Short Note

HELP: Interactive Programming System

KENJI KAIJIRI*

HELP has been designed as an interactive programming tool for the structured FORTRAN RATFOR-R, which assists stepwise construction of programs. HELP has the following facilities: refinement of abstract instructions, integration of program elements, modification of program elements, monitoring of program states, incremental parsing, and selective listing. HELP is implemented with RATFOR-R on ACOS-600S.

1. Introduction

Stepwise refinement is a topdown design approach to program development. In a stepwise refinement, a program is gradually developed in a sequence of refinement steps. This refinement requires so many housekeeping tasks to be done that programming tools which assist this approach are essential for use in interactive environments. Several programming environments have been proposed which assist the stepwise refinement: Program Synthesizer [1], Poesy [2], etc.

Synthesizer uses the template method and is only capable of refinement with syntactic categories (e.g., statements, expressions, etc.). With Poesy, on the other hand, a user is allowed to write special comments, which are later refined with refinement instructions. Only Synthesizer has a selective, though limited, listing facility; a user can attach a special comment to a syntactic category, and if a user has selected the ellipsis mode, this comment is displayed instead of the program element corresponding to the syntactic category.

We have designed an interactive programming environment HELP as a training tool for learning the stepwise refinement. It executes only the housekeeping tasks in the stepwise refinement, and it is the user that is responsible for the refinement. HELP has the following facilities:

(1) Abstract instruction
We can write abstract instructions as special comments within a program in the Poesy style.

(2) Integration of program elements
HELP integrates refined program elements into a complete program, and has the selective listing facility.

(3) Monitoring of program states
The user can monitor the refinement process and the usage of variables.

(4) Incremental parsing
HELP parses refined program elements incrementally.

*Faculty of Engineering, Sinshu University, Wakasato, Nagano 380, Japan.

2. The HELP System

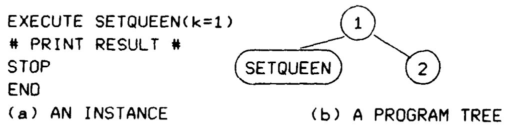
A program which is to be developed with HELP is a compound object composed of abstract and/or concrete instructions. We call abstract instructions PROCESSes, and refined abstract instructions and subprogram bodies their INSTANCEs. An INSTANCE may include PROCESSes. A PROCESS may be regarded as an open subroutine (macro). We can write the same PROCESS many times in a program. In HELP an abstract instruction is written as a comment as follows: #SOLVE AN EIGHT QUEEN PROBLEM#. An abstract instruction can be left as an ordinary comment in the head of a refined INSTANCE. There are three kinds of subprograms in RATFOR-R, i.e., subroutines, functions, and remote blocks.

A program is represented as a tree called a "program tree," whose node represents an INSTANCE and whose edge represents a PROCESS-INSTANCE relation (e.g., edge (1, 2) means that the INSTANCE "1" includes a PROCESS whose INSTANCE is "2"). The root of a program tree is the top level description of a given problem, i.e., the initial INSTANCE. An INSTANCE is parsed, and is stored as a parse tree. A number, called the entry code, is assigned to each PROCESS automatically, and a PROCESS and its INSTANCE are identified by this number. In Fig. 1(b), "1" and "2" are the entry codes. Fig. 1(b) shows a program tree for the program (INSTANCE) of Fig. 1(a). In Fig. 1(b), "1" represents the INSTANCE of Fig. 1(a) and "2" represents the INSTANCE of the PROCESS #PRINT RESULT# (when refined).

The commands of HELP are classified into six groups as follows:

(1) Program input assistance (PROGRAM, PROGRAM fname, SAVEL fname, SAVET fname, PROGEND, and BYE)

(2) Subprogram input assistance (REMOTE, REMOTE fname, REMEND, SUBPROG, SUBPROG fname, and SUBEND)



```
EXECUTE SETQUEEN(k=1)
# PRINT RESULT #
STOP
END
(a) AN INSTANCE
```

Fig. 1 An instance and its program tree.

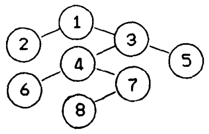


Fig. 2 An example of a program tree.

(3) Refinement (REFINE i, BACK, FORTH i, and TRANSFORM)

These commands relate to the scanning of a program tree and the incremental parsing. An INSTANCE is parsed by the "TRANSFORM" command and its parse tree is inserted into the corresponding node in the program tree.

(4) Selective listing (LIST, LIST C-, LIST C-i, LIST -C, LIST i-C, and LIST name)

These commands display several sectional lists around the current INSTANCE. For example, in Fig. 2, we suppose that "4" is the current INSTANCE being processed and that all the PROCESSes have been refined. The "LIST" command shows only the INSTANCE "4." The INSTANCEs "6" and "7" are displayed as PROCESSes, i.e., abstract instructions. The "LIST C-1" command shows the INSTANCEs "4", "6", and "7" in an integrated representation. The INSTANCE "8" is displayed as a PROCESS ("-1" means "one level down"). The "LIST 2-C" command shows the INSTANCEs "1" to "5." The INSTANCEs "6" and "7" are represented as PROCESSes ("2-" means "two level up").

(5) State monitor (STATE, ASTATE, and ASTATE ALL)

These commands display PROCESSes, subprograms, and identifiers which are referenced. The "STATE" command displays those which are referenced in the current INSTANCE. The "ASTATE" command displays those which are referenced but not declared in all INSTANCEs.

(6) Miscellaneous (EDIT and HELP)

The "EDIT" command invokes a simple line editor. The "HELP" command explains the set of HELP commands to users.

