THE LATTICE OF FLOW DIAGRAMS

by

Dana Scott

Oxford University Computing Laboratory

Programming Research Group

Oxford University Computing Laboratory
Oxford OX1 3QD

THE LATTICE
OF
FLOW DIAGRAMS

by

Dana Scott

Princeton University

Technical Monograph PRG-3

November 1970 (reprinted October 1978)

Oxford University Computing Laboratory,
Programming Research Group,
45 Banbury Road,
Oxford.

© 1970 Dana Scott

Department of Philosophy,
1879 Hall,
Princeton University,
Princeton, New Jersey 08540.

This paper also appears in *Symposium on Semantics of Algorithmic Languages*, Erwin Engeler (ed.), Lecture Notes in Mathematics Volume 188, Springer-Verlag, Heidelberg, 1971, and appears as a Technical Monograph by special arrangement with the publishers.

References in the literature should be made to the Springer Series, as the texts are identical and the Lecture Notes are generally available in libraries.

ABSTRACT

This paper represents a continuation of the program sketched in *Outline of a Mathematical Theory of Computation* (PRG 2). The language under consideration is the elementary language of flow diagrams where the level of analysis concerns the flow of control but not any questions of storage, assignment, block structure or the use of parameters. A new feature of the approach of this paper is the treatment of *both* syntax and semantics with the aid of complete lattices. This provides considerable unification of methods (especially in the use of recursive definitions) which can be applied to other languages. The main emphasis of the paper is on the *method* of semantical definition, and though the notion of equivalence of diagrams is touched upon a full algebraic formulation remains to be done.

CONTENTS

THE LATTICE

OF

FLOW DIAGRAMS

0. INTRODUCTION This paper represents an initial chapter in a development of a mathematical theory of computation based on lattice theory and especially on the use of continuous functions defined on complete lattices. For a general orientation, the reader may consult Scott (1970).

Let D be a complete lattice. We use the symbols:

$$\sqsubseteq,\ \bot,\ \top,\ \sqcup,\ \sqcap,\ \bigsqcup,\ \bigsqcap$$

to denote respectively the *partial ordering*, the *least* element, the *greatest* element, the *join* of two elements, the *meet* of two elements, the *join* of a set of elements, and the *meet* of the set of elements. The definitions and mathematical properties of these notions can be found in many places, for example Birkhoff (1967). Our notation is a bit altered from the standard notation to avoid confusion with the differently employed notations of set theory and logic.

The main reason for attempting to use lattices systematically throughout the discussion relates to the following well-known result of Tarski:

THE FIXED-POINT THEOREM. *Let* $f:D \to D$ *be a monotonic function defined on the complete lattice* D *and taking values also in* D. *Then* f *has a minimal fixed point* $p = f(p)$ *and in fact*

$$p = \bigsqcap\{x \in D : f(x) \sqsubseteq x\} .$$

For references and a proof see Birkhoff (1970), p. 115, and Bekić (1970). A function is called *monotonic* if whenever $x, y \in D$ and $x \sqsubseteq y$, then $f(x) \sqsubseteq f(y)$. Clearly, from the definition of p the element is $\sqsubseteq$ all the fixed points of f (if any). The only trick is to use the monotonic property of f to prove that p is indeed a fixed point. In the case of continuous functions we can be rather more specific.

Continuous functions preserve limits. It turns out that in complete lattices the most useful notion of limit is that of forming the join of a directed subset. A subset $X \subseteq D$ is called *directed* if every finite subset of X has an upper bound (in the sense of $\sqsubseteq$) belonging to X. This applies to the empty subset, so X must be non-empty. This also applies to any pair $x, y \in X$, so there must exist an element $z \in X$ with $x \sqcup y \sqsubseteq z$. An obvious example of a directed set is a *chain*:

$$X = \{x_0, x_1, \ldots, x_n, \ldots\}$$

where

$$x_0 \sqsubseteq x_1 \sqsubseteq \ldots \sqsubseteq x_n \sqsubseteq \ldots .$$

The *limit* of the directed set is the element $\bigsqcup X$. In the case of a chain (or any sequence for that matter) we write the limit (join) as:

$$\bigsqcup_{n=0}^{\infty} x_n .$$

A function $f:D \to D$ is called *continuous* if whenever $X \subseteq D$ is directed, then

$$f(\bigsqcup X) = \bigsqcup\{f(x): x \in X\} .$$

It is easy to show that continuous functions are monotonic. Note, too, that the definition also applies to functions $f:D \to D'$ between two different lattices; in which case we read the right-hand side of the above equation as the join-operation in the second lattice D'. In the case of continuous $f:D \to D$, the fixed point turns out to be:

$$p = \bigsqcup_{n=0}^{\infty} f^n(\perp)$$

where $f^0(x) = x$ and $f^{n+1}(x) = f(f^n(x))$.

This all seems very abstract, but there is a large variety of quite useful complete lattices, and the fixed-point theorem is exactly the right way in which to introduce functions defined by *recursion*. This has been known for a long time, but the novelty of the present study centers around the *choice* of the lattices to which this idea may be applied. In particular, we are going to show that the familiar flow diagrams can be embedded in a useful way in an interesting complete lattice, and *then* that the semantics of flow diagrams can be obtained from a continuous function defined with the aid of fixed points. Of course, this is only one small application of the method, but it should be instructive.

1. FLOW DIAGRAMS. Intuitively, a flow diagram looks very roughly like Figure 1. There is a distinguished *entry point* into which the input information "flows" and an *exit point* out of which the result or output will (hopefully) come. The main question, then, is what goes on inside the "black box". Now, the box may represent a *primitive operation* which we do not analyze further, or the box may be compounded from other diagrams.

A trivial example of compounding may be the combination of *no* diagrams whatsoever. The result is the "straight arrow" of Figure 2.

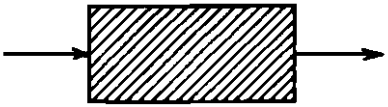


Figure 1
A FLOW DIAGRAM

Figure 2
THE IDENTITY

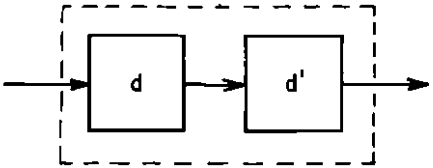


Figure 3
A PRODUCT

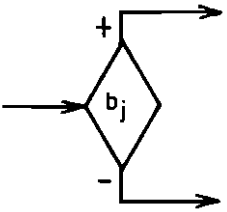


Figure 4
A SWITCH

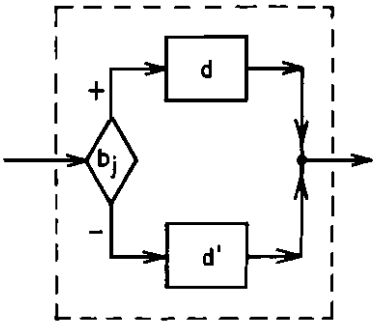


Figure 5
A SUM

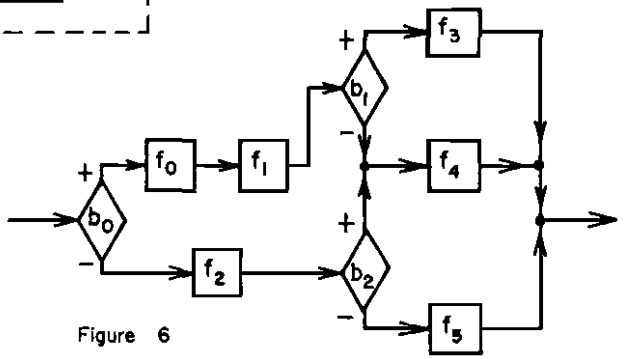


Figure 6
A LARGE DIAGRAM

The information flowing along such a channel exits untransformed; and so that diagram represents the *identity function*. A non-trivial compound is shown in Figure 3. In this combination, called a *product*, the output of the first box is fed directly into the input of the second with the obvious result.

With products alone not much useful could be done. As information flows, it must be tested and switched into proper channels according to the outcomes of the tests. For these switches we shall assume for simplicity in this paper that a *fixed* stock of primitive ones are given. This is not a serious restriction, and the method can just as well be applied when various forms of compounding of switches are allowed. We shall assume, by the way, that information flowing through a switch, though tested, exits untransformed. In diagrams a switch is represented as in Figure 4. In case the result of the test is positive, the information flows out of the top; if negative, from the bottom. A switch by itself is not a flow diagram because it has two exits. If these "wires" are attached to the inputs of the two boxes, and then if the outputs of the two boxes are brought together, we have a proper flow diagram. It is shown in Figure 5. We call this construction a *sum* (of the two boxes) for short, but it is also called a *conditional* because the outcome is conditional on the test.

Sums and products are the basic compounding operations for flow diagrams; iterating them leads to large diagrams such as the one shown in Figure 6. Here, the primitive boxes and switches have been labeled for reference and to distinguish them. The attentive reader will notice that we have cheated in the diagram in that the (-) and (+) leads from b_1 and b_2 have been brought together. The reason for doing this was to avoid duplicating box f_4. Strictly speaking, such shortcuts are *not* allowed: all repetitions must be written out. The diagrams will thus have a "tree" structure with switches at the

branch points and with strings of boxes (any number including zero) along the branches. (We draw these trees sideways.) At the "top" of the tree all the leads are brought together for the output.

What is wrong in Figure 6? That is to say, what is lacking? Obviously, the answer is that there are no *loops*, all good flow diagrams permit feedback around loops. The proper way to allow looping is discussed in the next section; first, we must connect flow diagrams in the intuitive sense with the mathematical theory of lattices.

Some notation will help. We have already used the notation

$$b_0, b_1, b_2, \cdots ,$$
$$f_0, f_1, f_2, \cdots$$

for the switches and boxes, respectively. (The "*b*" recalls *Boolean* or *binary*; while the "*f*" is used because the boxes represent *functions* on information.) For the identity (or "dummy") diagram we may use the notation I. Suppose d and d' are two diagrams, then the *product* is denoted by:

$$(d;d')$$

where the order is the same left-right order as in Figure 3. The *sum* is written:

$$(b_j \rightarrow d,d')$$

which is the familiar "conditional expression" used here in an adapted form for diagrams. The diagram of Figure 6 may now be written as:

$$(b_0 \rightarrow (f_0;(f_1;(b_1 \rightarrow f_3,f_4))),(f_2;(b_2 \rightarrow f_4,f_5))) .$$

This expression has many too many parentheses, but we shall have to discuss problems of equivalence before we can eliminate any. In any case, it is clear that instead of diagrams we may talk of *expressions* generated from the f_i and I by repeated applications of the various sum and product operations. The expressions may get long, but it is a bit more obvious what we are talking about.

The totality of all expressions obtained in the way described above is a natural and well-determined whole, but just the same, we are going to embed it in a much larger complete lattice by a method similar to the expansion of the rationals to the reals. The first step is to introduce a sense of *approximation*, and the second step is to introduce *limits*. In our particular case, a very convenient way to achieve the desired goal is to introduce *approximate* (or: *partial*) expressions which interact with the "perfect" expressions we already know in useful ways not directly analogous to the common notion of approximation in the reals. (There is an exactly parellel way to treat reals, however.) Existing between approximate expressions is a partial ordering relation $\sqsubseteq$ which provides the required sense of approximation of one expression by another. We now turn to the details of setting up this relation.

If the relationship

$$d \sqsubseteq d'$$

between partial diagrams is to mean that d *approximates* d', then it seems very likely that in a large number of cases d can approximate many *different* d'. In particular, we may as well also assume the existence of the *worst* (or most incomplete) diagram $\perp$ which approximates *everything*; that is,

$$\perp \sqsubseteq d'$$

will hold for all d'. In pictures we may draw $\perp$ as a "vague" box whose contents are undetermined. Now, these incomplete boxes may occur as *parts* of other diagrams, as has been indicated in Figure 7. The expression for Figure 7 of course would be written as:

$$(b_0 \rightarrow (f_0;\perp),(f_1;(b_1 \rightarrow f_2,I))) \ .$$

If we are going to allow incomplete parts of diagrams, then we must also allow ourselves the option of *filling in* the missing parts.

