Electronic Notes in Theoretical Computer Science 65 No. 5 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume65.html LastPage pages

Design of a Mobile Robot Controller using Esterel Tools

Arcot Sowmya [a,1,2] David Tsz-Wang So [a] Wan Hung Tang [a]

[a] *School of Computer Science and Engineering*
University of New South Wales
Sydney, Australia

Abstract

Robot controllers are often programmed using either standard sequential programming languages or a robot-specific language, which are then compiled to assembly language specific to the robot. Modern real-time programming languages, on the other hand, are more appropriate to program robots, as they better fit the real-time reactive model of robots. This paper reports on a project to program a non-trivial robot, the Rug Warrior, in the Artificial Intelligence Laboratory of UNSW, using Esterel, which is a real-time programming language. The approach is illustrated by *simulation of a colony of Siberian ants* using a group of Rug Warriors.

1 Introduction

Without software, a robot is a collection of mechanisms, sensors and actuators with no common purpose. Computer control programs tie these components together into an integrated system with the flexibility to perform a variety of tasks by rapid re-programming [10]. Our interest is in using a real-time programming language to write control programs for a robot. Such languages have the ability to control systems that acquire and emit data, or interact with their environment at precise times [7].

Traditionally, control software for robots is built in functional modules that are then combined into a single program [6]. Brooks proposed a *subsumption architecture* to control robots that is based on task-achieving behaviours rather than functions [2]. Other approaches to robot programming include discrete word recognition, teach-and-playback and high-level programming languages [3]. Of these, robot programming languages are used both to define the task that the robot is to

[1] The authors thank Professor S. Ramesh, IIT, Bombay, who suggested starting the translation from OC, and helped with understanding OC. They also thank Philip Byrnes-Preston, AI Lab, School of Computer Science and Engineering, UNSW, for his constant help in working with the Rug Warriors.
[2] Email: sowmya@cse.unsw.edu.au

perform, and to control the robot as it performs the task. General-purpose languages such as Fortran, Pascal and variants of C have been used to program robots; the problem with this approach is that it requires extensive recompilation and linking every time that a program is modified.

Previously we have reported on formal techniques for modelling mobile robots as real-time reactive systems, and for specifying and verifying their safety and liveness properties prior to actual design [12]. Richard and Mauras [9] reported on their experiments to control a Lego Mindstorms robot by writing Esterel programs. The ORCCAD environment provides tools for specifying, verifying and simulating hybrid systems [5]. In this paper, we discuss the development of control programs for a Rug Warrior robot using a real-time programming language called Esterel.

We first define the characteristics of a real-time programming language in section 2, and present the relevant features of Esterel. We then briefly describe the Rug Warrior robot in section 3. In section 4, we describe the design of a robot controller for the Rug Warrior, and its implementation. The testing and results for the Siberian ant simulation problem using Rug Warriors is then discussed in section 5.

2 Real-time Programming and Esterel

Real-time programming languages are used to program systems that reside in a real-time environment. Real-time systems are also often *reactive*, ie they maintain a permanent interaction with their environment and react to inputs from the environment in an open-ended way [4]. Esterel is a real-time programming language in which statements handle either classical assignable variables that are local to concurrent statements and cannot be shared, or *signals* that are used to communicate with the environment and between concurrent processes. More details are available from [1].

