MODES, VALUES AND EXPRESSIONS

Marvin Solomon

TR 74-219

November 1974

Department of Computer Science
Cornell University
Ithaca, New York 14853

Modes, Values and Expressions

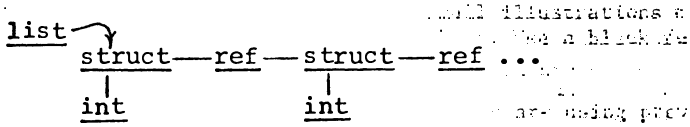
Marvin Solomon
Cornell University

0. Introduction:

Several modern programming languages (e.g. Algol 68 [1], Pascal [2], Basel [3], EL1 [4]) require each variable to have a declared mode or type, which limits the class of values which it may assume. In each of these languages, a value may contain a reference or pointer to a value of the same type, allowing linked lists. In this case, it is natural to define the mode of the value recursively -- that is, in terms of itself. For example, if list describes values which are structures containing two fields, one being an integer and the other a pointer to another value of type list, then we can summarize these facts by writing

(0.1) mode list = struct_2(int,ref list).

What is a mode? One point of view is that it is a set of values [5,6]. The Algol 68 Report [1] takes the slightly different view that a mode is a possibly infinite object that is associated with a set of values. For example, list is the infinite tree:



Building on these ideas, Lewis and Rosen [7], using methods of Scott [8,9], built a formal model SEQ, in which every meaningful definition like (0.1) denotes a unique element of SEQ. The question of which mode definitions should be considered equivalent (cf. [10,11]) is answered by saying that two definitions are equivalent if and only if they denote the same member of SEQ.

Lewis and Rosen were principally interested in answering the question of equivalence of modes in type systems which allow unions. We extend their work in a different direction by constructing a Scott-like model for values. The construction is similar to the model for modes, but involves certain new complications. Moreover, we investigate the relationship between modes and their associated sets of values. As an application of our technique, we consider certain mode definitions which are excluded from Algol 68, for example:

mode m1 = m1
mode m2 = struct(m2)
mode m3 = ref m3

Mode m1 is also disallowed by Lewis and Rosen, since it fails to denote a unique object in their system. (In fact, any object whatsoever satisfies the "defining property.") They give a syntactic characterization of those definitions which do have unique solutions. Mode m2 does have a unique solution, but we will show that the corresponding set of values is empty. We give a syntactic method of detecting such definitions, and it seems reasonable to disallow them.

Mode m3 is excluded for still more subtle reasons; by considering the allowed expressions in Algol 68 and their effect on values, we show that it is possible to construct a value of mode m3 which has "subvalues" (in a sense to be made more precise below) which cannot be accessed. These useless values can be eliminated by preventing definitions like that of m3. Again, we show how to detect such definitions. Removal of these modes entails no essential loss of expressivity from the language. Alternatively, a new operator may be added to the language so that such values are no longer useless.

The syntactic tests mentioned above are similar to the "declaration condition" [1, paragraph 4.4.4]. However, our conditions are less ad hoc, following from general considerations which may be applied to other programming languages as they are developed. Moreover, our semantic approach points out an anomaly in the Algol 68 definition:

mode m4 = struct(ref m4)

is legal, but

mode m5 = [1:1] ref m5

is not, despite the obvious similarity. Our considerations would allow m5 as a mode.

1. Modes

1.1 Syntax

We use the following definitional facility as a concrete example:

```
<definition>::=<declaraction>
            |<definition>;<declaraction>
<declaraction>::=mode<indicant>=
                 <declarer>
<declarer>::=int|<indicant>
   |ref<declarer>
   |struct_k(<declarer>,...,<declarer>)
      (k ≥ 1)      k declarers
```

(1.1.1)

We will denote terminal productions of <indicant> by underlined words. For notational simplicity, we use integers as field selectors rather than arbitrary character strings as in Algol 68. This has no essential effect on the theory. We will assume that every definition contains exactly one declaration for each indicant appearing in it.

1.2 Semantics

The model M we build for modes is inspired by [9] and closely resembles the one presented in [14] (cf. [7,8]). As mentioned above, modes are conceived of as (possibly) infinite trees. The nodes of these trees are labeled by the symbols $\{\underline{int}, \underline{ref}, \underline{struct}_1, \underline{struct}_2, \ldots\}$. We allow only those trees which are ranked, in the following sense: Define the rank of a symbol by

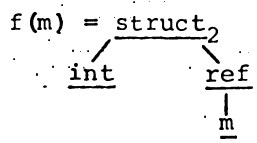
$$\begin{aligned} &\text{rank}(\underline{int}) = 0 \\ &\text{rank}(\underline{ref}) = 1 \\ &\text{rank}(\underline{struct}_k) = k \quad (k = 1,2,3,\ldots) . \end{aligned} \tag{1.2.1}$$

If S labels a node of a ranked tree, then that node has precisely rank(s) sons. For example, int appears only at leaves. We could completely specify our intended set of modes by saying that it consists of all ranked trees built from the ranked set of symbols given by (1.2.1), but for the sake of other definitions, it is preferable to have infinite modes built up inductively from finite pieces.

Rather than build our model specifically for the ranked set of symbols (1.2.1), we show how to construct such a model for any ranked set $\Omega = \Omega_0 \cup \Omega_1 \cup \ldots$. Appendix A contains the details. In this case, $\Omega_0 = \{\underline{int}\}$, $\Omega_1 = \{\underline{ref}, \underline{struct}_1\}$, and $\Omega_k = \underline{struct}_k$ for $k \geq 2$. We write M for $T(\Omega)$ with this particular choice of Ω.

Elements of M are called modes. Briefly, the facts about M are as follows: M is a type of ordered set, called a cpo, in which certain limits are defined. A function which preserves these limits is said to be continuous. The theorem of Tarski (A.6) guarantees that such a function f has fixed points, and shows how to compute a particular fixed point called the least fixed point of f, Yf.

Now consider a definition such as (0.1). The declarer $\underline{struct}_2(\underline{int}, \underline{ref}\ \underline{list})$ may be thought of as defining a function, where the indicant list is a free variable. More specifically, if $m \in M$, let



Then (0,1) may be thought of as saying that list is any element of M such that $m = f(m)$ -- i.e. a fixed point of f.

More formally, let D be the definition

$$\begin{aligned} &\underline{mode}\ \underline{m}_1 = d_1; \\ &\underline{mode}\ \underline{m}_2 = d_2; \\ &\quad\vdots \\ &\underline{mode}\ \underline{m}_n = d_n . \end{aligned}$$

For each declarer d_i, define a function $f_d : M^n \to M$ as follows:

If $d = \underline{int}$ then $f_d(\bar{x}) = \underline{int}$

If $d = m_i$ then $f_d(\bar{x}) = x_i$

If $d = \underline{ref}\ \hat{d}$ then $f_d(\bar{x}) = \text{ref}(f_{\hat{d}}(\bar{x}))$

If $d = \underline{struct}_k(d_1,\ldots,d_k)$ then

$f_d(\bar{x}) = \underline{struct}_k(f_{d_1}(\bar{x}),\ldots,f_{d_k}(\bar{x}))$

where $\bar{x} = \langle x_1,\ldots,x_n\rangle \in M^n$.

