

A generalized interpretation is a pair (I,T), where T is an ordered algebraic

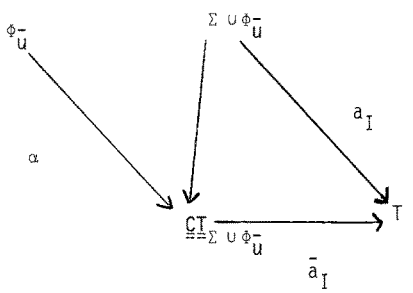
theory and $I : \Sigma \to T$ is a function assigning an arrow $I(f) : u \to s$ in T to each symbol f in $\Sigma_{u,s}$. When the theory T is fixed, we denote an interpretation (I,T) as I.

What is new in this definition is the fact that we do not necessarily assign functions to the symbols in Σ, but possibly more general operations, as long as they satisfy certain axioms (those of an ordered algebraic theory). Consequently, we are focusing on the properties that are required of the elements of an interpretation for the fixpoint-semantics approach to hold. This is also the motivation for introducing the new class of "recursion-closed" algebraic theories.

Given a scheme α of type $\bar{u}$ (with $\bar{u} = (u_1,s_1) \ldots (u_N,s_N)$) and an interpretation (I,T), we will define a functional α_I (really $\alpha_{I,T}$). For that purpose, we note that there is a bijection between the set of N-tuples $(a_1, \ldots, a_N)$ in $T(u_1,s_1) \times \ldots \times T(u_N,s_N)$ and the set of functions $a : \Phi_{\bar{u}} \to T(u_1,s_1) \times \ldots \times T(u_N,s_N)$. We abbreviate the Cartesian product $T(u_1,s_1) \times \ldots \times T(u_N,s_N)$ as $T^{\bar{u}}$. Then, the functional $\alpha_I : T^{\bar{u}} \to T^{\bar{u}}$ associated with the pair $(\alpha,(I,T))$ is defined as follows.

Definition 4.2

For any $a \in T^{\bar{u}}$, the unique function $a : \Phi_{\bar{u}} \to T^{\bar{u}}$ corresponding to a and the interpretation $I : \Sigma \to T$ define a function $a_I : \Sigma \cup \Phi_{\bar{u}} \to T$, and a_I has a unique homomorphic ω-continuous extension $\bar{a}_I$ as in the following diagram.



Abusing the notation slightly, we define $\alpha_I(a)$ the value of the functional α_I at a as $\alpha.\bar{a}_I$.[1]

We generalize slightly the concept of a scheme, and this generalization allows us to show that the class of (generalized) schemes has the structure of an algebraic theory. This allows us to define schemes of "higher types". We extend the definition of a scheme in the following way.

1. We denote composition from left to right, that is, $(f.g)(x) = g(f(x))$.

Definition 4.3

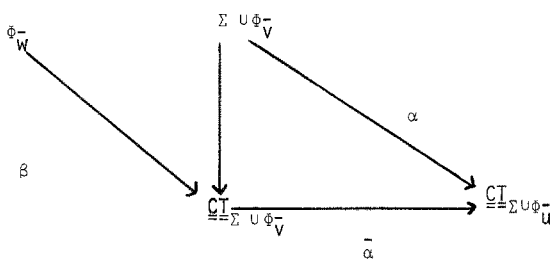
Given any pair of strings $\bar{u}$, $\bar{v}$ in $D(S)^+$ with $\bar{u} = (u_1,s_1) \ldots (u_n,s_n)$ and $\bar{v} = (v_1,r_1)\ldots(v_p,r_p)$, an extended scheme α of type $(\bar{u},\bar{v})$ is a function $\alpha : \Phi_{\bar{v}} \to \underline{\underline{CT}}_{\Sigma \cup \Phi_{\bar{u}}}$, such that each tree $\alpha_j = \alpha(F_j^{\bar{v}})$ is a tree of sort r_j in $\underline{\underline{CT}}_{\Sigma \cup \Phi_{\bar{u}}}(v_j,r_j)$ (if $\bar{u} = \lambda$ or $\bar{v} = \lambda$ then $\Phi_\lambda = \emptyset$).

This definition coincides with the previous one when $\bar{u} = \bar{v}$, in which case we say that α is a closed scheme. Also, for any interpretation (I,T), α and (I,T) define a functional $\alpha_I : T^{\bar{u}} \to T^{\bar{v}}$. The set of all extended schemes is denoted $CPRS_\Sigma$ (for Continuous Polyadic Recursion Schemes). We also have the sets PRS_Σ and $FPRS_\Sigma$ corresponding to the trees in T_Σ and FT_Σ.

