Computational Thinking in the Mathematics Classroom

Jonaki B Ghosh
jonakibghosh@gmail.com
Department of Elementary Education,
Lady Shri Ram College for Women, University of Delhi, New Delhi, India

Abstract

Computational thinking is an important aspect of mathematics learning. The ability to deal with challenging problems, represent ideas in computationally meaningful ways, create abstractions for the problem at hand, break down problems into simpler ones, assess the strengths and weaknesses of a representation system, engage in multiple parths of inquiry are some important aspects of computational thinking for mathematics learning. Though the importance of computational thinking has been recognised as an important skiil to be developed in school, finding appropriate tasks which help to develop and elicit such thinking is a key pedagogical challenge. This article described two studies where participants explored investigatory tasks with the researcher acting as a facilitator and engaged in various aspects of computational thinking. In one study pre-service teachers explored the Tower of Hanoi puzzle as a part of the mathematics course in a teacher education programme and in the secnd study a grade 12 student explored the topic of Cellular Automata using multiple representations and a computer algebra syatem. Both studies provided ample evidence of active student engagement in various aspects of computational thinking and paved the way for integrating such tasks in the school curriculum.

Introduction

In recent years computational thinking has been identified as one of the key analytical abilities required for mathematics and science learning. The rapidly changing nature of scientific and mathematical disciplines and the need to prepare students for careers in these disciplines have been the primary motivation for bringing computational thinking into classroom practices. Papert [5] was the first to stress on the importance of computational thinking by referring to the affordances of computational representations for highlighting powerful ideas. Over the decades many researchers have attempted to define computational thinking. According to Wing [6], computational thinking involves solving problems, designing systems and understanding human behaviour by drawing on the concepts fundamental to computer science. However operationalising computational thinking for the k – 12 classroom remains a key pedagogical challenge.

In this paper we shall argue that computational thinking can be developed by engaging participants in meaningfully designed tasks which can be easily incorporated in the school curriculum. The ability to deal with challenging problems, represent ideas in computationally meaningful ways, create abstractions for the problem at hand, break down problems into simpler ones, and assess the strengths and weaknesses of a representation system are some important aspects of computational thinking for mathematics learning. In order to encourage this kind of thinking, specific investigatory tasks need to be integrated into the curriculum. This article describes two research studies where participants engaged in various aspects of computational thinking by working on investigatory problems. In the first study, 30 pre-service teachers explored the Tower of Hanoi puzzle through multiple representations. As they

engaged with the puzzle through pictorial, numerical, symbolic and graphical representations, they displayed multiple paths of inquiry, used co-operative problem solving, dealt with recursive and explicit relations and developed greatly in their mathematical thinking. In the second study, a grade 12 student researched one – dimensional Cellular Automata using Mathematica, a computer algebra system. The exploration included generating multiple representations of the 256 Elementary Cellular Automata (ECA) and categorising them based on their evolutionary patterns. The studies highlight the nature and characteristics of mathematical tasks which require computational thinking and illustrate that such tasks need to find their place in school mathematics curricula.

Research study 1: Explorations with the Tower of Hanoi puzzle

Thirty pre-service teachers in the first year of their teacher education programme participated in this study. They came from diverse backgrounds in terms of their mathematical ability and interest. Among them, 12 had studied mathematics in school up to grade 10 and the rest up to grade 12. In the first year of the programme they studied a course on *Core Mathematics* in which they revisited various topics of the school mathematics curriculum through investigatory activities. The ToH puzzle was selected by the researcher (who was also their teacher) to introduce the participants of the study to the idea of recursive and explicit thinking and also to explore the puzzle using multiple representations. Some of the prerequisite concepts required for the activity were arithmetic and geometric sequences, laws of exponents, basics of fractal geometry and proof by mathematical induction. The participants had explored these topics prior to the ToH activity.

The aim of the study was to enable pre-service teachers to

- Engage in explicit and recursive reasoning.
- Explore the mathematical ideas encapsulated in the ToH puzzle using multiple representations, namely, pictorial, numerical, symbolic and graphical and also assess their strengths and weaknesses.

The Tower of Hanoi Puzzle – recursive and explicit formulae

To begin the activity wooden models of the puzzle were distributed to the participants who worked in groups of three. The puzzle comprises three vertical pegs fixed on a wooden base and six wooden circular discs of reducing radii placed on one of the pegs (see Figure 1). The problem lies in shifting the tower of discs from one peg to another using the third peg as an intermediary, subject to two constraints - *only one disc can be moved at a time,* and *a bigger disc can never be placed on top of a smaller one while making the moves*. The shift has to be achieved using a minimum number of moves

Figure 1. A wooden model of the ToH puzzle

Initially the participants used a trial and error approach to solve the puzzle and experienced difficulty in keeping track of the moves. To facilitate their exploration the researcher posed the following tasks:

Task 1: Let n represent the number of discs and T(n) the minimum number of moves required for shifting n discs. Compute T(n) for n = 2,3 and 4. Observe the pattern as n increases from 2 to 3, from 3 to 4 etc. Can you express T(n) in terms of T(n - 1)?