Let $f_D : M^n \to M^n$ by

$f_D(\bar{x}) = \langle f_{d_1}(\bar{x}),\ldots,f_{d_n}(\bar{x})\rangle$.

f_D is built up from operators in Ω so that by lemmas A.15 and A.7 in Appendix A, f_D is continuous. Hence, Tarski's theorem yields as a corollary

(1.2.2) Theorem: Each declaration D is satisfied by at least one $m \in M^n$. The set of $m \in M^n$ satisfying D is a cpo. The least element of this cpo is

$$\text{lub}\{f_D^k(\langle 1,\ldots,1\rangle) \mid k \geq 0\} .$$

1.3 Unique Modes

We have seen that each definition identifies at least one element of M^n. However, some definitions are satisfied by more than one. For example, the definition

(1.3.1) mode $\underline{m}$ = $\underline{m}$

is satisfied by any mode whatsoever. Otherwise stated, f_D is the identity function on M, so the lattice of fixed points of f_D is all of M. However, we can show that any definition with multiple fixed points is essentially a variation on (1.3.1). To characterize those "good" definitions which define unique fixed points, consider the following directed graph G_D constructed from a definition D. The nodes of G_D are the indicants of D. There is an arc from $\underline{m}$ to $\underline{n}$ if and only if the declaration mode $\underline{m}$ = $\underline{n}$ appears in D.

(1.3.2) Theorem (modified from Lewis and Rosen [7]): If G_D contains no directed cycles (i.e. if G_D is a directed acyclic graph) then the least fixed point Yf_D of f_D is maximal in M^n (i.e. there is no $\bar{n} \in M^n$ such that $\bar{n} > \bar{m}$). Conversely, if G_D contains a directed cycle, then f_D has a fixed point of which at least one component is $\perp$.

Proof (outline): Let $\bar{m}^0 = \langle \perp,\ldots,\perp \rangle \in M^n$, and let

$$\bar{m}^{k+1} = \langle m_1^{k+1},\ldots,m_n^{k+1} \rangle = f_D(\bar{m}^k).$$

By theorem A.6, the least fixed point Yf is given by

$$\bar{m} = \langle m_1,\ldots,m_n \rangle = \text{lub}\{\bar{m}^k \mid k \geq 0\}.$$

Suppose G is acyclic. For each $m \in M$ let $u(m)$ be the length of the shortest branch which terminates in $\perp$ if any such exists, and ∞ otherwise. Thus m is completely defined up to level $u(m)$. The idea is to show that as k increases, $u(m_i^k)$ increases without bound. This is because each iteration of f_D adds something to the root of m_i, for those indicants m_i which depend non-trivially (or not at all) on the other indicants (mode $\underline{m}_i$ = $\underline{m}_j$ is not in D). Each other component m_j^k of $\bar{m}^k$ depends on such an m_i^k in such a way that the value of $u(m_j^k)$ grows as fast as $u(m_i^k)$.

For the converse, suppose in G we have a path

$$m_{i_1} \to m_{i_2} \to \ldots \to m_{i_\ell} \to m_{i_1}.$$

Then for all $k \geq 0$ and all $j \leq \ell$, $m_{i_j}^k = \perp$. ■

(1.3.3) Corollary: If G_D is acyclic, then f_D has a unique fixed point. Hence D associates a unique mode $m_i \in M$ with each indicant $\underline{m}_i$ in D.

Proof: Let $\bar{m}$ = Yf and let $\bar{n}$ be any other fixed point. $\bar{m}$ is the least fixed point, so $\bar{m} \leq \bar{n}$. But $\bar{m}$ is maximal in M so $\bar{m} = \bar{n}$. ■

For the remainder of this paper, we will assume that all definitions mentioned have acyclic graphs and hence unique solutions. We will consistently use the notation m_i for the element of M associated with the indicant $\underline{m}_i$ by a definition.

1.4 Regular Modes

We have seen that every definition is associated with some element of M. We may wonder whether the converse holds. The answer is negative.

(1.4.1) Theorem: A mode $m \in M$ is associated with some indicant by some definition (satisfying the property of §1.3) if and only if m is a regular tree (see Appendix B).

Proof: Let D be a definition, and $s \in \Omega$. Modify D to form a grammar G as follows: The non-terminal symbols of G are the indicants of D. The terminal symbols are $1,\ldots,n$ where $n = \max\{k \mid s \in \Omega_k$ for some s occurring in D$\}$. For each declaration

mode $\underline{m}$ = e

in D, include the following productions in G:

If s occurs at node η of e and ADDRESS(η) = ξ, then include $\underline{m} \to \xi$ in G.

If $\underline{n}$ occurs at node η of e and ADDRESS(η) = ξ, include $\underline{m} \to \xi\underline{n}$.

Now for each indicant $\underline{m}$, it should be clear that if $\underline{m}$ is taken to be the start symbol of G, then G produces exactly $L_s(m)$. Since G is right linear ([15]), $L_s(m)$ is a regular set.

Conversely, let m be a regular mode. Let $s_1,\ldots,s_k$ be the symbols appearing in m and let $A_i = \langle Q_i, q_i^0, \delta_i, F_i \rangle$ be the minimal finite automaton recognizing $L_{s_i}(m)$. Let A be the parallel composition of the A_i -- i.e. $A = \langle Q, q^0, \delta, F \rangle$ where $Q = Q_1 \times \ldots \times Q_k$, $q^0 = \langle q_1^0,\ldots,q_k^0 \rangle$, and $\delta(\langle q_1,\ldots,q_k \rangle, a) = \langle \delta_1(q_1,a),\ldots,\delta_k(q_k,a) \rangle$ (the definition of F is irrelevant). Delete those states of A which are unreachable. Now for each remaining state $\langle q_1,\ldots,q_k \rangle$ there is exactly one k such that $q_k \in F_k$. This is because the reachability of $\langle q_1,\ldots,q_k \rangle$ implies that for some $x \in \{1,\ldots,k\}^*$, $\langle q_1,\ldots,q_k \rangle = \delta(\langle q_1^0,\ldots,q_k^0 \rangle, x) = \langle \delta_1(q_1^0,x),\ldots,\delta_k(q_k^0,x) \rangle$?

But $q_i = \delta_i(q_i^0,x) \in F_i$ implies that the node whose address is x is labeled by s_i. Clearly this is true for exactly one

i. Now introduce an indicant $\underline{q}$ for each state q of A and give the definition

$$\underline{\text{mode}}\ \underline{q} = s_i(\delta(q,1),\ldots,\delta(q,k))$$

where $q = \langle q_1,\ldots,q_k\rangle$ and s_i is the unique symbol such that $q_i \in F_i$. It may be verified that the mode corresponding to the indicant $\underline{q_0}$ is the given mode m. ■

2. Modes and Values

The central contribution of this work is the construction of a model for the set of possible values of an Algol 68-like language which provides a meaning for the, until now, purely formal notion of mode. In particular, each mode is associated with a set of values in such a way that distinct modes are associated with distinct sets of values. A value such as the one shown in figure 2.1 may be considered to be an infinite tree by "unrolling" it as shown in figure 2.2, or it may be defined by a recursive equation such as

$$v = \underline{\text{struct}}_2(5,\text{ref}(v)).$$

Both of these descriptions suggest a model V for values similar to the model M for modes. But in the case of values, an additional complication arises: whereas figure 2.2 indicates that some infinite trees should be allowed as values, we don't want infinite values such as

$$v = \underline{\text{struct}}_2(5,v)$$

since this would imply that v is a structure, one of whose fields contains v. Thus a more involved construction is required.