The operation of scheme-substitution gives a structure of algebraic theory to the class of schemes $CPRS_\Sigma$. As a consequence, we prove that every closed recursion scheme has a least fixpoint, which can be considered as the unfoldment of the scheme.

Definition 4.4

The substitution of a scheme α into a scheme β consists in a simultaneous substitution of the trees α_i forming α for the undefined function symbols occurring in the trees β_j forming β and performed in a homomorphic fashion. (This is rather different from tree-composition.) The result of substituting a scheme α of type $(\bar{u},\bar{v})$ into a scheme β of type $(\bar{v},\bar{w})$ is the scheme $\alpha \to \beta$ of type $(\bar{u},\bar{w})$ defined in the following diagram as $\alpha \to \beta = \beta.\bar{\alpha}$.



Theorem 1. The set $CPRS_\Sigma$ of extended schemes is an ω-continuous algebraic D(S)-theory under the operation of scheme-substitution.

Note that the complexity of the set of sorts increases. Starting with a set of sorts S, we obtain a D(S)-theory based on the set of sorts D(S). Noticing that the set CF(T) of all ω-continuous functions of the form $H : T^{\bar{u}} \to T^{\bar{v}}$ is an ω-continuous algebraic D(S)-theory under composition, we have the theorem:

Theorem 2. For every interpretation (I,T) where T is an ω-continuous algebraic

S-theory, the mapping $I : CPRS_\Sigma \to CT(T)$ which assigns the functional $\alpha_I : T^{\bar{u}} \to T^{\bar{v}}$ to a scheme α of type $(\bar{u},\bar{v})$ is a homomorphism of ω-continuous algebraic D(S)-theories.

Theorem 2 implies that the functionals α_I are ω-continuous and that scheme-substitution corresponds to the composition of functionals. For every pair of schemes α of type $(\bar{u},\bar{v})$ and β of type $(\bar{v},\bar{w})$, we have $(\alpha \to \beta)_I = \alpha_I . \beta_I$.

5. Fixpoint Solutions

From Theorem 2, every functional $\alpha_I : T^{\bar{u}} \to T^{\bar{v}}$ is ω-continuous. In particular, when α is a closed scheme ($\bar{u} = \bar{v}$), α_I has a least fixpoint $\alpha_I^\nabla = \bigcup_{n\in\omega} \alpha_I^n (\bot)$ (where $\bot$ is the least element of $T^{\bar{u}}$). The operation of scheme-substitution allows us to show that every closed scheme α of type $\bar{u}$ has a least fixpoint, in the sense that there is a scheme α^∇ of type $(\lambda,\bar{u})$ (which is an n-tuple of infinite trees not containing any of the symbols in $\Phi_{\bar{u}}$) which is the least solution of the "scheme equation" $\eta = \eta \to \alpha$. If $\bot$ denotes the scheme of type $\bar{u}$ composed of the trees $\bot$, $\alpha^\nabla = \bigcup_{n\in\omega} \alpha^{(n)}$, where $\alpha^{(0)} = \bot$ and $\alpha^{(n+1)} = \alpha^{(n)} \to \alpha$. α^∇ can be thought of as the unfoldment of the scheme α.

Theorem 2 yields a "Mezei-Wright" [19] type of result, namely that the identity $(\alpha^\nabla)_I = (\alpha_I)^\nabla$ holds: the least fixpoint of the functional α_I is equal to the image of the unfoldment α^∇ of the scheme α under the homomorphism $\bar{I}$.

6. Recursion-Closed Algebraic Theories

Goguen, Thatcher, Wagner and Wright [15] observed that the full power of ω-continuity was not necessary to compute fixpoint solutions and that certain difficulties stemmed from the question of ω-completeness. They introduced the notion of a rational algebraic theory: an ordered algebraic theory T is rational if every closed regular scheme has a fixpoint definition in T, that is, for every interpretation (I,T), the functional α_I has a least fixpoint in T (a scheme is regular if the undefined function symbols occurring in it are of arity λ, which implies that they can only label leaves of the tree). Unfortunately, functionals defined by non-regular schemes may fail to have a least fixpoint in some rational theories. If we choose the "free rational algebraic theory RT_Σ" generated by Σ (see [15]) as an interpretation, we claim that there exist non-regular schemes which do not have a least fixpoint in RT_Σ. This can be derived from a result due to Ginali [13]. Indeed, the elements of RT_Σ are n-tuples of infinite trees, and a suitable encoding of their sets of branches is a regular set [13]. The conclusion results from the fact that there exist recursion schemes whose least fixpoint is encoded by a non-regular language.