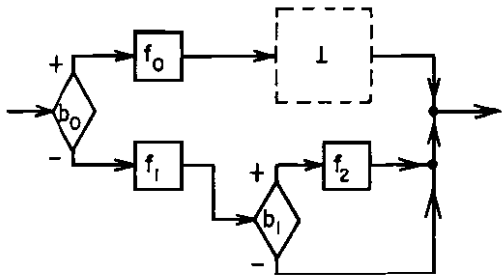


Figure 7
AN INCOMPLETE DIAGRAM

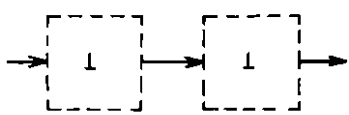


Figure 8
A PRODUCT OF
TWO INCOMPLETES

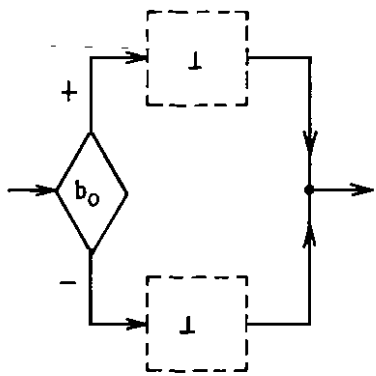


Figure 9
A SUM OF TWO
INCOMPLETES

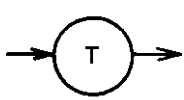


Figure 10
THE OVERDETERMINED
DIAGRAM

Thus, if d is incomplete, then a more precise reading of the relationship

$$d \sqsubseteq d'$$

is that d' is like d except that some of the parts left vague in d have been filled in. That reading is quite correct for the relationship $\perp \sqsubseteq d'$ that must always hold. In compound cases we can assure the desired results by assuming that sum and product formations are *monotonic* in the following precise sense:

$$\begin{aligned} &\textit{if } d_0 \sqsubseteq d_1 \textit{ and } d_0' \sqsubseteq d_1', \textit{ then} \\ &\quad (d_0;d_0') \sqsubseteq (d_1;d_1') \textit{ and} \\ &\quad (b_j \rightarrow d_0, d_0') \sqsubseteq (b_j \rightarrow d_1, d_1') \ . \end{aligned}$$

Besides this, the relation $\sqsubseteq$ must be assumed to be *reflexive*, *transitive*, and *antisymmetric* ($\sqsubseteq$ is a partial ordering).

As an illustration we could fill in the box of Figure 7 and prove by the above assumptions that:

$$(b_0 \rightarrow (f_0;\perp),(f_1;(b_1 \rightarrow f_2,I))) \sqsubseteq (b_0 \rightarrow (f_0;(f_1;f_0)),(f_1;(b_1 \rightarrow f_2,I)))$$

In working out these relationships it seems reasonable to assume in addition that:

$$(\perp;\perp) = \perp$$

but *not* to assume that:

$$(b_0 \rightarrow \perp,\perp) = \perp \ ,$$

as may be appreciated from the pictures in Figures 8 and 9.

For the sake of mathematical symmetry (and to avoid making exceptions in certain definitions) we also introduce an exceptional diagram denoted by $\top$ about which we assume:

$$d \sqsubseteq \top$$

for all d. We can think of $\perp$ as being the *underdetermined* diagram, and $\top$ as being *overdetermined*. The diagram $\top$ is something like a short circuit -- we will make its "meaning" quite precise in the section on semantics. We assume that

$$(\top;\top) = \top \ ,$$

but *not* that

$$(b_j \to \top,\top) = \top \ ,$$

again for reasons that will be semantically motivated. Other equations that might seem reasonable (say, $(d;\top) = \top$) are postponed to the discussion of equivalence.

Taking stock of where we are now, we can say that we begin with certain "atomic" symbols (representing elementary diagrams); namely:

$$\bot, f_0, f_1, \ldots, f_n, \ldots, I, \top \ .$$

Then, we form all combinations generated from these using:

$$(d;d') \text{ and } (b_j \to d,d').$$

These expressions are partially ordered by a relation $\sqsubseteq$ about which we demand first that

$$\bot \sqsubseteq d \sqsubseteq \top$$

for all d; and then which we subject to the reflexive, transitive, and monotonic laws (the so-generated relation will automatically be antisymmetric).

This is the "symbolic" method which is quite reasonable and is well motivated by the pictures. We could even pursue it further and make the totality of expressions into a lattice in the following way. The join and meet operations must satisfy these laws:

$$d \sqcup d' = d' \sqcup d \qquad d \sqcap d' = d' \sqcap d$$

$$d \sqcup d = d \qquad d \sqcap d = d$$

$$d \sqcup \bot = d \qquad d \sqcap \bot = \bot$$

$$d \sqcup \top = \top \qquad d \sqcap \top = d \ .$$

In addition for the atomic expressions other than $\bot$ and $\top$ we stipulate:

$$f_i \sqcup f_j = \top \qquad f_i \sqcap f_j = \bot$$

$$f_i \sqcup I = \top \qquad f_i \sqcap I = \bot \ ,$$

where $i \neq j$. For the case of products we have:

$$(\bot,\bot) = \bot \qquad (\top;\top) = \top \ ,$$

and in the following assume that the pair d,d' is not either of the exceptional pairs $\bot,\bot$ or $\top,\top$:

$$f_i \sqcup (d;d') = \top \qquad f_i \sqcap (d;d') = \bot$$
$$(b_j \to d_0,d_0') \sqcup (d;d') = \top \qquad (b_j \to d_0,d_0') \sqcap (d;d') = \bot$$
$$f_i \sqcup (b_j \to d_0,d_0') = \top \qquad f_i \sqcap (b_j \to d_0,d_0') = \bot$$
$$I \sqcup (d;d') = \top \qquad I \sqcap (d;d') = \bot$$
$$I \sqcup (b_j \to d_0,d_0') = \top \qquad I \sqcap (b_j \to d_0,d_0') = \bot$$

where d_0,d_0' is arbitrary. Moreover, for any two pairs d_0,d_0' and d_1,d_1' we assume:

$$(d_0;d_0') \sqcup (d_1;d_1') = (d_0 \sqcup d_1;d_0' \sqcup d_1')$$
$$(d_0;d_0') \sqcap (d_1;d_1') = (d_0 \sqcap d_1;d_0' \sqcap d_1')$$
$$(b_j \to d_0,d_0') \sqcup (b_j \to d_1,d_1') = (b_j \to d_0 \sqcup d_1,d_0' \sqcup d_1')$$
$$(b_j \to d_0,d_0') \sqcap (b_j \to d_1,d_1') = (b_j \to d_0 \sqcap d_1,d_0' \sqcap d_1') \ .$$

Finally, it *might* seem reasonable to assume:

$$(b_j \to d_0,d_0') \sqcup (b_k \to d_1,d_1') = \top$$
$$(b_j \to d_0,d_0') \sqcap (b_k \to d_1,d_1') = \bot$$

when $j \neq k$; but we postpone this decision.

This large number of rules allows us to compute joins and meets for any two expressions (in a recursive way running from the longer to the shorter expression), and it could be shown that in this manner the expressions do indeed form a lattice with the $\sqsubseteq$ relation as the partial ordering. The proof would be long and boring, however, as is always the case with symbolic methods. The reasons one must exercise care in this approach are in the main these two: one must be sure that all cases are covered, and one must be certain that different orders in carrying out symbolic operations do not lead to inconsistent

results. Now, it would be quite possible to do all this for our construction of the lattice of diagrams, but it is quite unnecessary because a better method is available.

The idea of the better approach is to work with structures that are *known* to be lattices from the very start; hence we shall never have to check the lattice laws except in some trivial cases. Next, some operations on structures are carried out which are known to transform lattices into lattices (in our case this will correspond to the formation of compound expressions). Finally, (and this is the main virtue of the approach) the extension to a *complete* lattice may be described in a neat way. The lattice of expressions to the extent to which it has been apprehended up to this point is *not* complete; and the adjunction of limits requires a certain amount of care: the structural approach will make the exercise of this care more or less automatic. It must be stressed, however, that after the desired structures are created as lattices a certain amount of argument is required to see that the structures conform to our intuitive ideas about expressions. Though necessary, this will not be difficult, as we demonstrate in the next section.

2. CONSTRUCTING LATTICES. The initial part of the lattice we are trying to construct corresponds to the atomic symbols $f_0, f_1, \ldots$ and I. Since these symbols play slightly different roles, we separate I from the others. Now, all we really know about the f_i is that they are pairwise distinct; hence it will be sufficient to represent them by elements of a lattice illustrated in Figure 11. In such pictures of lattices the partial ordering is represented by the ascending lines; the weaker (smaller) elements are below and the stronger (larger) elements are above. (By the way, a lattice is *not* a flow diagram; the two kinds of pictures should not be confused. We are trying to make flow diagrams *elements* of a lattice.) What the

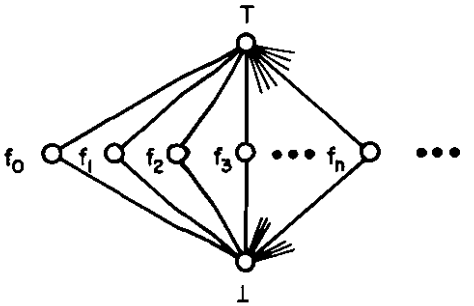


Figure 11

THE LATTICE F

Figure 12

THE LATTICE $\{I\}$

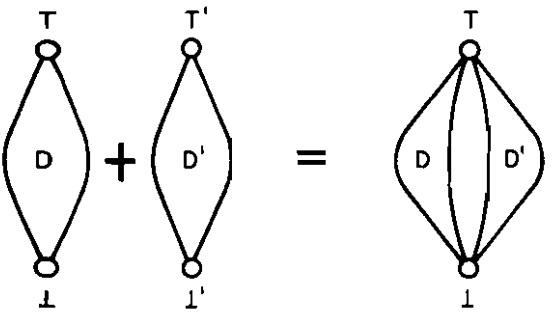


Figure 13

THE SUM OF TWO LATTICES

picture of the lattice F in Figure 11 shows is that the only partial ordering relations allowed are:

$$\perp \sqsubseteq f_i \sqsubseteq \top$$

for all i.

In Figure 12 we have a representation of the lattice $\{I\}$ which beside the $\perp$ and $\top$ elements has only one main element I. It should be mentioned in setting up these partial orderings that to check that they form complete lattices means that every subset of the partially ordered set must have a least upper bound (its join) in the sense of the partial ordering. In the two cases we have so far the result is obvious.

Suppose now that D and D' are two given complete lattices with partial orderings $\sqsubseteq$ and $\sqsubseteq'$, respectively. Inasmuch as it is only structure that is important, we may assume as sets of elements that D and D' are *disjoint*. We wish to combine D and D' together in one unified lattice: it will be called the sum of the two lattices and will be denoted by

$$D+D' \ .$$

Essentially, it is just the *union* of the two sets structured by the "union" of the two partial ordering relations. This partial ordering is not a lattice, however, because there is no largest and no smallest element. These could be adjoined from the outside, but a more convenient and more "economical" procedure is as follows. Let $\top,\top'$ and $\perp,\perp'$ be the largest and smallest elements of D and D', respectively. We have been regarding them as distinct (all the elements of D were to be distinct from the elements of D'), but now just these two pairs will be made equal. That is, we shall decree for D+D' that $\top=\top'$ and $\perp=\perp'$; though all the other elements are kept separate. The resulting partial ordering is easily seen to be a complete lattice. The process of forming this sum of lattices is illustrated in Figure 13.

The initial lattice of atomic expressions (diagrams) we wish to consider, then, is the lattice:

$$F+\{I\} .$$

It will be noted that the notion of sum just introduced could easily be extended to infinitely many factors. Thus, if we considered lattices $\{f_i\}$ that structurally were isomorphic to $\{I\}$ (but with different elements), then the lattice F could be defined by:

$$F = \{f_0\}+\{f_1\}+\ldots+\{f_n\}+\ldots .$$

Though they are not by themselves atomic expressions, the symbols b, will also be thought of as elements of a lattice B defined by:

$$B = \{b_0\}+\{b_1\}+\ldots+\{b_n\}+\ldots .$$

The lattices F, B, and $F+\{I\}$ are all isomorphic as lattices but are different because they have different elements. These, however, are very trivial lattices, and we need much more complicated structures.