2.1 The Model V

In Appendix A, we have all the machinery to produce a cpo $T(\Omega)$ from an arbitrary ranked set Ω of operators. Our approach will be to produce an appropriate set Σ of operators and let $V = T(\Sigma)$. Intuitively, an operator $\sigma \in \Sigma_n$ will be a storage cell having exactly n pointer fields and possibly some integer fields already initialized. Formally, we make the inductive definition

(2.1.1) $\Sigma_0 = Z$ (the set of integers)

$\underline{\text{ref}} \in \Sigma_1$

If $\sigma_i \in \Sigma_{n_i}$ for $i = 1,\ldots,k$, then

$\underline{\text{struct}}_k(\sigma_1,\ldots,\sigma_k) \in \Sigma_{n_1+n_2+\ldots+n_k}$.

Let $V = T(\Sigma)$. Intuitively, V contains all finite and infinite ranked trees with labels in Ω (cf. 1.2), perhaps with leaves labeled $\perp$, and satisfying the additional property that on each infinite branch, $\underline{\text{ref}}$ appears infinitely often.

2.2 The Map VALUE

Let 2^V denote the collection of all sets of values. Order 2^V by letting $S \leq T$ if and only if $T \subseteq S$ (note the change of direction). It is readily verified that 2^V becomes a cpo, with $\perp = V$. Define a function VALUE on finite modes inductively by:

(2.2.1)
$$\text{VALUE}(\perp) = V$$
$$\text{VALUE}(\underline{\text{int}}) = Z$$
$$\text{VALUE}(\underline{\text{ref}}\ m) = \{\underline{\text{ref}}\ v \mid v \in \text{VALUE}(m)\}$$
$$\text{VALUE}(\underline{\text{struct}}_k(m_1,\ldots,m_k)) = \{\underline{\text{struct}}_k(v_1,\ldots,v_k) \mid v_i \in \text{VALUE}(m_i) \text{ and } v_i \neq \perp,\ i = 1,\ldots,k\}.$$

This completely specifies VALUE for all finite modes. In fact, if we let VALUE_n denote the restriction of VALUE to M_n, then the set

$$\{\text{VALUE}_n : M_n \to 2^V \mid n \geq 0\}$$

satisfies all the hypotheses of Theorem A.13 as may easily be checked. Thus we have a unique extension of VALUE to infinite modes. By the proof of Theorem A.13, $\text{VALUE}: M \to V$ is given explicitly by

$$\text{VALUE}(m) = \bigcap_{n=0}^{\infty} \text{VALUE}_n(\pi_n(m)).$$

For example, consider the mode m defined by (0.1). The first few approximations to m and the value sets associated with them are shown in figure 2.2.1. Intuitively, each m_k gives more information about m than m_{k-1} and so eliminates some values as possible values of m.

Now, for some modes $m \in M$, $\text{VALUE}(m) = \emptyset$. For example, consider the definition

$$\underline{\text{mode}}\ \underline{m} = \underline{\text{struct}}_1(m).$$

This specifies a mode $m \in M$ whose projections are $m_i = \pi_i(m) = \underline{\text{struct}}_1^i \perp$.

$$\text{VALUE}(m_0) = \text{VALUE}(\perp) = V$$
$$\text{VALUE}(m_1) = \text{VALUE}(\underline{\text{struct}}_1 \perp) = \{\underline{\text{struct}}_1(v) \mid v \neq \perp\}.$$
$$\vdots$$
$$\text{VALUE}(m_i) = \{\underline{\text{struct}}_1^i(v) \mid v \neq \perp\}$$
$$\vdots$$

Each $v \in \text{VALUE}(m_i)$ has the property that $v = \sigma(v_1,\ldots,v_k)$ for some operator $\sigma \in \Sigma_k$ of depth at least i, where the depth of an operator is defined by

$$\text{depth}(\sigma) = 0 \text{ if } \sigma = \underline{\text{ref}} \text{ or } \sigma \in Z$$
$$\text{depth}(\underline{\text{struct}}_k(\sigma_1,\ldots,\sigma_k)) = 1 + \max\{\text{depth}(\sigma_i) \mid i = 1,\ldots,k\}.$$

Suppose $v \in \text{VALUE}(m) = \bigcap_{i=0}^{\infty} \text{VALUE}_i(m_i)$. Since $v \in V$, $v = \perp$ or $v = \sigma(v_1,\ldots,v_k)$ for some $\sigma \in \Sigma$, $v_1,\ldots,v_k \in V$ (Theorem A.14). $\perp \notin \text{VALUE}(m_1) \supseteq \text{VALUE}(m)$ so $v \neq \perp$. If $v = \sigma(v_1,\ldots,v_k)$, then for each i, $v_i \in \text{VALUE}(m_i^1)$, so $\text{depth}(\sigma) \geq i$. It follows that $\text{depth}(\sigma)$ is greater than any integer -- a patent absurdity.

The generalization of this proof is the following theorem. Given a definition D, we define a graph G_D similar to the graph constructed in §1.3, but now $\underline{m} \rightarrow \underline{n}$ if and only if D contains the declaration mode m = e where e is an expression containing an occurance of n not contained in the operand of any occurrence of ref.*/

Theorem: If G_D is acyclic then every mode defined by D (i.e. made to correspond to some indicant m in D) has a non-empty value set. Conversely, if G_D has a cycle, then D defines some $m \in M$ such that $\text{VALUE}(m) = \emptyset$.

Proof: Suppose G_D is acyclic. Let m be an indicant appearing in G_D and let its declaration be mode m = e for some expression e. Perform the following transformation on e:

Step 1: Replace all occurrences of int in e by 0, and all occurrences of ref e' (for some expression e') by ref $\perp$.

Step 2: If e has an occurrence of n for some indicant n, replace that occurrence by e', where the declaration of n is mode n = e', and go back to step 1; otherwise stop (e contains no indicants).

This procedure halts after a finite number of steps because G_D is acyclic. It is readily verified that a value $v \in V$ is produced such that $v \in \text{VALUE}(m)$.

The proof of the converse is similar to the above example, and is left as a tedious exercise. ∎

2.3 Regular Values

Because of the impossibility of structures, such as circular lists, whose "unrolled versions" are infinite, we have allowed infinite elements in V. But some elements of V are "actually" infinite, in that they require an infinite amount of storage. In fact, we have the following:

Theorem: For any value $v \in V$, v corresponds to some finite list structure if and only if v is a regular value (c.f. Appendix B; notice that the occurrence languages L_s are defined in terms of the operators $s \in \Sigma$, not the symbols of Ω).

Proof: (It may help the reader to look at the example, figure 2.3.1, while reading this proof) Given a finite list structure S, we construct a finite automaton ([15]), whose states correspond to the cells of S. There is a transition from state q_1 to q_2 under the symbol $k \geq 1$ if and only if the kth pointer out of the cell corresponding to q_1 points to the cell corresponding to q_2. For each cell c, if the corresponding state is made to be the unique final state, the automaton will recognize $L_c(s)$, the occurrence language of c in S. Thus S is (or, more precisely, corresponds to under unrolling) a regular value.

The converse construction is similar. ∎

3. Modes, Values, and Expressions

Suppose x is a variable with mode m and current value v. Let e[x] be an expression containing an occurrence of x. What values may occur as values of such expressions? Viewing v as a tree, we say w is a subvalue of v if w is the tree composed of some node of v and all its descendants. Now for any value of x, it would be desirable that there is some expression e for each subvalue of v. Otherwise, this subvalue is inaccessible and may as well be eliminated from v. This motivates the

(3.1) Definition: Let m be a mode, v a value in VALUE(m), x a variable of mode m. A subvalue w of v is accessible with respect to m if and only if there is some expression e[x] such that if x = v, then e[x] = w. v is connected with respect to m if and only if every subvalue of v is accessible with respect to m. m is connected if and only if every value $v \in \text{value}(m)$ is connected with respect to m.