In order to resolve this difficulty, we have defined a new class of algebraic theories which we have termed "recursion-closed". An ordered algebraic theory T is "recursion-closed" if, for every finite closed recursion scheme α and for every interpretation (I,T), the functional α_I has a least fixpoint in T.

We now proceed with the formal definition of a "recursion-closed" algebraic theory.

Definition 6.1

Given any ordered algebraic theory T, we define the S-ranked alphabet $T\Omega = \{T\Omega_{u,s}\}_{(u,s) \in S^* \times S}$ where $T\Omega_{u,s}$ is the set of symbols $\{\hat{\phi} \mid \phi \in T(u,s)\}$ in one to one correspondance with the set of arrows in T(u,s). Every symbol $\hat{\phi}$ is a name for the arrow $\phi : u \to s$ in T(u,s). We also define the interpretation $TI : T\Omega \to T$ such that $TI(\hat{\phi}) = \phi$, that is, TI is the function assigning to each name the arrow it represents.

The reason for defining $T\Omega$ and TI is the following. For any arbitrary S-ranked alphabet Σ, any arbitrary finite recursion scheme α over Σ and any arbitrary interpretation $I : \Sigma \to T$, there is a recursion scheme over $T\Omega$ denoted $T\alpha$ such that the functionals α_I and $(T\alpha)_{TI}$ are identical. Indeed, the scheme $T\alpha$ is the scheme obtained by renaming every symbol $f \in \Sigma_{u,s}$ with the symbol $\hat{I(f)}$ corresponding to the arrow I(f) assigned to f by I. This property yields immediately the following Lemma.

Lemma 3

Given any algebraic theory T, for every finite closed recursion scheme α (over an arbitrary S-ranked alphabet Σ) and for every interpretation $I : \Sigma \to T$, the functional α_I has a least fixpoint in T if and only if for every finite closed recursion scheme β over $T\Omega$, the functional β_{TI} (under the fixed interpretation TI) has a least fixpoint.

Noticing that for ω-continuous interpretations (I,T) the least fixpoint of a functional of the form α_I (where α is a finite closed scheme) is given by the identity $(\alpha_I)^\nabla = \bigcup \alpha_I^n(\perp)$ and the fact that in any ordered algebraic theory the identity $\alpha_I^n(\perp) = (\alpha^{(n)})_I$ holds (only monotonicity is needed), we see that the ω-chain $((\alpha^{(n)})_I)_{n\in\omega}$ has a least upper bound. We will require that in a recursion-closed algebraic theory, for every finite closed scheme α over $T\Omega$, the ω-chain $((\alpha^{(n)})_{TI})_{n\in\omega}$ has at least upper bound in T. We actually need the following slightly stronger conditions in order to prove the existence of free recursion-closed algebraic theories.

Definition 6.2

An ordered algebraic S-theory T is recursion-closed if the following conditions hold.

(1) (Completeness) For all finite schemes α and β over $T\Omega$, with β a closed scheme of type $\bar{u} = (w_1,s_1) \ldots (w_n,s_n)$ and α a (not necessarily closed) scheme of type $(\bar{u},\bar{v})$ where $\bar{v}$ is of the special form $(u,v_1) \ldots (u,v_p)$, the ω-chain $((\beta^{(i)} \to \alpha)_{TI})_{i\in\omega}$ has a least upper bound in $T(u,v)$ denoted $(\beta^\nabla \to \alpha)_{TI}$ (with $v = v_1 \ldots v_p$).

(2) (Right continuity) For all α and β as in (1), for all ϕ: $w \to u$ in T, we have $\phi \circ (\bigcup (\beta^{(i)} \to \alpha)_{TI}) = \bigcup (\phi \circ (\beta^{(i)} \to \alpha)_{TI})$.

(3) (Left continuity) For all α and β as in (1), for all ϕ: $v \to w$ in T, we have

$(\bigcup(\beta^{(i)} \to \alpha)_{TI}) \circ \phi = \bigcup((\beta^{(i)} \to \alpha)_{TI} \circ \phi)$.