Suppose D and D' are lattices whose elements represent "diagrams" we wish to consider. If we want to form products of diagrams, then according to the intuitive discussion in the last section, the partial ordering on products should be defined so that

$$(d_0;d_0') \sqsubseteq (d_1;d_1') \textit{ if and only if } d_0 \sqsubseteq d_1 \textit{ and } d_0' \sqsubseteq' d_1'$$

for all d_0, $d_1 \in D$ and all d_0', $d_1' \in D'$. Abstractly, we usually write $< d_0,d_0' >$ as an *ordered pair* in place of $(d_0;d_0')$, and then write:

$$D\times D'$$

for the set of all ordered pairs $< d,d' >$ with $d \in D$ and $d' \in D'$. The above biconditional defines a partial ordering on $D\times D'$ called the (cartesian) product ordering, and, as is well known, the result is again a complete lattice. The largest and smallest elements of $D\times D'$ are the pairs $< \tau,\tau' >$ and $< \iota,\iota' >$, respectively.

Let the lattice

$$D_0 = F+\{I\} .$$

be the lattice of atomic expressions. Then the lattice

$$D_0+(D_0\times D_0)$$

could be regarded as the lattice which in addition to the atomic expressions has compound expressions which can be thought of as products of two atomic expressions. In fact, there is no compelling reason to use the abstract notation $< d,d' >$: we can use the more suggestive $(d;d')$ remembering that lattice-theoretically this is just an ordered pair. Notice in this regard that by our definitions of sums and products of lattices we have the equations

$$(\bot;\bot) = \bot \qquad (\top;\top) = \top$$

automatically.

What about diagrams? Well, even though we wrote

$$(b_j \to d,d') \ ,$$

abstractly all we have is an ordered triple

$$< b_j,d,d' > \ .$$

This is just an element of the lattice

$$B\times D\times D \ .$$

(if the reader wants to be especially pedantic he can take $B\times D\times D$ to be $B\times(D\times D)$ and $< b_j,d,d' > = < b_j,< d,d' >>$, or he can introduce an independent notion of ordered triple. Structurally, all approaches give isomorphic lattices.) Hence, the next lattice we wish to consider would be

$$D_1 = D_0+(D_0\times D_0)+(B\times D_0\times D_0)$$

Again, there is no reason to use the abstract notation so that $(b_j \to d,d')$ can just as well stand for an ordered triple. Notice that we have in this way introduced some elements not considered as diagrams before:

$$(\top \to d,d') \text{ and } (\bot \to d,d') \ ;$$

but we shall find that it is easy to interpret them semantically, so that this extra generality costs us no special effort. If we

like, we can also use the more suggestive notation for the lattices themselves and write:

$$D_1 = D_0 + (D_0 ; D_0) + (B \to D_0 , D_0) \ ,$$

but for the time being it may be better to retain the abstract notation to emphasize the fact that we know all these structures as lattices.

Clearly, D_1 contains as elements only very short diagrams. To obtain the larger diagrams we must proceed recursively, iterating our compounding of expressions. Abstractly, this means forming ever more complex lattices:

$$D_{n+1} = D_0 + (D_n \times D_n) + (B \times D_n \times D_n) \ .$$

The way we are construing the elements of these lattices, D_0 is a *subset* of each D_n:

$$D_0 \subseteq D_n \ ;$$

and, in fact, D_0 is a *sublattice*. This means that partial ordering on D_0 is the *restriction* of the intended partial ordering on D_n (restricted to the subset). *And* besides, the join of any subset of D_0 formed within the lattice D_0 is exactly the same as the join formed within D_n. (This last is very important to remember.) The same goes for meets, but this fact is not so important.

Consider that

$$D_0 \subseteq D_1 \ ,$$

and that this implies that

$$D_0 \times D_0 \subseteq D_1 \times D_1$$

both as a subset *and* as a sublattice. Similarly, we have:

$$B \times D_0 \times D_0 \subseteq B \times D_1 \times D_1 \ .$$

It then follows that

$$D_0 + (D_0 \times D_0) + (B \times D_0 \times D_0) \subseteq D_0 + (D_1 \times D_1) + (B \times D_1 \times D_1)$$

both as a subset and as a sublattice. By definition we have:

$$D_1 \subseteq D_2 ,$$

and continuing in this way, we prove:

$$D_n \subseteq D_{n+1} .$$

Therefore,

$$D_n \subseteq D_m$$

whenever $n \leq m$.

What we have just done is to take advantage of general properties of the sum and product constructions on lattices as regards sublattices. These general properties about the comparisons of the partial orderings and the joins and meets are very simple to prove abstractly, and the reader is urged to work out the details for himself including the assertions of the last paragraph. As a result of these considerations it will be seen that the *union set*

$$\bigcup_{n=0}^{\infty} D_n$$

has a coherent partial ordering. Is this a lattice? It is not a complete lattice (we shall see why, later). On the other hand, many joins and meets do exist; in particular, the join of every *finite* subset exists in the union. (The reason is that any finite subset is wholly contained in one of the D_n.) So, the union of the lattices is a finitely complete lattice (a kind of structure that is ordinarily called just a lattice).

What are the elements of this union lattice? They are exactly all the finite combinations we desired generated from the atomic diagrams by means of the two modes of composition. Furthermore, the abstract lattice structure obtained in this way provides perfectly all the laws of computation we listed in the last section. Thus, the abstract approach gives us a structure which we know is a (finitely complete) lattice on the basis of simple, general principles. Then, by reference to the construction, the laws of computation are worked

out. Having worked them out in this case, we can see by inspection of cases that we have all we need because there are only a limited number of types of elements formed in an iterative fashion. The next step is to complete the lattice and then to figure out what is obtained.

3. COMPLETING THE LATTICE. Every lattice can be completed (as in Birkhoff (1967), p126), but we shall want to complete the lattice of flow diagrams in a special way that allows us to apprehend the nature of the limit elements very clearly. In particular, the notion of approximation will be made quite precise.

Roughly speaking, the elements of the lattice D_n are diagrams of "length" at most n. More exactly, they can be obtained from the generators by nesting the two modes of composition to a level of at most n. This suggests that the elements of D_{n+1} might be approximable by elements of D_n. Consider D_0 and D_1. If $d \in D_1$, then it may belong to $D_0 \subseteq D_1$ or it may not. If $d \in D_0$, then it is its own best approximation. If $d \notin D_0$, then since the elements of D_0 are not compounds (except in a trivial sense) the best we can do in D_0 is to approximate d by $\perp$. In other words, we have defined a mapping

$$\psi_0 : D_1 \to D_0 ,$$

where for $d \in D_1$ we have:

$$\psi_0(d) = \begin{cases} d & \text{if } d \in D_0 ; \\ \perp & \text{if not.} \end{cases}$$

As can easily be established this mapping is *continuous* (in fact, a more general theorem relating to sum formation of lattices is provable), and this is important as all the mappings we employ ought to be continuous.

Now, consider D_{n+2} and D_{n+1}. We wish to define

$$\psi_{n+1} : D_{n+2} \to D_{n+1} .$$

For $d \in D_{n+2}$, the element $\psi_{n+1}(d)$ will be the best approximation to d by an element in D_{n+1}. Recall that

$$D_{n+1} = D_0+(D_n\times D_n)+(B\times D_n\times D_n)$$

and

$$D_{n+2} = D_0+(D_{n+1}\times D_{n+1})+(B\times D_{n+1}\times D_{n+1}) \ .$$

Inductively, we may assume that we have already defined the mapping

$$\psi_n : D_{n+1} \to D_n \ .$$

Clearly, what is called for is this definition:

$$\psi_{n+1}(d) = \begin{cases} d & \text{if } d \in D_0 \ ; \\ (\psi_n(d');\psi_n(d")) & \text{if } d = (d';d") \ ; \\ (b \to \psi_n(d');\psi_n(d")) & \text{if } d = (b \to d',d") \ . \end{cases}$$

Now, these three cases are strictly speaking *not* mutually exclusive, but on the only possibilities of overlap we find agreement because $\psi_n(\tau) = \tau$ and $\psi_n(\iota) = \iota$. By a proof that need not detain us here, we show that ψ_{n+1} is continuous. Note also that we may prove inductively for all n that for $d \in D_{n+1}$ we have:

$$d \in D_n \text{ if and only if } \psi_n(d) = d \ .$$

The mapping $\psi_n : D_{n+1} \to D_n$ is easily illustrated. In Figure 14 two diagrams are given: the first belongs to D_6 and the second is the result of applying ψ_5 to the first. It will be noted that drawn diagrams are slightly ambiguous; this ambiguity is removed when one choses an expression for the diagram. In this example we chose to associate to the right and to interpret a long arrow without boxes as a single occurrence of I and not as a product of several I's. The upper diagram is complete; while what we might call its *projection* from D_6 into D_5 is necessarily incomplete. Clearly, we can recapture the upper figure by removing the vagueness in one position of the lower figure. This is the way approximation works. It is a very simple idea.

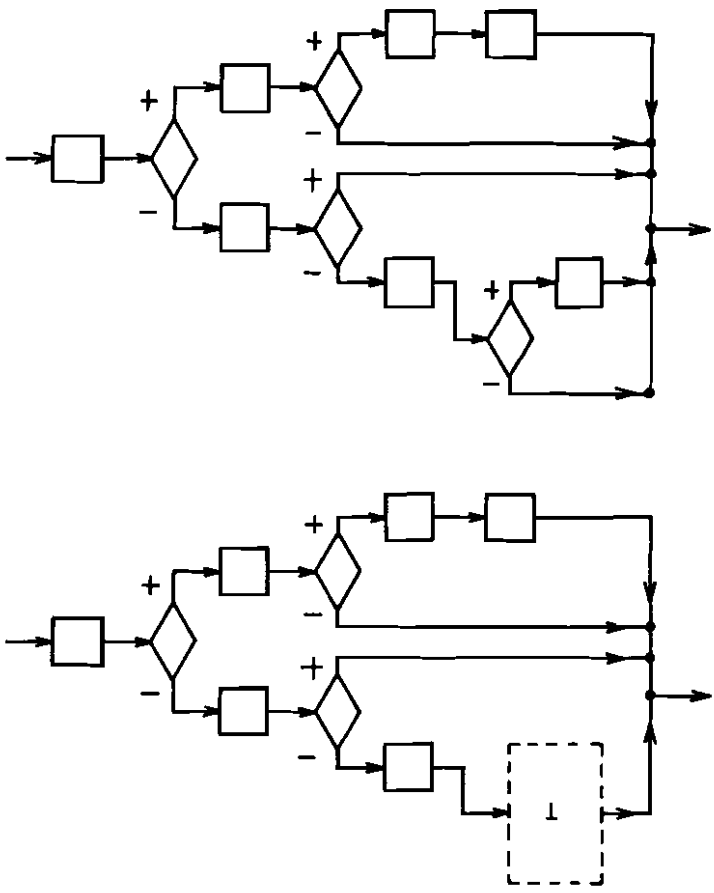


Figure 14

PROJECTING A DIAGRAM

Suppose $d' \in D_{n+1}$ and $d \in D_n$ and $d \sqsubseteq d'$. Now ψ_n is not only continuous, but also monotonic. Thus,

$$d = \psi_n(d) \sqsubseteq \psi_n(d') \sqsubseteq d' .$$

We have therefore shown that $\psi_n(d')$ is the *largest* element of D_n which approximates d'. This reinforces our conception of ψ_n as a projection of D_{n+1} upon D_n; the idea will now be carried a step further.

Assume that we already knew how to complete the union

$$\bigcup_{n=0}^{\infty} D_n$$

to a complete lattice D_∞. If we were to be able to preserve relationships, we ought to be able to project D_∞ successively onto each D_n, say by a mapping

$$\psi_{\infty n} : D_\infty \to D_n .$$

But these projections really should fit hand in glove with the projections we already have. One way of expressing the goodness of fit is by the functional equation

$$\psi_{\infty n} = \psi_n \circ \psi_{\infty(n+1)}$$

which means that the projection from D_∞ onto D_{n+1} followed by the projection from D_{n+1} onto D_n ought to be exactly the projection from D_∞ onto D_n. Suppose this is so.

Now, let $d \in D_\infty$ be any element of this ultimate lattice. Define a sequence of elements in the known lattices by the equation:

$$d_n = \psi_{\infty n}(d)$$

for all n. By what we have conjectured

$$\psi_n(d_{n+1}) = d_n$$

holds for all n, and so

$$d_0 \sqsubseteq d_1 \sqsubseteq \ldots \sqsubseteq d_n \sqsubseteq d_{n+1} \sqsubseteq \ldots .$$

How does the limit element d fit into the picture? Easy. We claim

$$d = \bigsqcup_{n=0}^{\infty} d_n \ .$$

Since $\psi_{\infty n}$ is a projection, we at least have $d_n \sqsubseteq d$ for all n; thus, the limit of the d_n must also approximate d. But why the equality? Well, since D_∞ is to be the completion of the union, each element of D_∞ is determined as the directed join (limit) of all elements of the union $\sqsubseteq$ it. (All elements of D_∞ must be approximable as closely as we please by elements from the union lattice.) If d' belongs to the union and $d' \sqsubseteq d$, then since $d' \in D_n$ for some n we must have $d' \sqsubseteq d_n$. Hence the equality.