Task 2: Write an explicit rule for T(n) in terms of n. (Hint: Find the nth term of the sequence of number of moves)

The first task was to lead the participants to the recursive rule for obtaining the minimum number of moves. To start with, most groups observed that the T(2) = 3. They labelled the pegs as A, B and C. After working with the discs manually, participants observed that 3 + 1 + 3 = 7 moves were required to shift three discs from one peg to the other. For example, if the discs had to be moved from A to B, the two smaller discs had to be moved from A to C (to release the largest disc) in three moves following which the largest disc could be shifted to B in one move. Finally the two smaller discs (on C) would require another three moves to be shifted back to B. This observation led them to generalise the pattern to larger number of discs. For example, T(4) = 7 + 1 + 7 = 2 × 7 + 1 = 15 and T(5) = 15 + 1 + 15 = 2 × 15 + 1 = 31 and so on. The researcher helped them to express the recursive relation symbolically as T(n) = 2 × T(n – 1) + 1.

To obtain the explicit rule, participants explored the sequence of minimum number of moves, that is, values of T(n), 1, 3, 7, 15, 31 and tried to find the nth term. The differences between successive terms - 2, 4, 8, 16… gave them the hint that the nth term has 'something to do with 2^n'. The observations T(2) = 3 = 2^2 – 1, T(3) = 7 = 2^3 – 1 helped them to arrive at the conjecture for the explicit rule, T(n) = 2^n – 1. To prove the conjecture mathematical induction was used. Participants were familiar with the steps of induction, as they had studied it in school. The following steps were used to prove the explicit rule:

Step 1: T(1) = 2^1 – 1 = 1.

Sep 2: Assuming T(k) = 2^k – 1, for a natural number k, to prove T(k + 1) = 2^{k+1} – 1

T(k + 1) = 2× T(k) + 1 = 2 × (2^k – 1) + 1 = 2^{k+1} - 2 + 1 = 2^{k+1} - 1.

ToH puzzle leading to counting problems and Hanoi Graphs

Apart from the solution for the minimum number of moves, the ToH puzzle, also leads to interesting counting problems. Participants were asked to explore the number of arrangements of n discs on three pegs and find a rule for it in terms of n. The exploration began with the number of arrangements of two discs on three pegs. Initially some felt that the number is 3 + 3 = 6 (three ways of placing each disc) but as they represented the arrangements pictorially, they realised that the number is 3 × 3 = 9. This provided the opportunity to reinforce the principle of multiplication by suggesting that the bigger disc can be placed in three ways and corresponding to each of these ways, the smaller disc can also be placed in three ways. This idea was generalised to conclude that the number of arrangements for n discs on three pegs 3^n. In general, for d discs and p pegs the number of arrangements is p^d.

Participants realised that pictorial representations of the arrangements of discs become cumbersome as the number of discs are increased. This motivated the researcher to introduce the concept of a *Hanoi graph* [7]. In a Hanoi graph, *vertices* represent states and two vertices are joined by an *edge* if it is possible to shift between those two states using only one move. Typically H_p^d is used to denote a Hanoi graph with p pegs and d discs. For drawing the Hanoi

graphs, the pegs were identified with numbers 0,1 and 2 while each arrangement of the discs on the pegs (referred to as a *state*) was assigned one two or three digits based on the number of discs and pegs in which the discs were positioned (starting with the position of the larger disc). Thus in H_3^2, there are nine vertices. Each vertex is assigned two digits. For example, 10 refers to the state in which the larger disc is on peg 1 and the smaller one is on peg 0. The researcher explained the drawing of H_3^1 where the vertices represent the three trivial states 0,1 and 2 respectively and all three vertices are mutually connected by edges leading to a 'triangle'. See Figure 2. Subsequently, participants were assigned the task of drawing H_3^2 and H_3^3. Some groups needed scaffolding for drawing H_3^2. Once the triangular structure of the Hanoi graph was decided upon, the corner vertices were numbered 00, 11 and 22. Participants referred to these as homogenous states and observed that each homogenous state can be connected to only two other states whereas non-homogenous states can be connected to three other states. Thus, while 00 can be joined by an edge to 01 and 02 only, 10 can be joined to 11, 12 and 20.