It should be noted that for all $i\varepsilon\omega$, $(\beta^{(i)} \to \alpha)_{TI}$ is an element of $T(u,v)$, because the special form of $\bar{v}$ implies that $T^{\bar{v}} = T(u,v)$, and therefore the above compositions are meaningful.

It can be easily shown that every recursion-closed algebraic theory is rational in the sense of [15] and it is obvious that every ω-continuous algebraic theory is recursion-closed. We will also need the concept of a homomorphism of recursion-closed algebraic theories.

Definition 6.3

A homomorphism $h: T_1 \to T_2$ between two recursion-closed algebraic theories T_1 and T_2 is a homomorphism of ordered theories such that for all pairs of schemes α and β as in the previous definition we have the identity, $h(\bigcup(\beta^{(i)} \to \alpha)_{TI}) = \bigcup h((\beta^{(i)} \to \alpha)_{TI})$.

The definition of a recursion-closed algebraic theory also allows us to provide the semantics of recursion schemes having a "main procedure".

Definition 6.4

A recursion scheme with a main procedure is a pair (α,β) where β is a finite closed scheme of type $\bar{u} = (w_1,s_1) \ldots (w_n,s_n)$ and α is a finite scheme of type $(\bar{u},(u,v))$ consisting of a unique tree. The scheme α is the main procedure and it contains procedure calls to the procedure names $F_i^{\bar{u}}$ defined by the "procedure declaration" β.

Note that a scheme with main procedure corresponds to the special case where in the pair (α,β) of the definition 6.2, the scheme α consists of a single tree.

The semantics of a scheme with main procedure (α,β) under an interpretation I are defined as $(\beta^{\nabla} \to \alpha)_I = \bigcup(\beta^{(i)} \to \alpha)_I$, which is defined for any recursion-closed interpretation T.

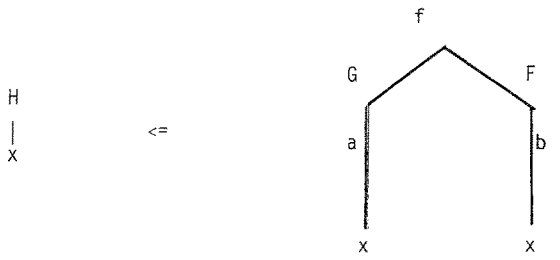
We also prove that the set of n-tuples of trees of the form $\beta^{\nabla} \to \alpha$ where α and β are schemes as in definition 6.2, form the free recursion-closed algebraic theory generated by the S-ranked alphabet Σ. We define the free recursion-closed algebraic theory $\underline{\underline{RCT}}_\Sigma$ in the following way.

Definition 6.5

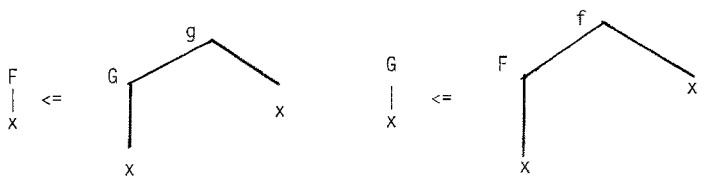
We define the subset $\underline{\underline{RCT}}_\Sigma$ of $\underline{\underline{CT}}_\Sigma$ as the set of all p-tuples of trees of the form $\beta^{\nabla} \to \alpha$, where β is a finite closed scheme of type $\bar{u} = (w_1,s_1) \ldots (w_n,s_n)$ and α is a finite scheme of type $(\bar{u},\bar{v})$ with $\bar{v} = (u,v_1) \ldots (u,v_p)$.

Example 2: A scheme with main procedure (α,β) and the unfoldment tree $\beta^{\nabla} \to \alpha$.

