



Learning computer programming: Implementing a fractal in a Turing Machine

Hernane B. de B. Pereira^{a,b,*}, Gilney F. Zebende^{a,c}, Marcelo A. Moret^{a,c}

^a Programa de Modelagem Computacional, SENAI Cimatec, Av. Orlando Gomes 1845, 41.650-010 Salvador, Bahia, Brazil

^b Departamento de Ciências Exatas, Universidade Estadual de Feira de Santana, Av. Transnordestina, S/N, 44036-900 Feira de Santana, Bahia, Brazil

^c Departamento de Física, Universidade Estadual de Feira de Santana, Av. Transnordestina, S/N, 44036-900 Feira de Santana, Bahia, Brazil

ARTICLE INFO

Article history:

Received 10 November 2009

Accepted 14 March 2010

Keywords:

Applications in subject areas
Cooperative/collaborative learning
Interdisciplinary projects
Programming and programming languages
Teaching/learning strategies

ABSTRACT

It is common to start a course on computer programming logic by teaching the algorithm concept from the point of view of natural languages, but in a schematic way. In this sense we note that the students have difficulties in understanding and implementation of the problems proposed by the teacher. The main idea of this paper is to show that the logical reasoning of computer programming students can be efficiently developed by using at the same time Turing Machine, cellular automata (Wolfram rule) and fractals theory via Problem-Based Learning (PBL). The results indicate that this approach is useful, but the teacher needs introducing, in an interdisciplinary context, the simple theory of cellular automata and the fractals before the problem implementation.

© 2010 Elsevier Ltd. All rights reserved.

1. Introduction

The main idea of this paper is to present a strategy for learning of computer programming in interdisciplinary classes. This strategy began with the assumption of how not to make boring teaching and at the same time inducing connections among the different areas of knowledge. In quest of that goal, we introduce the learning method based on problems (Section 4.1) with the main objective of the resolution of an interdisciplinary problem that involved Logic, computer programming language, cellular automata and fractals.

The proposed hypothesis is that a reduction in the repertoire of commands to solve an algorithmic problem was more stimulating for students throughout the process of solving the problem. But, there is a worry question: among the programming languages, what is it that “restricts” the basic set of commands? The response is to be found in abstract machines such as the Turing Machine and the Post Machine.

We have a proposition in Tenório (1991) chapter 4 that is the kernel of this paper: “How does the work of Turing and Post, in the same way that it happens in computer electronics, influence or may affect the relations of knowledge production, especially the relations of the subjects with the knowledge to be (re) produced in the school?”

Considering that the digital computer is based on the binary system, we believe that the inclusion of experiments that encourage students to think in a similar way to computers can facilitate their learning with respect to computer programming. In this paper, we present some reflections on the learning of computer programming, resulting from a teaching experiment that takes into account the implementation of a cellular automaton, Wolfram rule 90, in the Turing Machine. The goal here is to show that in a simple way, through an abstract machine and with a limited repertoire of instructions, we can more effectively develop the logical thinking of students of computer programming.

This paper is organized in the following way. Section 2 presents an overview of the Turing machine, so that it gives the reader a formal definition, the basic concepts and operation. Section 3 gives a brief introduction on the cellular automata as the basis for understanding the experiment. Section 4 describes the experiment, showing the method of teaching–learning used and detailing the experimental stages. Section 5 present the results obtained from a questionnaire and observations. Finally, in Section 6 we present our concluding remarks.

2. The Turing Machine

In 1936, the British mathematician, Alan Mathison Turing, many years before any modern digital computers existed, conceived of an abstract model of a computer, i.e. the conceptual structure, which can be used to represent algorithms. This abstract machine, which came to be known as the Turing machine (TM), covers just the logical aspects of its operation and not its physical implementation.

* Corresponding author. Programa de Modelagem Computacional, SENAI Cimatec, Av. Orlando Gomes 1845, 41.650-010, Salvador, Bahia, Brazil.

E-mail addresses: hbbpereira@gmail.com (H. B. de B. Pereira), gfzebende@hotmail.com (G. F. Zebende), mamoret@gmail.com (M. A. Moret).

