MODES, VALUES AND
EXPRESSIONS

Marvin Solomon

TR 74-219

November 1974

Department of Computer Science
Cornell University
Ithaca, New York 14853
0. Introduction:

Several modern programming languages (e.g. Algol 68 [1], Pascal [2], Baseline [3], ELL [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

\[(0.1) \text{mode list = struct}_2(\text{int, ref list}).\]

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:

```
list
  \text{struct}
    \text{ref}
      \text{struct}
        \text{ref}
          \text{int}
          \text{int}
```

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:

- \text{mode m1 = m1}
- \text{mode m2 = struct(m2)}
- \text{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 4.4.1]. 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:

- \text{mode m4 = struct(ref m4)}

is legal, but

- \text{mode m5 = (1:1) ref m5}
is not, despite the obvious similarity. Our considerations would allow \( m_5 \) as a mode.

1. Modes

1.1 Syntax

We use the following definitional facility as a concrete example:

\[
\begin{align*}
\text{	exttt{<definition>}} & \text{::= <declaration>} \\
\text{	exttt{<declaration>}} & \text{::= mode <indicant> = \\
\text{	exttt{<declarer>}} & \text{::= int <indicant> \\
\text{	exttt{<ref<declarer>}} & \text{::= struct}_k <\texttt{<declarer>}, \ldots,<\texttt{declarer}> (k \geq 1) \\
\end{align*}
\]

We will denote terminal productions of \(<\texttt{indicant}>\) 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 \{int, ref, struct_1, struct_2, \ldots\}. We allow only those trees which are ranked, in the following sense: Define the rank of a symbol by

\[
\begin{align*}
\text{rank}(\texttt{int}) & = 0 \\
\text{rank}(\texttt{ref}) & = 1 \\
\text{rank}(\texttt{struct}_k) & = k \ (k = 1, 2, 3, \ldots) .
\end{align*}
\]

If \( S \) labels a node of a ranked tree, then that node has precisely \( \text{rank}(S) \) 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, \( \Omega_0 = \{\texttt{int}, \texttt{ref}, \texttt{struct}_k\} \), and \( \Omega_k = \texttt{struct}_k \) for \( k \geq 2 \). We write \( M \) for \( T(\Omega) \) with this particular choice of \( \Omega \).

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 \( f \), \( y_f \).

Now consider a definition such as \( (0,1) \). The declarer \texttt{struct}(int,ref list) may be thought of as defining a function, where the indicant list is a free variable. More specifically, if \( m \in M \), let

\[
f(m) = \texttt{struct}_2 \begin{array}{c}
\texttt{int} \\
\texttt{ref}
\end{array}
\]

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

More formally, let \( D \) be the definition

\[
\begin{align*}
\text{mode } m_1 & = d_1; \\
\text{mode } m_2 & = d_2; \\
\ldots & \ldots \\
\text{mode } m_n & = d_n .
\end{align*}
\]

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

\[
\begin{align*}
\text{If } d = \texttt{int} & \text{ then } f_d(\tilde{x}) = \text{int} \\
\text{If } d = m_i & \text{ then } f_d(\tilde{x}) = x_i \\
\text{If } d = \texttt{ref} \hat{d} & \text{ then } f_d(\tilde{x}) = \texttt{ref}(f_{\hat{d}}(\tilde{x})) \\
\text{If } d = \texttt{struct}_{k}(d_1, \ldots, d_k) & \text{ then } \\
& f_d(\tilde{x}) = \texttt{struct}_k(f_{d_1}(\tilde{x}), \ldots, f_{d_k}(\tilde{x}))
\end{align*}
\]

where \( \tilde{x} = \langle x_1, \ldots, x_n \rangle \in M^n \).

Let \( f_D: M^n \to M^n \) by \( f_D(\tilde{x}) = \langle f_{d_1}(\tilde{x}), \ldots, f_{d_n}(\tilde{x}) \rangle \).