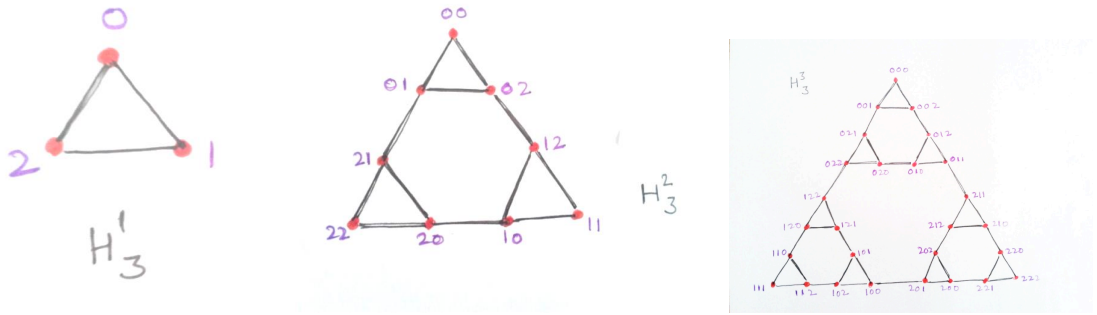

Figure 2. Hanoi graphs H_3^1 , H_3^2 and H_3^3 as represented by student teachers.

After drawing H_3^2, it took a while to develop H_3^3 but the process generated a great deal enthusiasm among the students. They identified a resemblance between H_3^2 and H_3^3 and the Sierspinski triangle. Using the notion of self-similarity they identified three copies of H_3^1 in H_3^2 and three copies of H_3^2 in H_3^3. The idea of drawing H_3^4 however seemed 'too complex'. At this point they were introduced to the Mathematica code for generating Hanoi graphs.

`GraphData[{"Hanoi",n}]` with a specified value of n (upto 5) will generate the Hanoi Graph H_3^n

`Table[GraphData[{"Hanoi",n}],{n,1,5}]` will display the graphs upto H_3^5 simultaneously. The Sierspinski - like structure of the graphs are evident from the output shown.

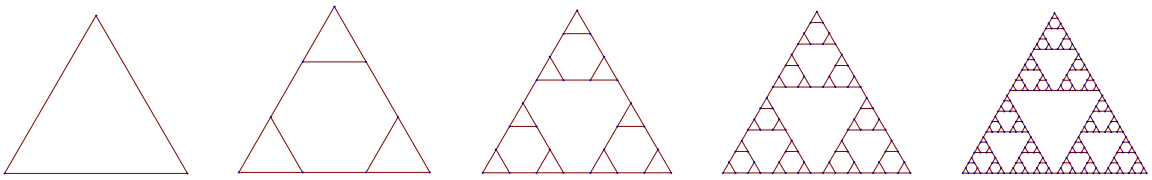

Figure 3. Mathematica generated Hanoi Graphs

Counting problems on Hanoi graphs

The Hanoi graphs themselves lead to interesting counting problems. The number of vertices of the Hanoi graph H_p^d is p^d, which represents the number of ways of arranging d discs on p pegs. The number of edges of H_p^d is less straightforward. The participants in this study devised an interesting way of counting the edges of H_3^n using the idea of self-similarity. They observed that H_3^1 has three edges. Since H_3^2 has three copies of H_3^1 and three edges interconnecting these copies, the total number of edges is $3 \times 3 + 3 = 12$. Similarly, H_3^3 has three copies of H_3^2 and three edges interconnecting these copies. This leads to a total of $12 \times 3 + 3 = 39$ edges. Finally the number of edges for H_3^n was generalised to the following recursive rule:

Number of edges in $H_3^n = 3 \times$ the number of edges in $H_3^{n-1} + 3$.

Much to the researcher's satisfaction, the participants realised the need for an explicit rule to represent the number of edges in H_p^d. The explanation for this led to a rich classroom discussion. In the general case of p pegs and d discs, let us assume that that among the p pegs, two are empty. This implies that the number of arrangements of the d discs on $(p - 2)$ pegs is $(p - 2)^d$. Thus for a fixed pair of pegs, the total number states in which the given pair is empty is $(p - 2)^d$ or $p^d - (p - 2)^d$ is the total number of states where we can move a disc between this pair of pegs. The number of ways this can happen = p_{C_2}. Hence the total number of edges in $H_p^d = \frac{1}{2} p_{C_2}[p^d - (p - 2)^d]$.