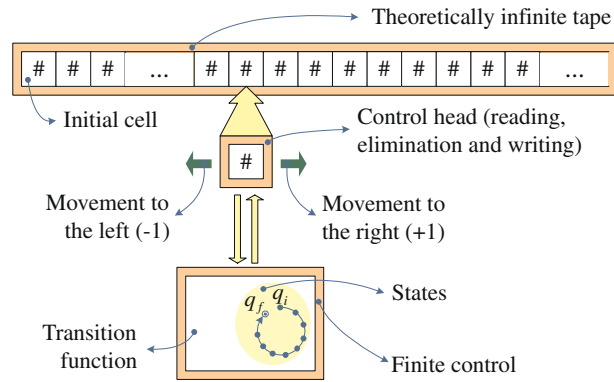


Fig. 1. Conceptual scheme of the Turing Machine.

This and others of a similar type (e.g. Emil L. Post's machine, better known as the Post Machine) enabled the development of computers – from the perspective of logic – in that they deal with, accurately, the concept of computability. This concept is strongly related to the definition of algorithms. Svaiter (2008), when presenting an introductory overview on complexity in computing, argues that:

“We can create an imaginary computer with unlimited memory and ask ourselves what are the problems that computer can solve, without worrying about the total time spent. (...) The tasks that can be performed by these computers are (problems or functions) computable.”

It is clear that a TM can model any digital computer. However, it is easier to explain what a TM is by presenting it as a machine in a physical sense. In this context, we can observe that these abstract models make it possible to set limitations for mechanical procedures. For example, in the literature (e.g. Aho, Hopcroft, & Ullman, 1974; Du & Ker-I, 2001), we find studies in which variations of TM (e.g. a strip of k -tape, etc.) are discussed. Following on, we briefly describe the simplest of variations of TM that exist, i.e. the TM of a tape, presenting its features, its formal definition and its operation.

The physical component of a TM is made up of three parts: (1) a theoretically infinite tape divided into cells or houses, with a symbol written in each cell, (2) a control head that moves along the tape, carrying out one or other of two possible moves (i.e. left or right), stopping at each house and performing the operations of reading and elimination of the contents of a cell and writing new content, which could be rewritten and (3) a finite control that manages the head above and contains a finite set of instructions that will be implemented during the program generating new states at each instant $t + 1$ (Fig. 1). Moreover, according to various authors, e.g. Aho et al. (1974), Campello and Maculan (1994), Du and Ker-I (2001) and Tenório (1991), this machine or abstract model, which performs basic functions, can be described by a tuple of 5–8 components. In this paper we have a tuple of 5 components ($Q, \Sigma, \Gamma, \delta, q_i$) (Table 1).

Besides these components, there is the final state represented by q_f , that here is assumed not to be part of Q ($q_f \notin Q$) and the white-space character that are both common to the TM. It is common to find a variation in terms of number of components that describe the machine (e.g. Campello & Maculan, 1994; Hopcroft, Motwani, & Ullman, 2002, among others).

There are many ways to conceive of the concrete operation of a TM, and therefore to structure its instructions. In general, a TM is deterministic, i.e. for every situation there is only one instruction that the machine has to follow. There are also non deterministic TM, but these machines are not described in this paper. For simplicity, we considered only the characters '0' and '1' in the TM. Fig. 2 shows two examples of notations for a typical instruction of a TM.

The set of instructions is the program that is stored in “memory” of the machine. When the machine stops, it displays a result. If a TM does not stop during the implementation of a program, it is likely that there is a defect or a possible lack of a decision (at least in finite time) in relation to the problem.

Thus, using the description of the TM presented above as a starting point, we observed that to formalize the procedure of a TM it is necessary to develop a notation for the settings or according to Hopcroft et al. (2002), instant descriptions.

In this paper, we use transition diagrams, which allow the representation of the transitions of a TM using a model similar to a graph. The states are the nodes and the movements of each state are the edges. For didactic reasons and to avoid this paper becoming too long, the exemplification of the formal procedure and the transition diagram is presented in Section 4.2, where a proposal for a result to the problem of the experiment is commented on.

Table 1
Description of TM components.

Components	Description
Q	A finite set of states.
Σ	A finite set of input alphabet not containing the white-space character (here represented by the character '#', although one can find other representations such as '_', 'B', 'Δ').
Γ	A finite set of tape alphabet with $\Sigma \cup \{\#\} \subseteq \Gamma$
δ	The state-transition function, i.e. instructions that describe the actions that will be carried out during the execution of a program.
q_i	The initial state.

