

UNDERSTANDING AND IMPROVING LISP PROGRAMS

Harald WERTZ
Université Paris 8
Département d'Informatique
75012 PARIS

We are currently investigating the creation of an didactic environment for the teaching of a programming language to approximately 1500 students each year. At this effect we are constructing a robust and perspicuous system : VISION & CAN & PHENARETE (GREUSSAY 1977, GOOSSENS 1977, WERTZ 1976), a system designed to help the individual apprentice in the process of developing, writing and debugging programs.

In this note we will describe some aspects of PHENARETE, the convivial part of the system : PHENARETE receives as input a student proposition of a program - which may contain as well syntactic as semantic errors - and will deliver as output her improved propositions of this program : a finite sequence of approximations.

With the help of our system, the learning of a programming language is done in a cycle :

user proposition	improvement and approximation system	propositions of PHENARETE
---------------------	---	---------------------------------

PHENARETE

Before constructing PHENARETE we observed for one year our students beginning to learn LISP, to see which kinds of errors are statistically the most current. There we found principally five sorts of errors :

- absence of variables
- Inversion of variables
- grouping errors
- absence of the conditionnal instruction
- non-représentation of the termination of a computation.

Our system is particularly efficient in the detection of these kinds of errors. It doesn't try to verify the program with respect to the intentions of the programmer, but it tries to understand the programming language constructs used in the program, to see the interaction of the different parts of the program and to detect inconsistencies, with a particular emphasis on the verification of the termination of the program. In some way PHENARETE may be considered as an intelligent interpreter : she does not execute but she interprets the program, and the result of an interpretation are some propositions (propositional interpreter). PHENARETE incorporates the programming knowledge of a programming apprentice.

Let me illustrate the reasoning involved in an interpretation of a simple LISP program (main steps only) : suppose a student has submitted the following program (the numbers are for references in the text).

- .1. (DE REV (X Y)
- .2. ((REV X (CONS (CAR X) Y)))
- .3. ((NULL Y) X))

First, (line 2 and 3) the grouping of the body of the function suggests that the user has omitted the COND-function call, so PHENARETE introduces (line 2-b) «COND». Then (line 2) we find as the first clause of the COND immediately a recursive call of the function REV, and following (line 3) another clause. Knowing that line 3 can never be attained during an execution of the program we can invert the two clauses. This gives :

- .1'. (DE REV (X Y)
- .2'. (COND
- .3'. ((NULL Y) X)
- .4'. (T (REV X (CONS (CAR X) Y))))

So far the surface improvements. Now a closer look at line 4' tells us that X and Y will be lists, when the function is invoked, and that the first argument (X) is inchanged and the second argument (Y) will grow longer. In line 3' the case second argument = NIL is solved, but this is the only case where the computation of REV will terminate. If Y is different of NIL, the only constructive computation done in the program is the creation of the new list argument 2. So, when hypothesising that the result of REV will be this list just created, we have to force a recursion stop. Let us try to call REV with the CDR of argument 1 :

(T(REV (CDR X) (CONS (CAR X) Y)))

but the recursion will not stop either. So let's try to introduce a supplementary test ((NULL X)). And, always under the hypothesis that Y will be the result of the computation, this gives us :

((NULL X) Y).

This will be the first proposition of PHENARETE ;

Proposition 1.

```
(DE REV (X Y)
  (COND
    ((NULL Y) X)
    ((NULL X) Y)
    (T (REV (CDR X) (CONS (CAR X) Y))))
```

But this new line is just the line 3' with X and Y inverted, so let's try without line 3'. Ok, that is also a recursive procedure which stops with the correct algorithm, so another proposition :

```
(DE REV (X Y)
  (COND
    ((NULL X) Y)
    (T (REV (CDR X) (CONS (CAR X) Y))))
```

Finding no more possible interpretation, PHENARETE will stop, after these two propositions. During the interpretation-phase she justifies all the modifications she proposes in a way similar we did.

The system is implemented in VLISP-10 and is currently used by about 1500 students.

REFERENCES :

- CHAILLOUX J., VLISP-10, Manuel de références, Dpt. Informatique, Université Paris 8, RT-17-76, 1976
GOOSSENS D., CAN, RT-03-77, 1977
GREUSSAY P., Différence et répétition dans les systèmes VISION,, RT-06-77, 1977
WERTZ H., PHENARETE, Springer Verlag, G-1.6 Jahrestagung, Stuttgart, oct. 1976, pp. 427-441.