Of course, this definition depends on what expressions are allowed. We choose the following subset of Algol 68 as an example:

(3.2) <expression>::=<variable>|<selection>|<coercion>

<selection>::=<positiveinteger> of <expression>

<corecion>::=<declarer>:<expression>

k of e is defined if and only if the mode of e begins with struct_ℓ for some $\ell \leq k$. Then the value of e must be of the form $\text{struct}_k(v_1,\ldots,v_c)$ and the value of k of e is of mode $\text{ref}^k m$ for some $k \geq 0$.

*/ See the remarks about notation at the beginning of Appendix A.

In this case the value of e is either $\underline{ref}^k v$ for some v, or $\underline{ref}^\ell \perp$ for $\ell < k$. In the former case d:e = v, in the latter d:e = ⊥. (In case the mode of e = $\underline{ref}^k m$ for more than one value of k, choose the least such k.) Mode m3 is not connected, as figure 3.1 illustrates: clearly, any expression involving only a variable x of mode m3 cannot contain any selections. Successive coercions can be combined, so any expression is equivalent to one of the form d:x. If the mode of x is $\underline{ref}^k m$ for some m, then m = m3, so the mode of x is m3 and the mode of d is m3. Thus the value of the expression is that of x. If the value of x is v, then w is inaccessible.

Let D be a definition. Again we construct a graph G_D as in §§1.3, 2.2. Here we have an arc from m to n if and only if the declaration of m is mode m = e where there is an occurrence of n in e which is not contained in any operand of any $\underline{struct}_k$.

(3.3) Theorem: Every mode m defined by a definition D is connected if and only if G_D is acyclic.

Proof (by pictures): Suppose G_D is acyclic. Let m be a mode defined by D, x a variable of mode m, v a value of x, and w a subvalue of v. Since G_D is acyclic, each branch of m (and hence of v) consists of alternating finite, non-empty sequences of structs and refs. The situation is given pictorially in figure 3.2. Then the value w may be described by the expression:

(3.4) $\underline{ref}^{p_{m_k}}:i_{kn_k}\underline{of}\ldots\underline{ofi}_{k1}\underline{ofm}_{k-1}: \ldots :$

$m_2:i_{2n_2}\underline{of}\ldots\underline{ofi}_{22}\underline{ofi}_{21}\underline{ofm}_1:$

$i_{1n_1}\underline{of}\ldots\underline{ofi}_{12}\underline{ofi}_{11}\underline{ofx}$.

Notice that each m_i is a mode which can be described by some declarer built up from the indicants appearing in D. Thus (3.4) becomes a valid expression if each m_i is replaced by the corresponding declarer.

Conversely, if G_D contains a cycle, then one of the modes m defined by D satisfies the equation $m = \underline{ref}^k m$ for some $k \geq 1$. The unique solution to this equation in M is the mode ref ref ref ... which, as we saw above, is not connected.

From one point of view, such a mode is anomalous simply because there are not enough expressions. If, for example, we allow an operator deref, then m3 becomes connected (in fact, all regular modes become connected). On the other hand, m3 may be replaced by m4 using m4: 1 of x to access the "second node" of x. Here 1 of x is of mode ref m4 which is not equal to m4, so the corecion has its desired effect.

4. Conclusion

We have given a model for modes and values, and used it to classify three types of undesirable declarations: those which do not define unique modes, those which define valueless modes, and those which define unconnected modes. We give criteria for detecting each kind of definition in terms of an easily checked property of a graph which can be constructed directly from the definition. We also consider regular trees and show that they correspond exactly to the definable modes and the representable values.

But we feel that the most important aspect of this paper is that it shows that the purely syntactic approach to programming languages is not always the best -- that a description in terms of intended meaning can be at the same time equally rigorous and more clear than the syntactic approach, if the semantics is properly formalized.

References

[1] van Wijngaarden, A. "Report on the Algorithmic Language Algol 68", Numerishe Mathematic, 14(1969), 79-218.

[2] Wirth, N. "The Programming Language Pascal", Acta Informatica 1,1(1971), 35-63.

[3] _____. Basel Language Programmer Manual. Massachusetts Computer Associates Inc., Wakefield Mass., 1970.

[4] Wegbreit, B. Studies in Extensible Programming Languages. Ph.D. Thesis, Harvard University, 1970.

[5] Reynolds, J.C. "A set-theoretic approach to the concept of type", Argonne National Laboratories, 1969.

[6] Morris, J.H.,Jr. "Types are not sets", ACM Symposium on Principles of Programming Languages, Boston, Mass. 1973

[7] Lewis, C. and B.K. Rosen. "Recursively defined data types, Part I", ACM Symposium on Principles of Programming Languages, Boston, Mass. 1973.

[8] Scott, D. "Continuous lattices", Proceedings of the Dalhousie Conference, Lecture Notes in Mathematics 274, Springer Verlag, Berlin, 1972.

[9] Scott, D. "The lattice of flow diagrams", in Semantics of Algorithmic Languages, E. Engeler, ed., Lecture Notes in Mathematics, 188, Springer Verlag, Berlin, 1971.

[10] Koster, C.H.A. "On infinite modes", Algol Bulletin 30 in SIGPLAN Notices 4,3 (March 1969).

[11] Kral, J. "The equivalence of modes and the equivalence of finite automata" Algol Bulletin 35 (March 1973).

[12] Tarski, A. "A lattice-theoretical fixpoint theorem and its applications", Pacific Journal of Mathematics 5(1955) 285-309.

[13] Rabin, M.O. "Automata on infinite objects and Church's problem", Regional Conference Series in Mathematics 13, American Mathematical Society, Providence, R.I., 1972.
[14] Wand, M. Mathematical foundation of formal language theory. Ph.D. Thesis, MIT Project MAC report TR-108, Cambridge, Mass. 1973.
[15] Hopcroft, J. and J.D. Ullman. Formal Languages and their Relation to Automata. Addison Wesley, Reading, Mass. 1969.
[16] Eilenberg, S. Automata, Languages and Machines. Academic Press, New York, 974.
[17] Engelfreit, J. "A note on infinite trees", Information Proc. Letters 1, 6 (December 1972), 229-232.
[18] Rosen, B.K. "Tree-manipulating Systems and Church-Rosser theorems", JACM 20,1 (January 1973), 160-187.
[19] MacLane, S. Categories for the Working Mathematician. Springer-Verlag, New York, 1971.

Appendix A

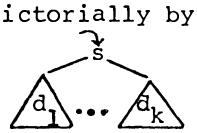
In this appendix, we show how to construct the cpo $T(\Omega)$ of all (ranked, finite or infinite, partially or completely defined) trees over an arbitrary ranked alphabet Ω and derive some of its properties

Suppose $\Omega = \Omega_0 \cup \Omega_1 \cup \Omega_2 \cup \ldots$ is any set of symbols. Without loss of generality, assume the Ω_i's are disjoint (otherwise replace $s \in \Omega_n$ by $\langle n,s\rangle$). Introduce a new symbol $\perp \notin \Omega$ and define:

A.1 Definition: $D_0 = \{\perp\}$

$$D_{n+1} = \{s(d_1,\ldots,d_k) \mid s \in \Omega_k \text{ and } d_1,\ldots,d_k \in D_n\} \cup \{\perp\}$$ ∎

[Here we pause to make a remark about notation. The elements of D_n are to be thought of as trees. $s(d_1,\ldots,d_k)$ is the tree given pictorially by:



In the case $s \in \Omega_0$, $s(\,) \in D_{n+1}$ where $s(\,)$ is a one-node tree: $\overset{s}{\cdot}$.

We will often refer to such a one-node tree by the label on its node. For example, in definition A.1 both occurrences of "$\perp$" denote one-node trees.