Notation 01		Notation 02	
Instruction01: 1, 0, L, Instruction02			1
		q_1	$(q_2, 0, L)$

Algorithm of Instruction01 or q_1 in both notations
(1) If the symbol in the cell is 1: (1.1) Write the symbol 0; (1.2) Move the control head to the left; (1.3) Pass to Instruction02.

Fig. 2. Examples of the format of a typical TM instruction.

3. Cellular automata

Between the years 40 and 60, von Neumann and Morgenstern (1944), von Neumann (1945, 1949) began to develop the theory of automata. He wondered whether a theory could be formalized mathematically and logically, that contributed in an effective way to the understanding of natural systems (natural automata) as well as the development of computers (artificial automata). Viewed in this way, two problems closely connected to the theory of automata came to his attention: the problem of reliability and the problem of self-reproduction.

In general, we can say that the application of cellular automata works in the study of dynamic systems which, for the most part, show an incredible simplicity in their definition and can simulate highly complex and nonlinear systems, such as fractals (Mandelbrot, 1982). Thus, the cellular automaton can be defined as a fixed rule of evolution, for a set of cells, where time is discrete and the state of a given cell depends on the states of its neighbors in a fixed number of earlier moments – in general, only the immediately preceding moment (Fig. 3).

The one-dimensional cellular automaton with interactions of first neighbors, with only one at any one time, two possibilities for each cell (0 or 1), was defined by Wolfram (2002) as an elementary cellular automaton with its own rules, or if we consider the cell with its two neighbors, a triplet, the next generation $2 \times 2 \times 2 = 8$ possible binary states of a total of $2^8 = 256$ elementary rules for the possible binary states. Fig. 4 shows the Rule 90, one of most beautiful and important rule, because if we save line under line each time $(t + 1, t + 2, t + 3, \dots, t + n)$, we can obtain interesting patterns like the Sierpinski Triangle (Fig. 5), the classic fractal.

In terms of relational operators used in computer programming, this rule can be easily represented by the Boolean operation XOR ('or' exclusively). The generalization to two and three dimensions is immediate, as is its application in various other problems.

4. The experiment

One of the things that prompted the selection and preparation of the problem was the shell shown in Fig. 6. Here we had the idea of connecting together some topics (logic programming, computational complexity, Turing Machine, cellular automata, fractals) of the course Modeling Techniques in Computer Science and thus try producing in the students a better learning, this in an integrated way.

Aside from the problems concerning the Programming Logic, there was the difficulty in choosing a suitable teaching style in the teaching–learning process – from a number of educational methods that have been utilized and discussed recently. After some consideration, the proposed experiment used the method of problem-based learning (PBL) that focuses on the student and a problem. This strategy contributes to the development of reasoning and discussion, and the consequent socialization of the subject involved.

The method, based on constructivist theories of Jean Piaget and Lev S. Vygotsky, involves facing the students with real world problems. The student is constantly encouraged to learn and be part of the construction of learning (Bound & Feletti, 1998; Deslile, 1997; Duch, Groh, & Allen, 2001). Although not the main focus of this article, some details on the operation of the PBL method are presented in order to accurately describe the experiment carried out.

4.1. The PBL method

The PBL method is made up of some activities and actors – set out by Pereira and Pinto (2004) – which are presented briefly below.

- **Tutorial Group:** The tutorial group is the basic method of learning in PBL. The group is usually made up of a tutor (course teacher – specialist of the module in question), and preferably 6 to 10 students. From these students, 2 are elected at the beginning to represent

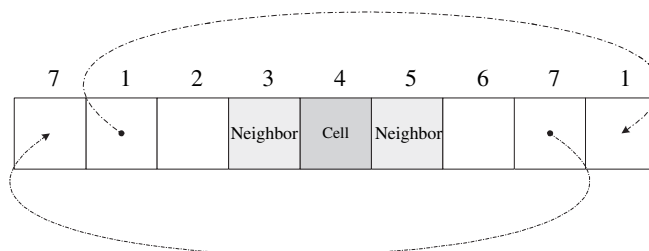


Fig. 3. Example of an one-dimensional cellular automaton subject to periodic boundary conditions. The first cell linked to the seventh and the seventh linked to the first can be used to simulate an infinite tape.