From pictorial to numerical representations

In the third session of the ToH activity participants were encouraged to make connections between the Hanoi Graphs and the minimum number of moves of the puzzle. They identified paths of varying lengths between different vertices of H_3^2 and H_3^3 and discovered that the shortest paths between two homogenous vertices match the minimum number of moves between the corresponding states. Hence the shortest path between 00 and 22 in H_3^2, consists of three edges (00 – 01 – 21 – 22) which is the same the minimum of three moves required for shifting two discs. In H_3^3, the path, 000 – 001 – 021 – 022 – 122 – 120 – 110 – 111, comprising 7 edges represents the 7 moves for shifting three discs. Following this, students took the initiative of looking for paths in H_3^4 and H_3^5 even though these hadn't been drawn. For example, 0000 – 0001 – 0021 – 0022 – 0122 – 0120 – 0110 – 0111 – 2111 – 2112 – 2102 – 2100 – 2200 – 2201 – 2221 – 2222 was identified as a path of 15 edges between the vertices 0000 and 2222 in H_3^4. By now participants had transitioned to numerical representations of the puzzle and were also able to connect these representations to their respective Hanoi graphs.

Some groups wanted to extend the problem to the 4-peg case. A few attempted to draw the Hanoi graphs H_4^1 and H_4^2. See figure 4. H_4^3 was far more challenging and was attempted by only one student. Others explored the 4 peg case numerically and tried to find T(n) for n = 2,3,4,5. T(2) was still equal to 3. However for three discs, the minimum number of moves is 5. For example one such path of 5 moves is 000 – 001 – 021 – 321 – 331 – 333. Similarly they conjectured that T(4) = 9 and T(5) = 13 . However they were unable to generalise to a an explicit rule.

Interacting with the students teachers as they engaged in the ToH investigatory activity, gave the researcher the opportunity to witness various aspects of computational thinking. In the first session, they engaged with the puzzle through the concrete model and tried to find recursive

and explicit rules for the minimum number of moves. By the second session they had moved to pictorial representations of the puzzle, that is, Hanoi graphs and had explored their self-similar structure. These graphs also led to interesting counting problems as students worked out rules for the number of vertices and edges. In the third session they explored the problem numerically and were able to think about the puzzle for a larger number of discs without having to draw Hanoi graphs. The participants demonstrated various aspects of computational thinking, which we shall elaborate upon in the conclusion of this article.

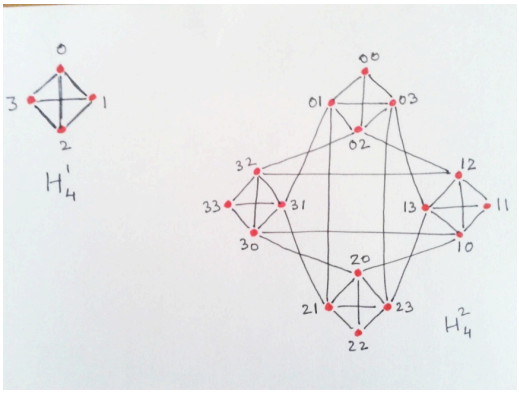

Figure 4. The Hanoi graphs H_4^1 and H_4^2.

Research Study 2: Explorations with Cellular Automata

Cellular automata (CA) is a powerful tool for modeling many kinds of systems. It is used to create mathematical models for systems in which simple components act together to produce complicated patterns of behavior. CA models are used to study physical, biological, chemical or social phenomenon involving interacting entities. The topic lends itself to interesting investigations well within the reach of high school students. Students, if given access to appropriate computer software can easily explore the ideas, which are simple and yet powerful.

Briefly defined, a *cellular automaton* is a collection of cells on a grid of a specified shape that evolves through discrete time steps according to a set of rules based on the state (or color) of the neighbouring cells. Cellular Automata may be one, two or three – dimensional. The one - dimensional *Elementary Cellular Automata* (ECA) was defined by Stephan Wolfram [8,9]. In this study we shall describe how a grade 12 student explored the ECAs using *Mathematica*, a Computer Algebra System. Mathematica's extensive numeric as well easy-to-use graphic capabilities along with inbuilt commands for cellular automata make it very conducive for exploring this topic.

The defining characteristics of a cellular automata are as follows

(i) A cellular automata develops on a grid of cells.

(ii) Each cell has a *state* – dead or alive. Live cells are coloured black or assigned a value 1 whereas dead cells are white and assigned a value of 0. Colours other than black or white may also be used.

(iii) Each cell in the grid has a *neighbourhood*. A neighbourhood of a given cell is a set of cells which are adjacent to it. This may be chosen in various ways. For example, if we consider a linear grid of square cells, then the neighbourhood of each cell may be the two adjacent cells – one to its left and the other to its right.

(iv) Finally every cellular automata must have a *defining rule* based on which it grows and evolves in discrete time steps. For example, in a square grid, each row of cells may be considered as a different generation of cells. Thus the first row is the initial

generation (or generation 0) where each cell has a state (0 or 1). The state of each cell in the second row must be a function of its neighboring cells in the row above it (that is the initial row). This may be written as (Cell state$_t$) = f (Neighboring Cell state$_{t-1}$)

The aim of the project was to

- Explore the evolution of the ECAs.
- Represent the ECA rules using multiple representations namely - decimal, binary, and Boolean forms.
- Classify all the ECAs into specific categories depending on the patterns emerging from their evolution.
- Explore the sensitivity of the ECAs to initial conditions.