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

(1.2.2) Theorem: Each declaration \( D \) is satisfied by at least one \( m \in M_0 \). The set of \( m \in M_0 \) satisfying \( D \) is a cpo. The least element of this cpo is

\[
\text{lub}\{D^k(<x_1, \ldots, x_l>) \mid k\geq 0\} .
\]

1.3 Unique Modes

We have seen that each definition identifies at least one element of \( M^0 \). However, some definitions are satisfied by more than one. For example, the definition
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 \( m \) to \( n \) if and only if the declaration 
\[
\text{mode } m = n
\]
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 \( Y_f \) of \( f_D \) is maximal in \( M^2 \) (i.e. there is no \( n' \in M^2 \) such that \( n > n' \)). Conversely, if \( G_D \) contains a directed cycle, then \( f_D \) has a fixed point of which at least one component is 1.

Proof (outline): Let \( m^0 = <1, \ldots, > \in M^2 \), and let

\[
m_k+1 = <m_{k+1}, ..., m_n^k> = f_D(m_k).
\]

By theorem A.6, the least fixed point \( Y_f \) is given by

\[
\bar{m} = <m_1, ..., m_n> = \text{sup}(m_k^k | k \geq 0).
\]

Suppose \( G \) is acyclic. For each \( m \in M \) let \( u(m) \) be the length of the shortest branch which terminates in \( 1 \) if any such exists, and \( 0 \) otherwise. Thus \( m \) is completely defined up to level \( u(m) \). The idea is to show that as \( k \) increases, \( u(m_k^k) \) increases without bound. This is because each iteration of \( f_D \) adds something to the root of \( m \), for those indicants \( m_i \) which depend non-trivially (or not at all) on the other indicants (mode \( m_i = m_j \) is not in \( D \)). Each other component \( m_j^k \) of \( m_k \) 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

\[
m_{i_1} \rightarrow m_{i_2} \rightarrow \ldots \rightarrow m_{i_l} \rightarrow m_{i_1}.
\]

Then for all \( k \geq 0 \) and all \( j \leq l, m_j^k = 1 \).

(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 \( m_i \) in \( D \).

Proof: Let \( \bar{m} = Y_f \) 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 \( \bar{m} = n \).

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 \( m_i \) 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 \( s \in A \). 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 \( n = \text{max}(\{k \in S_k \text{ for some } s \text{ occurring in } D\}) \). For each declaration

\[
\text{mode } m = e
\]
in \( D \), include the following productions in \( G \):

- If \( s \) occurs at node \( n \) of \( e \) and \( \text{ADDRESS}(n) = \xi \), then include \( m = \xi \) in \( G \).
- If \( n \) occurs at node \( n \) of \( e \) and \( \text{ADDRESS}(n) = \xi \), include \( m = \xi \).

Now for each indicant \( m \), it should be clear that if \( m \) 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 \( A_i = \{q_1^0, q_0^1, \ldots, q_i, q_k \} \) be the minimal finite automaton recognizing \( L_{s_i}(m) \).

Let \( A \) be the parallel composition of the \( A_i \) -- i.e., \( A = 0_1 \times \ldots \times 0_k, q^0_1 = q^0, \ldots, q^0_k \), and \( \delta(\langle q_1, q_2, \ldots, q_k \rangle, a) = \langle q_1(q_2(a)), \ldots, q_k(q_k(a)) \rangle \) (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 \( x \in (1, \ldots, k)^* \), \( \langle q_1, \ldots, q_k \rangle = \delta(\langle q_1^0, \ldots, q_k^0 \rangle, x) = \delta_1(q_1(x)), \ldots, \delta_k(q_k(x)). \)

But \( q_i = \delta_1(q_1(x)) \in F_i \) implies that the node whose address is \( x \) is labeled by \( s_i \). Clearly this is true for exactly one
1. Now introduce an indicant $q$ for each state $q$ of $A$ and give the definition

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

where $q = q_1, \ldots q_k$ 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 $q_0$ is the given mode $m_0$.