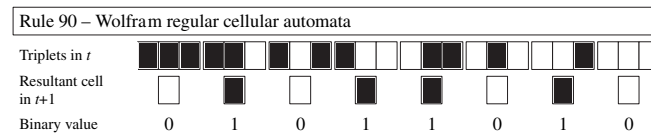


Fig. 4. The Wolfram Rule 90 of the regular cellular automaton – as we can see each rule can then be identified by a binary number of 8 bits.

respectively, the coordinator of the ensuing discussion and the secretary of the group. The actors in PBL all have clearly defined tasks and follow a rotation for the different problems, so at least once during the module/discipline each student carries out the duties of coordinator and secretary;

- *Actors involved:* The PBL method has a number of well-defined actors. Each of these actors has specific functions that are important to the effective development of the process. The tutor, student, coordinator, secretary, speaker and consultant are the main actors involved in the method of problem-based learning;
- *The Problem:* The problem becomes the core element in a PBL curriculum and is normally proposed for the development of studies on a specific topic of the curriculum. The goal of a problem is to bring about a productive discussion of the group tutorial, so it must be objective and motivate the student to participate. In some cases, there are false clues that may divert the students during the solving of the problem. An example of a problem can be seen in Fig. 6;
- *The Tutorial Session:* During a PBL session, the problem to be worked on is distributed, the coordinator and secretary designated, and the deadlines set for the handing in of the completed work and evaluation process. After the designation of the coordinator and the secretaries, the tutor hands the group a problem that should meet the curricular requirements and, within a thematic module, address a knowledge theme. The students then begin the discussion of the problem identifying the necessary aspects, i.e. ideas, facts, learning issues and goals, which could lead them to solving the problem. Second, after individual study conducted outside the tutorial session, students discuss the problem again in the light of the new knowledge acquired. These two moments have seven well-defined steps: (1) starting point, (2) brainstorming, (3) systematization, (4) formulation of questions, (5) learning goals, (6) evaluation of processes and (7) follow-up;
- *Theoretical session:* In addition to the tutorial sessions, the PBL method also involves the holding of theoretical sessions. The theoretical sessions are given by tutors or guest lecturers, usually by means of classes, seminars, lectures, debates, presentation panels, etc.

The evaluation process during the solving of the problem covers three stages. The first is the observation of student participation in tutorial sessions. They are scored by their contributions in the proposing of ideas, identification of facts, raising of learning issues, setting of goals, interaction among the group tutorial participants, and discussion emanating from the content generated and the shared theoretical–practical grounds. The second is the handing in of the report and the resultant product/service. The third is the presentation of each group's proposal for the solving of the problem, defending it in front of the other groups and/or tutors/teachers, with a general discussion of each proposal.

4.2. Solving of the problem

At the end of the PBL tutorial sessions the students should hand in, as a final product, a report that provides an in-depth introduction on Turing Machine (historical outline, mathematical foundations, applicability, etc.) and cellular automata. Furthermore, it should present the algorithm implemented in the Turing Machine, describing it in detail, as well as the source of the problem implemented and an “.TXT” file with the resultant tapes. The problem could have more than one acceptable response, based on the coding strategy used by students.

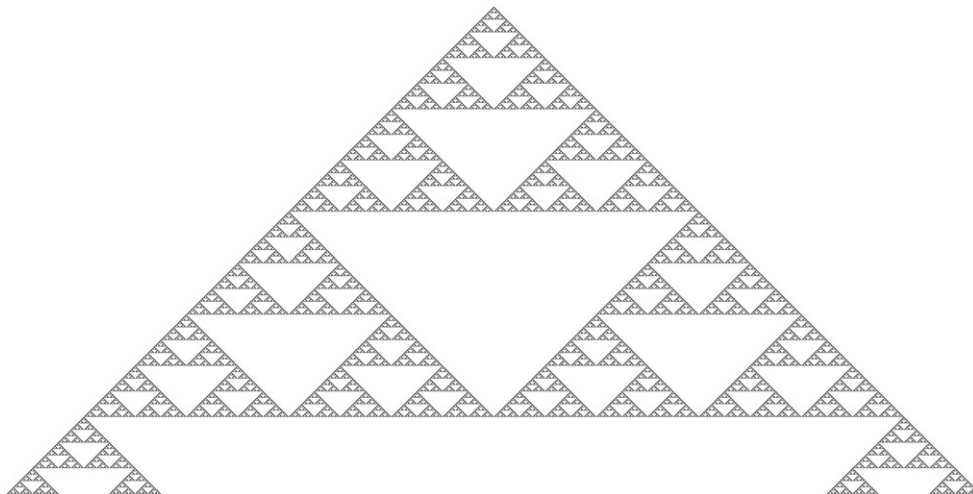


Fig. 5. Excerpt of the result of applying Rule 90.


