Add: replacement during procedure call is analagous.

Algol conference in Mainz, 14.-16. dec. 1959.

THE ARITHMETIC TYPES OF EXPRESSIONS.
THE MEANING OF THE ASSIGNMENT STATEMENT.

An attempt at a clarification. - Peter Naur.

Problem: What is to be the meaning of the assignment statement

V := E

where V and the identifiers, functions and numbers have been declared to be of different types?

In more detail the problem is this: on the basis of the type of the constituents of E it is possible, a priori and from purely mathematical rules, to determine the possible types of E. The types of E are, however, not unique, since they may depend on the actual numbers. This is shown in the following table for the pair of types integer (denoted i) and simple variable (denoted s):

E_1	i	i	s	s
E_2	i	s	i	s
$E_1 + E_2$	i	i,s	i,s	i,s
$E_1 - E_2$	i	i,s	i,s	i,s
$E_1 \times E_2$	i	i,s	i,s	i,s
E_1 / E_2	s,i	s,i	s,i	s,i

The question is now: what should we view the assignment statement what meaning do we attach to the assignment statement when in addition V is type-declared?

First possibility

Type "compression" is never allowed.

On this philosophy the widest possible type for the expression (which may of course be determined mechanically fairly simply, at least for the above pair of types) is compared with the declared type for the V. If the widest type of E is not "contained" within the type of V the assignment statement is considered an error. (to be rejected at compilation time).

Second possibility

The transfer function is automatically applied (Rutishauser AB 5.10).

On this philosophy a suitable transfer function, which will reduce the widest possible type of the expression to the type of V, is invariably applied to the expression before the assignment, i.e. the expression is interpreted as

V := transfer (E).

Third possibility

Type "compression" at users responsibility. Transfer function applied only if necessary to correct for imperfect arithmetics.

On this philosophy type compression is explicitly allowed in all cases where

the expression E is known for mathematical reason to yield a result within the type of V, while it is an error in all other cases.

From the point of view of translators this is to be understood as follows: A translator may be expected to perform an analysis of the expression. In simple cases this analysis may be sufficient to make sure that with the kind of arithmetics actually performed in the machine and on the assumption that the implied assertion of the programmer to the effect that the result is in fact Of the restricted type of V is correct the translator may be able to produce a program which is (again on the above assumption) correct without applying any transfer function. In more complicated cases the analysis performed by the translator may not be sufficient to give an answer, or the analysis may show that the arithmetics of the machine may not be able to yield the exact answer, and in this case the transfer function may be evoked. *)

It should be noted that this manner of treatment in different translators must be expected to give the same and correct answer if, and only if the expression does in fact yield an answer which is mathematically speaking of the correct type, (this is the case when the transfer function only changes the value of the expression slightly anyway), while expressions which do not do so may give completely different answers in different translators. (Warning against misuse).

Example: Program: integer (i, p); begin .. p := i/3 ; ..... end

First possibility: The expression is an error.

Second possibility: The correct rounding-off is always made.

Third possibility:

a) Crude translator: For simplicity reasons no analysis is performed and the rounding is always made applied in order to correct for the imperfection of the machine arithmetics.

b) Sophisticated translator for machine with inexact arithmetics in case of division: The translator discovers that the expression is a division and inserts the rounding to correct for the known machine characteristic.

c) Sophisticated translator for machine with exact division: The translator discovers that only a division is involved and consequently omits the application of the rounding function. If erroneously i is in practise not divisible by 3 this may lead to a completely wrong answer (the fractional part of i/3 may in the machine appear in positions where the exponent of p is supposed to be, or the like). In other words

Addenda to Third possibility: *) It is legitimate to let the translator work on the assumption that the type compression is in fact mathematically correct, even if it not equiped so as to check this assertion.