# EFFICIENT AXIOMS FOR ALGEBRA SEMANTICS

Mitchell Wand

Computer Science Department

Indiana University

Bloomington, Indiana 47401

TECHNICAL REPORT NO. 42

EFFICIENT AXIOMS FOR ALGEBRA SEMANTICS

MITCHELL WAND

NOVEMBER, 1975

Research Reported herein was supported in part by the National Science Foundation under grant number DCR75-06678.

# Efficient Axioms for Algebra Semantics

Mitchell Wand

Computer Science Department

Indiana University

Bloomington, Indiana 47401

Abstract

Algebra semantics is the specification of semantic models of computation by sets of algebraic identities. Sets of identities have a convenient, well-known model theory which can characterize the class of implementations of a given model and which can construct data structures for a canonical implementation of the corresponding formal system without recourse to lattice-theoretic or other external devices. In this paper we will consider how the control structures for a canonical implementation may be deduced from the axioms.

## 1. Introduction

Let us consider a simple example of how first order identities may be used as a defining language. Reynolds [ 6 ] provides a series of interpreters for a simple language with recursion. Each function in his interpreters consists of a conditional expression. Each test turns out to be a test for membership in a subtype, so the order of the tests in immaterial. The archetypical situation in the interpreter is:

$$\underline{\text{eval}} = \lambda(r,e)\ldots$$
$$\text{appl?}(r) \rightarrow \underline{\text{apply}}(\underline{\text{eval}}(\text{opr}(r),e),\underline{\text{eval}}(\text{opnd}(r),e)) \qquad \text{II.5}$$

If appl?(r) is true, then r must be of the form $\underline{\text{mk-appl}}[x_1,x_2]$ where $x_1$ and $x_2$ are of type EXP; then $\text{opr}(r) = x_1$ , $\text{opnd}(r) = x_2$ . We deduce that for all $x_1,x_2 \in \text{EXP}$ , $e \in \text{ENV}$ ,

$$\underline{\text{eval}}[\underline{\text{mk-appl}}[x_1,x_2],e] = \underline{\text{apply}}[\underline{\text{eval}}[x_1,e],\underline{\text{eval}}[x_2,e]] \qquad (*)$$

Equation (*) is nothing more than an axiom relating eval , apply , and mk-appl . Scott and Strachey [8] call such axioms semantic equations and use them to construct lattice-theoretic semantics. Goguen et al. [1] interpret (*) as saying that eval is the unique homomorphism from the "initial algebra" with mk-appl as a binary operator to another algebra A with the binary operator apply , the environment parameter e being eliminated by a clever choice of carrier for the algebra A .

We adopt the following interpretation for axiom (*) : A model for axiom (*) is a set A with three binary operations together satisfying (*) . So long as we restrict ourselves to first-order identities like (*) , the theory of universal algebras gives a general construction for such models.

By allowing any set of identities, we can model systems where the conventional technique of metacircular interpretation sheds less light than usual. One such system is the model of Hewitt [2]. In this semantic model of computation, the basic units, called actors, have the capability to send and receive messages in rather sophisticated ways. While they provide an attractive model of the macroscopic behavior of complex control and data flow regimes, there is some doubt as to whether such complex beasts exist in any reasonable ontology. Algebra semantics can provide a non-metacircular definition for actor-like objects.

Although we will present some brief examples of algebra semantics this paper is not primarily concerned with the programming implications of algebra semantics (q.v. [9]). It is, rather, concerned with a technical difficulty, which may be stated as follows: The original formulation of line II.5 has a fairly direct interpretation in an operational semantics, while axiom (*) does not. If we attempt to interpret (*) as some kind of rewriting system, then further difficulties arise: since identities are symmetric, we might want to use (*) from right to left. In this paper we will show how to interpret a set of identities as a rewriting system, and give sufficient conditions under which the system is unidirectional and close to deterministic.

In Section 2 we will give some algebraic preliminaries about theories and presentations. In Section 3 we will give a formal definition of algebra semantics and give some short examples. In Section 4 we will tackle the technical issues.