We have seen that each element $d \in D_\infty$ determines a sequence

$$< d_n >_{n=0}^{\infty}$$

such that

$$\psi_n(d_{n+1}) = d_n$$

holds for all n. Furthermore, distinct elements of D_∞ determine distinct sequences. (Because each d is determined as the limit of its corresponding sequence.) Suppose conversely that such a sequence of elements $d_n \in D_n$ is given and we *define* $d \in D_\infty$ by the equation

$$d = \bigsqcup_{n=0}^{\infty} d_n \ .$$

We are going to prove that for all n:

$$d_n = \psi_{\infty n}(d) \ .$$

In the first place, since these projections are continuous we have:

$$\psi_{\infty n}(d) = \bigsqcup_{m=0}^{\infty} \psi_{\infty n}(d_m) \ .$$

For $m \leq n$, since $d_m \in D_n$, it follows that

$$\psi_{\infty n}(d_m) = d_m \ .$$

For $m \geqslant n$, we are going to prove that

$$\psi_{\infty n}(d_m) = d_n \ .$$

This is true for $m = n$. We argue by induction on the quantity $(m-n)$. Having just checked it for the value 0, suppose the value is positive and that we know the result for the previous value. Thus, $m > n$. We use the equation relating the various projections and compute:

$$\begin{aligned} \psi_{\infty n}(d_m) &= \psi_n(\psi_{\infty(n+1)}(d_m)) \\ &= \psi_n(d_{n+1}) \\ &= d_n \ . \end{aligned}$$

Then since the required equation is proved we see:

$$\begin{aligned} \psi_{\infty n}(d) &= d_0 \sqcup d_1 \sqcup \ldots \sqcup d_n \sqcup d_n \sqcup d_n \sqcup \ldots \\ &= d_n \ . \end{aligned}$$

That is to say, in the infinite join all the terms after $n = m$ are d_n but the previous ones are $\sqsubseteq d_n$ anyway.

In other words, we have shown that there is a *one-one correspondence* between the elements of D_∞ and the sequences $\langle d_n \rangle_{n=0}^{\infty}$ which satisfy the equations

$$\psi_n(d_{n+1}) = d_n \ .$$

Mathematically, this is very satisfactory because it means that instead of *assuming* that we know D_∞ we can *construct* it as actually *being* the set of these sequences. In this way, our intuitive ideas are shown to be mathematically consistent. This construction is particularly pleasant because the partial ordering on D_∞ has this easy sequential definition:

$d \sqsubseteq d'$ if and only if $d_n \sqsubseteq d'_n$ for all n.

The other lattice operations also have easy definitions based on the sequences. But for the moment the details need not detain us. All we really need to know is that the desired lattice D_∞ does indeed exist and that the projections behave nicely. In fact, it can be

proved quite generally that each $\psi_{\infty n}$ is not only continuous but also is *additive* in the sense that

$$\psi_{\infty n}(\bigsqcup X) = \bigsqcup\{\psi_{\infty n}(x) : x \in X\}$$

for *all* $X \subseteq D_\infty$. Hence, we can obtain a reasonably clear picture of the lattice structure of D_∞. But D_∞ has "algebraic" structure as well, and we now turn to its examination.

4. THE ALGEBRA OF DIAGRAMS. Because the D_n were constructed in a special way, the complete lattice D_∞ is much more than just a lattice. Since we want to interpret the elements of D_∞ as diagrams, we replace the more abstract notation of the previous section by our earlier algebraic notation. Thus, by construction, if d, $d' \in D_n$ and if $b \in B$, then both

$$(d;d') \text{ and } (b \to d,d')$$

are elements of D_{n+1}. What if d, $d' \in D_\infty$? Will these algebraic combinations of elements make sense?

In order to answer this interesting question, we shall employ for elements $d \in D_\infty$ this abbreviated notation for projection:

$$d_n = \psi_{\infty n}(d)$$

Remember that we regard each $D_n \subseteq D_\infty$ and so d_n is the *largest element of* D_n *which approximates* d. If $d \in D_{n+1}$, then

$$d_n = \psi_n(d)$$

also.

Using these convenient subscripts, we may then define for d, $d' \in D_\infty$:

$$(d;d') = \bigsqcup_{n=0}^{\infty}(d_n;d'_n) \quad , \qquad \text{and} \qquad (b \to d,d') = \bigsqcup_{n=0}^{\infty}(b \to d_n,d'_n) \quad .$$

The idea is that the new elements of D_∞ will have the following projections:

$$(d;d')_{n+1} = (d_n;d'_n) \quad , \qquad \text{and} \qquad (b \to d,d')_{n+1} = (b \to d_n,d'_n) \quad .$$

(The projections onto D_0 behave differently in view of the special nature of Ψ_0 as defined in Section 3.) It can be shown that these operations $(d;d')$ and $(b \to d,d')$ defined on D_∞ are not only continuous but additive. (This answers the question at the end of Section 2.) Hence, D_∞ is a lattice enriched with algebraic operations (called *products* and *sums* and *not* to be confused with products and sums of *whole lattices*.)

Let $(D_\infty;D_\infty)$ be the totality of all elements $(d;d')$ with $d, d' \in D_\infty$. This is a sublattice of D_∞. Similarly, $(B \to D_\infty,D_\infty)$ is a sublattice of D_∞. In view of the construction of D_{n+1} from D_n we can show that in fact:

$$D_\infty = D_0+(D_\infty;D_\infty)+(B \to D_\infty,D_\infty) .$$

Because if $d \in D_\infty$ and if $d \notin D_0$, then we can find elements d'_n, $d''_n \in D_n$ such that either for all n:

$$d_{n+1} = (d'_n;d''_n)$$

or there is some $b \in B$ such that for all n:

$$d_{n+1} = (b \to d'_n,d''_n) .$$

Setting

$$d' = \bigsqcup_{n=0}^{\infty} d'_n \text{ and } d'' = \bigsqcup_{n=0}^{\infty} d''_n$$

we find that *either*

$$d = (d';d'') \text{ or } d = (b \to d',d'') .$$

(One must also check that the $\top$ and $\bot$ elements match.) Since there can obviously not be any partial ordering relationships holding between the three different types of elements, we thus see why D_∞ decomposes into the sum of three of its sublattices.

Inasmuch as D_∞ is our ultimate lattice of *expressions* for diagrams, it will look neater if we call it E from now on. Having obtained the algebra and the decomposition, we shall find very little need to refer back to the projections. Thus, we can write:

$$E = F+(I)+(E;E)+(B \to E;E) \ ,$$

an equation which can be read very smoothly in words:

Every expression is either a function symbol,
or is the identity symbol,
or is the product of two expressions,
or is the sum of two expressions.

These words are very suggestive but in a way are a bit vague. We show next how to specify additional structure on E that will turn the above sentence into a mathematical equation.

To carry out this last program we need to use a very important lattice: the lattice T of *truth values*. It is illustrated in Figure 15. Aside from the ubiquitous $\perp$ and $\top$ it has two special elemerts 0 (false) and 1 (true). Defined on this lattice are the Boolear operations $\wedge$ (and), $\vee$ (or), $\neg$ (not) given by the tables of Figure 16. For our present purposes, these operations are not too important however, and we discuss them no further. What is much more important is the *conditional*.

Given an arbitrary lattice D, the conditional is a function

$$\supset : T \times D \times D \to D$$

such that

$$\supset(t,d,d') = \begin{cases} d \sqcup d' & \text{if } t = \top ; \\ d & \text{if } t = 1 ; \\ d' & \text{if } t = 0 ; \\ \perp & \text{if } t = \perp . \end{cases}$$

The reason for the choice of this definition is to make $\supset$ an *additive* function on $T \times D \times D$. Intuitively, we can read $\supset(t,d,d')$ as telling us to test t. If the result is 1 (true), we take d as the value of the conditional. If the result is 0 (false) we take d'. If the result of the test is *underdetermined*, so is the value of the conditional. If the result is *overdetermined*, we take the join of the

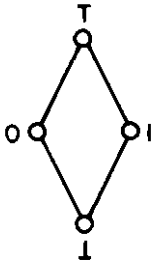


Figure 15

THE LATTICE $\mathbf{T}$

$\wedge$	$\perp$	0	1	T
$\perp$	$\perp$	0	$\perp$	0
0	0	0	0	0
1	$\perp$	0	1	T
T	0	0	T	T

$\vee$	$\perp$	0	1	T
$\perp$	$\perp$	$\perp$	1	1
0	$\perp$	0	1	T
1	1	1	1	1
T	1	T	1	T

$\neg$	
$\perp$	$\perp$
0	1
1	0
T	T

Figure 16

THE BOOLEAN OPERATIONS

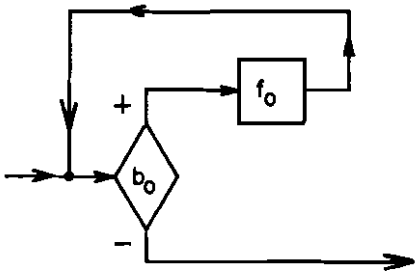


Figure 17

A SIMPLE LOOP

values we would have obtained in the self-determined cases. This last is conventional, but it seems to be the most convenient convention. It will be easier to read if we write

$$(t \supset d,d') = \supset(t,d,d') \ ,$$

and say in words:

$$\text{if } t \text{ then } d \text{ else } d' \ .$$

It is common to write $\rightarrow$ in place of $\supset$, but we have chosen the latter to avoid confusion with the conditional *expression* in E.

Returning now to our lattice E there are four fundamental functions:

$$\mathsf{func}:E \rightarrow T \ ,$$

$$\mathsf{idty}:E \rightarrow T \ ,$$

$$\mathsf{prod}:E \rightarrow T \ ,$$

$$\mathsf{sum}:E \rightarrow T \ .$$

All of these functions map τ to τ and $\perp$ to $\perp$. For elements with $d \neq \tau$ and $d \neq \perp$ we have

$$\mathsf{func}(d) = \begin{cases} 1 & \text{if } d \in F \ ; \\ 0 & \text{if } d \notin F \ . \end{cases}$$

$$\mathsf{idty}(d) = \begin{cases} 1 & \text{if } d \in \{I\} \ ; \\ 0 & \text{if } d \notin \{I\} \ . \end{cases}$$

$$\mathsf{prod}(d) = \begin{cases} 1 & \text{if } d \in (E;E) \ ; \\ 0 & \text{if } d \notin (E;E) \ . \end{cases}$$

$$\mathsf{sum}(d) = \begin{cases} 1 & \text{if } d \in (B \rightarrow E,E) \ ; \\ 0 & \text{if } d \notin (B \rightarrow E,E) \ . \end{cases}$$

These functions are all continuous (even:additive). They are the functions that correspond to the decomposition of E into four kinds of expressions.

Besides these there are five other fundamental functions:

$$\mathsf{first}:E \to E$$

$$\mathsf{secnd}:E \to E$$

$$\mathsf{left}:E \to E$$

$$\mathsf{right}:E \to E$$

$$\mathsf{bool}:E \to B$$

In case $d \in (E;E)$, we have:

$$d = (\mathsf{first}(d);\mathsf{secnd}(d)) \ ;$$

otherwise:

$$\mathsf{first}(d) = \mathsf{secnd}(d) = \bot .$$

In case $d \in (B \to E,E)$, we have:

$$d = (\mathsf{bool}(d) \to \mathsf{left}(d),\mathsf{right}(d)) \ ;$$

otherwise:

$$\mathsf{left}(d) = \mathsf{right}(d) = \bot \ (\text{in } E) \ ,$$

and

$$\mathsf{bool}(d) = \bot \ (\text{in } B) \ .$$

These functions are all continuous.