ECAs – Multiple Representations

To represent an ECA pictorially we need to begin with a linear grid of square cells. In figure 5, the grid has 8 cells where every cell has state 0 except the 5th cell which has a state 1.

Figure 5 A linear grid of 8 cells where the 5th cell is a live cell.

This will be referred to as **generation 0** (or row 0). The states of cells in **generation 1**(that is row 1) will be determined by the neighborhood of each cell in row 0 which comprises of the three cells just above it. Clearly the states of these three cells may be any one of the following

000 001 010 100 011 101 110 111

The fact that there are three cells and the state of each cell is either 0 or 1 implies that there are $2^3 = 8$ ways of colouring these cells. Thus there are 8 neighbourhood configurations described by the triples of 0's and 1's as shown above. However conventionally, while defining an ECA, these neighbourhoods are taken in the following specific order.

111 110 101 100 011 010 001 000

Each of these configurations will determine the state of the middle cell of the three cells just below it in the next row, which may again be either 0 or 1. However the state of the leftmost corner cell in row 1 will be determined by the state of the cell just above it in row 0, its right neighbour and the last cell in row 0. Similarly, the state of the rightmost corner cell in row 1 will be determined by the state of the cell just above it in row 0, its left neighbour and the first cell in row 0.

We can arbitrarily assign 0's and 1's to all 8 neighbourhood configurations as follows

111	110	101	100	011	010	001	000
0	1	1	0	1	1	1	0

Pictorially this may be represented as

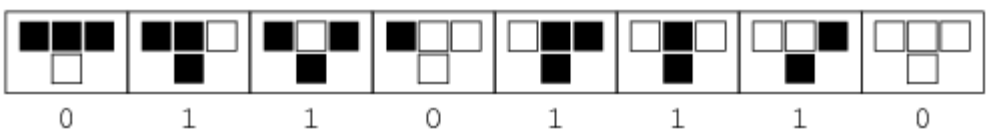

Figure 6 A rule set for a one-dimensional cellular automata

Binary and Decimal representations

This arbitrary assignment (also known as the *rule set*) will be the *defining rule* which will determine how this particular automaton will evolve. Note that this defining rule **01101110** may be treated as a binary number whose decimal representation may be obtained as follows

$0 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 110$

This kind of a rule set generates an *elementary cellular automaton*. The ECA which evolves from this rule set is referred to as **Rule 110**. However a different assignment of 0's and 1's would lead to a different rule set and a different ECA. Since each of the 8 groups of three cells may be assigned 0 or 1, this leads to $2^8 = 256$ possible assignments. Thus, in all, there are 256 ECA rules.

Boolean representations of ECA Rules

Boolean expressions for ECA rules may be obtained using the concept of truth tables and Karnaugh maps [2]. In this example three input variables can be combined in 8 different ways. Thus a truth table with three arguments p, q and r as input variables will have 8 rows. The 8 possible configurations of the truth values of p, q and r (where 1 corresponds to true and 0 corresponds to false) correspond to the 8 neighbourhood configurations of rule 110 as discussed above. Further the truth value of each row corresponds to the binary digits of 110 as follows:

Table 1: Truth table for rule 110

Row number	P	q	R	f(p,q,r)
1	1	1	1	0
2	1	1	0	1
3	1	0	1	1
4	1	0	0	0
5	0	1	1	1
6	0	1	0	1
7	0	0	1	1
8	0	0	0	0

Karnaugh maps [3,4] were used to simplify the Boolean expressions(the details of which are omitted here). The simplified Boolean expression for rule 110 was obtained as

p'r + q'r + qr'

Using the 'and', 'or' and 'not' notations of logic denoted by the symbols $\wedge$, $\vee$, and $\sim$ respectively the above expression may also be expressed as

$(\sim p \wedge r) \vee (\sim q \wedge r) \vee (q \wedge \sim r)$

(note that '+' is replaced by $\vee$, ' . ' is replaced by $\wedge$ and ' is replaced by $\sim$)

which may be further simplified using the XOR notation as

$(\sim p \wedge r) \vee (q \oplus r)$

(XOR is referred to as the *exclusive or* argument in logic and is represented by the symbol ⊕. A ⊕ B is true when *either* A *or* B is true (or has value 1)).

Mathematica exploration of the ECAs

The high point of the study was the Mathematica exploration, which was used to obtain the graphic (pictorial) representations of all the 256 ECAs. The aim was to observe the evolutionary pattern of each ECA through the first 100 iterations and to categorise them into specific classes based on the patterns manifested by them.