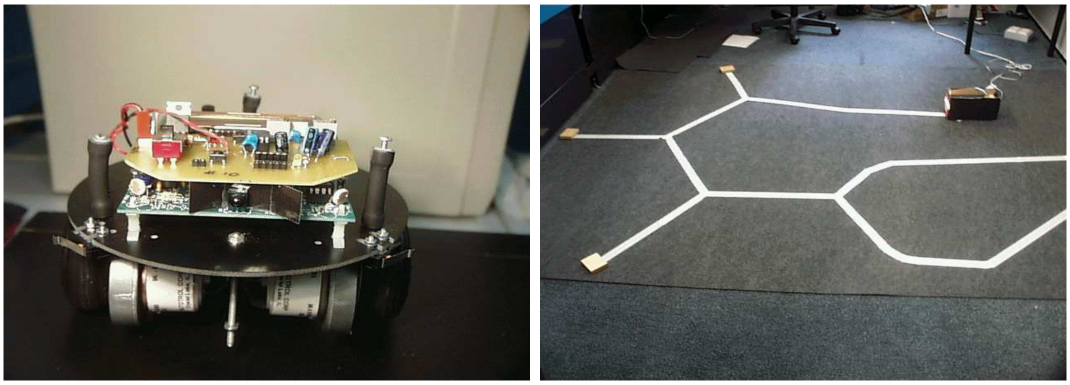
The Esterel compiler can generate executable code in C and Java, or produce intermediate code in five different formats. Of these, the OC file contains intermediate automaton code which documents the program structure; in fact, it is designed to be used by post-processors translating an automaton into a sequential programming language. A drawback of OC, though, is the state explosion for large programs. Within the Esterel environment, the executable C code is generated from the OC code, and OC code therefore is a good candidate for translating into Interactive C, the language used to program the Rug Warrior.

The objects referred to by the automaton fall into several classes, eg. types, variables etc, each associated with a table. Within each class, there may be several pre-defined objects, eg. the type $integer$. There are 13 tables that appear in a fixed order, and contain instances, types, constants, functions, procedures, signals, implications, exclusions, variables, tasks, execs, actions and halts. The automaton is fully described by the transitions of the various states, and each transition consists of a series of actions. A state table lists all possible states for the automaton.

3 The Rug Warrior

The Rug Warrior Pro (Figure 2(a)) is a microprocessor controlled mobile robot based on the MC68HC11A0 chip. It comes with 256 bytes of memory, 8 A-to-D converters and a timer-counter system. It also has 32K of external memory. The sensors on the Rug Warrior include a bump sensor that resolves collisions into one of six possible directions, dual light sensors that measure the light intensity in the robot's environment, two infrared emitters and one infrared detector, a microphone and amplifier to monitor sounds and shaft encoders to measure how far each wheel has turned. The actuators include a piezo buzzer that can produce tones in a wide range of frequencies, and a motor driver. The Rug Warrior is programmed in a high

Fig. 1. Rug Warrior Experiments



(a) Rug Warrior Pro (b) Binary Tree on Carpet

level language called Interactive C (IC), developed by Sargent and Martin of the MIT Media Laboratory. IC is a C dialect consisting of an interactive command line compiler and debugger, and a run-time language module. IC implements a subset of C including control structures (for, while, if, else), local and global variables, arrays, pointers etc [8].

4 Controller Design and Implementation

Out goal was to program the Rug Warrior to perform higher-level tasks, using Esterel at the top level. This is sought to be achieved by writing and simulating Esterel control programs within the Esterel environment, translating the program into the intermediate OC code, and finally mapping the OC code to IC programs that can execute directly on the Rug Warrior. The user first writes an Esterel program using pre-defined input and output signals that correspond to different sensors and actuators on the robot. The program may then be compiled and simulated using the xes tools within the Esterel environment. Once the simulation indicates that the program behaves correctly, the Object Code (OC) is produced, which is then translated to IC code, which can be down loaded onto the Rug Warrior for direct execution.

Table 1
Mapping of Sensors to Esterel Input and Output Signals

Signal	Input/Output	Sensor
BUMPER_L	input	Left Bump Switch
BUMPER_R	input	Right Bump Switch
BUMPER_B	input	Back Bump Switch
LIGHT_L	input	Left Photo Cell
LIGHT_R	input	Right Photo Cell
MICROPHONE	input	Microphone
LIGHT_SENSOR_L	input	Left Ground Photo Cell
LIGHT_SENSOR_M	input	Middle Ground Photo Cell
LIGHT_SENSOR_R	input	Right Ground Photo Cell
MOTOR_SHAFT_L	input	Left Shaft Encoder
MOTOR_SHAFT_R	input	Right Shaft Encoder
IR_L	input	IR Detector (for left emitter)
IR_R	input	IR Detector (for right emitter)
TONE-—IN	input	Digital Tone Detector
MOTOR_L	output	Left Motor
MOTOR_R	output	Right Motor
TONE_OUT	output	Digital Tone Emitter