## 2. Algebra Preliminaries

We presume familiarity with the concepts of category, functor, and algebraic theory [3,4,5]. Except for the definition of actor theory below, this section sets forth notational conventions for standard material.

If $C$ is a category, $C(a,b)$ is the set of arrows or morphisms from object $a$ to object $b$. If $f \in C(a,b)$ and $g \in C(b,c)$, their composition, a member of $C(a,c)$, is denoted $g.f$. If $f \in C(a,b)$ then $dom(f) = a$ and $cod(f) = b$.

Let $\omega$ denote the nonnegative numbers $0,1,2,\ldots$. A ranked set is a map $\Omega:S \to \omega$ for some set $S$. If $s \in S$, we say $\Omega s$ is the rank of $s$. Alternatively, if $\Omega s = n$, we say $s$ is an n-ary member of $S$. When no ambiguity results, we will write $\Omega$ for $S$: e.g., if $s \in \Omega$, then $\Omega s \in \omega$. We may specify $\Omega$ by its graph: e.g., $\Omega = \{(+,2),(e,1)\}$.

An algebraic theory is a category $T$ whose objects are the non-negative numbers $0,1,2,\ldots$ and in which the object $n$ is the categorical product of the object $1$ taken $n$ times. If $T$ is a theory, and $f_1,..,f_n \in T(k,1)$, then the product morphism in $T(k,n)$ is denoted $[f_1,..,f_n]$. We write $e_i$ for the projection morphisms. If $\Omega$ is a ranked set, then the free theory $T_\Omega$ may be constructed by standard methods. If $T_\Omega$ is a free theory, the ranked set $Pairs(\Omega)$ is

$$\{((f,f'),n) \mid f,f' \in T_\Omega(n,1)\}$$

An $\Omega$-identity is an element of $Pairs(\Omega)$. If $\Delta \subseteq Pairs(\Omega)$ we will often write $(f,f') \in \Delta$ for $((f,f'),n) \in \Delta$. A theory-functor is a product-preserving functor between theories.

If $\Delta$ is a set of identities, we construct a formal system $E_\Delta$, whose formal objects are pairs $(f,f')$ such that $f,f' \in T_\Omega(n,m)$ for some $n$ and $m$. We denote this formal object $(f,f'):n \to m$ in order to make $n$ and $m$ explicit. The system $E_\Delta$ is defined as follows:

Axioms: If $((f,f'),n) \in \Delta$, then $\vdash(f,f'):n \to 1$ $E\Delta$

For any $f \in T_\Omega(n,m)$, $\vdash(f,f):n \to m$ ER

Rules: $$\frac{(f,g):n \to m}{(g,f):n \to m}\ ES \qquad \frac{(f,g):n \to m \quad (g,h):n \to m}{(f,h):n \to m}\ ET$$

$$\frac{(g,g):m \to k \quad (f,f'):n \to m \quad (h,h):p \to n}{(g.f.h,g.f'.h):p \to k}\ EC$$

$$\frac{(f_1,f'_1):m \to 1,\ldots,(f_n,f'_n):m \to 1}{([f_1,..,f_n],[f'_1,..,f'_n]):m \to n}\ EP$$

A theory may be presented by $(\Omega,\Delta)$, where $\Omega$ is a ranked set (of generators) and $\Delta$ is a set of $\Omega$-identities. $(\Omega,\Delta)$ presents the theory $T$ where $T(n,m) = T_\Omega(n,m)/E_\Delta(n,m)$, where $E_\Delta(n,m) = \{(f,f') \mid (f,f'):n \to m$ is a theorem of $E_\Delta\}$.

It is easily confirmed that $E_\Delta$ preserves composition and products, and so $T$ is a theory, and the functor $F:T_\Omega \to T$ sending each morphism to its equivalence class is a full theory functor. $T$ is often denoted $T_\Omega/\Delta$.

If $T$ is a theory, a T-algebra is a product-preserving functor $A:T \to Sets$. The underlying set of the algebra is $A(1)$; to each morphism $f \in T(n,m)$ there is a map $Af:A(n) \to A(m)$; since $A$ is product-preserving, $Af:A^n \to A^m$.