The inbuilt Mathematica command for exploring cellular automata is

```
CellularAutomaton[rule,init,t]
```

where **rule** stands for the decimal representation of the rule set in binary form, **init** represents the initial condition or generation 0 and **t** denotes the number of steps.

For example, the **rule 30** elementary cellular automata [1] after 10 iterations, with an initial condition comprising 21 cells, with one live cell in the centre may be computed as follows

```
CellularAutomaton[30,{{1},0},10]//TableForm
```

The output shows 10 rows (iterations) with an initial row comprising a single live cell 1 having 10 dead cells (value 0) on either side.

```
000000000010000000000
000000001110000000000
000000011001000000000
000000110111110000000
000001100100011000000
000011011111011100000
000110010000010010000
001101111100111111000
011001000111000001100
110111101100100011110
100100001011110111001
```

The visual image of the automata is obtained using the `ArrayPlot` command. The `ColorRules` option within the `ArrayPlot` command can be used to add colour to the automata.

```
ArrayPlot[CellularAutomaton[30,{{1},0},100]
```

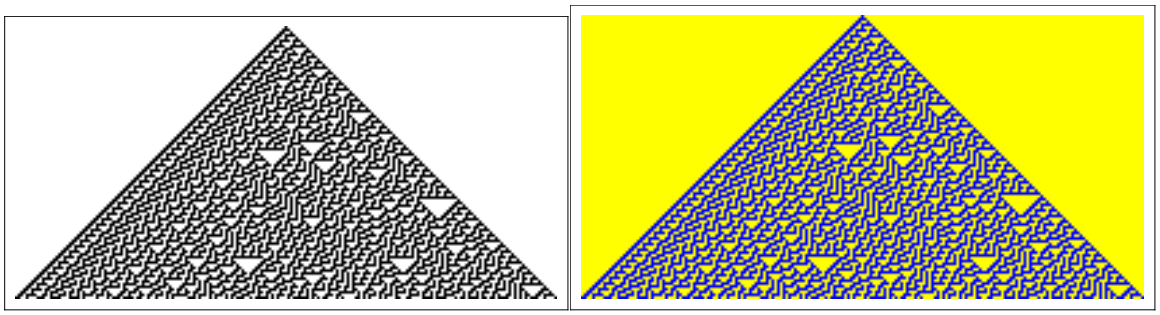

Figure 7. Mathematica generated graphic of Rule 30.

Classification and sensitivity analysis of the ECAs

The 256 ECA rules were classified them into four major categories. These are also mentioned in the research literature associated with cellular automata.

1. **Uniform**: where all cells are either black or white.
2. **Repetitive**: Some patterns are repetitive having a regular alternating pattern or a block of cells which repeat themselves throughout.
3. **Nested or Fractal –like**: These automata lead to Sierspinski triangle like fractal patterns exhibiting clear self- similarity or other nested patterns.
4. **Random or chaotic**: These are patterns which cannot be placed in any of the above three categories. There is no fixed pattern in these automata and their evolution is highly unpredictable.

The following graphical outputs of the ECAs were obtained using Mathematica. These evolve from one live cell in the centre of the top row of the grid

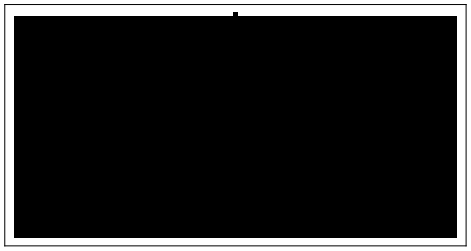

Figure 8. Rule 151: Uniform – all cells are black

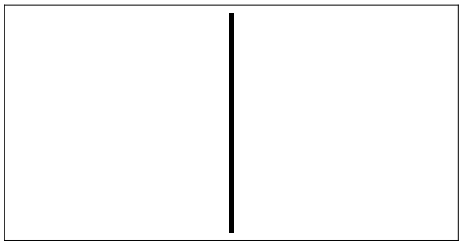

Figure 9. Rule 4: Repetitive: stationary – all cells are black in the same location

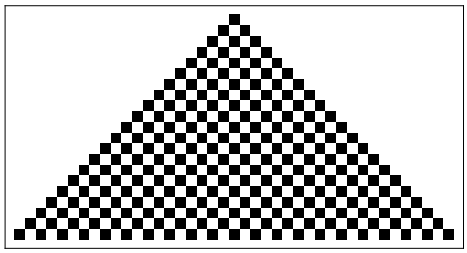

Figure 10. Rule 50: Repetitive: alternating black and white cells remain the same throughout

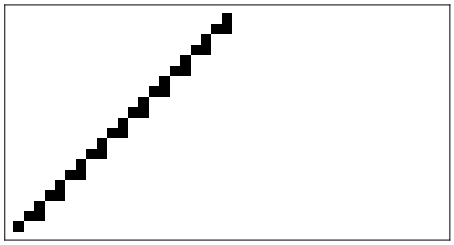

Figure 11. Rule 6: Repetitive: non-stationery: the same pattern is repeated in a different location.

