Copyright Notice

On substitution, replacement, and the notion of a function.

With respect to substitution and replacement, two extreme attitudes seem to prevail. Either the mechanisms are deemed so simple that the author uses them without any explanation or statement of the rules, or the author tries to give a precise statement of the most general rules and ends up with 200 pages of small print in which all simplicity has been lost. Neither is entirely satisfactory for the purposes of this monograph.

The first extreme has to be rejected in view of our stated goal of presenting fairly "calculational" arguments. The "mindless" manipulation of uninterpreted formulae that is implied by "letting the symbols do the work" is clearly impossible without a sufficiently clear view of the permissible manipulations.

The second extreme has to be rejected because this is not a treatise on formal logic and would probably provide much more than needed for the limited scope of the theory we have set out to develop.

In this section we shall try to sail between Scylla and Charibdis by mainly confining ourselves to the identification of the sources of complication, as an awareness of them should suffice to protect the human formula manipulator from making the "silly" mistakes.

* * *

It all starts innocently enough. Given

(0) $13 = 5+8$ and

(1) $x > 13$,

we all feel pretty confident in concluding

(2) $x > 5+8$;

the equality (0) tells us that in (1) we may replace 13 by 5+8 without changing (1)'s value. We are, however, courting trouble as soon as we view the transition from (0) to (2) purely as a matter of string manipulation. Given

(3) $y < 131$ and

(4) $z = q - 13$,

we should conclude neither

(5) $y < 5 + 81$

from (0) and (3), nor

(6) $z = q - 5 + 8$

from (0) and (4). The moral of the story is that there is more to it than just the manipulation of strings of characters.

Though strings of characters are what we write down and see, the only strings we are interested in are formulae that satisfy a formal grammar. The rôle of this grammar goes beyond defining which strings represent well-formed formulae, it also defines how a well-formed formula may be parsed. Indicating permissible

parsings with parentheses, we would get

(2') $x > (5+8)$

(5') $y < (5 + (81))$

(6') $z = ((q - 5) + 8)$.

Comparing (2') with (5') and (6') reveals what went wrong. "Replacing equals by equals" should be a reversible operation, i.e. from (0) and (2) we should be able to conclude (1), as, indeed, we are. Whereas in (2') 5+8 occurs as syntactic unit of an admissible parse, in (5') and (6') it does not. The moral of the story is that, apart from the internal structures of the subexpressions exchanged, the two formulae equated should admit the same parsings. The consequences of this requirement obviously depend on the grammar adopted: with a grammar with operators of different binding power, for instance, a parenthesis pair may have to be introduced.

Remark We don't require the grammar to be unambiguous. For instance, the associativity of the addition might be rendered grammatically in a production rule like

<sum> ::= <addendum> | <sum> + <sum> .

(End of Remark.)

* * *

We get another complication with grammars in which variables may have a limited scope, such as the dummy i in the quantied expression

$(\bar{A}i :: Q.i)$. Here the outer parenthesis pair delineates the scope of the dummy i , which is therefore said to be "local to this expression"; a variable that occurs in the text of an expression, but is not local to it or to one of its subexpressions is said to be "global to that expression". (For "local" and "global", also the terms "bound" and "free" respectively are used.) Sometimes it is convenient to regard "global to an expression" as "local to that expression's environment".

Variables are place holders in the sense that their main purpose is to get something substituted for them. More precisely: when we substitute something for a variable, each occurrence of that variable is replaced by that same something. Identifying different variables by different identifiers is a means for indicating which places have to be filled with the same something in the case of substitution. From

$$[2 \cdot X = X + X] \quad \text{for any } X$$

we may thus derive

$$[2 \cdot Y = Y + Y] \quad \text{and}$$

$$[2 \cdot (Y - Z) = (Y - Z) + (Y - Z)] \ .$$

whereas the "conclusion"

$$[2 \cdot (Y - Z) = (Y - Z) + X]$$

is unjustified, not all occurrences of X having been replaced.

In the absence of local variables, all variables are global and life is simple. There are as many variables as we have identifiers, they are in one-to-one correspondence to each other to the extent that we hardly need to distinguish between the variable and the identifier identifying it.

But consider now an expression like

(7) $\quad (\underline{A}\, i :: P.i) \vee (\underline{E}\, j :: Q.j)$.

In this expression we have four variables: P, Q, i and j. The first two are global, i.e. expression (7) is deemed to occur in an environment that may have more occurrences of P and Q, and substitution of something for P (or Q) is deemed to apply to all occurrences of P (or Q), both in (7) and in its environment. As long as the environment of an expression like (7) is left open, for each of the variables global to that expression the set of its occurrences is open-ended.

The other two, i and j, are sealed off, so to speak: their sets of occurrences are closed. The first one, introduced after the quantifier $\underline{A}$, is local to the first disjunct, and the second one, introduced after $\underline{E}$ is local to the second disjunct. Their occurrences are by definition confined to their respective scopes as delineated by the parenthesis pairs.

In the environment of (7), P and Q play different rôles: P refers to the term in the universally quantified expression and Q to the

term ~~in the term~~ in the existentially quantified expression. In contrast, the variables i and j play in the environment no rôle at all. As a result it should not be very relevant for the environment which identifiers have been used to denote these local variables. In fact, instead of (7), we could equally well have written

(8) $(\underline{A}i :: P.i) \vee (\underline{E}k :: Q.k)$

or even

(9) $(\underline{A}i :: P.i) \vee (\underline{E}i :: Q.i)$.