These nine functions together with the notions of products and sums of elements of E give a complete analysis of the structure of E. In fact, we can now rewrite the informal statement mentioned previously as the following equation which holds for all $d \in E$:

$$\begin{aligned} d = {} & (\mathsf{func}(d) \supset d \ , \\ & (\mathsf{idty}(d) \supset I \ , \\ & (\mathsf{prod}(d) \supset (\mathsf{first}(d);\mathsf{secnd}(d)) \ , \\ & (\mathsf{sum}(d) \supset (\mathsf{bool}(d) \to \mathsf{left}(d),\mathsf{right}(d)),\bot)))) \end{aligned}$$

Another way to say what the result of our construction is would be this: the lattice E replaces the usual notions of syntax. This lattice is constructed "synthetically", but what we have just verified is the basic equation of "analytic" syntax. All we really need to know about E is that it is a complete lattice that decomposes into

a sum of its algebraic parts. These algebraic parts are either generators or products and sums. The complete analysis of an element (down *one* level) is provided by the above equation which shows that the algebraic terms out of which an element is formed are uniquely determined as *continuous* functions of the element itself.

Except for stressing the lattice-theoretic completeness and the continuity of certain functions, this sounds just like ordinary syntax. The parallel was intended. But our syntax is *not* ordinary; it is an essential generalization of the ordinary notions as we now show.

5.LOOPS AND OTHER INFINITE DIAGRAMS. In Figure 17 we have the most well-known construction of a flow diagram which allows the information to flow in circles: the so-called while-loop. It represents, as everyone knows, one of the very basic ideas in programming languages. Intuitively, the notion is one of the simplest: information enters and is tested (by b_0). If the test is positive, the information is transformed (by f_0) and is channeled back to the test in preparation for recirculation around the loop. While tests turn out positive, the circulation continues. Eventually, the cumulative effects of the repeated transformations will produce a negative test result (*if* the procedure is to allow output), and then the information exits.

None of our finite diagrams in E (that is, diagrams in any of the D_n lattices) has this form. It might then appear that we had overlooked something. But we did not, and that was the point of making E complete. To appreciate this, ask whether the diagram involving a loop in Figure 17 is not an abbreviation for a more ordinary diagram. There are many shortcuts one can take in the drawing of diagrams to avoid tiresome repetitions; we have noted several previously. Loops may just be an extreme case of abbreviation. Indeed,

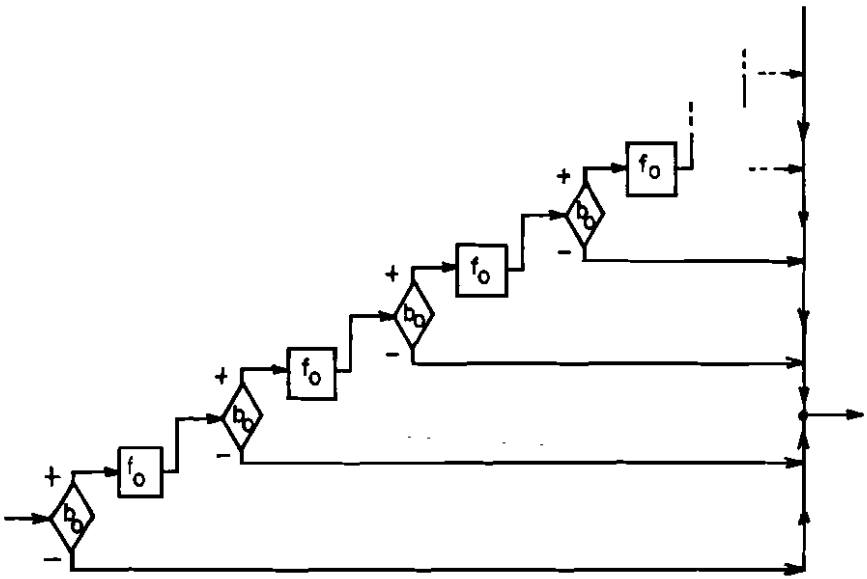


Figure 18

THE INFINITE VERSION OF THE LOOP

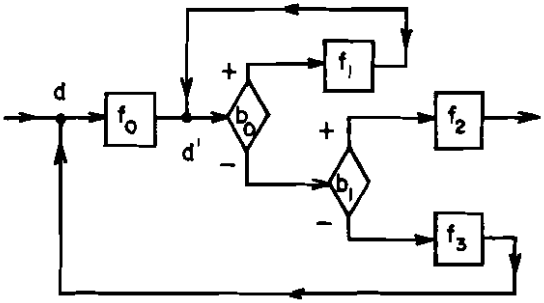


Figure 19

A DOUBLE LOOP

instead of bending the channel back around to the front of the diagram, we could write the test again. And, after the next transformation, we could write it out again. And again. And again, and again, and ... . The beginning of the *infinite* diagram that will thereby be produced is shown in Figure 18. Obviously, the infinite diagram will produce the same results as the loop. (Actually, this assertion requires proof.)

Does what we have just said make any sense? Some symbolization will help to see that it does. We have symbols for the test b_0 and the transformation f_0. Let the diagram we seek be called d. Look again at Figure 18. After the first test and transformation the diagram repeats itself. This simple pattern can easily be expressed in symbols thus:

$$d = (b_0 \rightarrow (f_0;d),I) \ .$$

In other words, we have a test with an exit on negative. If positive, on the other hand, we compound f_0 with the same procedure immediately following. Therefore, the diagram contains *itself* as a part.

That is all very pretty, but does this diagram d really exist in E? To see that it does, recall that all our algebraic operations are *continuous* on E. Consider the function Φ:E $\rightarrow$ E defined by the equation:

$$\Phi(x) = (b_0 \rightarrow (f_0;x),I) \ .$$

The function Φ is evidently a continuous mapping of diagrams. *Every continuous function on a complete lattice into itself has a fixed point.* In this case, we of course want d to be the *least* fixed point:

$$d = \Phi(d) \ ,$$

because the diagram should have no other quality aside from the endless repetition. The infinite diagram d *does* exist. (It cannot be finite, as is obvious.) We can now see why we did not introduce loops in the beginning: their existence follows from completeness

and continuity. In any case, they are very special and only one among many diverse types of infinite diagrams.

Figure 19 shows a slightly more complex example with a double loop. We shall not attempt to draw the infinite picture, since that exercise is unnecessary. The figure with loops is explicit enough to allow us to pass directly to the correct symbolization. To accomplish this, label the two re-entry points d and d'. Following the flow of the diagram we can write these equations:

$$d = (f_0;d')$$

and

$$d' = (b_0 \rightarrow (f_1;d'),(b_1 \rightarrow f_2,(f_3;d)))$$

Substituting the first equation in the second we find:

$$d' = (b_0 \rightarrow (f_1;d'),(b_1 \rightarrow f_2,(f_3;(f_0;d')))) .$$

Now, the "polynomial"

$$\Psi(x) = (b_0 \rightarrow (f_1;x),(b_1 \rightarrow f_2,(f_3;(f_0;x))))$$

is a bit more complex than the previous $\Phi(x)$, but just the same it is continuous and has its least fixed point d'. Thus, d', and therefore d, does exist in E.

Sometimes, the simple elimination procedure we have just illustrated does not work. A case in point is shown in Figure 20. The loops (whose entry points are marked d and d') are so nested in one another that each fully involves the other. (By now, an attempt at drawing the infinite diagram is quite hopeless.) The symbolization is easy, however:

$$d = (b_0 \rightarrow f_0,(b_1 \rightarrow (f_1;d),(f_2;d')))$$

and

$$d' = (b_2 \rightarrow f_3,(b_3 \rightarrow (f_4;d),(f_5;d'))).$$

In this situation any substitution of either equation in the other leaves us with an expression still containing *both* letters d and d'.

That is to say, the two diagrams called d and d' have to be constructed *simultaneously*. Is this possible? It is. Consider the fact that $E \times E$ is also a complete lattice. Introduce the function

$$\Theta : E \times E \to E \times E$$

defined as follows:

$$\Theta(\langle x,y \rangle) = \langle (b_0 \to f_0, (b_1 \to (f_1;x), (f_2;y))), (b_2 \to f_3, (b_3 \to (f_4;x), (f_5;y))) \rangle$$

Now, this function Θ is continuous and has a least fixed point:

$$\langle d,d' \rangle = \Theta(\langle d,d' \rangle) ,$$

and this pair is exactly the pair of diagrams we wanted.

This method can now be seen to be flexible and of wide applicability. For example, if using our algebra on E, we write down any system of polynomials in several variables:

$$\Pi_0(x_0,x_1,x_2,\ldots), \Pi_1(x_0,x_1,x_2,\ldots), \Pi_2(x_0,x_1,x_2,\ldots), \ldots ,$$

then on a suitable product lattice:

$$E \times E \times E \times \ldots$$

we can solve for fixed points:

$$\begin{aligned} d_0 &= \Pi_0(d_0,d_1,d_2,\ldots) \\ d_1 &= \Pi_1(d_0,d_1,d_2,\ldots) \\ d_2 &= \Pi_2(d_0,d_1,d_2,\ldots) \\ &\ldots . \end{aligned}$$

Diagrams constructed in this way may be called *algebraic* elements of E. The finite diagrams in the union of the D_n may be called *rational*. This classification does not by far exhaust the elements of E: there are besides a continuum number of *transcendental* elements. (The reader may construct one from Figure 18 by replacing the sequence of boxes f_0, f_0, f_0, ... by the sequence f_0, f_1, f_2, ... or by some other nonrepeating sequence.) Whether these other elements of E are of any earthly good remains to be seen. They are there, in any case. If you do not care to look at them, you need not do so. It will be your loss not theirs.

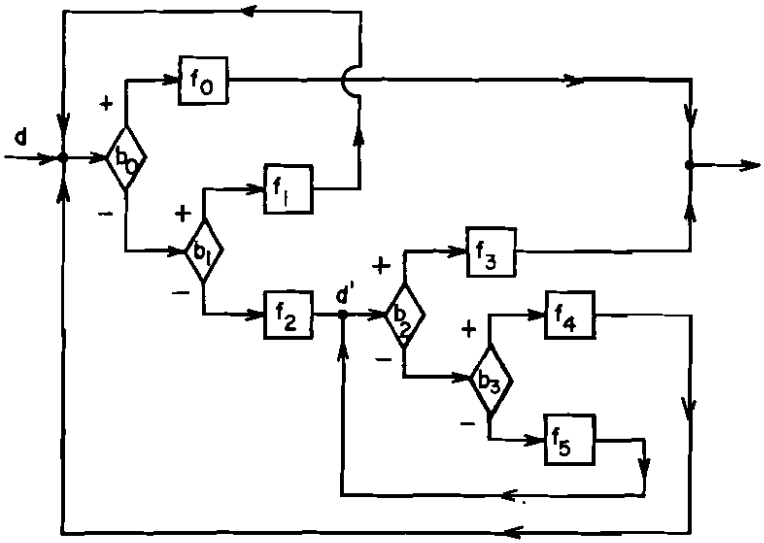


Figure 20
NESTED LOOPS

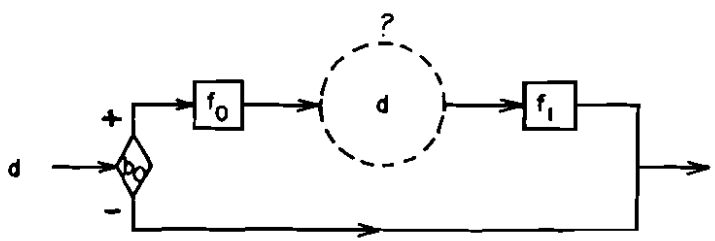


Figure 21
A DIFFICULT DIAGRAM

It is not too easy to draw pictures of some of the algebraic elements of E. Take, for example, this defining equation:

$$d = (b_0 \rightarrow (f_0;(d;f_1)),I)$$

A first and an unsatisfactory attempt to draw this as a diagram is shown in Figure 21. The question is what to fill in the middle. We need another copy of d itself; but this involves still another copy of d. And, so on. There seem to be no shortcuts available. Any attempt to introduce loops will not make it clear that in any one tour of the channels the *same* number of f_0 boxes as f_1 boxes must be visited. But this is a failure of the picture language. The algebraic language is unambiguous (hence, better!). Nevertheless, this example does suggest that there is a classification of the *algebraic* elements of E that needs additional thought.

Now that we see something of the scope of E, we can organize the study of its elements with the aid of further notations. For example, the while-loop is so fundamental that it deserves its own notation:

$$(b*d) \ ,$$

which stands for the least fixed point of the function

$$(b \rightarrow (d;x),I) \ .$$

It can be easily shown that $*$ is a continuous function on B×E into E. There are many others.