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

$$v = \text{struct}_k(5, \text{ref}(v))$$

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

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

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 $T(\Sigma)$ from an arbitrary ranked set $\Sigma$ of operators. Our approach will be to produce an appropriate set $\Sigma$ of operators and let $v = T(\Sigma)$. Intuitively, an operator $\sigma \in \Sigma_n^k$ will be a storage cell having exactly $n$ pointer fields and possibly some integer fields already initialized. Formally, we make the inductive definition

$$\begin{align*}
0 & = 2 \text{ (the set of integers)} \\
\text{ref} & \in \Sigma_1 \\
\text{If } \sigma_i & \in \Sigma_{n_i}^k \text{ for } i = 1, \ldots, k, \text{ then} \\
\text{struct}_k(\sigma_1, \ldots, \sigma_k) & \in \Sigma_{n_1 + n_2 + \ldots + n_k} \\
\end{align*}$$

Let $V = T(\Sigma)$. Intuitively, $V$ contains all finite and infinite ranked trees with labels in $0$ (cf. 1.2), perhaps with leaves labeled $1$, and satisfying the additional property that on each infinite branch $\text{ref}$ 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 $\bot = V$. Define a function VALUE on finite modes inductively by:

$$(2.2.1) \text{VALUE}(1) = V$$

$$\text{VALUE}(\text{int}) = \mathbb{Z}$$

$$\text{VALUE}(\text{ref } m) = \{ \text{ref } v \mid v \in \text{VALUE}(m) \}$$

$$\begin{align*}
\text{VALUE}(\text{struct}_k(m_1, \ldots, m_k)) & = \\
& \{(\text{struct}_k(v_1, \ldots, v_k) \mid v_i \in \text{VALUE}(m_i) \text{ and } v_i \neq \bot, \\
& i = 1, \ldots, k) \} \\
\end{align*}$$

This completely specifies VALUE for all finite modes. In fact, if we let $\text{VALUE}_n$ denote the restriction of VALUE to $M_n$, then the set

$$\{ \text{VALUE}_n(m) \mid n \geq 0 \}$$

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, VALUE : $M + V$ is given explicitly by

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

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

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

This specifies a mode $m \in M$ whose projections are $m_i = \pi_i(m) = \text{struct}_1^1 \bot$.

$$\begin{align*}
\text{VALUE}(m_0) & = \text{VALUE}(1) = V \\
\text{VALUE}(m_1) & = \text{VALUE}(\text{struct}_1) = \\
& \{ \text{struct}_1(v) \mid v \neq \bot \} \\
\vdots \\
\text{VALUE}(m_n) & = \{ \text{struct}_1(v) \mid v \neq \bot \} \\
\end{align*}$$

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

$$\text{depth}(\sigma) = 0 \text{ if } \sigma = \text{ref } \text{ or } \sigma \in \mathbb{Z}$$

$$\text{depth}(\text{struct}_k(\sigma_1, \ldots, \sigma_k)) = \\
1 + \max\{ \text{depth}(\sigma_i) \mid i = 1, \ldots, k \}.$$
Suppose \( v \in \text{VALUE}(m) \) = \( \bigoplus_{i=0}^{n} \text{VALUE}_{i}(m_{i}) \).

Since \( v \in V \), \( v = \bot \) or \( v = s(v_{1}, \ldots, v_{k}) \) for some \( s \in S, v_{1}, \ldots, v_{k} \in V \) (Theorem A.14).

If \( \bot \notin \text{VALUE}(m_{i}) \supseteq \text{VALUE}(m) \) so \( v \neq \bot \). If \( v = s(v_{1}, \ldots, v_{k}) \), then for each \( i, v_{i} \in \text{VALUE}(m_{i}) \), so depth \( (v) \geq i \). It follows that depth \( (v) \) 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 \( m + n \) if and only if \( D \) contains the declaration mode \( m = e \) where \( e \) is an expression containing an occurrence of \( n \) not contained in the operand of any occurrence of ref.4

**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 \( \text{Int} \) in \( e \) by \( 0 \), and all occurrences of \( \text{ref} e' \) (for some expression \( e' \)) by \( \text{ref} \).