Throughout the paper, we use the "tree" and the "term" or "expression" terminology interchangeably. For example each symbol occurring in d_i in the above example is in the ith subtree of s and in the ith operand of s.]

A.2 Definition: Introduce a partial order on D_{n+1} as follows:

$x \leq y$

if and only if

$x = \perp$ or $x = s(d_1,\ldots,d_k)$, $y = s(d_1',\ldots,d_k')$

for some k, some $s \in \Omega_k$ and some $d_1,\ldots,d_k$, $d_1',\ldots,d_k' \in D_n$ such that $d_i \leq d_i'$ in D_n for $i = 1,\ldots,k$. ∎

A.3 Definition: Let C be a set and $\leq$ a relation on C (a subset of $C \times C$). $\langle C, \leq \rangle$ is a cpo (complete partially ordered set) if and only if

(i) $\leq$ partially orders C (i.e. $\leq$ is transitive, reflexive and anti-symmetric)

(ii) There is an element $\perp \in C$ such that $\forall x \in C, \perp \leq x$.

(iii) Each chain $x_0 \leq x_1 \leq x_2 \leq \ldots$ of elements of C has a least upper bound $x = \text{lub}\{x_i\}$ -- i.e. an element satisfying the properties

a) $x_i \leq x \quad \forall i$

b) if $x_i \leq y \quad \forall i$ then $y \leq x$. ∎

Notice that if C is finite, then (iii) holds trivially.

A.4 Lemma: Each D_n defined by A.1 and A.2 is a cpo. ∎

A.5 Definition: Let A, B be cpo's, $f: A \to B$ Then f is monotone if and only if

for each $x,y \in A$, with $x \leq y$,

$f(x) \leq f(y)$.

f is continuous if and only if f is monotone and

for each chain $x_0 \leq x_1 \leq \ldots$ in A,

$f(\text{lub}\{x_i\}) = \text{lub}\{f(x_i)\}$. ∎

Notice that the monotone property of f insures that $f(x_0) \leq f(x_1) \leq \ldots$ so the definition of continuity makes sense. Notice also that in a finite cpo (or more generally, in one in which all chains are finite) every monotone function is continuous.

A.6 Theorem: (Modified from Tarski [12]) Let C be a cpo and $f: C \to C$ continuous. Let $B \subseteq C, B = \{x \mid f(x) = x\}$, the set of fixed points of f. Then

(i) $B \neq \emptyset$

(ii) B is a cpo

(iii) The least element $\perp_B$ of B is denoted Yf and may be computed by

$Yf = \text{lub}\{f^n(\perp) \mid n \geq 0\}$. ∎

We list several easy properties of cpo's and continuous functions.

A.7 Lemma: Let A, B, C be cpos, $f:A \to B$, $g:B \to C$ continuous.

(i) $g \circ f:A \to C:x \mapsto g(f(x))$ is continuous

(ii) $id_A:A \to A:x \mapsto x$ is continuous.

(iii) if $A \subseteq B$ then $\subseteq:A \to B:x \mapsto x$ is continuous.

(iv) $A \times B$ is a cpo under component-wise ordering: $\langle x_1,x_2\rangle \leq \langle y_1,y_2\rangle$ if and only if $x_1 \leq y_1$ and $x_2 \leq y_2$.

(v) $\pi_1:A \times B \to A:\langle x,y\rangle \mapsto x$ and $\pi_2:A \times B \to B:\langle x,y\rangle \mapsto y$ are continuous

(vi) Similar to (iv) and (v) but for arbitrary products.

(vii) if $s \in \Omega_k$, then $s:D_n^k \to D_{n+1}$ is continuous. ∎

A.8 Definition: The projection $p_n:D_{n+1} \to D_n$ is defined by

$$p_0(x) = \perp \ \forall x$$
$$p_{n+1}(\perp) = \perp$$
$$p_{n+1}(s(d_1,\ldots,d_k)) = s(p_n(d_1),\ldots,p_n(d_k)).$$

∎

A.9 Lemma: $\forall n$, $D_n \subseteq D_{n+1}$,

p_n is continuous for all n. Moreover, $p_n(x) \leq x$ in D_{n+1} and if $y \in D$, $y \leq x$, then $y \leq p_n(x)$. ∎

The intuitive idea of all this is as follows:

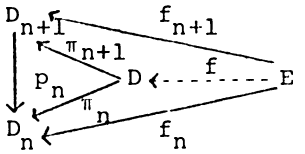
Let D be the set of all finite or infinite ranked trees over $\Omega \cup \{\perp\}$ where $\perp$ has rank 0. If $t_1,t_2 \in D$, define $t_1 \leq t_2$ if and only if t_2 can be obtained from t_1 by replacing occurrences of $\perp$ by other trees. It may be verified that for finite trees, this is exactly the ordering defined in A.2. Now if $t \in D$ is an infinite tree, define the sequence $\langle t_0,t_1,\ldots\rangle$ as follows: t_i comes from t by chopping off all branches after the first i levels and replacing removed subtrees by $\perp$. It may be readily verified that $t_i \in D_i$ and $t_i = p_i(t_{i+1})$ for all i. But it may be shown that the sequence thus constructed corresponds to a unique tree $t \in D$. Thus we define t to be the sequence $\langle t_0,t_1,\ldots\rangle$.

A.10 Definition: Let $D = T(\Omega)$ be the set of all sequences $\langle d_0,d_1,\ldots\rangle \in D_0 \times D_1 \times D_2 \times \ldots$ such that $d_n = p_n(d_{n+1})$. Order D by $\langle d_i\rangle \leq \langle d_i'\rangle$ if and only if $d_n \leq d_n'$ for each n.

Let $\pi_n:D \to D_i:\langle d_i\rangle \mapsto d_n$. ∎

A.11 Theorem: D is a cpo. $\pi_n:D \to D_n$ is continuous and satisfies $\pi_n = p_n \circ \pi_{n+1}$. If E is a cpo and $\{f_i:E \to D_i\}$ is a set of continuous functions such that $f_n = p_n \circ f_{n+1}$ then there is a unique $f:E \to D$ such that $f_n = \pi_n \circ f$ for all n.

Pictorially:



∎

Theorem A.11 may be summarized by saying that D is the inverse limit of $\{D_i\}$ [19]. D is also the direct limit of $\{D_i\}$ as we show below:

A.12 Definition: Let $i_n:D_n \to D_{n+1}:x \mapsto x$ $\iota_n:D_n \to D:x \mapsto \langle d_i\rangle$ where

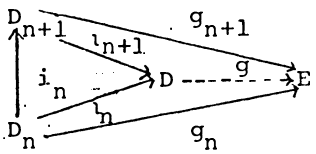
$$d_i = \begin{cases} x & \text{if } i \geq n \\ p_i(p_{i+1}(\ldots(p_{n-1}(x))\ldots)) & \text{if } i < n. \end{cases}$$

∎

A.13 Theorem: Each $\iota_n:D_n \to D$ is continuous and $\iota_n = \iota_{n+1} \circ i_n$.

If t is a cpo and $\{g_n:D_n \to E\}$ continuous functions such that $g_n = g_{n+1} \circ i_n$ then there is a unique $g:D \to E$ such that $g_n = g \circ i_n$.

Pictorially,



Proof: We may show that each $d = \langle d_i\rangle$ satisfies the property that

$$d = \text{lub}(\{\iota_n(d_n) \mid n \geq 0\}).$$

From this it follows easily that g may be defined by

$$g(d) = \text{lub}(\{g_i(d_i)\}).$$

∎

We conclude with a recursive characterization of D and two lemmas for constructing continuous functions on D.