This is the place to clear up a continuing notational confusion. Since, in order to communicate mathematical facts, we need to write formulas involving symbols, we have to be clear about the distinction between a symbol and what it denotes. This distinction becomes particularly critical when we study the theory of syntax, as we have been doing here. So, let us be very pedantic about the nature of the constructions we have been discussing. What are the elements of E actually? Either they are elements of F or of $\{I\}$ or they are

pairs or triples of pairs or triples of ... of elements of B and E. Or they are limit points of these, which strictly speaking, are infinite ("convergent") sequences of rational elements of E. Alas, E contains no *symbols*, only mathematical constructs.

But defined on E is a whole array of functions and constants:

$$f_0, f_1, \ldots, I, (x;y), (b \to x, y) ,$$

$$\mathsf{func}(x), \ldots, \mathsf{first}(x), \ldots, b*x ,$$

etc.

Thus, such things as subscripts, capital letters, parentheses, semicolons, arrows, commas, bold-face letters, and stars do not actually occur as parts of any of the elements of our "expression" space E. Rather, E is to be regarded as *a mathematical model of a theory of expressions*. It is only one of many similar models. Or, if you like, E is a model for a theory of *geometric diagrams*, and a quite satisfactory theory at that. The lattice E does not care what applications you care to make of it. E is abstract. E gives you a fixed structure to guide your thoughts. It is the same with the theory of the real numbers and analytic geometry. These structures are "pure": it is up to us to supply the plot and to write exciting stories about them using a careful choice of language (that is, functions, relations, etc.). In the case of E, however, we can ask not only what it *is*, and what its elements *do*, but also what do they *mean*.

6. THE SEMANTICS OF FLOW DIAGRAMS. We have spoken all along of the flow of information through a diagram. It is intuitively clear what is meant, but eventually one must introduce some precise definitions if he ever hopes to get any definite results. In other words, it is now time to present in detail a mathematical model of the concept of *flowing*. Up to this point, everything is static: the elements of E do not move; they do not light up, make noise, or otherwise show signs of life. We have sketched many pictures of elements

of E and *on the paper*, on these diagrams, we can move *our* fingers or shift *our* eyes back and forth. The abstract elements of E remain impassive, however, and must remain so, frozen in the eternal realm of ideas. But they neither expect or want our pity. And, we are free to study them, to talk about them as we do of works of art.

Clearly, the first requirement in the study of the meaning of the artifacts in E is a theory of *information*. Disappointingly enough, in this paper we shall not make a very deep study of this essential notion. We shall take it as axiomatic that *the quanta of information form a lattice called*:

$$S .$$

If you prefer, you can also consider the lattice S as being the lattice of *states* (states of "nature"). Where the lattice comes from, we do not say. We shall give some examples, by and by, but shall not be able to discuss lattices in general here. It should be reasonably evident from the success we have had in constructing lattices with useful properties, that this assumption is no loss of generality. Indeed, it can be argued that the requirement is a *gain* of generality.

In order to specify the meanings of the elements E, we must begin with the $f_i \in F$. Here, we have great freedom: their meanings can be determined at will -- within certain limits. The limits are set by this reasoning: as information passes through a box it is transformed. If the box is labeled with the symbol f_i, then the meaning of f_i is this transformation. That is, corresponding to each f_i is a *function*

$$\mathfrak{F}(f_i): S \to S$$

which provides the means of transforming S. Note that the transformation depends only on the label and not on the context of occurrence of a box, because we intend like labeled boxes to perform the same transformation. Since we have gone to the trouble of saying that

S is a complete lattice, we will also require each function $\mathfrak{F}(f_i)$ to be *continuous*.

Think for a moment of the collection of all continuous functions from S into S. If u and v are such, there is a most natural way of defining what it means for u to approximate v:

$$u \sqsubseteq v \text{ if and only if } u(\sigma) \sqsubseteq v(\sigma) \text{ for all } \sigma \in S \ .$$

It can easily be established that the set of continuous functions becomes in this way a complete lattice itself. We denote this lattice by $[S \to S]$. (*Caution*: do not confuse this notation with the earlier $(B \to E,E)$, which is a certain sublattice of E.)

In a highly useful short-hand way we can say that

$$\mathfrak{F}:F \to [S \to S] \ .$$

We even require $\mathfrak{F}$, as a mapping, to be *continuous*. Thus,

$$\mathfrak{F} \in [F \to [S \to S]] \ .$$

In this manner, we indicate succinctly what is called the *logical type* of $\mathfrak{F}$ as a mapping. Attention paid to logical types is attention well spent.

The next project is to attach meanings to elements of B. If $b \in B$ it designates some test that may be applied to elements of S. The outcome of a test is a truth value. For us, that means an element of the lattice T. Hence, to have meanings is to have a (continuous) function

$$\mathcal{B}:B \to [S \to T]$$

Both $[S \to S]$ and $[S \to T]$ have largest elements (both are lattices). In $[S \to S]$ it is the constant function τ (obviously, a continuous function). We should write

$$\tau_{[S \to S]} \in [S \to S]$$

where for all $\sigma \in S$:

$$\tau_{[S \to S]}(\sigma) = \tau_S \ .$$

But we drop the subscripts and write $\tau(\sigma) = \tau$. Similarly, for $\perp(\sigma) = \perp$. The same slightly ambiguous notation is used for $[S \to T]$. For simplicity, we require both $\mathfrak{F}$ and $\mathfrak{B}$ to have the property that

$$\mathfrak{F}(\tau) = \tau \ , \quad \mathfrak{F}(\perp) = \perp \ ,$$
$$\mathfrak{B}(\tau) = \tau \ , \quad \mathfrak{B}(\perp) = \perp \ ,$$

where it is left to the reader to determine to which lattices each of the τ's and $\perp$'s belong.

The functions $\mathfrak{F}$ and $\mathfrak{B}$ may be chosen freely within their respective logical types -- but that is all the freedom we have. The meanings of all the other elements of E are uniquely determined relative to this choice of $\mathfrak{F}$ and $\mathfrak{B}$.

To show how this works out, we shall determine a function $\mathfrak{V}$ (again:continuous) such that

$$\mathfrak{V}:E \to [S \to S] \ .$$

If $d \in E$, then $\mathfrak{V}(d)$ is the "value" of d (given $\mathfrak{F}$ and $\mathfrak{B}$). The intention is that if $\sigma \in S$ is the initial state of the information entering the flow diagram d, then

$$\mathfrak{V}(d)(\sigma)$$

is the final state upon exiting. We thus do not teach you how to swim through the channels of the flow diagram, but content ourselves with telling you what you will look like when you come out as a function of what you looked like when you jumped in. The transformation is, of course, continuous, And, merely knowing this transformation (over all d and all σ) is sufficient for a mathematical theory of *flowing*.

The precise definition of $\mathfrak{V}$ is obtained by simply writing out an equation that corresponds to what you yourself would do in swimming through a diagram. We write it first and then read it:

$$\mathfrak{V}(d)(\sigma) = (\mathrm{func}(d) \supset \mathfrak{F}(d)(\sigma)\ ,$$
$$(\mathrm{idty}(d) \supset \sigma\ ,$$
$$(\mathrm{prod}(d) \supset \mathfrak{V}(\mathrm{secnd}(d))(\mathfrak{V}(\mathrm{first}(d))(\sigma))\ ,$$
$$(\mathrm{sum}(d) \supset (\mathfrak{B}(\mathrm{bool}(d))(\sigma) \supset \mathfrak{V}(\mathrm{left}(d))(\sigma), \mathfrak{V}(\mathrm{right}(d))(\sigma)), \perp))))$$

(One small point: we may regard $\mathfrak{F}$ as being of type $\mathfrak{F}:E \to [S \to S]$ because $\mathfrak{F}(d) = \perp$ is a good value if $d \notin F$. Or, we should replace $\mathfrak{F}(d)$ by $\mathfrak{F}(|d|)$ where $|d| = d$ if $d \in F$, and $|d| = \perp \in F$ if $d \notin F$.)

The translation of the above equation runs as follows:

> *To compute the outcome of the passage of σ through d, first ask whether d is a function symbol. If it is, the outcome is $\mathfrak{F}(d)(\sigma)$. If it is not, ask whether d is the identity symbol. If it is, then the outcome is σ. If it is not, ask whether d is a product. If it is, find the first and second terms of d. Pass σ through the first term of d obtaining the proper outcome. Take this outcome and pass it through the second term of d. That gives the desired final outcome. If d is not a product, ask whether it is a sum. If it is, find the boolean part of d and test σ by it. Depending on the result of the test, pass σ through either the left or the right branch of d, obtaining the desired outcome. If d is not a sum (this case will not arise), the outcome is $\perp$.*

One soon learns to appreciate equations. And, the equations are more precise as well as being more perspicuous -- though sometimes they become so involved as to be unreadable. Note, for example, how our equation for $\mathfrak{V}$ tells us exactly what to do in case

$$\mathfrak{B}(\mathsf{bool}(d))(\sigma) = \bot \text{ or } = \top \ .$$

This would be rather tiresome to put in words. The question we need to ask now, however, is whether this equation really defines $\mathfrak{V}$. Obviously, it is not an *explicit* definition because $\mathfrak{V}$ occurs on both sides of the equation. Hence, we cannot claim straight off that $\mathfrak{V}$ exists. To prove that it does, some fixed points must be found in some rather sophisticated lattices.

It was not just an idle remark to point out that

$$[S \to S]$$

is a complete lattice. Knowing this, we have by the same token that

$$[E \to [S \to S]]$$

is also complete. And this lattice gives the logical type of $\mathfrak{V}$:

$$\mathfrak{V} \in [E \to [S \to S]] \ .$$

To find this $\mathfrak{V}$, then, as a fixed point, we would need a function

$$\Xi \in [[E \to [S \to S]] \to [E \to [S \to S]]]$$

which is a lattice somewhat removed from everyday experience. But that does not matter: we know all the general definitions.

Here is the specific principle we need. In the following expression the variables $\mathfrak{X}$, d, and σ occur of types $[E \to [S \to S]]$, E, and S respectively. The expression is:

$(\mathsf{func}(d)\supset \mathfrak{F}(d)(\sigma)$,

$(\mathsf{idty}(d)\supset\sigma$

$(\mathsf{prod}(d)\supset \mathfrak{X}(\mathsf{secnd}(d))(\mathfrak{X}(\mathsf{first}(d))(\sigma))$,

$(\mathsf{sum}(d)\supset(\mathfrak{B}(\mathsf{bool}(d))(\sigma)\supset \mathfrak{X}(\mathsf{left}(d))(\sigma), \mathfrak{X}(\mathsf{right}(d))(\sigma)), \bot))))$

This is a function of three variables. We can prove, just by looking at it, that it is *continuous* in its three variables.

Forget about the exact form of the above expression and imagine any such continuous expression:

$$(\ldots\mathfrak{X}\ldots d\ldots\sigma) \ .$$

Holding $\mathfrak{X}$ and d *fixed* we have a function of σ. The logical type of

the value of the expression is also S. Thus, there is a function $\Xi'(\mathfrak{X},d)$ depending on given $\mathfrak{X}$, d such that

$$\Xi'(\mathfrak{X},d)(\sigma) = (\ldots\mathfrak{X}\ldots d\ldots\sigma) .$$

The logical type of $\Xi'(\mathfrak{X},d)$ is $[S \to S]$. In other words, $\Xi'(\mathfrak{X},d)$ is an "expression" whose value depends on $\mathfrak{X}$ and d. We can show that $\Xi'(\mathfrak{X},d)$ is *continuous* in $\mathfrak{X}$ and d. Going around again, there must be a function (a uniquely determined function) $\Xi(\mathfrak{X})$ such that

$$\Xi(\mathfrak{X})(d) = \Xi'(\mathfrak{X},d) ,$$

so that

$$\Xi(\mathfrak{X}) \in [E \to [S \to S]] .$$

But this correspondence is *continuous in* $\mathfrak{X}$. So, really

$$\Xi \in [[E \to [S \to S]] \to [E \to [S \to S]]] .$$

All continuous functions have fixed points (when they map a lattice into itself), and so our $\mathcal{V}$ is given by

$$\mathcal{V} = \Xi(\mathcal{V})$$

with the understanding that we take the least such $\mathcal{V}$ (as an element of the lattice $[E \to [S \to S]]$).