**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:

\[
*/ \text{See the remarks about notation at the Beginning of Appendix A.}
\]

**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 \( s \in S \), not the symbols of \( S \).

**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 \( k > 1 \) if and only if the \( k \)th 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 \( \mathcal{w} \) is a subvalue of \( \mathcal{v} \) if \( \mathcal{w} \) is the tree composed of some node of \( \mathcal{v} \) 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 \( \text{VALUE}(m) \), \( x \) a variable of mode \( m \). A subvalue \( \mathcal{w} \) of \( \mathcal{v} \) is accessible with respect to \( m \) if and only if there is some expression \( e[x] \) such that if \( x = \mathcal{v} \), then \( e[x] = \mathcal{w} \). \( v \) 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) \(<\text{expression}> ::=<\text{variable}>|<\text{selection}>|<\text{coercion}>\)

\(<\text{selection}> ::=<\text{positiveinteger}>\) of

\(<\text{coercion}> ::=<\text{declarator}>:<\text{expression}>\)

\(k\) of \( e \) is defined if and only if the mode of \( e \) begins with \text{struct} \( k \) for some \( k \geq k \).

Then the value of \( e \) must be of the form \text{struct} \( (v_{1}, \ldots, v_{k}) \) and the value of \( k \) of \( e \) is of mode \text{ref}_m \) for some \( k \geq 0 \).
In this case the value of $e$ is either $ref^k v$ for some $v$, or $ref^l$ for $l < k$. In the latter case $die = v$, in the latter $die = l$. (In case the mode of $e = ref^m$ for more than one value of $k$, choose the least such $k$.) Mode $3$ 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 $ref^m$ 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 $struct^k$.

(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) $\text{ref}^P_{m_k} i_1 \text{of} \ldots \text{of} i_{k-1} m_1 : \ldots : m_2 i_2 \text{of} \ldots \text{of} i_{2} m_1 : \ldots : m_i i_n \text{of} \ldots \text{of} i_{n} f_x$.

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 $m = ref^m$ for some $k > 0$. The unique solution to this equation in $M$ is the mode $ref ref ref \ldots$ 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 $m4: l \text{of} x$ to access the "second node" of $x$. Here $l \text{of} x$ is of mode $ref m4$ which is not equal to $m4$, so the coercion 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


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

$x \leq y$

if and only if

$x = \bot$ or $x = s(d_1, \ldots, d_k)$,

$y = s(d'_1, \ldots, d'_k)$

for some $k$, some $s \in \mathcal{O}_k$ 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 $C \times C$). $\leq$ 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 $\bot \in C$ such that $\forall x \in C, \bot \leq x$.

(iii) Each chain $\chi_0 \leq \chi_1 \leq \chi_2 \leq \cdots$ of elements of $C$ has a least upper bound $x = \text{lub}(\chi_i)$ -- i.e. an element satisfying the properties

a) $\chi_i \leq x \forall i$

b) if $\chi_i \leq y \forall i$ 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, $f : A \rightarrow B$

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 $\chi_0 \leq \chi_1 \leq \cdots$ in $A$, $f(\text{lub}(\chi_i)) = \text{lub}(f(\chi_i))$.

Notice that the monotone property of $f$ insures that $f(\chi_0) \leq f(\chi_1) \leq \cdots$ 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 $f : C \rightarrow C$ continuous.

Let $B \subseteq C, B = \{x | f(x) = x\}$, the set of fixed points of $f$. Then

(i) $B \neq \emptyset$

(ii) $B$ is a cpo

(iii) The least element $\bot$ of $B$ is denoted $\text{Yf}$ and may be computed by

$\text{Yf} = \text{lub}(f^n(\bot) | n \geq 0)$.

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 $A \leq B$ then $\leq:A \to B:x \mapsto x$ is continuous.

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

(v) $\pi_1:A \times B \to A:<x,y> \mapsto x$ and $\pi_2:A \times B \to B:<x,y> \mapsto y$ are continuous.

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

(vii) if $s \in \mathcal{C}_k$, then $s:D_k \to D_n+1$ is continuous.

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

$$
p_0(x) = 1 \forall x
$$

$$
p_{n+1}(1) = \bot
$$

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

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 $y \in D_n, y \leq x$, 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 $\mathcal{G} \cup \{\bot\}$ where $\bot$ 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 $\bot$ 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 $<t_0,t_1,\ldots>$ as follows: $t_1$ comes from $t$ by chopping off all branches after the first $i$ levels and replacing removed subtrees by $\bot$. 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 $<t_0,t_1,\ldots>$. |

A.10 Definition: Let $D = T(N)$ be the set of all sequences $<d_0,d_1,\ldots> \in D_0 \times D_1 \times D_2 \times \ldots$ such that $d_i = p_n(d_{i+1})$. Order $D$ by $<d_i> < <d_i>$ if and only if $d_i < d_i$ for each $n$.