A.14 Theorem: $D = \{\perp\} \cup \{s(d_1,\ldots,d_k) \mid s \in \Omega_k, d_i \in D\}$. ∎

A.15 Lemma: for each $s \in \Omega_k$, $s:D^k \to D$ is continuous.

Proof: $s:D_n^k \to D_{n+1}$ is continuous by A.7. Using A.11 and A.13, s can be extended to

a continuous function $s:D^k \to D$.

Our last lemma uses the particular inductive structure of the Ω constructed in §2.1 to build a continuous map similar to the one in A.14.

A.16 Lemma: Let $k \geq 0$ and suppose that for each $n_1,\ldots,n_k$,

$$f:\Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \to \Omega_{n_1+n_2+\ldots+n_k}.$$

Then there is a unique function

$$\hat{f}:D^k \to D$$

such that

$$\hat{f}(d_1,\ldots,d_k) = \perp \text{ if } d_i = \perp \text{ for any } i$$

and

$$\hat{f}(s_1(d_{11},\ldots,d_{1n_1}),\ldots,s_k(d_{k1},\ldots,d_{kn_k})) = f(x_1,\ldots,s_1)(d_{11},d_{12},\ldots,d_{kn_k}) .$$ ∎

Appendix B

In this appendix we introduce the notion of regular tree. This idea has appeared in many places in the literature under many guises. See, for example, [11,13,14,16,17]. The notion of regularity is reasonably robust in that different definitions yield essentially equivalent results. We present only the version that seems particularly convenient for our purposes.

Let $d \in D = T(\Omega)$. We wish to give a notation for the nodes of d (cf. [18]). Informally, if η is a node of d, then there is a unique path ξ from the root of d to η. ξ may be described by giving the sequence of choices made in passing from the root to η. If the k sons of a node are labeled 1 to k, this assigns to each node η a finite string ADDRESS(η) of positive integers. (see figure B.1) Then for $s \in \Omega$, let the occurrence language of s in d be the set

$$L_s(d) = \{\text{ADDRESS}(\eta) \mid \eta \text{ is labeled by } s\} .$$

More formally,

B.1 Definition: ([14]) For each $s \in \Omega_{Z_+^*}$ and $\eta \geq 0$, define a mapping $L_s:D_\eta \to 2^{Z_+^*}$ as follows:*/

$$L_s(\perp) = \emptyset$$

$$L_s(\hat{s}(d_1,\ldots,d_k)) = \begin{cases} \bigcup_{i=1}^{k} i\cdot L_s(d_i) & \text{if } \hat{s} \neq s \\ \{\varepsilon\} \cup \bigcup_{i=1}^{k} i\cdot L_s(d_i) & \text{if } \hat{s} = s \end{cases}$$

where $i\cdot L_s(d_i)$ denotes the result of prefixing each string in $L_s(d_i)$ by the number i. ∎

$2^{Z_+^*}$ is a cpo under the subset ordering (the converse of the ordering in §2.2) and L_s is continuous and satisfies the hypotheses of A.13. Thus we can extend L_s to a function from D to $2^{Z_+^*}$. The reader may verify that this formal definition corresponds to the informal definition above.

We are particularly interested in the case where $L_s(d)$ is a regular set. Suppose only finitely many $s \in \Omega$ appear in d. Then there is an n such that any s appearing in d may be found in some Ω_k for $k \leq n$. Hence, $L_s(d) \subseteq \{1,\ldots,n\}^*$ for each s in Ω.

B.2 Definition: $d \in D$ is regular if and only if $L_s(d) = \emptyset$ for all but finitely many s and $L_s(d)$ is a regular subset of $\{1,\ldots,n\}^*$ for all s (where $n = \max\{k \mid s \in \Omega_k \text{ and } L_s(d) \neq \emptyset\}$). ∎

*/ Z_+^* is the set of finite strings of positive integers.

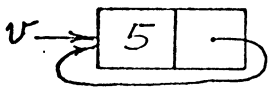


Figure 2.1

A Value

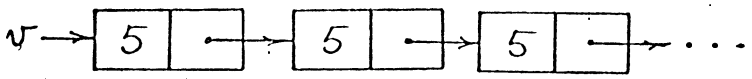
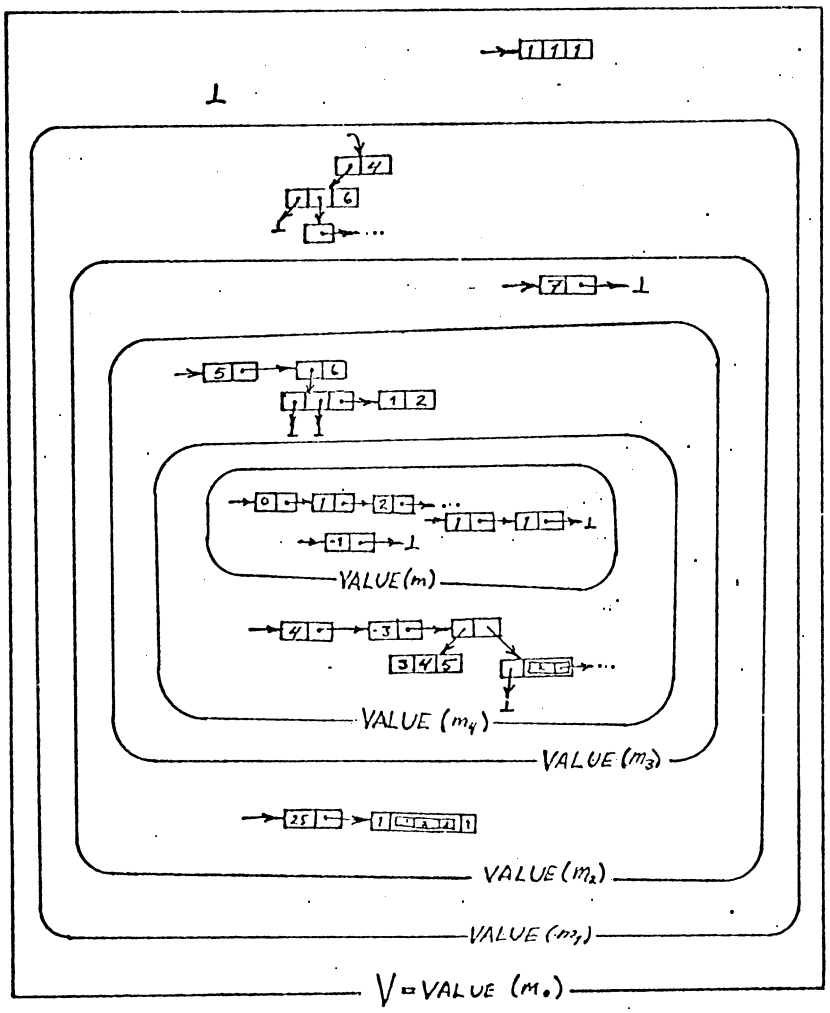


Figure 2.2

The value of figure 2.1 unrolled



$m_0 = \bot$

$m_1 = \underline{\text{struct}}_2(\bot, \bot)$

$m_2 = \underline{\text{struct}}_2(\underline{\text{int}}, \underline{\text{ref}}\ \bot)$

$m_3 = \underline{\text{struct}}_2(\underline{\text{int}}, \underline{\text{ref}}\ \underline{\text{struct}}_2(\bot, \bot))$

$m_4 = \underline{\text{struct}}_2(\underline{\text{int}}, \underline{\text{ref}}\ \underline{\text{struct}}_2(\underline{\text{int}}, \underline{\text{ref}}\ \bot))$

$\vdots$

$m = \underline{\text{struct}}_2(\underline{\text{int}}, \underline{\text{ref}}\ \underline{\text{struct}}_2(\underline{\text{int}}, \underline{\text{ref}}\ \underline{\text{struct}}_2(\ \ldots\ )))$

Figure 2.2.1

Some typical members of the cpo V of Values.