Definition: Let $X$ be any set. The free theory of actors with primitive actors $X$ is the free theory generated by $\{(x,0)\ x\in X\} \cup \{(\underline{send},2)\}$ , and is denoted $A_X$ .

An $A_X$-algebra $C$ is a set (of "actors"), with distinguished elements $Cx$ for each $x \in X$ and a binary operation $C\underline{send}$ (transmission) such that $C\underline{send}(a,b)$ is the actor which results from sending the message $b$ to target $a$ . We write $\langle a\ b\rangle$ for $\underline{send}(a,b)$ and we make the convention that transmission associates to the left; thus $\langle a_1 \ldots a_n\rangle$ means $\langle\langle\cdots\langle a_1\ a_2\rangle\ a_3\rangle\cdots a_n\rangle$ .

Definition: An actor theory is a quotient theory of a free actor theory. The actor theory presented by $(X,\Delta)$ is $A_X/\Delta$ .

## 3. Examples of Algebra Semantics

An algebra semantics is a full theory-functor $F:T_\Omega \to T$ from a free theory $T_\Omega$ to some other theory $T$. We identify T-algebras $X:T \to Sets$ with implementations. The underlying set $X(1)$ of $X$ is the set of values in the implementation $X$. A denotation is a morphism in $T_\Omega(0,1)$. If $d$ is a denotation, then $XF(d) \in X(1)$ is the value denoted by $d$ in the implementation $X$. Note that $F(d_1) = F(d_2)$ iff $d_1$ and $d_2$ denote the same value in every implementation. We call $T_\Omega$ the theory of denotations and $T$ the semantic theory of $F$. The set of programs is a subset (often proper) of the set of denotations. (See Figure 3.1.)

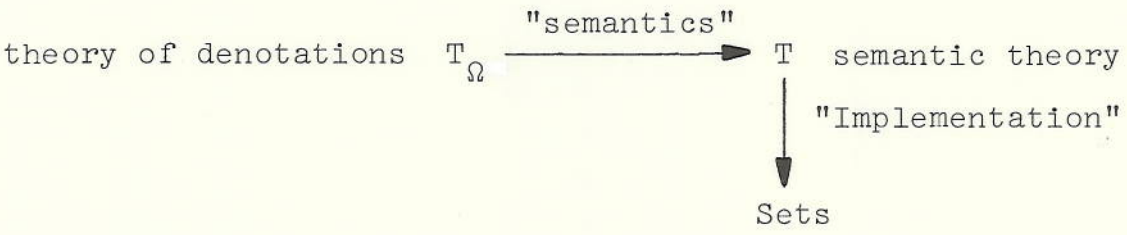


Figure 3.1

An algebra semantics may be presented by $(\Omega,\Delta)$ where $\Omega$ is a ranked set and $\Delta$ is a set of $\Omega$-identities. $(\Omega,\Delta)$ presents the algebra semantics $F:T_\Omega \to T_\Omega/\Delta$ where $F$ is the canonical functor. A computation in $(\Omega,\Delta)$ is a derivation in the formal system $E_\Delta$. Let $H \subseteq T_\Omega(0,1)$ be a set of denotations regarded as "known" (e.g., representations of integers). If $d$ is a denotation, we say $d$ halts relative to $H$ iff there is an $h \in H$ such that $(d,h) \in E(\Delta)$.

A typical choice of $H$ is the set of denotations $d$ such that for no $d'$ is $(d,d')$ a substitution instance of an identity in $\Delta$.

We will give two examples of algebra semantics. Both use actor theories, though this use is not mandatory.

Example 1. A language with assignment, sequencing, and while [10]. Here, <list $S_1$ $S_2$> is the denotation for $(S_1;S_2)$ ; <while B S> denotes while B do S ; and <assign I EXP> denotes I := EXP ; and if S is a statement and ENV an environment, <S ENV> denotes the environment produced by executing S in ENV . For a more detailed discussion of the axioms, see [9].

