CULET: Cubelet Lego Turing Machine

Ricardo Q. Figueroa  
Artificial Life Robotics Lab, Escuela Superior de Cómputo, Instituto Politécnico Nacional, México

Daniel A. Zamorano  
Artificial Life Robotics Lab, Escuela Superior de Cómputo, Instituto Politécnico Nacional, México

Genaro J. Martínez  
Artificial Life Robotics Lab, Escuela Superior de Cómputo, Instituto Politécnico Nacional, México. Unconventional Computing Laboratory, University of the West of England, Bristol, United Kingdom

Andrew Adamatzky  
Unconventional Computing Laboratory, University of the West of England, Bristol, United Kingdom

E-mail: qf7.ricardo@gmail.com, daz23ayala@gmail.com, genaro.martinez@uwe.ac.uk, andrew.adamatzky@uwe.ac.uk

Abstract

We describe a design and implementation of CULET - a modular and distributed robot that physically simulates functioning of a 2-symbols Turing machine. CULET is constructed with Cubelets (small autonomous robot-cubes used for teaching basic robotics and programming for kids) and Lego bricks. Cubelets allow for a high level programming: instead of the traditional code, robots are programmed by assembling various elementary blocks of robots together. We illustrated how reading and writing operations are physically implemented by CULET.

Keywords: Turing machine, Robotics, Cubelets, Lego, Cellular Automata.

1. Introduction

In the history of Turing machines a large number of physical mechanical devices have been developed. For example, in 2012, to celebrate Turing’s 100th birthday, LEGO built an autonomous mechanical Turing machine adding binary numbers\(^1\). Inspired by the idea of machines emerged from the theory of self-reproducing cellular automata\(^2\), decided to make a robot based in small robots-cube Cubelets and some pieces Lego. We called this robot CULET. A key feature of CULET is the first ever implementation of the machine constructed with robots and bricks. This robotic machine is capable for simulating the function of other machine, machines simulating machines. Cubelets, enhanced with pieces of Lego bricks, demonstrate how a robotic machine simulates an abstract computing machine. A distributed controller of the robot is in the cube which can read and other two cubes that move the head and the arm (which implements write operation). These cubes need to be reprogrammed for each table of transitions.

For the implementations we programmed two special Turing machines. The first one is a universal Turing machine able to simulate the behaviour of a complex elementary universal cellular automaton known as Rule 110\(^3\). The second Turing machine programmed in
2. Preliminaries

2.1 The Cubelets

The Cubelets are small cube-shaped robots designed to teach fundamentals of robotics and programming to kids. Cubes can have different functionality. An idea is to construct robots only through the concatenation of different cubes. This way Cubelets can be seen as a formal language, where every valid construction is a valid expression in the language, with virtually unlimited number of possible combinations. This can be interpreted as a high level of programming: the child does not modify the original program of each cube, but instead, use this program together with the interaction among cubes for produce a new robot. The different cubes are classified in three categories:

- Input sensors: cubes used to obtain data from the environment. The most common are the proximity and light sensors.
- Actuators: cubes used to execute movements, e.g. drives or rotors.
- Thinkers: cubes to implement a computation, e.g. inverters, maximum, minimum, etc.

A single cube communicates with his neighbours through a shared channel that is established in the concatenation of blocks. The input sensors-cubes distribute their readings thought the channels; the thinkers-cubes obtain these readings, process the information and make a decision, and send the commands to the actuators.

Also, each cube can be reprogrammed to re-assign its functionality. This reprogramming is done in C or Scratch-like code. CULET, which is described in the next section, follows this approach. The API for programming the cubes is described in Ref 5.

2.2 Universal Turing machines

The Turing machines were proposed by Turing in 1936 to solve problems about completeness and decidability. A Turing machine is a finite-state machine with an external memory medium, an infinite tape divided into squares. The machine moves on the tape and its head reads the value in each square sequentially. Formally, the Turing machine is a set of quintuples of the form

\[(q_0, s_0, q_{i+1}, s_{i+1}, d)\]  

Where \(q\) is the set of states, \(s\) is the set of symbols and \(d\) indicates the direction of movement of the head in the tape (right or left); \(i\) is the time step. So, the first two elements of the quintuple indicate the current state and symbol, the next two elements indicate the following state and symbol, and the last one indicates the direction of movement.

One of the greatest achievements of Turing’s work is the conceptualization of the so called universal machines. A universal machine \(U\) can do everything that any machine can do. \(U\) receives as input two things: a description of a machine \(A\) and the input for that machine \(M\). The machine \(U\) writes on the tape the same string of symbols that the machine \(A\) would write in its response to the input \(M\).

2.3 Elementary cellular automata

Cellular automata were proposed by von Neumann in Ref 2 as means to study decentralized systems (which later led to designs of distributed computing, ironically called non-von-Neumann-style computers). The cellular automata consist of an array of cells, each of which is a finite-state automaton (the same rules for all the cells) and where the state of each cell in the next generation depends on its current neighbourhood. Von Neumann developed a very complex automaton with 29 states evolving in two dimensions.

Another important event in cellular automata history is the popularization of Conway’s Game of Life in the 70’s. This is a cellular automaton with only two states and a neighbourhood of eight cells (Moore neighbourhood). The automaton is popular because it shows how a complex and unpredictable behaviour emerges from simple rules. Conway demonstrated the universality of this automata by constructing a register machine in Ref 10.

In the 1980’s years, Wolfram described the elementary cellular automata to understand and formalise the
dynamics of cellular automata\textsuperscript{11}. The space is one-
dimensional (a row of cells) and a cell's neighbourhood is two closest cell-neighbours of the cell. Wolfram hypothesis was that if we cannot understand such minimal models, then we could never understand bigger ones. The interesting part comes when even in those apparently simple automata there are intriguing patterns of a behaviour.