Let $\pi_n:D \to D_i:<d_i> \mapsto d_i$.

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 $i_n:D_n \to D_{n+1}; x \mapsto x$

$$
i_n:D_n \to D_{n+1}; x \mapsto d_i
$$

where

$$
d_i = \left(\left.\begin{array}{c} x \kern1cm | \kern1cm i > n
\end{array}\right\right) \kern1cm \left(\left.\begin{array}{c} p_i(p_{i+1}(\ldots(p_{n-1}(x))\ldots) \\kern1cm | \kern1cm i < n
\end{array}\right\right)
$$

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

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 $d = <d_i>$ satisfies the property that

$$
d = \text{lub}\{(i_n(d_n)| n \geq 0)\}.
$$

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: $D = \{1\} \cup \{s(d_1, \ldots, d_k) | \kern1cm s \in \mathcal{C}_k, d_1 \in D\}$.

A.15 Lemma: for each $s \in \mathcal{C}_k, s:D^k \to D$ is continuous.

Proof: $s:D^k \to D_{n+1}$ 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 $\Omega$ 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 $n_1, \ldots, n_k$,

$$f : \Omega_n \times \Omega_n \times \cdots \times \Omega_n \to \Omega_n^+ n_1 n_2 \cdots n_k^+$$

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

$$\hat{f}(s_1(d_1, \ldots, d_{n_1}), \ldots, s_k(d_{kn_1}, \ldots, d_{kn_k})) = f(x_1, \ldots, s_1)(d_{11}, d_{12}, \ldots, d_{k n_k})$$

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 $\eta$ is a node of $d$, then there is a unique path $\xi$ from the root of $d$ to $\eta$. $\xi$ may be described by giving the sequence of choices made in passing from the root to $\eta$. If the $k$ sons of a node are labeled 1 to $k$, this assigns to each node $\eta$ a finite string ADDRESS($\eta$) of positive integers. (see figure B.1) Then for $s \in \Omega$, let the occurrence language of $s$ in $d$ be the set

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

More formally,

B.1 Definition: ([14]) For each $s \in \Omega^*$ and $n \geq 0$, define a mapping $L_s : D_n \to 2^*$ as follows:

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

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

*/ $2^*$ is the set of finite strings of positive integers.
Figure 2.1
A Value

Figure 2.2
The value of figure 2.1 unrolled

\[ m_0 = \bot \]
\[ m_1 = \text{struct}_2(\bot, \bot) \]
\[ m_2 = \text{struct}_2(\text{int}, \text{ref } \bot) \]
\[ m_3 = \text{struct}_2(\text{int}, \text{ref } \text{struct}_2(\bot, \bot)) \]
\[ m_4 = \text{struct}_2(\text{int}, \text{ref } \text{struct}_2(\text{int}, \text{ref } \bot)) \]
\[ \vdots \]
\[ m = \text{struct}_2(\text{int}, \text{ref } \text{struct}_2(\text{int}, \text{ref } \text{struct}_2(\ldots))) \]

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 3.2
A connected mode m, a value v of mode m, and a subvalue w of v of mode n

Figure 3.1
A value v which is not connected and an inaccessible subvalue w

addresses are shown in parentheses
ADDRESS(int) = (1, 2, 11, 2, 3)

Figure B.1
The ADDRESS function on modes and on symbols

ADDRESS(struct2) = {2^{k-1} \mid 0 \leq k \leq k}

Figure B.2
A mode which is not regular
ERRATA

Modes, Values, and Expressions†
by Marvin Solomon

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

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

ERRATA

Modes, Values, and Expressions†
by Marvin Solomon

Page Column Line For Read

2 left 8-10 declaration 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 coercion coercion
5 right -2 struct_k(v_1, ..., v_e) struct_\ell(v_1, ..., v_\ell)
5 right -1 is of mode \text{ref}^k_m is v. The only coercion we
for some k>0. 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 \text{ref}^k_m for some k>0.

