Remaining Trouble Spots in Oberon

Niklaus Wirth 22.3.2008

Constant Parameters

Constant parameters were introduced in Revised Oberon (Oberon-07), because strings could not be used as actual parameters, if the corresponding formal parameter was a value parameter. If the formal parameter therefore was specified to be a VAR parameter, assignments could be made to strings, which supposedly are constants.

In the Oberon compiler for the ARM, constant parameters are handled like value parameters, if their type is scalar, and like VAR parameters otherwise. In any case, they are treated as read-only objects.

This seems quite satisfactory. However, a purist will not be satisfied. Consider the example

```
VAR a: ARRAY n OF CHAR;
PROCEDURE Q; BEGIN a[i] := "%" END Q;
PROCEDURE P(CONST s: ARRAY OF CHAR); BEGIN … ; Q; … END P;
BEGIN … P(a) …
```

Q will change the value of *a* and thereby of *s* which is considered to be a constant.

Character Arrays and Strings

Strings are considered as sequences of characters that can be assigned to arrays. This seems quite satisfactory but bears a problem, because short strings can be assigned to longer arrays. When comparing two arrays, the question arises, whether the entire arrays or only the strings should be compared. Remember that strings are always terminated by a null character. The point is illustrated by the following example:

```
VAR s0, s1: ARRAY 8 OF CHAR;
s0 := "abc"; s1 := "abc";
s0 = s1?    true
s1[4] := "?"; s0 = s1   still true?
```

The ARM implementation gives the answer "true", because comparison stops, when a null character is encountered (s[3] = 0X). Note that arrays cannot be compared, except when the elements are characters, because only then we postulate an order.

In the case of assignment of an array to another array variable, the entire array will be copied, regardless of whether it contains null characters or not.

Character vs. String

Since the times of Pascal there exists the question of how to denote a string of length 1. The trouble is that "?" denotes a character constant, whereas "" and "01" denote strings of length 0 and 2 respectively. Several "solutions" have been considered, but any change in this matter was deemed not worth the trouble. For example, Mesa used 'A to denote the single character, and "A" to denote the string of length 1. (In Oberon, we now let "x" denote the singleton string, whereas 'x' denotes the character constant *x*).

The For Statement

It's a story as old as Algol: What is the value of the control variable after a for statement?

```
FOR i := 1 TO n DO sum := sum + i END
```

Is the terminating value *n* or *n+1*, or what else? It is probably wisest to leave the question unanswered, because then the implementer can choose what suits him best, and the programmer knows he must avoid using *i* after the for statement.

A better solution would, however, have been to specify in the language definition that the for clause declares a fresh variable. This implies that *i* is unknown outside the for statement.

These may seem to be many remaining trouble spots, but actually they are few. The inclined reader is referred to

D.E. Knuth. The Remaining Trouble Spots in Algol 60. *Comm. ACM, 10,* 10, 611-618, (Oct. 1967)