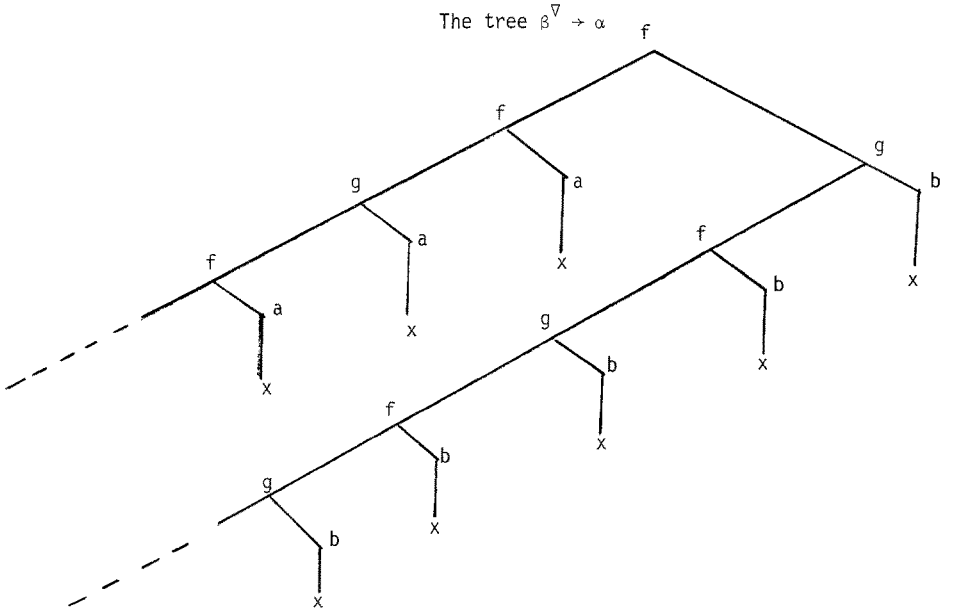
Scheme α (main procedure)



Scheme β (Prodecure declaration)



The tree $\beta^{\nabla} \to \alpha$



Note that β^{∇} is a n-tuple of trees without undefined function symbols and that $\beta^{\nabla} \to \alpha$ is a p-tuple of trees without undefined function symbols and belongs to $\underline{\underline{CT}}_{\Sigma}(u,v)$.

Theorem 4

$\underline{\underline{RCT}}_{\Sigma}$ is the free recursion-closed algebraic theory generated by Σ. That is, for every recursion-closed algebraic theory T and for every function $h : \Sigma \to T$, there exists a unique homomorphism $\bar{h} : \underline{\underline{RCT}}_{\Sigma} \to T$ extending h, and such that for every pair of schemes (α,β) as in definition 6.5, we have $\bar{h}(\beta^{\nabla} \to \alpha) = \cup \bar{h}(\beta^{(i)} \to \alpha)$.

The proof consists in showing that $\underline{\underline{RCT}}_{\Sigma}$ is closed under composition, tupling, that it is recursion closed, and that it has the unique extension property stated in Theorem 4. The verification that $\underline{\underline{RCT}}_{\Sigma}$ is closed under the desired operations is accomplished by constructing appropriate schemes and using properties of fixpoints.

Following the terminology of [15], we can say that $\underline{\underline{RCT}}_{\Sigma}$ consists of the "behaviors" of recursion schemes defined by pairs (α,β) as in definition 6.5.

Given any program $((\alpha,\beta),I)$ where (α,β) is a scheme with main procedure and $I : \Sigma \to T$ is an interpretation with T a recursion-closed algebraic theory, the unique homomorphism $\bar{I}$: $\underline{\underline{RCT}}_{\Sigma} \to T$ extending I gives a fixpoint semantics $(\alpha,\beta)_I = \bar{I}(\beta^{\nabla} \to \alpha) = \cup (\beta^{(i)} \to \alpha)_I$ to the program $((\alpha,\beta),I)$. This implies that two pairs of schemes (α_1,β_1) and (α_2,β_2) are equivalent in all recursion-closed interpretations, if and only if they are "tree equivalent", that is, if the trees $\beta_1^{\nabla} \to \alpha_1$ and $\beta_2^{\nabla} \to \alpha_2$ are identical. The "Mezei-Wright" result also holds, since for every finite closed recursion scheme α we have, $(\alpha_1)^{\nabla} = \cup \alpha_I^n(\perp) = \cup \alpha_I^{(n)} = (\alpha^{\nabla})_I$ from Theorem 4.

We have also proved that there exists an encoding of the set of branches of the tree $\beta^{\nabla} \to \alpha$ which consists of a finite set of deterministic context-free languages. This is analogous to a result of Courcelle [6]. However, our proof is more direct and uses a different encoding due to Ginali [13]. Also, we construct a DPDA whereas Courcelle constructs a strict deterministic grammar. This last result will appear elsewhere.

Acknowledgments:

I would like to thank Professor R.V. Book for many helpful suggestions. It is also a pleasure to thank Professor J.A. Goguen for many illuminating discussions and Dr. J.W. Thatcher for his encouragement.