The system consists of three components: *Simulation*, *Translation* and *Execution* modules. The *Simulation* module uses the Esterel compiler, simulator and other tools within the Esterel environment. Our task is to map signals in Esterel to sensors and actuators on the robot. There are two classes of signals in Esterel, namely the visible signals that are used to control the interface between the module and its environment, and the invisible signals that are used for internal communication. The visible signals could be input, output, inputoutput or return signals within Esterel. In our application, we use only input and output signals. Each sensor on the robot is mapped to an input signal and each actuator to an output signal. The *Translation* module separates each table in the OC code and translates data objects them into IC code based on the object structure.

To execute the translated program, the *Execution* module uses two approaches. The Rug warrior contains 32K bytes of memory, of which the pcode interpreter interprets user code occupies about 16K. Therefore only 16K memory is available for compiled IC programs in the form of pcode- this corresponds to about 50K of original IC code from our experiments. Upto this limit, the IC code generated from translation from OC can be down loaded and executed directly on the robot. For larger programs, we down load a command interpreter onto the Rug Warrior, and compile the translated IC code together with a serial communication driver which can send serial commands to communicate with the command interpreter on the robot.

We briefly touch upon some implementation issues. The set of input/output signals that map the sensors and actuators of the robot are shown in Table 1, and insist that users program the Rug Warrior using this pre-defined set. If the Esterel program contains any other signals, the compiled OC file will not be translated successfully into an IC program. The Execution Module deals with execution of a generated IC program directly on a Rug Warrior, as well as executing a larger C program on a host computer. For the first, signal functions in the generated IC program must be mapped to library functions. The reason for this is that some robot sensors can be accessed by a single function, whereas all signals in an Esterel program are independent of each other. The implementation of functions for each defined signal in Esterel must be down loaded into the robot prior to down loading an IC program. For the second part of the Execution module, a protocol for communication between a host computer and a Rug Warrior was implemented. A mapping of functions to communicable commands is defined. The C program is executed on the host, and function calls converted to commands are sent down a serial line to the Rug Warrior. A command interpreter sitting on the Rug Warrior receives a command, interprets it and returns the result.

5 Simulation of Siberian Ant Colony

The test problem chosen is simulation of a colony of Siberian ants, which is described by Michie [11]. A study of these ants provided convincing evidence that they are able to communicate the path to a food source. A scout ant which locates food, subsequently returns to mobilise foraging teams of ants and engages in contact sessions. At end of these sessions, the foragers proceed to the exact spot where the food was earlier found by the scout.

We simulate the ant behaviour using two robots, a scout and a forager. The scout robot will leave from an arbitrary start position in a simulated tree and locate the food source. The tree is made up of white paper tape, and laid out on a black carpet; the tree is binary, all branches are separated by 120 degrees (see Figure 2(b)). A robot is able to detect a branch using its line following capability. The end of a branch is demarcated by a small wooden block that will make a solid bump, and therefore detectable using the bump sensors. The food source is a bright light which will also provide a solid bump; it is arranged at a convenient height for the light sensors on the robot and so as not to flood the field with light. After detecting the food, the scout will return to the starting point, and communicate the location of the food to the forager. In order to simulate the scout and forager, we must simulate the following behaviours:
(i) follow a line on the carpet.
(ii) search the tree for the light source and record the path from starting point to the 'food'.
(iii) travel in the tree given a predefined path
(iv) communicate the path from the scout to the forager.
We implemented these behaviours in three different ways: by programming directly in IC, by programming in Esterel and simulating within the Esterel environment, and by translating the Esterel programs into IC using our tools and then testing the robots.