Problem	Design Patterns	(Algorithms, Turing Machine, Celular Automata)
<p>On an evening walk along a beach during their vacation, a student of computer science finds a shell with the following patterning:</p>  <p>At this precise moment he associates the shell with some of the elementary cellular automata which he has just studied in one of the disciplines on the course. Immediately he has the idea of implementing one of the rules in a Turing machine, i.e., Rule 90, which he suspects is the closest to the patterning observed. He spends the holidays thinking about this. What can we do to help him?</p>		

Fig. 6. Problem “Design Patterns”. The image used in this problem description was originally presented by Oliveira (2008) and entitled “Figure 42. Patterns observed in a real shell”.

The initial configuration of the tape shown in Fig. 7 can be considered to be the starting point for the construction of the proposed cellular automaton. As can be seen, the initial configuration of the tape in the TM, used for the generation of patterning based on Rule 90 of the elementary cellular automata, is established from the use of two colors: white and black. Thus, conversion to the binary system is very simple, where white is equivalent to ‘0’ and black to ‘1’.

Taking into account Rule 90, the pattern to be developed (Fig. 4) and initial configuration of the tape shown in Fig. 7, we suggested to the students two tools that simulate the operation of a TM. The first developed by Prof. John Kennedy, is entitled JKTuring Program and the second entitled Visual Turing, developed by Cristian Cheran; both programs are freeware. In Fig. 8 we give examples of a sub-machine that evaluates the triplet and calculates the next element in $t + 1$.

The implementation of TM from Rule 90 suggests that if “infinite” iterations are made and each resultant tape is saved one under the previous one, the patterning shown in Figs. 5 and 6 too.

The transitions from one state to another of a TM can be represented through state-transition function (Fig. 8) or illustrations, called transition diagrams, the mathematical structure of which is similar to that of a directed graph.

A transition diagram (Fig. 9) consists of a directed graph, where N is the set of nodes, i.e. the states of TM, and A the set of arcs corresponding to the connections between the nodes (e.g. the state q_3 is connected to the states q_2, q_4 and q_9). The arcs are labeled considering the following chain of characters “sign/symbol, meaning” where $symbol \in \Gamma$ (e.g. #/1, R). This notation states that the first symbol corresponds to the value that was read on the tape and which will be assessed. If the test returns true, the reading and recording head of the TM writes the second symbol. The meaning represents the next movement of the head, represented by the letters R (right) or L (left).

We present in Fig. 6 an example of the solution to the problem of “Design Patterns” (Fig. 6). This teaching proposal served as a subject for discussion during the presentation by the students of their results. Fig. 6 presents the solution to the same problem by a group of students and shows an example of the cognitive capacity of students in optimizing the state-transition function of the TM.

5. Methodology

The experiment was carried out in five classes (2005–2009) divided into eleven groups in the discipline Modeling Techniques in Computer Science in interdisciplinary postgraduate programs on computational modeling.

5.1. Method and results

In order to obtain data for a quantitative analysis, we elaborate a questionnaire performed at the end of each class, after the delivery of reports relating to the resolution of the problem (Fig. 6). A total of 69 questionnaires (100% response rate) were answered by the students. Questionnaires (23%) were collected on the first year (2005), 19% on the second year (2006), 23% on the third year (2007), 15% on the fourth year (2008), and 20% questionnaires were collected on the fifth year (2009).

The proposed questionnaire is composed of two parts. The first part asks about student profile (Fig. 10) and the second part expresses the students opinions with respect the problem. For our purpose only the following characteristics have been asked for: student age, gender and formation/activity knowledge areas. Specifically for formation and activity knowledge areas, the current classification by CNPq (Brazilian Federal Grant Agency) has been used: Exact Sciences, Social Sciences, Health Sciences, Biological Sciences, Arts, Applied Social Sciences, Engineering, Agrarian Sciences and Linguistics and Languages.