Yes, the argument is abstract, but then it is very general. The easiest thing to do is simply to accept the existence of a continuous (minimal) $\mathcal{V}$ and to carry on from there. One need not worry about the lattice theory -- as long as he is sure that all functions that he defines are continuous. Generally, they seem to take care of themselves. Intuitively, the definition of $\mathcal{V}$ is nothing more than a recursive definition which gives the meaning of one diagram in terms of "smaller" diagrams. Such definitions are common and are well understood. In the present context, we might only begin to worry when we remember that a portion of an infinite diagram is not really "smaller" (it may even be *equal* to the original). It is this little worry which the method of fixed points lays

to rest. Let us examine what happens with the while-loop.

7. THE MEANING OF A WHILE-LOOP. Let $b \in B$ and $d \in E$. Recall the definition of

$$(b*d) .$$

It is the least element of E satisfying the equation:

$$x = (b \to (d;x),I) .$$

We see that $(b*d) \in (B \to E,E)$ and

$$\text{bool}\ ((b*d)) = b$$

$$\text{left}\ ((b*d)) = (d;(b*d))$$

$$\text{right}\ ((b*d)) = I .$$

Hence, by the definition of $\mathcal{V}$, for $\sigma \in S$ we have:

$$\mathcal{V}((b*d))(\sigma) = (\ \mathcal{B}(b)(\sigma) \supset \mathcal{V}((d;(b*d)))(\sigma),\sigma)$$

But $(d;(b*d)) \in (E;E)$ and

$$\text{first}((d;(b*d))) = d$$

$$\text{secnd}((d;(b*d))) = (b*d) .$$

So we find that:

$$\mathcal{V}((b*d))(\sigma) = (\ \mathcal{B}(b)(\sigma) \supset \mathcal{V}((b*d))(\ \mathcal{V}(d)(\sigma)),\sigma) .$$

The equation is too hard to read with comfort. Let $w = (b*d)$ and

$$\overline{b} = \mathcal{B}(b) \in [S \to T] ;$$

and

$$\overline{d} = \mathcal{V}(d) \in [S \to S] .$$

We may suppose that $\overline{b}$ and $\overline{d}$ are "known" functions. The diagram is the while-loop formed from b and d, and the semantical equation above now reads:

$$\mathcal{V}(w)(\sigma) = (\overline{b}(\sigma) \supset \mathcal{V}(w)(\overline{d}(\sigma)),\sigma)$$

The equation is still too fussy, because of all those σ's. Let

$$\overline{I} \in [S \to S]$$

be such that for all $\sigma \in S$:

$$\overline{I}(\sigma) = \sigma \ .$$

For any two functions $u, v \in [S \to S]$, let

$$u \cdot v \in [S \to S]$$

be such that for all $\sigma \in S$:

$$(u \cdot v)(\sigma) = v(u(\sigma)) \ .$$

(This is functional composition, but note the order.) For any $p \in [S \to S]$, and $u, v \in [S \to S]$, let

$$(p \rhd u,v) \in [S \to S]$$

be such that for all $\sigma \in S$:

$$(p \rhd u,v)(\sigma) = (p(\sigma) \supset u(\sigma),v(\sigma)) \ .$$

Now, we have enough notation to suppress all the σ's and to write:

$$\mathcal{V}(\omega) = (\overline{b} \rhd (\overline{d} \cdot \mathcal{V}(\omega)),\overline{I}) \ ,$$

at last an almost readable equation.

It is important to notice that $\cdot$ and $\rhd$ are *continuous* functions (of several variables) on the lattices $[S \to T]$ and $[S \to S]$. Indeed, we have a certain parallelism:

E	$[S \to S]$
B	$[S \to T]$
f_i	$\overline{f}_i$
I	$\overline{I}$
b_j	$\overline{b}_j$
$(x;y)$	$(u \cdot v)$
$(b \supset x,y)$	$(p \rhd u,v)$

We could even say that $[S \to S]$ is an algebra with *constants* $\overline{f}_i$ and $\overline{I}$, with a *product* $(u \cdot v)$ and with *sums* $(\overline{b}_j \supset u,v)$ (and, if they are of interest, also $(\tau \supset u,v)$ and $(\bot \rhd u,v)$ where $\tau, \bot \in [S \to T]$ are the obvious functions). The function $\mathcal{V}: E \to [S \to S]$ then proves to be a *continuous algebraic homomorphism*. It is *not* in general a lattice homomorphism since it is continuous and not join preserving.

It does, however, preserve all products and sums - as we have illustrated in one case.

Let us make use of this observation. If we let $\Phi: E \to E$ be such that

$$\Phi(x) = (b \to (d;x), I) \quad ,$$

then our while-loop $w = (b * d)$ is given by

$$w = \bigsqcup_{n=0}^{\infty} \Phi^n(\bot) \quad .$$

The function Φ is an algebraic operation on E; we shall let $\overline{\Phi}: [S \to S] \to [S \to S]$ be the corresponding operation such that

$$\overline{\Phi}(u) = (\overline{b} \to (\overline{d} \cdot u), \overline{I}) \quad .$$

From what we have said about the continuity and algebraic properties of $\boldsymbol{\nu}$, it follows that

$$\boldsymbol{\nu}(w) = \bigsqcup_{n=0}^{\infty} \overline{\Phi}^n(\bot) \quad .$$

This proves that $\boldsymbol{\nu}(w)$ is the least solution $u \in [S \to S]$ of the equation

$$u = (\overline{b} \blacktriangleright (\overline{d} \cdot u), \overline{I})$$

Thus, $\boldsymbol{\nu}$ preserves while; more precisely, there is an operation on $[S \to T] \times [S \to S]$ analogous to the $*$ operation on $B \times E$, and we have shown that $\boldsymbol{\nu}$ is also a homomorphism with respect to this operation.

Actually, the solution to the equation

$$x = (b \to (d;x), I)$$

is *unique* in E. It is not so in the algebra $[S \to S]$ that the equation

$$u = (\overline{b} \blacktriangleright (\overline{d} \cdot u), \overline{I})$$

has only one solution. But we have shown that $\boldsymbol{\nu}$ picks out the least solution. This observation could be applied more generally also.

In any case, we can now state definitely that the quantity

$$\mathcal{V}(\omega)(\sigma)$$

is computed by the following iterative scheme: first $\overline{b}(\sigma)$ is computed. If the result is 1 (true), then $\overline{g}(\sigma) = \sigma'$ is computed and the whole procedure is started over on

$$\mathcal{V}(\upsilon)(\sigma') \ .$$

If $\overline{b}(\sigma) = 0$, the result is σ at once. (If $\overline{b}(\sigma) = \perp$, the result is $\perp$. If $\overline{b}(\sigma) = \tau$, the result is $\mathcal{V}(\omega)(\sigma') \sqcup \sigma$, which generally is not too interesting.) The *minimality* of the solution to the equation in $[S \to S]$ means that we get *nothing more* than what is strictly implied by this computation scheme.

This result is hardly surprising; it was not meant to be. What it shows is that our definition is *correct*. Everyone computes a **while** in the way indicated and the function $\mathcal{V}(\omega)$ gives us just what was expected: no more, no less. We can say that $\mathcal{V}$ is the semantic function which maps the diagram ω to its "value" or "meaning" $\mathcal{V}(\omega)$. And, we have just shown that the meaning of a while-loop is exactly a function in $[S \to S]$ to be computed in the usual while-manner. The meaning of the diagramatic while is the while-process. No one would want it any other way.

It is to be hoped that the reader can extend this style of argument to other configurations that may interest him.

8. EQUIVALENCE OF DIAGRAMS. Strictly speaking, the semantical interpretation defined and illustrated in the last two sections depends not only on the choice of S but on that of $\mathcal{F}$ and $\mathcal{B}$. Indeed, we should write more fully:

$$\mathcal{V}_S(\mathcal{F})(\mathcal{B})(d)(\sigma) \ ,$$

and take the logical type of $\mathcal{V}$ to be:

$$\mathcal{V}_S \in [[F \to [S \to S]] \to [[B \to [S \to T]] \to [E \to [S \to S]]]] \ .$$

Of course, $\mathcal{V}_S(\mathcal{F})(\mathcal{B})$ is continuous in $\mathcal{F}$ and in $\mathcal{B}$.

If we like, we can call the set S the set *states of a machine*. The functions $\mathfrak{F}$ and $\mathfrak{B}$ give the behavior of the "hardware" of a machine. Thus, the lattice

$$[F \to [S \to S]] \times [B \to [S \to T]]$$

may be called the *lattice of machines* (relative to the given S). This is obviously a very superficial analysis of the nature of machines: we have not discussed the "content" of the states in S, nor have we explained how a function $\mathfrak{F}(f)$ manages to produce its values. Thus, for example, the functions have no estimates of "cost" of execution attached to them (e.g. the time required for computation or the like). The level of detail, however, is that generally common in studies in automata theory (cf. Arbib (1969) as a recent reference), but it is sufficient to draw some distinctions. Certainly, lattices are capable of providing the structure of finite state machines with partial functions (as in Scott (1967)), and much more: the uses of *continuous* functions on certain infinite lattices S are more subtle than ordinary employment of point-to-point functions. The demonstration that the present generalization is really fruitful will have to wait for future publications, though. (Cf. also Bekić, and Park (1969))

Whenever one has some semantical construction that assigns "meaning" to "syntactical" objects, it is always possible to introduce a relationship of "synonymity" of expressions. We shall call the relation simply: *equivalence*, and for $x, y \in E$ write:

$$x \equiv y$$

to mean that for *all* S and all $\mathfrak{F}$ and $\mathfrak{B}$ relative to this S we have:

$$\mathfrak{V}_S(\mathfrak{F})(\mathfrak{B})(x) = \mathfrak{V}_S(\mathfrak{F})(\mathfrak{B})(y) .$$

This relationship obviously has all the properties of an equivalence relation, but it will be the "algebraic" properties that will be of

more interest. In this connection, there is one algebraic relationship that suggests itself at once. For $x, y \in E$ we write:

$$x \sqsubseteq y$$

to mean that for all S and all $\mathfrak{F}$ and $\mathfrak{B}$ relative to this S we have:

$$\mathfrak{V}_S(\mathfrak{F})(\mathfrak{B})(x) \sqsubseteq \mathfrak{V}_S(\mathfrak{F})(\mathfrak{B})(y) .$$

These relationships are very strong -- but not as strong as *equality*, as we shall see. The $\approx$ and $\sqsubseteq$ are related; for, as it is easy to show, $\sqsubseteq$ is *reflexive* and *transitive*, and further

$$x \approx y \text{ if and only if } x \sqsubseteq y \text{ and } y \sqsubseteq x .$$

But these are only the simplest properties of $\approx$ and $\sqsubseteq$.

For additional properties we must refer back to the exact definition of $\mathfrak{V}$ in Section 6. In the first place, the definition of $\mathfrak{V}$ was tied very closely to the *algebra* of E involving products and sums of diagrams. The meaning of a product turned out to be *composition* of functions; and that of a sum, a *conditional* "join" of functions. The meanings of $\approx$ and $\sqsubseteq$ are *equality* and *approximation* in the function space $[S \to S]$, respectively. Hence, it follows from the monotonic character of compositions and conditionals that:

$$x \sqsubseteq x' \text{ and } y \sqsubseteq y' \text{ implies } (x;y) \sqsubseteq (x';y')$$
$$\text{and } (b \to x,y) \sqsubseteq (b \to x',y') .$$

In view of the connection between $\approx$ and $\sqsubseteq$ noted in the last paragraph, the corresponding principle with $\sqsubseteq$ replaced by $\approx$ also follows.

A somewhat more abstract way to state the fact just noted can be obtained by passing to equivalence classes. For $x \in E$, we write

$$x/\approx \; = \{x' \in E : x \approx x'\}$$

for the *equivalence class* of x under $\approx$. We also write

to denote the set of all such equivalence classes, the so-called *quotient algebra*. And the point is that $E/\equiv$ *is* an algebra in the sense that products and sums are well defined on the equivalence classes, as we have just seen.

We shall be able to make E seem even more like an algebra, if we write:

$$x \dagger y = (\tau \to x, y)$$

for all $x, y \in E$. Now, in Section 6 we restricted consideration to those $\mathcal{B} \in [B \to [S \to T]]$ such that

$$\mathcal{B}(\tau) = \tau \ .$$

Thus

$$\mathcal{V}_S(\mathcal{F})(\mathcal{B})(x \dagger y)(\sigma) = \mathcal{V}_S(\mathcal{F})(\mathcal{B})(x)(\sigma) \sqcup \mathcal{V}_S(\mathcal{F})(\mathcal{B})(y)(\sigma) \ ;$$

that is the meaning of $x \dagger y$ is the lattice-theoretic *join* (or full sum) of the functions assigned to x and to y. This, of course, seems very special. As a diagram we would draw $x \dagger y$ as in Figure 21. The intended interpretation is that flow of information is directed through *both* x and y and is "joined" at the output. The sense of "join" being used is that of the join in the lattice S.