8 right -1 Using A.11 and A.13 Using A.11 with E = D^F and
f_{n+1} = s \circ (\pi_{n'}, ..., \pi_n)
9 left 8 \mathit{f:}\Omega_{n_1} \times \Omega_{n_2} \times ... \times \Omega_{n_k} \mathit{f:}\Omega_{n_1} \times \Omega_{n_2} \times ... \times \Omega_{n_k}
+ \Omega_{n_1}+\Omega_{n_2}+...+\Omega_{n_k} \rightarrow \Omega_{n_1+n_2+n_3+...+n_k}
9 left 15 f(x_1, ..., s_1) f(s_1, ..., 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
ERRATA

Modes, Values, and Expressions†
by Marvin Solomon

Page  Column Line   For                              Read

2 left   8-10       declaration                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          coercion                  coercion
5 right  -2          \text{struct}_k(v_1,\ldots,v_e) \text{struct}_k(v_1,\ldots,v_e)
5 right  -1          is of mode \text{ref}^k_m is v. The only coercion we
                   for some k>0. 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 \text{ref}^k_m for some k>0.
6 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:Ω_{n_1} \times f_{n_2} \times \cdots \times Ω_{n_k} f:Ω_{n_1} \times Ω_{n_2} \times \cdots \times Ω_{n_k}
                   \rightarrow \Omega_{n_1+n_2+\cdots+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
### ERRATA

*Modes, Values, and Expressions*†

by Marvin Solomon

<table>
<thead>
<tr>
<th>Page</th>
<th>Column</th>
<th>Line</th>
<th>For</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>left</td>
<td>8-10</td>
<td>declaration</td>
<td>declaration</td>
</tr>
<tr>
<td>4</td>
<td>left</td>
<td>-16</td>
<td>let v =</td>
<td>let V =</td>
</tr>
<tr>
<td>4</td>
<td>right</td>
<td></td>
<td>After line 16 add:</td>
<td>(c.f. Lemma A.16)</td>
</tr>
<tr>
<td>5</td>
<td>left</td>
<td>-7</td>
<td>impossibility</td>
<td>possibility</td>
</tr>
<tr>
<td>5</td>
<td>right</td>
<td>-6</td>
<td>coercion</td>
<td>coercion</td>
</tr>
<tr>
<td>5</td>
<td>right</td>
<td>-2</td>
<td>$\text{struct}_k(v_1, \ldots, v_e)$</td>
<td>$\text{struct}_k(v_1, \ldots, v_k)$</td>
</tr>
<tr>
<td>5</td>
<td>right</td>
<td>-1</td>
<td>is of mode $\text{ref}_m^k$</td>
<td>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 $\text{ref}_m^k$ for some k&gt;0.</td>
</tr>
<tr>
<td>8</td>
<td>right</td>
<td>-1</td>
<td>Using A.11 and A.13 Using A.11 with E = $D^F$ and</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>left</td>
<td>8</td>
<td>$f: \Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \to \Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k}$</td>
<td>$f: \Omega_{n_1} \times \Omega_{n_2} \times \ldots \times \Omega_{n_k} \to \Omega_{n_1+n_2+n_3+\ldots+n_k}$</td>
</tr>
</tbody>
</table>