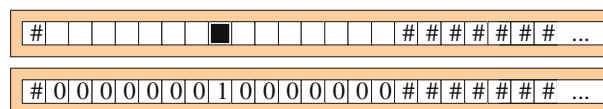


Fig. 7. Initial Configuration of the tape for Rule 90 of the Elementary Cellular Automata.

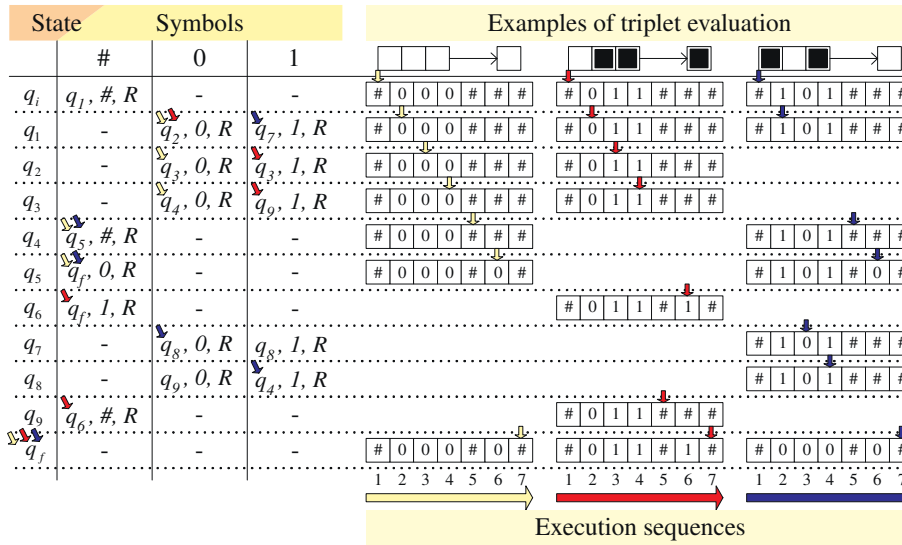


Fig. 8. State-transition function of the Turing Machine that evaluates and identifies, starting from the triplet, the value of the next cell in the instant $t + 1$.

As can be seen, Fig. 10 shows the number of students participating in the experiment by gender, i.e. of the respondents, 81% were men and 19% were women (Fig. 10a); by age, which the average age was 35.3 years, $\sigma = 9.93$ (Fig. 10b), and by formation/activity areas (Fig. 10c) that are concentrated in five major areas of knowledge: Exact Sciences, Social Sciences, Health Sciences, Applied Social Sciences and Engineering.

Table 2 shows the percentage of professional activity in formation knowledge areas of the students. We can observe that students which formation knowledge area is Health Sciences do not change of area in their professional activity. On the other hand, engineers change more to other areas different from their formation knowledge area.

Now, in the second part of the questionnaire, we have eight questions on the student opinion with respect to the proposed problem and its resolution by means the PBL method.

1. Degree of difficulty of the problem
2. Interaction among the members of the tutorial groups
3. Student knowledge of the subject before solving the problem
4. Student knowledge of the subject after solving the problem
5. Speed in solving the problem

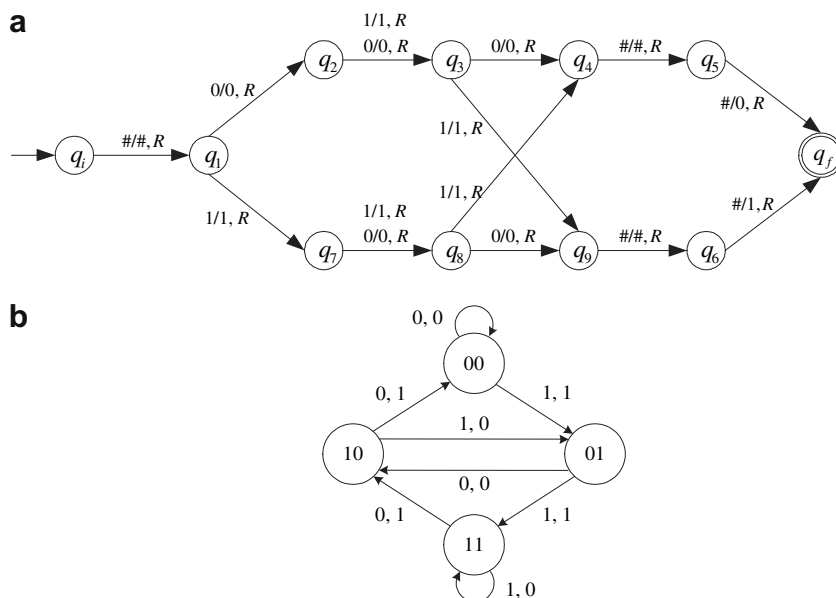


Fig. 9. Transition diagrams for the state-transition function. a: Preliminary proposal by teachers. b: Proposal by one of the groups of students.