Appendix: Review of definitions: labeled trees, algebraic theories.

Sorts (or types). By a set of sorts (or types), we understand a set S of data types in some programming language. For example, S = {integer, real, boolean, character} is a set of sorts.

S-ranked alphabet. An s-ranked alphabet Σ is a family $(\Sigma_{u,s})_{(u,s)\in S^*\times S}$ of sets $\Sigma_{u,s}$ indexed by the pairs (u,s) in S* x S. Intuitively, if $u = u_1 \ldots u_n$, each symbol f in $\Sigma_{u,s}$ represents an operation taking n arguments each of sort u_i and yielding an element of sort s. Symbols in $\Sigma_{\lambda,s}$ are called constants of sort s. We say that a symbol f in $\Sigma_{u,s}$ is of sort s and has artiy u. In the rest of this paper, we will assume that a special symbol denoted $\perp$ is adjoined to every S-ranked alphabet Σ ($\perp$ is of arity λ).[1]

Σ-trees. A Σ-tree t is a (finite branching, ordered, possibly infinite) tree whose nodes are labeled with symbols from an S-ranked alphabet Σ in a way which is consistent with the sorts and arities of the symbols in Σ. By this, we mean that for any node v in the tree having exactly n successors $v_1, \ldots, v_n$, the symbol f labeling v must have some arity $u_1 \ldots u_n$ and each symbol f_i labeling v_i must be of sort u_i. In particular, the leaves of the tree are labeled with constants.

A tree whose root is labeled with a symbol of sort s is called a tree of sort s. The set of all trees of sort s is denoted CT^s_Σ and the set of all trees is denoted CT_Σ. A tree is total if the label $\perp$ (of arity λ) does not occur in the tree and otherwise it is partial. The set of total finite trees is denoted T_Σ and the set of partial and total finite trees is denoted FT_Σ. The tree consisting of a unique node labeled $\perp$ is also denoted $\perp$. There is a partial ordering $\leq$ defined on CT_Σ (and FT_Σ) as follows. For every pair of trees $t_1, t_2 \in CT_\Sigma$, the relation $t_1 \leq t_2$ holds if the unlabeled tree t_1 is a subtree of the unlabeled tree t_2 and for every node w in t_1 not labeled with $\perp$, the corresponding node in t_2 has the same label.

Tree-composition. The relevant operation here is that of tree-composition. We introduce for every string $u \in S^*$ the set of variables $X_u = \{x^u_1, \ldots, x^u_n\}$[2] (with $X_\lambda = \emptyset$). The variables x^u_i are used as markers indicating the leaves where the substitution operation takes place. Given a tree t in $CT_{\Sigma \cup X_u}$ and an n-tuple $(t_1, \ldots, t_n)$ of trees in CT_Σ with each tree t_i a tree of sort u_i, the result of composing $(t_1, \ldots, t_n)$ with t denoted $(t_1, \ldots, t_n) \circ t$ is the tree obtained by substituting the tree t_i for each leaf labeled x^u_i in t.

Algebraic theories. We explain the notion of an algebraic theory based on a set of sorts, S (an S-theory). Such a structure T is a set of vectors of operations ϕ, where each vector ϕ has a finite number of "scalar" components $\phi_1, \ldots, \phi_p$ (for some $p \geq 1$), each component ϕ_i having some common "input sort" $u \in S^*$ and some "output sort" $v_i \in S$ (not necessarily the same for each ϕ_i). Every vector ϕ is considered as an "arrow" with some source u and some target $v = v_1 \ldots v_p$ and is usually denoted

---

1. The empty string is denoted λ.

2. $n = |u|$.