For these examples, we will drop the requirement that projections be named $e_1,e_2,\ldots$ . Instead, lower case italic identifiers will be used for the projections. Boldface will be used for generators.

Let I be a finite set of identifiers. Let H be the actor theory specified by:

<list first rest env> = <rest <first env>> H.1

<while b s env> = <choose <b env> <while b s <s env>> env> H.2

<assign i exp env> = <ext i <exp env> env> H.3

<choose true $x_1$ $x_2$> = $x_1$ H.4

<choose false $x_1$ $x_2$> = $x_2$ H.5

<identifier z <ext i v env>> =

<choose <eq z i> v <identifier z env>> H.6

for each $i \in I$ ,

<eq i i> = true HS.7

and for each $i,i' \in I$ such that $i \neq i'$ ,

<eq i i'> = false HS.8

Since there are no rules for expressions or booleans in the original, any "useful" computation must occur in some particular H-algebra. For instance, let X be an H-algebra, $b,s,e \in X(1)$ , and in X let <b e> = true , <b <s e>> = true , and <b <s <s e>>> = false . Then in X ,

```
<while b s e> = <choose <b e> <while b s <s e>> e>
              = <choose true <while b s <s e>> e>
              = <while b s <s e>>
              = <choose <b <s e>> <while b s <s <s e>>> <s e>>
              = <choose true <while b s <s <s e>>> <s e>>
              = <while b s <s <s e>>>
              = <choose <b <s <s e>>>
                        <while b s <s <s <s e>>>>
                        <s <s e>>>>
              = <choose false <while b s <s <s <s e>>>>
                              <s <s e>>>
              = <s <s e>>
```

Example 3.2. A simple applicative language [6]. Let I be a set of identifiers, with {zero,suce,equal} ⊆ I , and let R be th the actor theory presented by the following axioms:

```
<identifier id env> = <env id>                                          R.1
<apply opr opnd env> = <opr env <opnd env>>                             R.2
<cond pred thenpart elsepart env> = <choose <pred env>
                                            <thenpart env>
                                            <elsepart env>>             R.3
<letrec vble pbody body env> = <body <label vble pbody env>>            R.4
<lambda vble pbody env arg> = <pbody <ext vble arg env>>                R.5
<ext vble value oldenv probe> =
          <choose <eq vble probe> value <oldenv probe>>                 R.6
<label vble pbody oldenv probe> =
          <choose <eq vble probe>
                  <pbody <label vble pbody oldenv>>
                  <oldenv probe>>                                       R.7
```

<choose true $x_1$ $x_2$> = $x_1$ R.8

<choose false $x_1$ $x_2$> = $x_2$ R.9

For each $i \in I$ ,

<eq i i> = true RS.10

For each $i,i' \in I$ such that $i \neq i'$ ,

<eq i i'> = false RS.11

<initenv succ> = succ R.12

<initenv equal> = equal R.13

<initenv zero> = zero R.14

<equal zero zero> = true R.15

<equal <succ x> <succ y>> = <equal x y> R.16

<equal <succ x> zero> = false R.17

<equal zero <succ x>> = false R.18

Again, for a discussion of the programming issues involved in this choice of axioms, see [9].

## 4. Theory Presentations and Rewriting Systems

In this section we will see how an arbitrary set of axioms may be interpreted as a rewriting system (in fact, a subtree replacement system [7]). With fairly natural restrictions on the axioms, the resulting system has the Church-Rosser property, with pleasant consequences. Under some additional plausible conditions, we will show that "outermost" ("topmost", "leftmost") derivations suffice to perform all useful computations.

Let $A$ be a set, $R \subseteq A \times A$, and let $R^*$ be the reflexive transitive closure of $R$. We say $R$ has the Church-Rosser property iff for every $x,y,z \in A$, if $(x,y) \in R^*$ and $(x,z) \in R^*$, there is a $t \in A$ such that $(y,t) \in R^*$ and $(z,t) \in R^*$. We say $t$ is an R-normal form of $x$ iff $(x,t) \in R^*$ and there is no $u$ such that $(t,u) \in R$. If $R$ has the Church-Rosser property, then every $x \in A$ has at most one R-normal form. Let $R\#$ denote the reflexive, symmetric, transitive closure of $R$.