3. Example

We show the program development process using HELP for the well-known eight queen problem. On starting HELP, we will be asked with COMMAND?" as shown in Fig. 3. If we are planning to develop a new program, we type "PROGRAM," whereas if we are planning to modify an existing program, we type "PROGRAM fname." When we type "PROGRAM," line numbers will sequentially be displayed. We type in an INSTANCE of the top level PROCESS (in this case, this INSTANCE can be something like #SOLVE AN EIGHT QUEEN PROBLEM#) line by line following these numbers. In Fig. 3, the INSTANCE includes one subprogram call and one PROCESS (EXECUTE SETQUEEN and #PRINT RESULT#). A CR key just following a line number indicates the end of the INSTANCE, which allows HELP to return to the command level (COMMAND?).

Next the INSTANCE will be parsed and transformed into a parse tree. HELP has a facility of monitoring the state of the current INSTANCE to allow the user to control the refinement process. The "STATE" command does this task as shown in Fig. 3. With three kinds of entries, i.e., subprograms, identifiers, and PROCESSes, their states are displayed. "US" means "unspecified."

```
*********** RATFOR HELP SYSTEM ************
** COMMAND ? PROGRAM
  10 EXECUTE SETQUEEN(K=1)
  20 # PRINT RESULT #
  30 STOP;END
  40
** COMMAND ? TRANSFORM
....................................
** COMMAND ? STATE
**MAIN PROGRAM**
*** REMOTE BLOCK ***
SETQUEEN      US
*** IDENTIFIER ***
K             US
*** PROCESS ***
  3   US   STATEMENT        # PRINT RESULT #
** COMMAND ? REMOTE
  10 REMOTE SETQUEEN
  20 SAVE K
  30 J=1
  40 REPEAT
  50 [IF(#K-TH QUEEN IN J-TH ROW ?#)
  60 # SET QUEEN #
  70 J=J+1
  80 ] UNTIL(J>8)
  90 FLAG=0
 100 RETURN;ENDBLOCK
 110
....................................
** COMMAND ? REFINE 4
 PROCESS CONDITION    #K-TH QUEEN IN J-TH ROW ?#
  10 Y(J)==0 & RD(K+J-1)==0 & LD(K-J+8)==0
  20
....................................
** COMMAND ? BACK
....................................
** COMMAND ? LIST
  10    REMOTE SETQUEEN
  20    SAVE K
  30    J=1
  40    REPEAT
  50      [
  60       IF(#K-TH QUEEN IN J-TH ROW ?#)
  70         # SET QUEEN #
  80       J=J+1
  90      ]
 100    UNTIL(J>8)
....................................
** COMMAND ? REMEND
....................................
** COMMAND ?ASTATE
*** IDENTIFIER ***
K          US  IN  MAIN
J          US  IN  MAIN
....................................
*** REMOTE BLOCK ***
SETQUEEN    IS  IN  MAIN
*** PROCESS ***
STATEMENT   US   # PRINT RESULT #
STATEMENT   IS   # SET QUEEN #
....................................
** COMMAND ? EDIT
-5 INTEGER I,J,K,FLAG
-6 INTEGER Y(8),LD(15),RD(15),P(8)
-
....................................
** COMMAND ? PROGEND
** PROGRAM IS SPECIFIED **
```

Fig. 3 An example of program development by HELP.

"Specified" means that (1) a PROCESS is refined, (2) a subprogram is declared, or that (3) an identifier is declared. Entry codes are also displayed with PROCESSes. Next we will refine the remote block SETQUEEN. In this case we first type "REMOTE," and type in an INSTANCE (a remote block body) line by line. This INSTANCE includes two PROCESSes (#K-TH QUEEN IN J-TH ROW?# and #SET QUEEN#). In order to refine the first PROCESS, we type "REFINE 4," where "4" is the entry code of the first PROCESS. After the refinement we go back to the ancestor INSTANCE by the "BACK" command. We can get a pretty-printed version of the current INSTANCE by the "LIST" command. After completion of the refinement of the remote block SETQUEEN, the "REMEND" command terminates the remote block refinement process.

As a final step we insert the declaration of variables into the top level INSTANCE using the "EDIT" command. The current HELP system is capable of modifying an INSTANCE only through line insertion, exchange, and deletion. Fig. 3 gives an example of line insertion. Programming has been completed. The "PROGEND" command terminates the program input and indicates whether the developed program has been specified or not, i.e., whether all the entries in this program have been specified or not. In this example the program has completely been specified.

4. Conclusions

We have been using HELP to develop several programs such as a simple lexical analyzer and a calculator. Our experience in using HELP has led to a conclusion that HELP requires us to design a program hierarchically, i.e., we must design the overall program structure using PROCESSes and subprograms beforehand (a high level program description). Thus the program should have a hierarchical structure, causing any control transfer beyond the scope of an INSTANCE to be seldom used. The programming process and/or attitude is much influenced by the programming environment, and the programming metholodogy cannot really be used until its assisting tools have become available. In view of HELP being an experimental system, we would like to investigate effects of programming environments like this. In order to make HELP applicable to programming-in-general, we must add the following facilities to HELP in the future: transformation between an abstract instruction and a subprogram: a RATFOR-R interpreter; an editor in token level.

References

1. Teitelbaum, T. and Reps, T. The Cornell Program Synthesizer: A Syntax-Directed Programming Environment, *Comm. ACM* **24**, 9 (September 1981), 563–573.
2. Miyamoto, E. and Asami, K. A Text Editor Having Editing Facilities Based on Program Structure. *Trans. IPSJ* **20**, 6 (1979), 474–480.

(Received December 10, 1981; revised October 18, 1983)