The Nabla Trick

The other day I invited my CS class to develop, with only minor prompting from my side, a program for the computation of the greatest common divisor of two given numbers X and Y. The problem was specified as follows:

$$\{1 \leq X \wedge 1 \leq Y\}$$
$$\text{Euclid}$$
$$\{x = X \underline{\text{gcd}}\ Y\}\ .$$

(Admittedly, the arbitrary identifier "Euclid" had been chosen somewhat too indicative, at least for those students who knew Euclid's algorithm.)

We quickly discovered that the simplest program meeting the functional specification was the assignment $x := X \underline{\text{gcd}}\ Y$. But for the sake of the argument and the fun we discarded it. It then was suggested that we might need a program with a repetition and therefore had to think about an invariant relation.

At this stage the class started to show an overwhelming enthusiasm and energy by dictating me various complete program texts. Needless to report on the quality of the suggestions. We then agreed that chaos had got the upper hand over systematic manners. We also agreed that we had better try a simpler programming problem first.

So I proposed

$$\{1 \le X \ \wedge \ 1 \le Y\}$$
Nabla
$$\{x = X \nabla Y\} .$$

Needless to say that the class was shocked (by so much ruthlessness on my part?) ; it dropped back into silence and even failed to suggest $x := X \nabla Y$. After a while some students wanted to know about ∇ -- not everything, but just something simple. We then agreed that us going into ∇-theory would pretty much be unavoidable, but that we had better investigate first how far we could go without knowing anything about ∇ at all.

The theorem of invariance for repetitions being what it is, the postcondition should follow from the conjunction of the invariant and the negation of the guards. Disallowing ourselves ∇-expressions in the program text, we concluded that therefore the postcondition had to follow from the invariant. This was the moment to reveal ∇'s idempotence

$$x \nabla x = x .$$

Then it did not take long before the postcondition got rewritten as

$$x \nabla x = X \nabla Y$$

(, which was the only thing the class could do), and the invariant was chosen to be

$$x \nabla y = X \nabla Y$$

(, which for the sake of its initializability was the simplest thing the class could do). The negation of the guards then had better imply $x = y$. Etcetera. Thus, Euclid's algorithm was derived.

+

The trick, being as simple as it is, turned out to be amazingly effective. It commanded the separation of concerns needed for an orderly design of the algorithm. As long as the "meaningful" identifier <u>gcd</u> was on the blackboard, all the unwanted associations, connotations, and complexity coming along with it distracted attention from what was relevant for the development.

I would not have reported the Nabla Trick if it were merely a trick played on a class of novices. We can be pretty sure that the trick of replacing (too) familiar names or symbols by something meaningless can also serve mature mathematicians or computing scientists, especially in those circumstances when a design tends to become complicated (, the latter being usually taken as a defence for using meaningful identifiers).

Austin, 5 March 1988

W.H.J. Feijen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 - 1188
USA