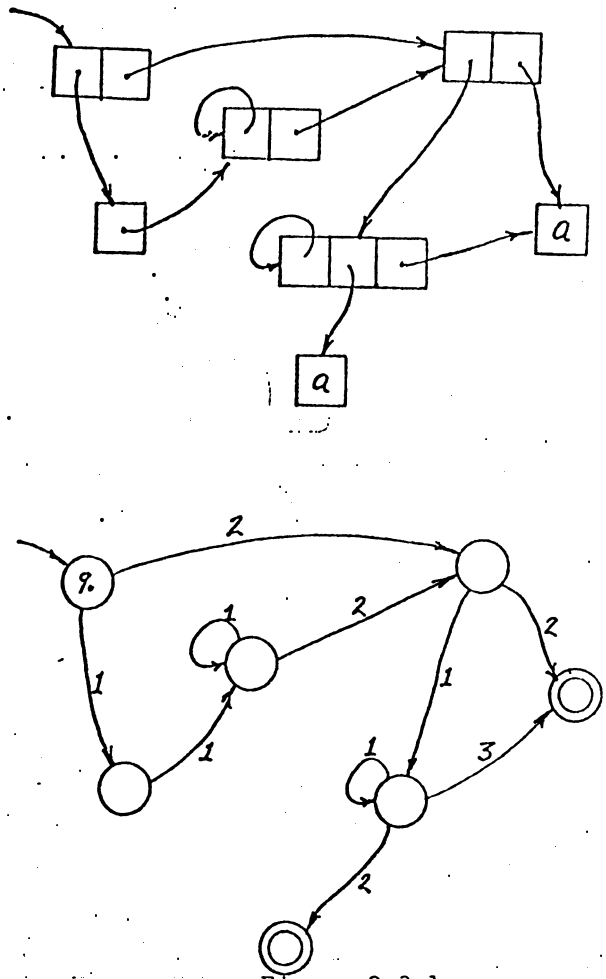


Figure 2.3.1

A list structure and its finite automaton

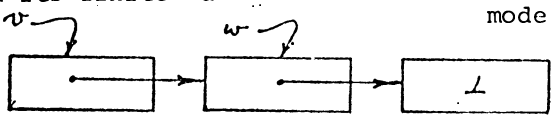
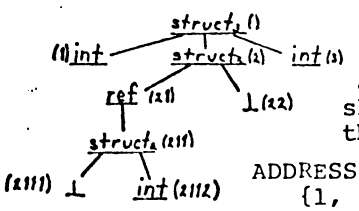


Figure 3.1

A value v which is not connected and an inaccessible subvalue w

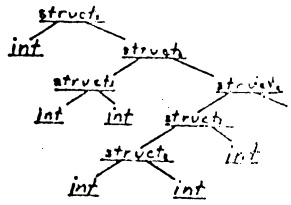


addresses are shown in parentheses

ADDRESS(int) = {1, 2$\overline{112}$, 3}

Figure B.1

The ADDRESS function on modes and on symbols



ADDRESS(struct$_2$) = $\{2^k 1^\ell \mid 0 \leq \ell \leq k\}$

Figure B.2

A mode which is not regular

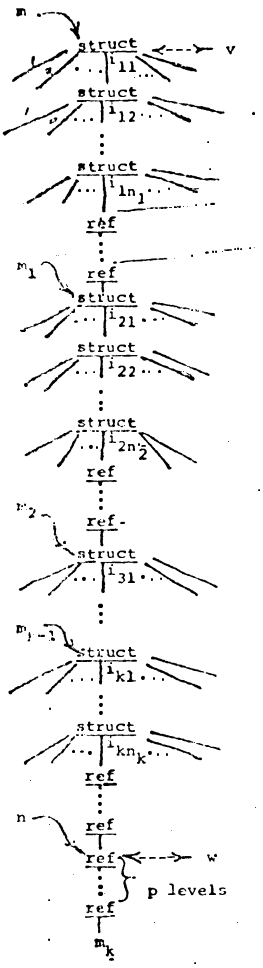


Figure 3.2

A connected mode m, a value v of mode m, and a subvalue w of v of mode n

ERRATA

Modes, Values, and Expressions[†]

by Marvin Solomon

Page	Column	Line	For	Read
2	left	8-10	declaraction	declaration
4	left	-16	let v =	let V =
4	right	After	line 16 add:	(c.f. Lemma A.16)
5	left	-7	impossibility	possibility
5	right	-6	corecion	coercion
5	right	-2	struct $(v_1,\ldots,v_e)$	struct$_\ell$ $(v_1,\ldots,v_\ell)$
5	right	-1	is of mode ref^{k}m for some $k\geq 0$.	is v. The only coercion we will allow is dereferencing. Thus the expression d:e is defined if and only if d is of some mode m and e is of mode ref^{k}m for some $k>0$.
8	right	-1	Using A.11 and A.13	Using A.11 with $E = D^k$ and $f_{n+1} = s \circ (\pi_n,\ldots,\pi_n)$
9	left	8	$f:\Omega_{n_1} \times f_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1+\Omega n_2+\ldots+\Omega n_k}$.	$f:\Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1+n_2+n_3+\ldots+n_k}$.
9	left	15	$f(x_1,\ldots,s_1)$	$f(s_1,\ldots,s_k)$

Thanks to Barry Rosen for pointing out many of these errors to me.

[†]Presented at the Second ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Palo Alto, Calif., January 20-22, 1975.

ERRATA

Modes, Values, and Expressions[†]

by Marvin Solomon

Page	Column	Line	For	Read
2	left	8-10	declaraction	declaration
4	left	-16	let v =	let V =
4	right	After	line 16 add:	(c.f. Lemma A.16)
5	left	-7	impossibility	possibility
5	right	-6	corecion	coercion
5	right	-2	$\underline{\text{struct}}_k(v_1,\ldots,v_e)$	$\underline{\text{struct}}_\ell(v_1,\ldots,v_\ell)$
5	right	-1	is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.	is v. The only coercion we will allow is dereferencing. Thus the expression d:e is defined if and only if d is of some mode $\underline{m}$ and e is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.
8	right	-1	Using A.11 and A.13	Using A.11 with $E = D^k$ and $f_{n+1} = s \circ (\pi_n,\ldots,\pi_n)$
9	left	8	$f:\Omega_{n_1} \times f_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1}+\Omega_{n_2}+\ldots+\Omega_{n_k}$.	$f:\Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1+n_2+n_3+\ldots+n_k}$.
9	left	15	$f(x_1,\ldots,s_1)$	$f(s_1,\ldots,s_k)$

Thanks to Barry Rosen for pointing out many of these errors to me.

[†]Presented at the Second ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Palo Alto, Calif., January 20-22, 1975.