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

---

## ERRATA

*Modes, Values, and Expressions*†
by Marvin Solomon

<table>
<thead>
<tr>
<th>Page</th>
<th>Column</th>
<th>Line</th>
<th>For</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>left</td>
<td>8-10</td>
<td>declaration</td>
<td>declaration</td>
</tr>
<tr>
<td>4</td>
<td>left</td>
<td>-16</td>
<td>let v =</td>
<td>let V =</td>
</tr>
<tr>
<td>4</td>
<td>right</td>
<td>After line 16 add:</td>
<td>(c.f. Lemma A.16)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>left</td>
<td>-7</td>
<td>impossibility</td>
<td>possibility</td>
</tr>
<tr>
<td>5</td>
<td>right</td>
<td>-6</td>
<td>coercion</td>
<td>coercion</td>
</tr>
<tr>
<td>5</td>
<td>right</td>
<td>-2</td>
<td>(\text{struct}_{k}(v_1,\ldots,v_e))</td>
<td>(\text{struct}_{k}(v_1,\ldots,v_k))</td>
</tr>
<tr>
<td>5</td>
<td>right</td>
<td>-1</td>
<td>is of mode (\text{ref}^k_m)</td>
<td>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 (\text{ref}^k_m) for some (k&gt;0).</td>
</tr>
<tr>
<td>8</td>
<td>right</td>
<td>-1</td>
<td>Using A.11 and A.13 Using A.11 with (E = D^F) and (f_{n+1} = s \circ (\pi_{n'},\ldots,\pi_n))</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>left</td>
<td>8</td>
<td>(f: \prod_{n_1} \times \cdots \times \Omega \rightarrow \prod_{n_1} \times \cdots \times \Omega )</td>
<td>(f: \prod_{n_1} \times \cdots \Omega \rightarrow \prod_{n_1} \times \cdots \Omega )</td>
</tr>
<tr>
<td>9</td>
<td>left</td>
<td>15</td>
<td>(f(s_1,\ldots,s_k))</td>
<td>(f(s_1,\ldots,s_k))</td>
</tr>
</tbody>
</table>

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

ERRATA

Modes, Values, and Expressions†
by Marvin Solomon

Page Column Line  For Read

2 left 8-10 declaration 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 coercion coercion
5 right -2 $\text{struct}_k(v_1, \ldots, v_e)$ $\text{struct}_\ell(v_1, \ldots, v_\ell)$
5 right -1 is of mode $\text{ref}^k_m$ is v. The only coercion we
for some $k>0$. 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 $\text{ref}^k_m$ for some $k>0$.
8 right -1 Using A.11 and A.13 Using A.11 with $E = D^F$ and
\[ f_{n+1} = s \circ (\pi_n', \ldots, \pi_n) \]
9 left 8 $f: \Omega_1 \times f_2 \times \cdots \times \Omega_n$ $f: \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$
\[ \rightarrow \Omega_1^{n_1} \Omega_2^{n_2} \cdots \Omega_n^{n_k} \]
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
ERRATA

Modes, Values, and Expressions†
by Marvin Solomon

Page | Column | Line | For | Read
--- | --- | --- | --- | ---
2 | left | 8-10 | declaration | declaration
4 | left | -16 | let v = | let V =
4 | right | After line 16 add: | (c.f. Lemma A.16) | possibility
5 | left | -7 | impossibility | coercion
5 | right | -6 | coercion | coercion
5 | right | -2 | $\text{struct}_k(v_1, \ldots, v_e)$ | $\text{struct}_\ell(v_1, \ldots, v_\ell)$
5 | right | -1 | is of mode $\text{ref}^k_m$ | is v. The only coercion we will allow is dereferencing.
 | | | for some $k > 0$. | Thus the expression $d:e$ is defined if and only if $d$ is of some mode $m$ and $e$ is of mode $\text{ref}^k_m$ 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 | $f: \Omega_{n_1} \times f_{n_2} \times \cdots \times \Omega_{n_k}$ | $f: \Omega_{n_1} \times \Omega_{n_2} \times \cdots \times \Omega_{n_k}$
 | | | + $\Omega_{n_1} \Omega_{n_2} \cdots \Omega_{n_k}$ | + $\Omega_{n_1+n_2+n_3+\cdots+n_k}$
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.

ERRATA

Modes, Values, and Expressions†
by Marvin Solomon

Page Column Line For Read
2 left 8-10 declaration 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 coercion coercion
5 right -2 struct_k (v_1, ..., v_e) struct_g (v_1, ..., v_g)
5 right -1 is of mode ref^k_m is v. The only coercion we
for some k>0. 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 o (\pi_n', ..., \pi_n)
9 left 8 f: \Omega_{n_1} \times f: \Omega_{n_2} \times \cdots \times \Omega_{n_k} \times f: \Omega_{n_1} \times \Omega_{n_2} \times \cdots \times \Omega_{n_k}
+ \Omega_{n_1}^{n_2} + \cdots + \Omega_{n_k}^{n_1+n_2+n_3+ \cdots + n_k}
9 left 15 f(x_1, ..., s_1) f(s_1, ..., 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