$\phi : u \to v$. The components ϕ_i are denoted $\phi_i : u \to v_i$. For technical reasons, there is a unique arrow denoted $0_u : u \to \lambda$ for every $u \in S^*$. In addition, there exists a composition operation o, an operation [ ] called tupling which allows the creation of vector arrows from scalar arrows, and projection arrows $x_i^u : u \to u_i$ which extract the scalar components of vector arrows. More precisely, given any two arrows $\phi : u \to v$ and $\psi : v \to w$ where the source of ψ is the target of ϕ, $\phi \circ \psi : u \to w$ is the the arrow obtained by composing ϕ and ψ. The composition operation o is associative and has identities I_u for all $u \in S^*$. Also, for any p scalar arrows $\phi_i : u \to v_i$ ($u \in S^*$, $v_i \in S$), $[\phi_1, \ldots, \phi_p] : u \to v$ (with $v = v_1 \ldots v_p$) is the arrow from u to v having the ϕ_i as components, and for any vector arrow $\phi : u \to v$ ($v = v_1 \ldots v_p$), $\phi \circ x_i^v = \phi_i$ is the i-th component of ϕ. In summary, the following two identifies hold: $[\phi_1, \ldots, \phi_p] \circ x_i^v = \phi_i$ and $[\phi \circ x_1^v, \ldots, \phi \circ x_p^v] = \phi$. (In an axiomatic definition, these identities are taken as axioms.) The set of all arrows from u to v ($u, v \in S^*$) is denoted $T(u,v)$.

We will actually be interested in algebraic theories equipped with a partial ordering.

Ordered algebraic theories. An algebraic theory T is ordered if each set $T(u,v)$ is partially ordered and has a least element $\perp_{u,v}$, and composition and tupling are monotonic. It is required that for all $u, v \in S^*$, $0_u \circ \perp_{\lambda,v} = \perp_{u,v}$. This implies that for all $\phi : u \to v$, we have $\phi \circ \perp_{v,w} = \perp_{u,w}$.

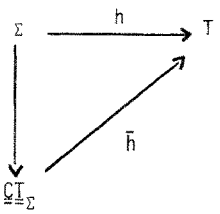
ω-continuous algebraic theories. An ordered algebraic theory T is ω-continuous if each $T(u,v)$ is ω-complete and composition is ω-continuous. Continuity of tupling follows from the other axioms.

There is an obvious notion of a homomorphism of algebraic theories. If T_1 and T_2 are algebraic theories, a homomorphism $h : T_1 \to T_2$ maps every arrow $\phi : u \to v$ in T_1 onto an arrow $h(\phi) : u \to v$ in T_2 and preserves composition, tupling, identities and projections. In addition, for ordered algebraic theories, h is monotonic on each $T(u,v)$ and preserves least elements, and for ω-continuous theories, h is ω-continuous on each $T(u,v)$.

One of the reasons for being interested in algebraic theories comes from the fact that "free algebraic theories generated by an S-ranked alphabet" exist, and that they consist of trees. Furthermore, free algebraic theories are characterized by a universal extension property which proves to be a very useful tool, as we shall see in the next section. Without giving details, the set of n-tuples of Σ-trees can be made into the free ω-continuous algebraic theory $\underline{\underline{CT}}_\Sigma$, and similarly for the set of n-tuples of partial and total finite trees and the set of n-tuples of total finite trees, yielding the free ordered algebraic theory $\underline{\underline{FT}}_\Sigma$ and the free algebraic theory $\underline{T}_\Sigma$. (In the rigorous definition, we actually deal with trees possibly labeled with variables and with tuples of trees of the form $(t_1, \ldots, t_p)$, where each t_i is a tree of sort v_i in $CT_{\Sigma \cup X_u}$.) The composition operation is tree-composition extended to tuples of trees.

The following theorem expresses the "freeness" of the algebraic theory $\underline{\underline{CT}}_\Sigma$ (and we have similar theorems for $\underline{\underline{FT}}_\Sigma$ and $\underline{T}_\Sigma$).

Theorem [15,16,26]. For every ω-continuous algebraic theory T, for every function $h : \Sigma \to T$ assigning an arrow $h(f) : u \to s$ to each symbol $f \in \Sigma_{u,s}$, there exists a unique homomorphism of ω-continuous algebraic theories $\bar{h}$ extending h as in the following diagram:



REFERENCES

1. Arnold, A., Systemes d'equations dans le magmoide. Ensembles Rationels et algebriques d'arbres, These d'Etat, Universite de Lille, (March 1977).

2. Arnold, A. and Dauchet, M., Theorie des Magmoides, Publications du Laboratoire de Calcul, Universite de Lille I, (January 1977).

3. Arnold, A. and Nivat, M., Nondeterministic Recursive Program Schemes. IRIA Technical Report No. 262, Domaine de Voluceau, Le Chesnay, France (November 1977).

4. Arnold, A. and Nivat, M., Algebraic semantics of nondeterministic recursive program schemes, Technical Report no. 78-4, Universite Paris VII, (Feb. 1978).