Since a variable local to an expression plays by definition no rôle in that expression's environment, we can always choose a fresh identifier to denote it. In (9) we have carried that freedom to its ultimate consequence and have used the same identifier for both local variables. No confusion arises as it is still quite clear which variable is universally quantified and which is existentially quantified.

Note that we have taken a major step. The one-to-one correspondence is no longer between variable and identifier, but between variable and identifier together with its scope.

It is customary to go one step further. It so happens that (7) is equivalent to

(10) $(\underline{A}i :: P.i \vee (\underline{E}j :: Q.j))$.

In view of (9), we would like to permit

(11) $(\underline{A}i :: P.i \vee (\underline{E}i :: Q.i))$

as well. In fact we do.

Scopes, being delineated by matching parenthesis pairs, are either disjoint, as in (7), (8), and (9), or nested, as in (10). But what about (11)? Well, the scopes are still nested: the inner scope pertains to the existentially quantified variable, the outer scope to the universally quantified variable. More precisely, for any occurrence of a variable we find the scope of that variable by looking for the smallest scope (i) that introduces the identifier used to denote the variable and (ii) whose delineating parentheses surround the occurrence. A consequence of this rule is that, in (11), the double usage of the identifier i has created a "hole" in its outer scope: the scope of the universally quantified i extends in principle -as in (10)- from the outer opening parenthesis until the outer closing parenthesis, but in (11), over "(E i:: Q.i)", though within the outer parenthesis pair, the identifier i does not refer to the universally quantified variable, but to the existentially quantified one.

The things to remember are
(i) that no identifier may occur outside its scope, and
(ii) we are always free to replace, all through its scope, all occurrences of a local variable by a fresh identifier.
We assume that the reader remembers this and shall often not mention explicitly its specific consequences. For instance, in the

presentation of the axiom

$$[(\underline{A}i:: Q \vee P.i) \equiv Q \vee (\underline{A}i:: P.i)]$$

we feel free to omit the standard caveat "provided i is not global to Q": we should remember that we are free to view i as a fresh identifier and that, were i global to Q, its occurrence in the second Q would be outside its scope. Hence we prefer to consider the caveat as "obvious".

Remark That last desire is perhaps very naive in view of the amount of effort logicians have spent on explaining "bound" versus "free" variables. They consider perhaps more complicated manipulations than we do. From the moment that we decided to delineate the scopes of local variables explicitly by an obligatory parenthesis pair and to adopt a syntax that leaves no doubt as to which local variable(s) a scope pertains – and this decision was taken many years ago – we never made an error due to clashing identifiers. One manipulates in the constant awareness that each local variable could have been identified by a fresh identifier; it is a permanent supervision not unlike the physicist's awareness that his formulae should be invariant under change of units or of coordinate system. (End of Remark.)

* * *

We have to say a word or two about functions because, depending on the ways permitted for their definitions, it may be ques-

tionable whether they are well-defined at all. Fortunately, our functions will be simple.

First of all, they will be total functions of arguments of familiar types, such as boolean or integer - which we deem well-understood - and structures thereof. Their values will be of similarly simple types: a new function is always a mapping from a previously introduced domain to a previously introduced domain.

Aside A, say, integer function that accepts itself as argument, is of a type α that satisfies the type equation

$$\alpha = \alpha \rightarrow \text{int} \quad .$$

With such a recursively defined type it is no longer obvious whether a function definition makes sense or can lead to a contradiction. Such functions, fortunately, fall outside the scope of this monograph. (End of Aside.)

Secondly, our ways of defining functions are very simple. They are only two. Firstly we define functions by means of expressions (in their arguments), built from previously defined primitives. An example would be the definition of square (as a mapping of type $\text{int} \rightarrow \text{int}$) by

$$\text{square}.x = x^2 \quad .$$

Furthermore we shall define functions as solution of an equation that contains the argument(s) of the function being defined as

parameter(s). In that case we take upon ourselves to prove that such an equation has for any value of the parameter(s) a unique solution.

Note We could have defined square.x as the solution of

$$y: (y = x^2)$$

– here the prefix y: denotes that the following boolean expression is considered an equation in the unknown y –. The above equation, however, is somewhat special in the sense that the task of solving it is empty. (End of Note.)

Our main use of the notion of a function will be that, for f a function, we appeal to

$$(x0 = x1) \Rightarrow (f.x0 = f.x1) \quad ,$$

a rule for which we give credit to Leibniz by naming it after him. Since the argument might be a structure and since the function value might be a structure, the above should be formulated as

$$[x0 = x1] \Rightarrow [f.x0 = f.x1] \quad .$$

But now a slight problem emerges, that is most easily shown with a simple example. Let us define the function f, mapping two boolean structures on a boolean structure, all three on the same space, by

$$[f.X.Y \equiv [X] \wedge Y] \quad .$$

Leibniz's rule gives us immediately

$$[X_0 \equiv X_1] \Rightarrow [f.X_0.Y \equiv f.X_1.Y]$$

$$[Y_0 \equiv Y_1] \Rightarrow [f.X.Y_0 \equiv f.X.Y_1] \quad .$$

Both conclusions are correct; the trouble with the last one is that it follows from the stronger

$$[\,(Y_0 \equiv Y_1) \Rightarrow (f.X.Y_0 \equiv f.X.Y_1)\,] \quad ,$$

which is also correct, whereas the similar strengthening of the first conclusion would be incorrect. Evidently, the ways in which f depends on its two arguments are fundamentally different.

To denote the difference, we call $f.X.Y$ a spatial function of X and a punctual function of Y. We shall return to the distinction later.

Austin, 21 April 1986

prof. dr. Edsger W. Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1111
United States of America