A Rule-Based Visual Language for Small Mobile Robots

Joseph J. Pfeiffer, Jr.
Department of Computer Science
New Mexico State University
Las Cruces, New Mexico, USA
pfeiffer@cs.nmsu.edu
http://www.cs.nmsu.edu/~pfeiffer

Abstract

Altaira is a visual language intended for the control of small mobile robots. It is a rule-based language, combining information from sensors, navigation, and current state to determine actions to be taken by the robot and changes required in the state. The language, implemented sring the Spring of 1997, is proving to be a powerful flexible language for robot control, and has also been useful for providing an introduction to programming for children.

1. Introduction

Robot applications may be broadly divided into two classes: industrial robots, in which a stationary robot functions as an intelligent tool in the tightly constrained environment of the assembly line, and autonomous mobile robots, in which a mobile robot maneuvers in much less constrained environments such as offices, hallways, or even the surface of another planet. Using a rule-based reactive control strategy to combine sensor inputs with state information to produce motor control outputs and state changes has become a very common approach to controlling mbile robots.

A visual representation is especially well-suited for this domain, for reasons relating both to the domain and to the language model adopted. First, there is a natural match between the objects required in the domain and visual representations. This is equally true for the robot's inputs and its outputs: visual sensor inputs have a very natural representation as light or dark circles, visual representations of direction are extremely common, and the local environment of the robot is naturally represented as an icon; while motor controls and direction updates also have clear representations. Second, rule-based languages have a great deal of independence between rules, which is easily mapped to showing a single rule in a window.

2. Background

Altaira may be regarded as the intersection of two research areas: reactive robot control and rule-based visual languages.

Robots, and especially mobile robots, are one of the traditional testbeds for problem solving, planning, and other areas of artifical intelligence. The founding paper on using a layered reactive control architecture for mobile robots is [1]. An approach to implementing a reactive strategy for planetary explorers is in [2]. An excellent introduction to developing reactive control strategies for mobile robots (specifically in the context of robots constructed from LEGO building bricks) may be found in [4, Chapter 5]).

A number of rule-based visual languages have been developed over the past several years, including BITPICT [3] and Cocoa (formerly known as KidSim) [5].

3. The Altaira Language

Figure 1 shows a screen dump of Altaira's execution engine during the interpretation of a ruleset. In this section, we will describe the rule inputs and outputs in the figure.

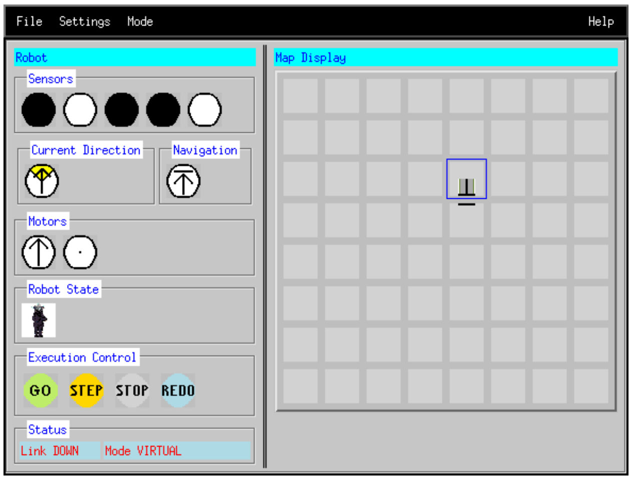

Figure 1: Altaira Execution Engine

3.1. Inputs

The four inputs to a rule consist of:
Sensors At present, Altaira only supports simple binary (*eg* light-dark or contact-no contact sensors). In the configuration shown, there are five sensors (shown as five circles in the upper left corner). These include three central sensors used to follow a line, and two outboard sensors used to detect markings that encode tile types. In the example, the robot has veered slightly to the left (so the center sensor and the right side inboard sensor are both dark), and a tile type encoding has been detected by the left outboard sensor.
Direction A dead-reckoning navigation system maintains the robot's current direction (on the left, directly below the sensors). This is partitioned into four possible direction indexes for rule lookup. Here, the robot is heading straight ahead.
Tile State The robot's field of operation is represented as a map divided into square tiles, in order to maintain local information. The state of the current local tile (highlighted in the map, occupying the right side of the figure) is used as a third input for rule lookup. Up until the current execution cycle, we have not been able to determine what type of tile we are on, so the current tile state reflects only that we are on a tile and have left the tile boundary (rules which fired on previous execution cycles left the tile in this state).
Robot State A second state input, used to maintain subgoals and global information, is the robot state. The "Robby" state is used to denote the default robot state.

Rules require each of the four inputs to be completely specified (there are no don't-care conditions), in order to simplify the execution engine. The ruleset is represented as a four-dimensional array, and by translating each of the four inputs into an index, rule lookup is reduced to an array access.

3.2. Outputs

The rule also has four outputs, which include:
Navigation Commands Commands to the dead-reckoning navigator are also provided by the rule. Commands may instruct the navigator that the robot is proceeding straight ahead, veering to the left or right, or changing to a new tile, for instance. There are a total of ten navigator commands. The navigation commands may be regarded as the intention behind the motor commands. Once we encounter tile markings, we can know that the robot is indeed heading straight forward. The navigation command (directly to the right of the current direction) reflects this.
Motor Controls The drive motors may independently be set to go forward, stop or reverse. In order to stay on course, the robot will have to turn to the right. Consequently, the right motor is turned off, while the left motor is set to go forward.
Tile State The state of the current tile may be changed as a result of information made available in a rule (consequently, the action of Altaira on a single tile is very similar to a finite state machine. On the next execution cycle, the tile state will have been changed.
Robot State Similar, the robot's state may be changed as well. The robot state should be regarded as similar to the state of a Turing Machine. In this situation there is no reason to change the robot state, so it will still be in the Robbie state on the next cycle.

The difference between the tile state and the robot state is that the tile state is tied to a particular tile (and there is no way to directly communicate the state of a tile to a neighboring tile), while the robot state is global, affecting rule lookup regardless of the robot's position on the map.

4. Conclusions and Acknowledgements

We have been able to use Altaira to implement a number of rulesets, performing line following, exploration, and Turing Machine simulation. It is proving to be a flexible, easy to use tool for these tasks. Children as young as nine years old have also been able to implement line-following algorithms with Altaira.

Altaira was implemented by a team of graduate and undergraduate students at NMSU. Hardware support was provided by VPC'97.

5. References

1. Brooks, R. A. A robust layered control system for a mobile robot, in *IEEE Journal of Robotics and Automation* RA-2, pp 14-23, 1986.
2. Gat, E., R. Desai, R. Ivleve, J. Loch, and D. P. Miller. Behavior control for robotic exploration of planetary surfaces, in *IEEE Transactions on Robotics and Automation*, 10(4), pp 490-503, 1994.
3. Furnas, G. Formal models for imaginal deduction, in *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pp 662-669.
4. Martin, F. *The Art of Robotics: An Introduction to Engineering.* Addison Wesley Longman, forthcoming.
5. Smith, D. C., A. Cypher and J. Spohrer, KidSim: programming agents without a programming language, in *Communications of the ACM*, 37(7), pp. 54-67, 1994.