5. Burstall, R.M., and Thatcher, J.W., The algebraic theory of recursive program schemes, Symposium on Category Theory Applied to Computation and Control. Lecture Notes in Computer Science, Vol. 25, Springer-Verlag, New York, 1975, 126-131.

6. Courcelle, B., A Representation of trees by Languages, Part I and Part II, Theoretical Computer Science. Part I: Vol. 6, pp. 255-279; Part II: Vol. 7, pp. 25-55, (1978).

7. Courcelle, B., On the Definition of Classes of Interpretations, IRIA Technical Report No. 236, Domaine de Voluceau, Le Chesnay, France (May 1977).

8. Courcelle, B. and Nivat, M., Algebraic families of interpretations, Proc. 17th IEEE Symp. on Foundations of Comp. Sci., Houston, Texas (October 1976), 137-146.

9. Eilenberg, S. and Wright, J.B., Automata in general algebras, Inf. and Cont. 11 (1967), 452-470.

10. Elgot, C.C., Monadic computation and iterative algebraic theories. In H.E. Rose and J.C. Shepherdson (Eds.), Logic Colloquium '73, Studies in Logic, Vol. 80, North-Holland, Amsterdam, 1975, 175-230.

11. Elgot, C.C., Algebraic Theories and Program Schemes. Symposium on Semantics of Algorithmic Languages, (Ed. E. Engeler), Springer-Verlag (1971), 71-88.

12. Elgot, C.C., Structured programming with and without GOTO statements. IBM Research Report, RC-5626 (1975); IEEE Transactions on Software Engineering SE1 (1976), 41-53.

13. Ginali, S., Iterative Algebraic Theories, Infinite Trees and Program Schemata. Ph.D. Thesis, Department of Mathematics, University of Chicago, Chicago, Illinois, June 1976.

14. Goguen, J.A., Thatcher, J.W., Wagner, E.G., and Wright, J.B. An Introduction to Categories, Algebraic Theories and Algebras. IBM Report RC-5369, Yorktown Heights, New York (1975).

15. Goguen, J.A., Thatcher, J.W., Wagner, E.G., and Wright, J.B. Rational algebraic theories and fixed-point solutions. Proc. 17th IEEE Symp. on Foundations of Comp. Sci., Houston, Texas (October 1976). 147-158.

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

17. Guessarian, I. Schemas de Programmes Recursif Polyadiques: Equivalence Semantiques et Classes d'Interpretations. These d'Etat, Universite de Paris VII, 1975.

18. Manes, E.G. Algebraic Theories. Graduate Texts in Mathematics, Vol. 26, Springer-Verlag, New York, 1976.

19. Mezei, J. and Wright, J.B. Algebraic automata and context-free sets, Inf. and Cont. 11 (1967), 3-29.

20. Nivat, M. On the interpretation of recursive polyadic program schemes, Symposia Mathematica, Vol. 15, Academic Press, New York, 1975, 255-281.

21. Nivat, M. and Arnold, A., Calculs infinis, interpretations metriques et plus grands points fixes, Technical Report no. 78-19, Universite Paris VII, (May 1978).

22. Scott, D. Outline of a Mathematical Theory of Computation. Technical Monograph PRG-2, Oxford University Computing Laboratory, Programming Research Group (1970).

23. Scott, D. The Lattice of Flow Diagrams. Technical Monograph PRG-3, Oxford University Computing Laboratory, Programming Research Group (1971).

24. Scott, D. Continuous Lattices. Technical Monograph PRG-7, Oxford University Computing Laboratory, Programming Research Group (1971).

25. Scott, D. Data types as lattices, SIAM J. Comp. 5 (1976), 522-587.

26. Thatcher, J.W., Wagner, E.G. and Wright, J.B., Programming languages as mathematical objects, to appear in Mathematical Foundations of Computer Science, '78.

27. Thatcher, J.W., Wagner, E.G. and Wright, J.B., Free continuous theories, Technical Report RC 6906, IBM T.J. Watson Research Center, Yorktown Heights, New York, (December 1977).

28. Wagner, E.G. Languages for defining sets in arbitrary algebras: Proceedings, 12th IEEE Symposium on Switching and Automata Theory, East Lansing, Michigan (1971).

29. Wagner, E.G. An algebraic theory of recursive definitions and recursive languages. Proceedings, 3rd Annual ACM Symposium on Theory of Computing, Shaker Heights, Ohio (1971).

---

* This research has been partially supported by the National Science Foundation under Grant #MCS77-11360.