Theorem 4.1: If $R$ has the Church-Rosser property, $(x,y) \in R\#$ and $y$ is R-normal, then $(x,y) \in R^*$.

Proof: We will show that if $(x,y) \in R\#$, then there is a $z$ such that $(x,z) \in R^*$ and $(y,z) \in R^*$. If $(x,y) \in R\#$, then there exist $x_0 = x, y_0, x_1, y_1, .., x_k, y_k = y$ such that $(x_j,y_j) \in R^*$ and $(y_j,x_{j+1}) \in (R^{-1})^*$. (Figure 4.1.) We proceed by induction on $k$. If $k = 0$ the claim is trivial. Assume the claim for $k - 1$. Then for some $z$, $(x,z) \in R^*$ and $(y_{k-1},z) \in R^*$. Now $(x_k,y_{k-1}) \in R^*$, so $(x_k,z) \in R^*$ and $(x_k,y_k) \in R^*$. By the Church-Rosser property, there is a $z'$ such that $(z,z') \in R^*$ and $(y_k,z') \in R^*$. So $(x,z') \in R^*$ and $(y,z') \in R^*$. To complete the proof, note that if $y$ is R-normal and $(y,z) \in R^*$, then $y = z$. ∎

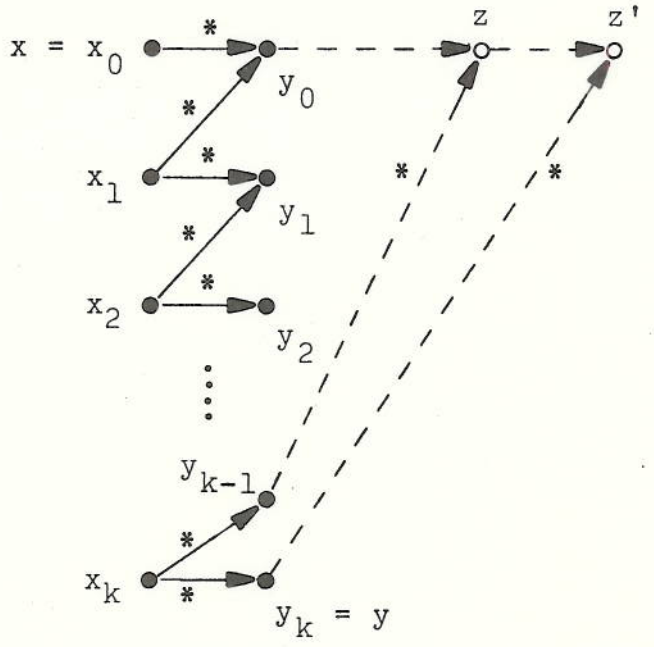


Figure 4.1

We now consider the case where the set $A$ is a set of morphisms in a free algebraic theory.

Definition: If $\Delta$ is a set of identities and $k \in \omega$, the set of $\Delta$-moves on $T_\Omega(k,1)$, denoted $m_\Delta(k)$ is $\{(g.h, g.h') \mid g \in T_\Omega(p,1),\ h,h' \in T_\Omega(k,p)$ and for each $i$ either $e_i.h = e_i.h'$ or $e_i.h = f.q$ and $e_i.h' = f'.q$ where $(f,f') \in \Delta\}$.

In Rosen's terminology, if $\Delta$ is a set of rule schemata, $m_\Delta(k)$ is the general replacement system induced by $\Delta$ on $T_\Omega(k,1)$. If $k = 0$, $m_\Delta(k) = \{(g.f.h, g.f'.h) \mid (f,f') \in \Delta\}$; for $k > 0$, this simpler definition does not work. Note that parallel rewritings are permissible but not mandatory.