2.4 Universal ECA: rule 110

In Ref. 12 Cook proofed that the elementary cellular automaton Rule 110 is universal. The proof is based on showing how, with particle interactions\textsuperscript{13} typical of the rule, one can emulate the operation of a cyclic tag system. This one is the smallest cellular automaton demonstrated to be computationally universal by simulating a very sophisticated sequential machine via particles collisions in one dimension. Such a new machine was invented to simulate a computable function known as cyclic tag system in this automaton, for details see Ref. 12 and 14. Rule 110 can simulate a cyclic tag system\textsuperscript{12}, cyclic tag systems can simulate tag systems\textsuperscript{12,14}, tag systems can simulate Turing machines\textsuperscript{7}, and a Turing machine (developed by Eppstein in Ref. 12) can simulate Rule 110 and therefore such a Turing machine is universal. Finally, CULET can simulate the universal Turing machine which simulates the behaviour of Rule 110, complex behaviour, and collisions of particles\textsuperscript{15,16}.

Table 1. 7x2 Rule 110 Turing Machine.

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_{00}</td>
<td>(0, T_{00}, R)</td>
<td>(1, T_{01}, R)</td>
</tr>
<tr>
<td>S_{01}</td>
<td>(1, T_{01}, R)</td>
<td>(1, T_{11}, R)</td>
</tr>
<tr>
<td>S_{11}</td>
<td>(1, T_{11}, R)</td>
<td>(0, T_{10}, R)</td>
</tr>
<tr>
<td>S_L</td>
<td>(0, T_{0L}, L)</td>
<td>(1, T_{0L}, L)</td>
</tr>
<tr>
<td>T_{00}</td>
<td>(1, S_{00}, R)</td>
<td>(0, S_{0}, L)</td>
</tr>
<tr>
<td>T_{01}</td>
<td>(1, S_{01}, R)</td>
<td>X</td>
</tr>
<tr>
<td>T_{11}</td>
<td>(1, S_{11}, R)</td>
<td>X</td>
</tr>
</tbody>
</table>

The Table 1 shows the transition table of the machine implemented. Every generation in which the head of the machine reaches a right end, a new generation of the rule 110 is computed. There are some considerations with respect to this machine: the initial condition and the representation of the cellular automata symbols. The initial condition of the machine must be, from left to right, an infinite string of zeros, the encoded version of the initial that the machine is going to emulate, and an infinite repetition of the string “10”. A “1” of the cellular automata is represented like “11” in the machine, and a “0” like “00”; the symbol “10” is used like a blank symbol.

2.5 A duplicator Turing machine

A duplicator Turing machine was used by Rendell as part of the Turing machine constructed in the Game of Life cellular automaton, for details see Ref. 17. We implement this machine in CULET. In Fig. 1, we show a 3-symbols version: it starts advancing to the right, for each one scanned, it put another one in the left until there is no more ones to the right. For the implementation in the CULET we used a 2-symbols version of this machine.

3. CULET Design

3.1 General description

In the Fig. 2.1 we show a snapshot of CULET. The tape is represented by an array of Lego blocks on rectangles of a black paper; the position of the block (front or back) indicates the symbol in the cell (0 or 1, respectively). The Cubelet car moves through the tape reading and writing in the tape; the read operation is achieved with a distance sensor that tests the position of the Lego block under scanning; the writing is done with a lever made of Lego and a rotate cube: depending on the desired symbol, the lever pushes or pulls the block of the tape. The movement across the tape is done with a couple of wheels that move in right or left direction. In the backwards of the car is attached a rail that helps to maintain the alignment of the car with respect to the tape.

3.2 Hardware

In the Fig. 2.2 we show a picture of the cubes needed by the CULET: a couple of batteries, a couple of drives, two distance sensors, one rotator, one passive cube and one

Figure 1. Duplicator machine.

Figure 2. (1) The CULET; (2) and (3) Cubelets and Lego.
light. In Fig 2.3 we show the Lego components needed (from left to right): a rail of 4 blocks width; a stabilizer car (it must go over the rail); a lever; the symbols of the cells.

3.3 Software

We can reprogram some of the cubes to functionalize their behaviours. A central cube, the one that stores the production rules of the Turing machine, is the rotor. In the state diagram of the Fig. 3 we show the stages of the program of this cube. The scan operation is achieved by reading values of the sensor that is pointing to the cell; the other sensor, the one that is pointing up, controls the movement of the lever: it decides when to stop movement of the rotor; both sensor-cubes were used as supplied, without re-programming. Other two cubes that have custom programs are the motors; they receive instructions from the rotor: if they receive a symbol 1, they move to the right; if they receive a symbol 2, they move to the left; a symbol 0 indicates the command ‘halt’.

The car starts with a cell below the reading sensor; it scans the position of the block and determines the next action: to write 1, to write 0, or to do nothing; then, it moves to the left or to the right and starts over again. The decisions of this automaton (which symbols to write and which direction to move) are determined accordingly to the automaton of the machine that is being executed. In any moment, there are two descriptions of the program working at different levels: the low level one, that determines the behaviour of the hardware components (move left, move right, scan, write); and the high level one, which is given by the rules of the Turing Machine.

4. Final Remarks

We developed a robotic Turing machine named CULET, able to simulate any 2-symbols Turing machine. CULET is constructed with autonomous cube-robots Cubelets and Lego bricks. A video showing the function of the Turing machine simulating Rule 110 with CULET is available in Ref. 18 and the duplicator Turing machine in Ref 19. The full code to reprogram Cubelets is available from Ref. 20. In the future work, we will develop machines reading more than two states.

References