Line Following: This requires the robot to maintain its moving direction on the line, and make adjustments whenever it detects that it is not on the line. Table 2 provides the logic for the direct IC solution and the independent Esterel simulation. The three inputs in Table 2 correspond to the ground light detecting sensors, the meaning refers to the position of the robot relative to the line and the behaviour corresponds to the robot action performed. In the Esterel simulation, the values

Table 2
Line Following, IC Program and Esterel Simulation

Input	Meaning	Behaviour	Output
Left Sensor = Low Middle Sensor = Low Right Sensor = Low	on top of junction	turn right	MTR_L=110 MTR_R=-30
Left Sensor = Low Middle Sensor = Low Right Sensor = High	right sensor not on top of line	turn left	MTR_L=-30 MTR_R=-110
Left Sensor = Low Middle Sensor = High =-30t Sensor = Low	on top of junction	turn right	MTR_L=110 MTR_R=-30
Left Sensor = Low Middle Sensor = High Right Sensor = High	only left sensor on top of line	turn left	MTR_L=110 MTR_R=-30
Left Sensor = High Middle Sensor = Low Right Sensor = Low	left sensor not on top of line	turn right	MTR_L=110 MTR_R=-30
Left Sensor = High Middle Sensor = Low Right Sensor = High	only middle sensor on top of line	move forward	MTR_L=110 MTR_R=-30
Left Sensor = High Middle Sensor = High Right Sensor = Low	only right sensor on top of line	turn right	MTR_L=110 MTR_R=-30
Left Sensor = High Middle Sensor = High Right Sensor = High	no sensor on top of line	stop	MTR_L=110 MTR_R=-30

for the light sensor signals obtained by experimentation and observation are a low value of 40 and a high value of 100. The output signal values are obtained from the simulation based on input signal values.

The translated program was down loaded into Rug Warrior and executed, and its behaviour followed the simulated Esterel behaviour quite closely. It was able to follow different types of lines, including straight, curved, angled lines and line junctions. The failures occurred when the testing field was filled with light, which affected the sensors; this is a problem with robot sensor calibration rather than the translation.

Tree Searching: Here, the robot must maintain its position on the tree and detect an end point through bumper contact, or a junction through ground light detecting sensors. At that time, it should be able to compute its current position and traverse a branch not previously visited. We use inorder tree searching where the search order is (left subtree, node, right subtree). The test tree, using inorder traversal, is:
junction1, junction2, endpoint1, junction2, junction3, endpoint2, junction3, endpoint3, junction3, junction2, junction1, junction4, junction5, food source
where *junction* is a node in the tree and *endpoint* is a leaf node. Table 3 shows the results for testing tree searching behaviour by programming in IC and Esterel. The left, middle and right sensors correspond to the ground light detecting sensors, the left and right eye sensors to the left and right light detecting sensors, and the left and right bumpers to the left and right bump switches. In the Esterel simulation, an input value of 40 for all ground light sensor signals corresponds to the robot being at a junction in the tree; an input value of ON for the bumper sensor signals corresponds to the robot being at an endpoint, and a value of 10 for the eye light

