The everywhere operator once more.

Let X, Y, Z be boolean functions defined on some space; let M, N be integer functions defined on that same space; let that same space be the understood range of the dummy s. Then, for instance, the equality of the functions X and Y would be expressed by

$$(\underline{A}s::\ X.s \equiv Y.s) \quad ;$$

similarly, M and N being the same function would be expressed by

$$(\underline{A}s::\ M.s = N.s) \quad ;$$

M being everywhere less than N would be given by

$$(\underline{A}s::\ M.s < N.s) \quad .$$

Besides functions denoted by identifiers, we allow functions denoted by expressions; when applied (to s), these function expressions are enclosed by parentheses. They are defined as one would expect, e.g.

$$(\underline{A}s::\ (\neg X).s \equiv \neg X.s)$$
$$(\underline{A}s::\ (X \vee Y).s \equiv X.s \vee Y.s)$$
$$(\underline{A}s::\ (X \wedge Y).s \equiv X.s \wedge Y.s)$$
$$(\underline{A}s::\ (X \Leftarrow Y).s \equiv X.s \Leftarrow Y.s)$$
$$(\underline{A}s::\ (X \equiv Y).s \equiv X.s \equiv Y.s)$$
$$(\underline{A}s::\ (M + N).s = M.s + N.s)$$

$(\underline{A}s:: (M < N).s \equiv M.s < N.s)$.

Note that we also allow constants in function expressions, e.g.

$(\underline{A}s:: (true).s \equiv true)$
$(\underline{A}s:: (X \equiv false).s \equiv X.s \equiv false)$
$(\underline{A}s:: (N+2).s = N.s + 2)$.

Function expressions enable us to reduce the number of function applications, e.g. instead of rendering the symmetry of the conjunction by

$(\underline{A}s:: X.s \wedge Y.s \equiv Y.s \wedge X.s)$

we can write

$(\underline{A}s:: (X \wedge Y).s \equiv (Y \wedge X).s)$

or even

(*) $(\underline{A}s:: (X \wedge Y \equiv Y \wedge X).s)$.

With a sufficiently rich repertoire of function expressions the number of applications can be reduced to 1 . We now propose to render in a formula like (*) the initial "$(\underline{A}s:: ($" by "[" and the final "$).s)$" by "]", yielding for (*) $[X \wedge Y \equiv Y \wedge X]$.

Similarly, M and N being the same function would be expressed by $[M = N]$.

Notice that in the transition from the universal quantification to the square brackets, all that has been lost is the irrelevant name of the dummy.

* * *

Above, we have introduced the square brackets as an abbreviation, as a "mere" notational device. Having done so, however, we have achieved more: the dummy s being no longer visible, we don't need to remember any more that once it was there! We don't need to remember that X, Y, Z, M, N started as functions: we can treat them as variables of new types. The only thing we need to remember is that the type of X, Y, Z has boolean connotations — in the sense that true and false are possible values and the logical operators are defined on them — and that the type of M, N has integer connotations — in the sense that 0, 1, and, say, 7 are possible values and that the arithmetic operations are defined on them.

For lack of a better name, we called them boolean and integer "structures" respectively. (Had I have to name them now, I might have chosen "profiles".) More important than their names is that we can postulate the algebraic properties of the operators introduced for them.

In this exercise, the boolean structures and the square brackets play a fundamental rôle. The square brackets now denote an operator —called "the everywhere operator"— of the type

boolean structures → {true, false}

or, calling true and false the two "boolean scalars", of the type

boolean structures → boolean scalars .

The fundamental properties of the everywhere operator are

(i) true and false are the 2 solutions of the equation

$$X: \quad [X] \equiv X \quad ,$$

(ii) $[(\underline{A}X :: X)] \equiv (\underline{A}X :: [X])$.

From (i) we deduce

- $[[X]] \equiv [X]$, i.e. the everywhere operator is idempotent;
- because false is disjunction's unit element and true its zero element

$$[Y \vee [X]] \equiv [Y] \vee [X] \quad ,$$

and similarly

$$[[X] \Rightarrow Y] \equiv [X] \Rightarrow [Y] \quad .$$

With trading and the above we derive from (ii)

- $[(\underline{A}X{:}\,[r.X]{:}\,X] \equiv (\underline{A}X{:}\,[r.X]{:}\,[X])$,

where we have written $[r.X]$ with a pair of square brackets to express explicitly that the range of X is given by an expression of type boolean scalar. A special instance is

- $[X \wedge Y] \equiv [X] \wedge [Y]$.

An immediate consequence is the monotonicity of the everywhere operator, i.e.

- $[X \Rightarrow Y] \Rightarrow ([X] \Rightarrow [Y])$.

* * *

A major advantage of the above algebraic introduction of boolean structures is that it creates a universe of discourse more abstract than set theory. The comparison with set theory is suggested because, in the functional view with which we started, X and Y can be viewed as the characteristic functions of subsets —to function X corresponds the set $\{s \mid X.s\}$— ; function $X \wedge Y$ is then the characteristic function of their intersection, etc. . Since sets are defined as composed of elements, it is hard to talk about sets without mentioning the elements as well. With the transition from the boolean functions to the boolean structures the dummy s disappeared from our formulae, but not only that: we have got rid of the underlying space with its individual points!