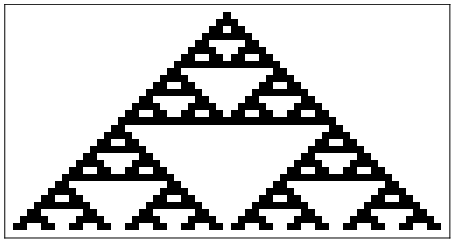

Figure 12. Rule 126: Nested: looks like the Sierspinski triangle pattern

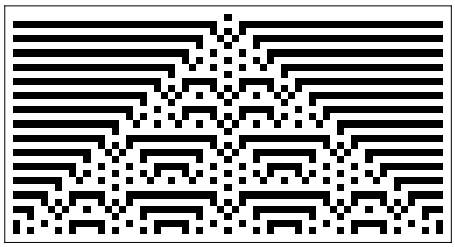

Figure 13. Rule 105: Nested: nested behaviour appearing in symmetrical patterns

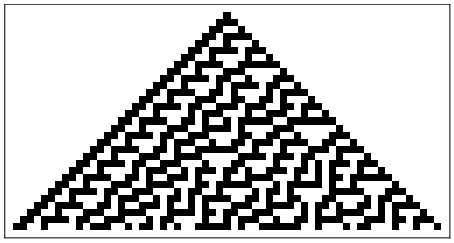

Figure 14. Rule 30: Random: completely random behaviour

Sensitivity Analysis of the ECAs

The high point of the project was when the student embarked on a sensitivity analysis of the ECAs. He tried to explore the sensitivity of each ECA rule to the specific initial conditions. For the investigation, he fixed the initial condition to a row of white cells having one black cell positioned in the centre. The objective was to observe if making a slight variation in the initial condition (such as changing the state of one cell) significantly impacted the evolution of the ECA.

For example, Rule 151 with one black cell in its initial row leads to all black cells. However if we add another black cell to the initial condition we get a Sierspinski- like nested pattern

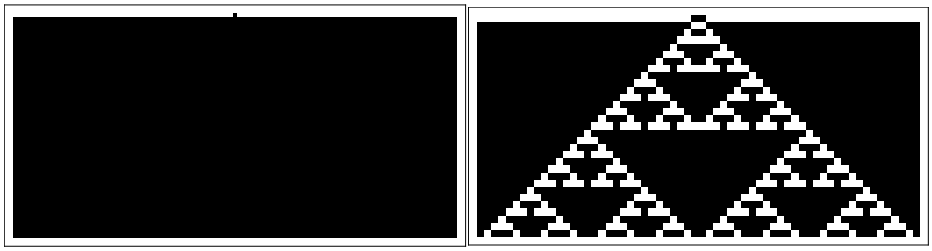

Figure 15. Rule 151 for different initial conditions which differ by only by one black cell.

Similarly Rule 106 displays a repetitive non-stationary evolution where the same pattern is repeated in a different location. However by making a slight change in its initial condition we get a random behaviour.

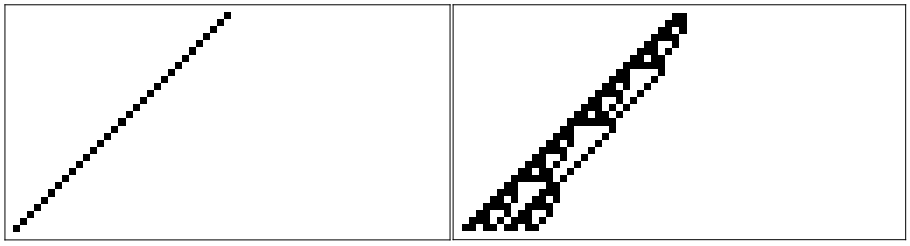

Figure 16. Rule 106 for different initial conditions which differ by only by one black cell.

All 256 ECAs fwere tested for sensitivity. The findings have been summarised in Table 2. However, all the rule numbers could not be included within the table due to paucity of space.

Table 2. Categorisation and sensitivity analysis of the 256 ECAs.

Category	**Rule Number(s)**	**Sensitivity to Initial Conditions**
Uniform All cells are white or all cells are black	0,8,32,40,64,72,96,… … 151,159,183,191, ...	Not sensitive to initial conditions. Rule 151 is very sensitive and leads to randomness and Sierpinski-like structures. Rule 183 leads to nestedness.
Repetitive(Stationery) The same pattern is repeated in the same location.	1,4,5,7,12,19,21,23, …….	Same structure was retained with minor variations.
Repetitive (Single Triangular pattern) A single triangle is formed in which the same pattern continues throughout although inner variations may occur.	50,54,58,77,94,109, 114,122,133, 147,158,163,177, 178,179,186,190, 214,222,242,246, 250,254	109 and 133 are very sensitive to initial conditions, and lead to randomness. Rule 122 leads to Sierpinski-like structure.

