Algorithm Animation with Symbol Processing Robots

Bard S. Crawford
Stage Harbor Software
9 Patriots Drive
Lexington, MA 02420 USA
+1 781 862 3613
bscrawford@aol.com

ABSTRACT

This experience report demonstrates several running programs with visual, animated, colorful displays of "algorithms in action." A common element of all of the programs is the use of "symbots" (symbol processing robots), which conceptually move about over a two-dimensional grid of cells. Each program is a "teaching example" illustrating selected aspects of computer science, object oriented program specification and design, or collaborative engineering. The Situation Driven Modeling Language (SDML) created by Cherry [1] played a key role in developing these examples. (See web site http://sdml.com for more details.)

I. INTRODUCTION

Four examples of algorithm animation based on the use of "symbots" (symbol processing robots) have been developed and are demonstrated in this presentation. As a symbot moves it can read or write the content (a single character) of the cell below it. It can also change the color of the cell. The examples are:

Palindrome detector: single-symbot version
Palindrome detector: two-symbot version
Simulated human addition of multi-digit numbers
Knight's Tour solutions

The first two examples employ a single row of the two-dimensional grid of cells. In both cases the purpose of the symbot-control algorithm is to detect whether a sequence of characters is a palindrome (the same sequence reading from left-to-right or right-to-left). The third example employs four rows of the grid: a carry row, an augend row, an addend row, and a result row. The Knight's Tour example employs a recursive algorithm used to search for solutions to this classic problem. In all four examples the solution algorithms are based entirely on the symbot's reading and writing of characters. Colors are used only to help the user visualize and follow the evolving solution.

II. PROGRAM ARCHITECTURE

The first three examples share an identical architecture, including several reused components. All four examples employ three subsystems: GUI, Model, and Display. GUI depends upon Model, and Model depends upon Display. GUI contains a main procedure and a package, generated by the "WYSIWYG" GUI Builder (supplied by Aonix as part of its ObjectAda for Windows™ environment). Display consists of a single package, much of which was also automatically generated by the GUI Builder then "cut and pasted" into a different file. Model contains four packages, one of which includes a procedure that contains the symbot-control algorithm. This package is the only Model component that varies among the first three examples. The Knight's Tour Model is slightly different from the others because of the recursive nature of the solution. Its design could be reused for other recursive algorithms such as an Eight Queens problem solution.

III. SITUATION-REACTION TABLES

An important element of all examples presented is the use of the Situation Driven Modeling Language (SDML) for specification and design introduced by G.W. Cherry. [1] An artifact of this language is a tabular form known as a "Situation-Reaction Table." Situation-Reaction Tables are used to define the semantic specifications of needed objects or classes of objects, in a complete, consistent and compact manner. Each row of the table corresponds to a numbered rule, and each rule is then made to correspond to an identified slice of code in the program's final implementation. The table itself becomes very useful in the process of validating system specifications, and the correspondence between rules and code slices provides a firm basis for code verification. (See example in Table 1.)

IV. CONCLUDING COMMENTS

Situation-Reaction Tables, as illustrated in Table 1, played a key role in creating and communicating complete and correct algorithm specifications and in verifying the Ada-coded implementations. The use of a modern "WYSIWYG" GUI builder, such as that supplied with Aonix's ObjectAda for Windows™ system saves much time and effort in this kind of interactive system development. The fact that Ada tools and environments now facilitate such small, educational, and entertaining projects is a very encouraging sign.

ACKNOWLEDGEMENTS

George W. Cherry created the concept of symbots and did the initial specification work on all examples presented. The work was sponsored by the U. S. Army/CECOM under an Innovative Research and Development Contract.

REFERENCES

[1] Cherry, G. W. and Crawford, B. S., The Situation in Object Oriented Specification and Design, Proceedings of Tri-Ada'96 Conference (November 1996) ACM Press.

#	Situation	Reaction
0	Stimulus = Initialize	Color all cells white (one row only) Enter test sequence of characters Place symbot at first (far left) cell Color cell yellow, Read character Set Goal to "Find first non-blank character"
1	Goal = "Find first non-blank character"	
1a	Character = blank	Move symbot right Color cell yellow, Read character
1b	Character not = blank	Set Remembered Value to Character Write blank in cell, Color cell blue Set Goal to "Find right-most character" Move symbot right, Color cell yellow, Read character
2	Goal = "Find right-most character"	
2a	Character not = blank	Move symbot right Color cell yellow, Read character
2b	Character = blank	Move symbot left, Color cell blue, Read character Write blank in cell Set Goal to "Compare with Remembered Value"
3	Goal = "Compare with Remembered Value"	
3a	Character = Remembered Value and Character not = blank	Set Goal to "Find left-most character" Move symbot left, Color cell green, Read character
3b	Character = blank	Put message "This is a palindrome" Exit this run
3c	Character not = Remembered Value and Character not = blank	Put message "Sorry, this is not a palindrome" Exit this run
4	Goal = "Find left-most character"	
4a	Character not = blank	Move symbot left Color cell green, Read character
4b	Character = blank	Move symbot right, Read character Set Remembered Value to Character Write blank in cell, Color cell blue Set Goal to "Find right-most character" Move symbot right, Color cell yellow, Read character

Table 1 Situation-Reaction Table for Single-Symbot Palindrome Detector