Pushing the algebraic analogy a bit further we can write certain conditionals as *scalar products*:

$$b \cdot x = (b \to x, \bot) \ ,$$

and

$$(1\text{-}b) \cdot y = (b \to \bot, y) \ .$$

These two compounds are diagramed in Figure 22. The first passes information through x provided b is true; the second, through y provided b is false. Now, our first really "algebraic" result is this equivalence:

$$(b \to x, y) \equiv (b \cdot x) \dagger (1\text{-}b) \cdot y \ .$$

That is, up to equivalence, the conditional sum can be "defined" by the full sum with the aid of scalar multiples. A fact that can

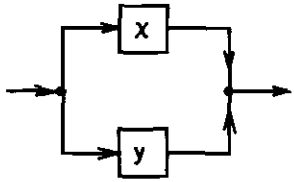
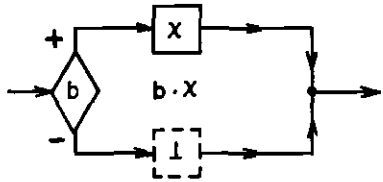


Figure 21
A(FULL) SUM



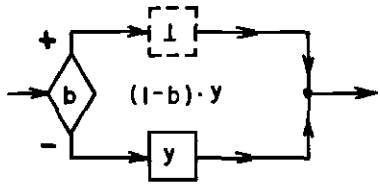


Figure 22
TWO SCALAR PRODUCTS

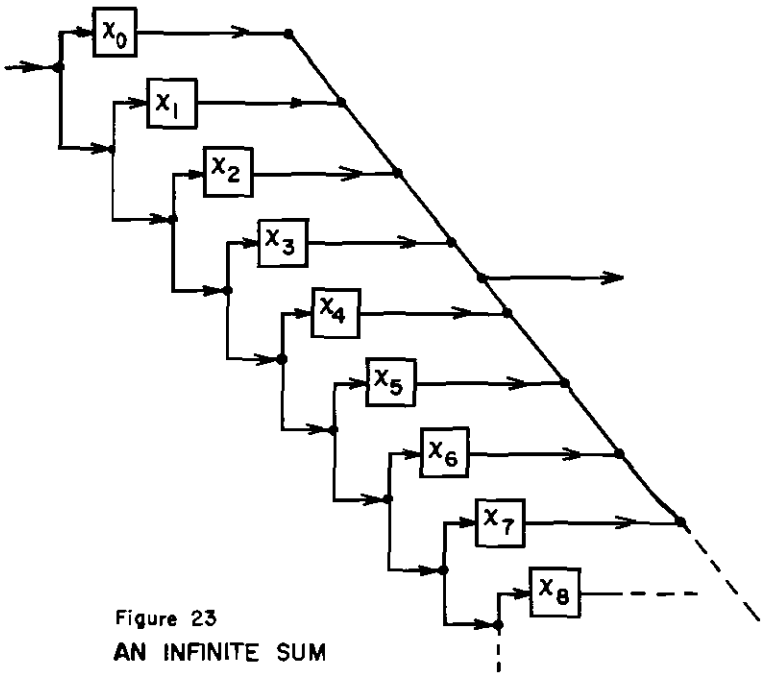


Figure 23
AN INFINITE SUM

be easily appreciated from the diagrams. This would not be very interesting if we did not have further algebraic equivalences, but note the following:

$$b\cdot(x\dagger y) \equiv b\cdot x\dagger b\cdot y$$

$$b\cdot(b\cdot x) \equiv b\cdot x$$

$$b\cdot(c\cdot x) \equiv c\cdot(b\cdot x)$$

$$b\cdot x \sqsubseteq x$$

(If we had introduced some algebra into B, These results would be even more regular. But we chose here not to algebraicize B.) One must take care to remember that T is not a Boolean algebra; thus, while it is correct that:

$$b\cdot x\dagger(1\text{-}b)\cdot x \equiv x \ ,$$

it is *not* correct that:

$$b\cdot(1\text{-}b)\cdot x \equiv \bot \ .$$

The reason being that $\mathcal{B}(b)(\sigma) = \tau$ is possible.

Having sufficient illustration of the properties of scalar multiples, we turn now to products. First, there is one distributive law:

$$x;(y\dagger z) \supseteq (x;y)\dagger(x;z)$$

that is correct; but the opposite

$$(y\dagger z);x \not\supseteq (y;x)\dagger(z;x)$$

is *not* correct. The reader may carry out the semantical analysis of these two proposed laws. The correctness of the first turns on the fact that for f, $f' \in [S \to S]$ and $\sigma \in S$ we have

$$(f \sqcup f')(\sigma) = f(\sigma) \sqcup f'(\sigma) \ .$$

The incorrectness of the second is a consequence of the failure in general of the equation

$$f(\sigma \sqcup \sigma') = f(\sigma) \sqcup f(\sigma')$$

for $f \in [S \to S]$ and $\sigma, \sigma' \in S$. Similarly, one must take care to note that *neither* of the following are correct:

$$(b\cdot x);y \approx b\cdot(x;y),$$

$$x;(b\cdot y) \approx b\cdot(x;y)$$

However, the *associative law* for ; is valid:

$$(x;y);z \approx x;(y;z) .$$

Returning to consideration of the operation $\dagger$, we remark that it is associative up to equivalence also; and since it is a join operation, we can prove

$$x\dagger y \lesssim z \text{ if and only if } x \lesssim z \text{ and } y \lesssim z .$$

This means that $E/\approx$ is algebraically a *semi-lattice* with $\dagger$ as the join. Whether $E/\approx$ is a lattice, the author does not know at the moment of writing. However, we can define *countably infinite* joins in the partially ordered set $E/\approx$ as follows: Given a sequence x_n of elements of E, there is a unique element we shall call

$$\sum_{n=0}^{\infty} x_n ,$$

which is characterized by the equation

$$\sum_{n=0}^{\infty} x_n = x_0 \dagger \sum_{n=1}^{\infty} x_n$$

In pictorial form the diagram is illustrated in Figure 23. This type of combination is clearly only of theoretical interest, but it does show why $E/\approx$ is countably complete. It may be possible that $E/\approx$ is a complete lattice, but the author doubts it.

As examples of equivalences involving infinite sums we have:

$$(x; \sum_{n=0}^{\infty} y_n) \approx \sum_{n=0}^{\infty} (x;y_n) .$$

In case $y_n \lesssim y_{n+1}$ holds for all n, we would also have:

$$(\sum_{n=0}^{\infty} y_n);x) \approx \sum_{n=0}^{\infty} (y_n;x)$$

as a consequence of *continuity*. There are many other similar laws.

9. CONCLUSION. Starting with very simple-minded ideas about flow diagrams as actual diagrams, we introduced the idea of *approximation* which led to a *partially ordered* set of diagrams. A rigorous, mathematical construction of this set produced what proved to be a *complete lattice* -- the lattice of flow diagrams. Defined on this lattice were several *algebraic operations* and the lattice as a whole satisfied an equation that connected it with the approach of *analytic syntax*. But the *limits* available in a complete lattice introduced something new into the picture: *infinite diagrams*. In particular, these infinite diagrams provided solutions to algebraic equations (the solutions were *algebraic elements*) which could be identified with the intuitive concepts of *loops* and other "*recursive*" diagrams (i.e. with feedback). So much for *syntax*.

Semantics of flow diagrams entered when the mapping of *evaluation* was defined from the algebra of diagrams into the *algebra of functions* on a *state space* (which was also a lattice). A bit of argument was required to see that evaluation captured the intuitive idea of *flow* in a diagram, but it became clearer in the example of a while-loop. From there, it was safe to introduce the notion of *equivalence* of diagrams and to study the resulting algebra. The reason for working out the equivalence algebra is, of course, to *formalise* some general facts about semantics of flow diagrams.

Much remains to be done before we have a perfect understanding even of this elementary area of the theory of computation. For one thing, only a start on the systematization of the algebra under equivalence was made in Section 8. It may be that equivalence is *not* at all the most important notion, for there may be *too few* equations between diagrams holding as equivalences. A more useful notion is the *conditional equation*. That is, we might write

$$x = x' \vdash y = y'$$

for x, y, x', $y' \in E$ to mean that for all S and all $\mathcal{F}$, $\mathcal{B}$, and all $\sigma \in S$, *if* it is the case that

$$\mathcal{V}_S(\mathcal{F})(\mathcal{B})(x)(\sigma) = \mathcal{V}_S(\mathcal{F})(\mathcal{B})(x')(\sigma) \quad ,$$

then (as a *consequence*) we have

$$\mathcal{V}_S(\mathcal{F})(\mathcal{B})(y)(\sigma) = \mathcal{V}_S(\mathcal{F})(\mathcal{B})(y')(\sigma) \quad .$$

Note that this is not an implication between two equivalences, but from each *instance* of one equation to the corresponding instance of the other. Similarly, we could write:

$$x \sqsubseteq x' \vdash y \sqsubseteq y' \quad .$$

Also it is useful to be able to write:

$$x_0 \sqsubseteq x_0', x_1 = x_1', \ \ldots \vdash y = y'$$

to mean that in each instance in which all the *hypotheses* on the left are true, the conclusion on the right follows. Many important algebraic laws can be given such a form.

Thus, in order to have a really systematic and useful algebra of flow diagrams, one should study the consequence relation $\vdash$ and attempt to axiomatize all of its laws. An effective axiomatization may only be possible for equations involving a *restricted* portion of E, because E contains so many infinite diagrams. But it is an interesting question to ponder.

References

[1] M. Arbib, **Theories of Abstract Automata**, Prentice-Hall Series in Automatic Computation (1969)

[2] H. Bekič, *Definable Operations in General Algebras, and the Theory of Automata and Flowcharts*, Private Communication.

[3] G. Birkhoff, **Lattice Theory**, American Mathematical Society Colloquium Publications, Vol. 25, Third (New) Edition (1967).

[4] D. Park, *Fixpoint Induction and Proofs of Program Properties*, in **Machine Intelligence 5 (1969)**, Edinburgh University Press, pp. 59-78.

[5] D. Scott, *Some Definitional Suggestions for Automata Theory*, **Journal of Comnuter and System Sciences**, Vol. 1, No. 2 (1967), Academic Press, pp. 187-212.

[6] D. Scott, *Outline of a Mathematical Theory of Computation* in Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems (1970), pp. 169-176.

Programming Research Group Technical Monographs

August 1978

This is a series of technical monographs on topics in the field of computation. Further copies may be obtained from the Programming Research Group, (Technical Monographs), 45 Banbury Road, Oxford, OX2 6PE, England. (The cost indicated includes surface postage. If faster delivery is required it should be indicated and an additional 30 per cent sent.)

PRG-1 (Out of Print)

PRG-2 Dana Scott.
Outline of a Mathematical Theory of Computation
(£0.50)

PRG-3 Dana Scott.
The Lattice of Flow Diagrams
(£1.00)

PRG-4 (Cancelled)

PRG-5 Dana Scott.
Data Types as Lattices
(£2.00)

PRG-6 Dana Scott and Christopher Strachey.
Toward a Mathematical Semantics for Computer Languages
(£0.60)

PRG-7 Dana Scott.
Continuous Lattices
(£0.60)

PRG-8 Joseph Stoy and Christopher Strachey.
OSS - An Operating System for a Small Computer
(£1.00)

PRG-9 Christopher Strachey and Joseph E. Stoy.
The Text of OSPub
(£3.50)

PRG-10 Christopher Strachey.
Varieties of Programming Language
(£0.50)

PRG-11 Christopher Strachey and Christopher P. Wadsworth.
Continuations: A Mathematical Semantics for Handling Full Jumps
(£0.60)

PRG-12 Peter Mosses.
The Mathematical Semantics of Algol 60
(£1.00)

PRG-13 Robert Milne.
The Formal Semantics of Computer Languages and their Implementations

Available as:
Technical Microfiche TCF-2.
A set of 6 Microfiche (£4.00)
or
Photocopy PRG-X13 (£10.00)

PRG-14 Shan S. Kuo, Michael H. Linck and Sohrab Saadat.
A Guide to Communicating Sequential Processes
(£1.00)