We say $\Delta$ is monic iff if $(f,f') \in \Delta$, then $f$ is a monomorphism (i.e., $f'$ has no variables not in $f$). We say $\Delta$ is linear iff if $(f,f') \in \Delta$ then $f$ has no repeated variables.† We

† We leave as an exercise the algebraic characterization of linearity.

say $\Delta$ has no common substitution instances iff if $(f,f'),(g,g') \in \Delta$ and for some $h$ and $h'$, $f.h = g.h'$, then $(f,f') = (g,g')$.

Theorem 4.2 (Rosen [7]): If $\Delta$ is monic, linear, and has no common substitution instances, then $m_\Delta(0)$ has the Church-Rosser property.

We define a formal system $M_\Delta$ as follows:

Axioms: If $(f,f') \in m_\Delta(k)$, then $\vdash (f,f'):k \to 1$ MΔ

If $f \in T_\Omega(k,1)$, then $\vdash (f,f):k \to 1$ MR

Rules: $$\frac{(f,g):k \to 1}{(g,f):k \to 1}\ \text{MS} \qquad \frac{(f,g):k \to 1 \quad (g,h):k \to 1}{(f,h):k \to 1}\ \text{MT}$$

$$\frac{(f_1,g_1):k \to 1, \ldots, (f_n,g_n):k \to 1}{([f_1,..,f_n],[g_1,..,g_n]):k \to n}\ \text{MP}$$

Define $m_\Delta^{\#}(k,n) = \{(f,f') \mid (f,f'):k \to n \text{ is a theorem of } M_\Delta\}$.

Lemma 4.1: $M_\Delta \vdash (f,g):k \to n$ iff $M_\Delta \vdash (e_i.f,e_i.g):k \to 1$ for each $i$ $(1 \le i \le n)$.

Proof: If $n > 1$, then $(f,g)$ can only come from an application of rule MP. ∎

Lemma 4.2: If $M_\Delta \vdash (f,f'):k \to 1$, then $M_\Delta - MP \vdash (f,f'):k \to 1$ ∎

Hence $m_\Delta^{\#}(k,1)$ is just $[m_\Delta(k)]^{\#}$. We write $m_\Delta^{\#}(k)$ for this set. Similarly we write $m_\Delta^{*}(k)$ for $[m_\Delta(k)]^{*}$.

Lemma 4.3: If $M_\Delta \vdash (f_i,f_i'):k \to 1$, then

$M_\Delta \vdash (g.[f_1,..,f_i,..,f_n],g.[f_1,..,f_i',..,f_n]):k \to 1$.

Proof: By induction on the proof of $(f_i,f_i')$ in $M_\Delta$. If $(f,f_i')$ is an axiom, so is $(g.[f_1,..,f_i,..,f_n],g.[f_1,..,f_i',..,f_n])$. If the end of the proof of $(f_i,f_i')$ is

$$\frac{(f_i, f_i'') \qquad (f_i'', f_i')}{(f_i, f_i')} \text{ MT}$$

Then by the induction hypothesis,

$M_\Delta \vdash (g.[f_1,..,f_i,..,f_n], g.[f_1,..,f_i'',..,f_n])$ and

$M_\Delta \vdash (g.[f_1,..,f_i'',..,f_n], g.[f_1,..,f_i',..,f_n])$ , so the result follows by MT. MS is similar; MP is irrelevant by Lemma 4.2. ∎

Lemma 4.4: If $M_\Delta \vdash (f,f'):k \to l$ , then $M_\Delta \vdash (f.h, f'.h):p \to l$ .

Proof: Another easy induction on proofs in $M_\Delta$ - MP . ∎

Theorem 4.3: $E_\Delta(n,p) = m_\Delta^\#(n,p)$ .

Proof: (i) $m_\Delta^\#(n,p) \subseteq E_\Delta(n,p)$ . Every axiom of $M_\Delta$ is provable in $E_\Delta$ , and every rule in $M_\Delta$ is a rule in $E_\Delta$ .

(ii) $E_\Delta(n,p) \subseteq m_\Delta^\#(n,p)$ . Proof by induction on proofs in $E_\Delta$ . Every axiom in $E_\Delta$ is an axiom of $M_\Delta$ . We have one case for each rule in $E_\Delta$ .