Table 3
Tree Searching, IC Program and Esterel Simulation

Order	Input	Meaning	Behaviour	Output
1	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction1	turn right	MOTOR_L = 110 MOTOR_R = -30
2	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction2	turn right	MOTO R_L = 110 MOTO R_R = -30
3	Left Bumper = Press	at endpoint1	turn around	MOTO R_L = 70 MOTO R_R = -70
4	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction2	turn right	MOTO R_L = 110 MOTO R_R = -30
5	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction3	turn right	MOTO R_L = 110 MOTO R_R = -30
6	Right Bumper = Press	at endpoint2	turn around	MOTO R_L = 70 MOTO R_R = -70
7	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction3	turn right	MOTO R_L = 110 MOTO R_R = -30
8	Left Bumper = Press	at endpoint3	turn around	MOTO R_L = 70 MOTO R_R = -70
9	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction3	turn right	MOTO R_L = 110 MOTO R_L = -30
10	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction2	turn right	MOTO R_L = 110 MOTO R_R = -30
11	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction1	turn right	MOTO R_L = 110 MOTO R_R = -30
12	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction4	turn right	MOTO R_L = 110 MOTO R_R = -30
13	Left Sensor = Low Middle Sensor = Low Right Sensor = Low	at junction5	turn right	MOTO R_L = 110 MOTO R_R = -30
14	Left Eye Sensor = Low Right Eye Sensor = Low	at food source	stop	MOTO R_L = 0 MOTO R_R = 0

sensor signals corresponds to the robot being at a food source.

When the translated IC file is down loaded into Rug Warrior and executed, its behaviour follows the simulated result. The robot is able to locate the light source when the light is moved to a different position. When the light source is taken away from the tree and wooden blocks are placed at the endpoints. the robot is able to search through the tree and return to its starting point.

Communication: Here, the scout and forager robots must send and receive a path on the tree using tone emitter and detector respectively. Communication relies heavily on time measurements and not so much on environmental interactions, as the solution we adopted did not involve other sensors and actuators. We wrote a function called *measure()* in IC to measure the digital value from the tone detector in a given time, and used it to differentiate between left and right turns in a path list. Within Esterel, the communication process will require an external function to measure the digital value of the tone detector, and therefore is only as good as that external function. When the translated program was tested on the robot, we also used the same *measure()* function, and obtained similar results.

Path Traversal: Both scout and forager robots need to traverse the tree. The robot needs to know the direction of travel at a junction, and traversal is a simple extension of line following and simpler than tree search. The results here are similar to those obtained for line following and tree searching.

The combined behaviours in the translated version is over 200K bytes and could not be down loaded wholly onto the Rug Warrior. Here recourse to serial communication between a host executing the generated C program, and the Rug Warrior executing a command interpreter, is needed and we are currently working on it.

References

[1] G. Berry. *The Esterel v5 Primer Version 5.21 release 2.0*. INRIA, Sophia-Antipolis, 1999.

[2] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, 1986.

[3] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics Control, Sensing, Vision and Intelligence*. McGraw Hill, New York, 1987.

[4] D. Harel and A. Pnueli. On the development of reactive systems: logic and models of concurrent systems. In *Proc. NATO Advanced Study Institute on Logics and Models for Verification and Specification of Concurrent Systems, NATO ASI Series F*, volume 13, pages 477–498. Springer Verlag, Berlin, 1985.

[5] Borrelly J-J., Coste-Maniere E., Espiau B., Kapellos K.and Pissard-Gibollet R., Simon D., and Turro N. The orccad architecture. *International Journal of Robotics Research*, 17(4):338–359, 1998.

[6] J. L. Jones and A. M. Flynn. *Mobile Robots: Inspiration to Implementation*. A. K. Peters Ltd, Wellesley, Mass., 1993.

[7] P. D. Lawrence and K. Mauch. *Real-Time Microcomputer Sustem Design: an introduction*. McGraw-Hill, New York, 1987.

[8] F. Martin. *Interactive C Unser's Guide*. MIT Media Laboratory, Newton Research Labs, 1997.

[9] C. Mauras and M. Richard. How to use synchronous languages to play with the lego mindstorms? *http://www.emn.fr/richard/lego/*, March 2000.

[10] P. J. McKerrow. *Introduction to Robotics*. Addison Wesley, Sydney, 1991.

[11] D. Michie. The numerate ants of siberia, seminar notes, university of edinburgh. *http://www.cs.york.ac.uk/seminars/Past/98Summer/1michie8sept.html*, July 1998.

[12] A. Sowmya and S. Ramesh. Extending statecharts with temporal logic. *IEEE Transactions on Software Engineering*, 24(3):216–231, 1998.