We can introduce the points of the space into the world of the boolean structures by postulating the existence of special boolean structures called "point predicates". (As derivations of characteristic functions they would be derived from the characteristic functions of the singleton sets.) With x ranging over the point predicates, the notion of point predicates can now be captured by postulating

(iii) $[(\underline{E}x :: x)]$

(iv) $[x \Rightarrow \neg X] \equiv \neg[x \Rightarrow X]$ for all x, X .

(Set-theoretically, (iii) states that the union of all singleton sets is the universe, and (iv) that, given a set and a singleton set, the latter is a subset of either the former or the former's complement.)

The great thing about (iii) and (iv) is that we can abstain from postulating them, thereby creating a formal system that is intrinsically simpler to work with. (For a discussion of the same simplification in connection with the relational calculus, see [0].)

The usual name for boolean structures is "predicates"; this the name that will be used in the sequel.

* * *

Leibniz's Principle states that a function applied to equal arguments yields equal values, formally usually expressed as

$$p=q \Rightarrow f.p=f.q$$

or, equivalently,

$$p=q \wedge f.p=k \equiv p=q \wedge f.q=k \quad .$$

Because the arguments p and q could be structures —in which case $p=q$ would be a predicate and not necessarily a boolean scalar— and because f could be structure-valued, we reformulate Leibniz's Principle as

(v) $$[p=q] \Rightarrow [f.p=f.q]$$

or, equivalently,

$$[p=q \wedge f.p=k] \equiv [p=q \wedge f.q=k] \quad .$$

Formula (v) holds for all p,q,f of the appropriate types. There are functions, known as <u>punctual functions</u>, for which the stronger (vi) holds:

(vi) $$[p=q \Rightarrow f.p=f.q]$$

or, equivalently,

$$[p=q \wedge f.p=k \;\equiv\; p=q \wedge f.q=k] \quad .$$

The logical connectives — i.e. $\equiv \wedge \vee \Rightarrow \Leftarrow \neg$ — are all punctual in all their arguments, e.g.

$$[(p\equiv q) \Rightarrow (p\wedge r \equiv q \wedge r)] \quad ,$$

this in contrast to the relational operators: for composition we have only general Leibniz (v)

$$[p\equiv q] \Rightarrow [p;r \equiv q;r] \quad .$$

<u>Remark</u> Matrix calculus provides an environment in which punctuality can be identified. Negation is punctual — $(-A)_{ij} = -(A_{ij})$ — , the transpose — $(A^T)_{ij} = A_{ji}$ — is not. The matrix sum is punctual, the product is not. This is not too surprising, since the relational calculus can be modelled as a boolean matrix calculus with for (addition, multiplication, matrix multiplication) the triple (disjunction, conjunction, composition). (End of Remark.)

* * *

There is a wide-spread inclination to render the application of the everywhere operator by special infix operators, e.g.

$$X == Y \qquad \text{for} \qquad [X\equiv Y]$$

$$X \leqslant Y \qquad \text{for} \qquad [X\Rightarrow Y] \qquad ,$$

but this way of eliminating square brackets leads to notational conflicts. Performing the elimination on the Golden Rule

$$[X \wedge Y \equiv X \equiv Y \equiv X \vee Y]$$

destroys the associativity and leads - see [1] - to 5 different "theorems" of set theory. A 6th form would be

$$X \wedge Y \equiv X \equiv Y \equiv X \vee Y == true \quad ,$$

a device that can be used to render for instance

$$[X \vee \neg X] \quad \text{by} \quad X \vee \neg X == true \quad ,$$

but the device is not recommended. It would lead to rendering (ii) by

$$((\underline{A}X :: X) == true) \equiv (\underline{A}X :: X == true) \quad ,$$

but this has now become a very special property of the constant true , for in general

$$((\underline{A}X :: X) == Y) \text{ and } (\underline{A}X :: X == Y)$$

are unrelated.

Besides generating the need for more and more symbols, the convention of special infix operators generates the need for more theorems. In the case of the $\leq$ introduced above we would need

$$(\underline{E}X :: X) \leq Y \equiv (\underline{A}X :: X \leq Y)$$
$$X \leq (\underline{A}Y :: Y) \equiv (\underline{A}Y :: X \leq Y) \quad .$$

The great notational advantage of the square brackets is that, once they have been accepted, they free us from the dilemma whether to introduce a special operator like $==$ or $\leq$. Giving us all the combinatorial freedom we need, the square brackets solve these problems once and for all.

[0] Alfred Tarski, On the Calculus of Relations, Journal of Symbolic Logic, 6 (3); pp. 73-89; 1941

[1] Edsger W. Dijkstra & Carel S. Scholten, Predicate Calculus and Program Semantics, Springer-Verlag, p. 60; 1990

Austin, 26 November 1990

prof. dr. Edsger W. Dijkstra,
Department of Computer Sciences,
The University of Texas at Austin,
Austin, TX 78712-1188
USA