(ES): If $M_\Delta \vdash (f,g):n \to m$ then for each $i$ , $M_\Delta \vdash (e_i.f, e_i.g):n \to l$ . By MS , $M_\Delta \vdash (e_i.g, e_i.f):n \to l$ , and then by MP , $M_\Delta \vdash (g,f):n \to m$ .

(ET): Similar.

(EP): Identical to MP .

(EC): We must show that if $M_\Delta \vdash (f,f'):n \to m$ , $g \in T_\Omega(m,k)$ and $h \in T_\Omega(p,n)$ , then $M_\Delta \vdash (g.f.h, g.f'.h):p \to k$ .

If $m = 1$ , then the result follows from Lemmas 4.3 and 4.4.

Assume $m > 1$ . Then construct morphisms $q(j'):n \to m$ for each $j'(0 \le j' \le m)$ so that for each $j(1 \le j \le m)$

$$e_j.q(j') = \begin{cases} e_j.f. & \text{if } j \le j' \\ e_j.f'. & \text{if } j > j' \end{cases}$$

For each $i(1 \le i \le k)$ and $j(0 \le j \le m)$, $M_\Delta \vdash (e_i.g.q(j), e_i.g.q(j+1)):n \to 1$ by Lemma 4.3. By repeated use of MT (noting that $q(0) = f$ and $q(m) = f'$, $M_\Delta \vdash (e_i.g.f, e_i.g.f'):n \to 1$. Hence by Lemmas 4.1 and 4.4 $M_\Delta \vdash (g.f.h, g.f'.h):p \to k$. ∎

A set of identities $\Delta$ is <u>nonoverlapping</u> iff for every $(f,f') \in \Delta$ and $(f.g,h) \in m_\Delta(k)$ either $h = f'.g$ or $h = f.g'$. Roughly speaking, $\Delta$ is nonoverlapping iff rewrite sites are always disjoint.

If $\Delta$ is nonoverlapping, define the set of <u>outermost</u> moves in $m_\Delta(k)$ as the set of all moves $(g.h, g.h')$ such that

(i) $g \in T_\Omega(p,1)$ and $g$ is $\Delta$-normal

(ii) $h,h' \in T_\Omega(k,p)$

(iii) for each $i$, either $e_i.h$ is a projection or $e_i.h = f.q$ for some $(f,f') \in \Delta$.

(iv) for each $i$, either $e_i.h = e_i.h'$ or $e_i.h = f.q$ and $e_i.h' = f'.q$ for some $(f,f') \in \Delta$.

Since $T_\Omega$ is a free theory, to any non-normal morphism $f$ there is exactly one $g$ (up to isomorphism) which fits the definition; it consists of the tree down to the top of the first rewritable spot on each branch. An outermost move consists of rewriting some of those "outermost" or "topmost" sites.

<u>Lemma 4.5</u>: If $(f,f')$ is outermost, so is $(f.h, f'.h)$. ∎

<u>Lemma 4.6</u>: If $\Delta$ is nonoverlapping, $g$ is not normal, and $(g.h,f)$ is outermost, then $f = g'.h$ for some outermost move $(g,g')$.

<u>Proof</u>: In Figure 4.2, consider what regions of $g.h$ may be rewritten. If the rewritten section lies entirely within $h$ (as in Figure 4.2(b)), then the move cannot be outermost. The rewritten section cannot overlap (as in (c)), since that would violate the

nonoverlapping property of $\Delta$ . The only remaining possibility is (d), which corresponds to the conclusion of the lemma. ∎

Note that this lemma depends on the fact that T is a free theory.