ERRATA

Modes, Values, and Expressions[†]

by Marvin Solomon

Page	Column	Line	For	Read
2	left	8-10	declaraction	declaration
4	left	-16	let v =	let V =
4	right	After	line 16 add:	(c.f. Lemma A.16)
5	left	-7	impossibility	possibility
5	right	-6	corecion	coercion
5	right	-2	$\underline{\text{struct}}_k(v_1,\ldots,v_e)$	$\underline{\text{struct}}_\ell(v_1,\ldots,v_\ell)$
5	right	-1	is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.	is v. The only coercion we will allow is dereferencing. Thus the expression d:e is defined if and only if d is of some mode $\underline{m}$ and e is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.
8	right	-1	Using A.11 and A.13	Using A.11 with $E = D^k$ and $f_{n+1} = s \circ (\pi_n,\ldots,\pi_n)$
9	left	8	$f:\Omega_{n_1} \times f_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1+\Omega_{n_2}+\ldots+\Omega_{n_k}}$.	$f:\Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1+n_2+n_3+\ldots+n_k}$.
9	left	15	$f(x_1,\ldots,s_1)$	$f(s_1,\ldots,s_k)$

Thanks to Barry Rosen for pointing out many of these errors to me.

[†]Presented at the Second ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Palo Alto, Calif., January 20-22, 1975.

ERRATA

Modes, Values, and Expressions[†]

by Marvin Solomon

Page	Column	Line	For	Read
2	left	8-10	declaraction	declaration
4	left	-16	let v =	let V =
4	right	After	line 16 add:	(c.f. Lemma A.16)
5	left	-7	impossibility	possibility
5	right	-6	corecion	coercion
5	right	-2	$\underline{\text{struct}}_k(v_1,\ldots,v_e)$	$\underline{\text{struct}}_\ell(v_1,\ldots,v_\ell)$
5	right	-1	is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.	is v. The only coercion we will allow is dereferencing. Thus the expression d:e is defined if and only if d is of some mode $\underline{m}$ and e is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.
8	right	-1	Using A.11 and A.13	Using A.11 with $E = D^k$ and $f_{n+1} = s \circ (\pi_n,\ldots,\pi_n)$
9	left	8	$f:\Omega_{n_1} \times f_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1}+\Omega_{n_2}+\ldots+\Omega_{n_k}$.	$f:\Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1+n_2+n_3+\ldots+n_k}$.
9	left	15	$f(x_1,\ldots,s_1)$	$f(s_1,\ldots,s_k)$

Thanks to Barry Rosen for pointing out many of these errors to me.

[†]Presented at the Second ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Palo Alto, Calif., January 20-22, 1975.

ERRATA

Modes, Values, and Expressions[†]

by Marvin Solomon

Page	Column	Line	For	Read
2	left	8-10	declaraction	declaration
4	left	-16	let v =	let V =
4	right	After	line 16 add:	(c.f. Lemma A.16)
5	left	-7	impossibility	possibility
5	right	-6	corecion	coercion
5	right	-2	$\underline{\text{struct}}\ (v_1,\ldots,v_e)$	$\underline{\text{struct}}_\ell(v_1,\ldots,v_\ell)$
5	right	-1	is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.	is v. The only coercion we will allow is dereferencing. Thus the expression d:e is defined if and only if d is of some mode $\underline{m}$ and e is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k>0$.
8	right	-1	Using A.11 and A.13	Using A.11 with $E = D^{\overline{k}}$ and $f_{n+1} = s \circ (\pi_n,\ldots,\pi_n)$
9	left	8	$f:\Omega_{n_1} \times f_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1}+\Omega_{n_2}+\ldots+\Omega_{n_k}$.	$f:\Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1+n_2+n_3+\ldots+n_k}$.
9	left	15	$f(x_1,\ldots,s_1)$	$f(s_1,\ldots,s_k)$

Thanks to Barry Rosen for pointing out many of these errors to me.

[†]Presented at the Second ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Palo Alto, Calif., January 20-22, 1975.

ERRATA

Modes, Values, and Expressions[†]

by Marvin Solomon

Page	Column	Line	For	Read
2	left	8-10	declaraction	declaration
4	left	-16	let v =	let V =
4	right	After	line 16 add:	(c.f. Lemma A.16)
5	left	-7	impossibility	possibility
5	right	-6	corecion	coercion
5	right	-2	struct $(v_1,\ldots,v_e)$	$\underline{\text{struct}}_{\ell}(v_1,\ldots,v_{\ell})$
5	right	-1	is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.	is v. The only coercion we will allow is dereferencing. Thus the expression d:e is defined if and only if d is of some mode $\underline{m}$ and e is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k>0$.
8	right	-1	Using A.11 and A.13	Using A.11 with $E = D^{\overline{k}}$ and $f_{n+1} = s \circ (\pi_n,\ldots,\pi_n)$
9	left	8	$f:\Omega_{n_1} \times f_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1}+\Omega_{n_2}+\ldots+\Omega_{n_k}$.	$f:\Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1+n_2+n_3+\ldots+n_k}$.
9	left	15	$f(x_1,\ldots,s_1)$	$f(s_1,\ldots,s_k)$

Thanks to Barry Rosen for pointing out many of these errors to me.

[†]Presented at the Second ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Palo Alto, Calif., January 20-22, 1975.

ERRATA

Modes, Values, and Expressions[†]

by Marvin Solomon

Page	Column	Line	For	Read
2	left	8-10	declaraction	declaration
4	left	-16	let v =	let V =
4	right	After	line 16 add:	(c.f. Lemma A.16)
5	left	-7	impossibility	possibility
5	right	-6	corecion	coercion
5	right	-2	$\underline{\text{struct}}_k(v_1,\ldots,v_e)$	$\underline{\text{struct}}_\ell(v_1,\ldots,v_\ell)$
5	right	-1	is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.	is v. The only coercion we will allow is dereferencing. Thus the expression d:e is defined if and only if d is of some mode $\underline{m}$ and e is of mode $\underline{\text{ref}}^k\underline{m}$ for some $k\geq 0$.
8	right	-1	Using A.11 and A.13	Using A.11 with $E = D^{\overline{k}}$ and $f_{n+1} = s \circ (\pi_n,\ldots,\pi_n)$
9	left	8	$f:\Omega_{n_1} \times f_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1}+\Omega_{n_2}+\ldots+\Omega_{n_k}$.	$f:\Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \rightarrow \Omega_{n_1+n_2+n_3+\ldots+n_k}$.
9	left	15	$f(x_1,\ldots,s_1)$	$f(s_1,\ldots,s_k)$

Thanks to Barry Rosen for pointing out many of these errors to me.

[†]Presented at the Second ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Palo Alto, Calif., January 20-22, 1975.

ERRATA

Modes, Values, and Expressions[†]

by Marvin Solomon

Page	Column	Line	For	Read
2	left	8-10	declaraction	declaration
4	left	-16	let v =	let V =
4	right	After	line 16 add:	(c.f. Lemma A.16)
5	left	-7	impossibility	possibility
5	right	-6	corecion	coercion
5	right	-2	struct $(v_1,\ldots,v_e)$	struct$_\ell(v_1,\ldots,v_\ell)$
5	right	-1	is of mode ref^{k}m for some k≥0.	is v. The only coercion we will allow is dereferencing. Thus the expression d:e is defined if and only if d is of some mode m and e is of mode ref^{k}m for some k>0.
8	right	-1	Using A.11 and A.13	Using A.11 with E = $D^{\overline{k}}$ and $f_{n+1} = s \circ (\pi_n,\ldots,\pi_n)$
9	left	8	$f:\Omega_{n_1} \times f_{n_2} \times \ldots \times \Omega_{n_k} \to \Omega_{n_1}+\Omega_{n_2}+\ldots+\Omega_{n_k}$.	$f:\Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \to \Omega_{n_1+n_2+n_3+\ldots+n_k}$.
9	left	15	$f(x_1,\ldots,s_1)$	$f(s_1,\ldots,s_k)$

Thanks to Barry Rosen for pointing out many of these errors to me.

[†]Presented at the Second ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, Palo Alto, Calif., January 20-22, 1975.