Repetitive (Non-Stationery) The same pattern is repeated in a different location.	2,3,6,9,10,11,14,15, 16,17,20,24,25,27, 34,35,38,39,41,42,….	Rules 106,120 are very sensitive and lead to randomness.
One-Sided Pattern Forms a definite shape on only one side.	13,28,60,69,70,78, 79,92,93,102,110, 124,137,141,153,……	Some rules, such as 124 and 137 displayed chaotic behaviour.
Nested Shows nested patterns	89,105,150	
Sierpinski Triangle Lead to a Sierpinski-Triangle structure	18,22,26,82,90,126, 129,146,154,161, 165,167,181,182, 210, 218	Rules 22 and 182 are very sensitive to initial conditions and leads to randomness.
Multiple Patterns Contains two patterns in the same rule. Many of these are symmetric.	57,62,73,99,118,131, 145	In Rule 73, the symmetry disappears.
Random The evolution seems to be random.	30,45,75,86,89,101, 135,149,169,225	Randomness is retained

Conclusion

The studies discussed in this paper highlight investigatory activities, which provide ample opportunity to engage in various aspects of computational thinking. The tasks within these activities are meaningful from a pedagogical perspective and can be easily integrated into the senior school mathematics curriculum. In the first study based on explorations of the ToH puzzle, the researcher observed a strong evidence of the use of multiple representations throughout participants' investigations – concrete, pictorial, symbolic and numerical. During their exploration they broke down the problem into simpler parts working with only two or three discs at a time to arrive at a pattern. As they tabulated the minimum number of moves they abstracted a pattern, which enabled them to find recursive as well as explicit rules for the solution. The result thus obtained was later proved using mathematical induction. Participants also compared the various representations and considered the pros and cons for each. For example they felt that the Hanoi graphs are appropriate when number of discs are three or less but for larger number of discs the numerical representation of the moves was handier. They also recognised the intricate self-similar nature of the Hanoi graphs but felt these would be 'too complex' for larger number of discs. They succeeded in representing the problem in computationally meaningful ways. They dealt with counting problems based on the arrangement of discs and the number of vertices and edges oh Hanoi graphs. Some students

even attempted to extend the problem to the 4 – peg case and obtained the minimum number of moves upto 5 discs.

The second study describing the exploration of the Elementary Cellular Automata (ECA) attempted by a grade 12 student, once again highlights the use of multiple representations in representing mathematical ideas. ECAs can be described using decimal, binary, graphical (pictorial) as well as Boolean representations. Abstracting the relationships among these representations and assessing their strengths and weaknesses is an important aspect of computational thinking. In this project, Mathematica played a critical role in exploring the graphical representations and sensitivity of the ECAs to initial conditions. At the end of the exploration the student developed a Java program to generate ECAs and also tried to define a new automata based on a different set of rules.

The level of student engagement and motivation in both the studies described in this paper have provided convincing evidence that computational thinking tasks can be easily designed and incorporated in the school curriculum. The tasks described in this paper are meaningful from a pedagogical perspective and if incorporated within a module on discrete mathematics and computer science, they can go a long way in enabling the student to explore the problems and develop computational thinking.

References

[1] Gage, D., Laub, E., McGarry, B. Cellular Automata: Is rule 30 random? extracted from https://www.cs.indiana.edu/~dgerman/2005midwestNKSconference/dgelbm.pdf

[2] Georgiadis, E., (2007). A Note on Minimal Boolean Formula Size of One-Dimensional Cellular Automata, Massachusetts Institute of Technology, Cambridge, MA 02139, U.S.A

[3] House, A. (2003). Introduction to Karnaugh Maps extracted from http://www.eecg.toronto.edu/~ahouse/mirror/engi3861/kmaps.pdf

[4] Karnaugh maps retrieved from https://en.wikipedia.org/wiki/Karnaugh_map

[5] Papert S. (1980) Mindstorms: children, computers, and powerful ideas. Basic books, New York.

[6] Wing J.M. (2006). Computational thinking. Commun ACS. 49(3); 33 – 35.

[7] Wolfram MathWorld. http://mathworld.wolfram.com/HanoiGraph.html

[8] Wolfram, S., (1986). Cellular Automata and Cryptography In Crypto '85 Proceedings, Lecture Notes in Computer Science, 218, 429-432. Springer-Verlag.

[9] Wolfram, S. (2002). A new kind of science. 1st edn. Wolfram Media. Tokyo.