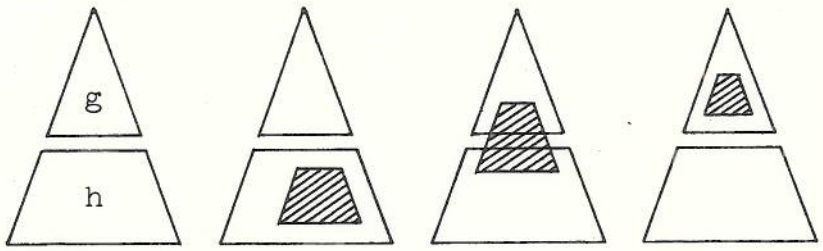


Figure 4.2

Since $m^*_\Delta(k,l)$ is the reflexive transitive closure of $m_\Delta(k)$ , every nontrivial proof in $m^*_\Delta(k,l)$ may be represented by a string of members of $m_\Delta(k)$ , each step representing a rewriting.

Theorem 4.4: If $(f,g) \in m^*_\Delta(k)$ and $\Delta$ is nonoverlapping, then there is a derivation of (f,g) in which every outermost move precedes every non-outermost move.

Proof: Consider a non-outermost move which precedes an outermost move:

$$(g.h,g.h'),(g.h',f)$$

where g,h, and h' are as in the definition of a move. Since the first move is non-outermost, g is non-normal. By Lemma 4.5, $f = g'.h'$ for some outermost move $(g,g')$ . Now we change the derivation to

$$(g.h,g'.h),(g'.h,g'.h')$$

putting the outermost move first. Repeat until all violations are eliminated. ∎

Corollary 4.1: If $(f,g) \in m^*_\Delta(k)$ , $\Delta$ is non-overlapping, and $g$ is $\Delta$-normal, then there is a derivation in which every move is outermost.

Proof: Apply Theorem 4.4 to put all non-outermost moves last; but if $g$ is normal, the last move must be outermost. ∎

We may summarize our results as follows:

Theorem 4.5: If $\Delta$ is linear, monic, non-overlapping, and has no common substitution instances, then a denotation $d$ halts relative to the set of $\Delta$-normal denotations iff there is derivation $(d,h) \in m^*_\Delta(0)$ where $h$ is $\Delta$-normal and every move is outermost. Furthermore, $h$ is unique.

Proof: From Theorems 4.1, 4.2, 4.3, and Corollary 4.1. ∎

This says that without loss of generality, we may always compute in the forward direction only, and with only outermost moves. In most cases, the limited degree of nondeterminism is acceptable, since usually the root of the tree is being rewritten (and there is therefore only one outermost more possible).

(We have a fair amount of intuition about when a completely deterministic strategy is possible; we hope to include this in the final draft of this paper.)

## 6. Conclusions

The initial attraction of algebra semantics was that the model theory of first-order identities gave an immediate handle on the data structures for an implementation. In this paper we have shown how algebra semantics gives a reasonable control structure for an interpreter in terms of subtree replacement systems.

Typed by Christopher Charles

References

1. Goguen, J.A., and Thatcher, J.W. Initial algebra semantics. Proc. 15th IEEE Conference on Switching and Automata Th., New Orleans (1974).

2. Hewitt, C.; Bishop, P.; and Steiger, R. A universal modular actor formalism for artificial intelligence. Proc. IJCAI 3, San Francisco (1973).

3. Lawvere, F.W. Functional semantics of algebraic theories. Proc. NAS USA 50 (1963), 869-872.

4. MacLane, S. Categories for the Working Mathematician, Springer-Verlag, New York (1971).

5. Pareigis, B. Categories and Functions, Academic Press, New York (1970).

6. Reynolds, J.C. Definitional interpreters for higher-order programming languages. Proc. ACM National Conference (1972), 717-740.

7. Rosen, B.K. Tree-manipulating systems and Church-Rosser theorems. JACM 20 (1973), 160-187.

8. Scott, D., and Strachey, C. Toward a mathematical semantics for computer languages. Computers and Automata, J. Fox (Ed.), Wiley, New York (1972), 19-46.

9. Wand, M. First order identities as a defining language. Technical Report No. 29, Computer Science Department, Indiana University, Bloomington (1975).

10. Hoare, C.A.R., and Lauer, P. Consistent and complementary formal theories of the semantics of programming languages. Acta Informatica 3 (1974), 135-153.