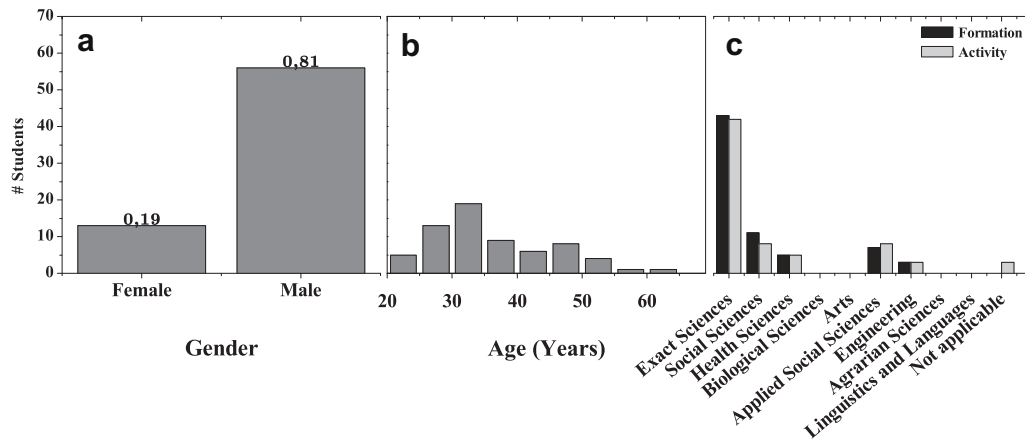


Fig. 10. Number of students by (a) gender, (b) age and (c) formation/activity area.

Table 2

Percentage of student changing profile related to student formation activity knowledge areas.

Student formation knowledge areas	% of activity in student formation knowledge areas
Engineering	67
Social sciences	73
Applied social sciences	86
Exact sciences	93
Health sciences	100

6. Level of response demanded
7. Participation of tutorial group members in solving the problem
8. Personal satisfaction after solving the problem

In order to quantify the student opinions on the proposed experiment, we applied a questionnaire based on Likert scale with a typical format of five-level, e.g. 1. Strongly disagree, 2. Disagree, 3. Neither agree nor disagree, 4. Agree and 5. Strongly agree. Moreover, we added 0 for Not applicable. We used carefully a set of scales that include adjective pairs of positive and negative types in order to capture overall evaluation in the best possible way.

We observed that all the students performed highly throughout the rest of the discipline; the approach thus gaining their approval (see mean of variable 8 in Table 3 and mainly Fig. 11). Furthermore, the students indicated to colleagues and other teachers of the courses their satisfaction with the learning acquired, and that it had given them a sense of security in the teaching of subjects related to computer programming, since most of the class were already teaching in institutions of higher education.

The next step was to try establish a relationship between the difficulty and the speed with which the problem is resolved. Thus, in the same Fig. 12, we put the relationship between student formation knowledge areas by degree of difficulty (Fig. 12a) and speed in solving the problem (Fig. 12b). Fig. 12a shows that, although the engineers' opinion on difficulty of the problem is high to very high, all students' opinions are from adequate to high. With respect to the *Speed in solving the problem*, Fig. 12b depicts that, independently of student formation knowledge area, the students' opinion give an indication of a well projected problem.

With the aim of measuring whether or no correlation between the eight variables, we made a Table 4, which gives the coefficient of Pearson for all possibility crosses.

In order to facilitate the reading of Pearson correlations, Fig. 13 shows the Pearson coefficient in ascending order of exponent r . With the support of the Table 5, we can note the three size ranges of correlation between the variables. In the large size range (0.7–1.0; –0.7 to –1.0), there are nine positive correlations and two negative correlations. In the medium size range (0.3–0.7; –0.3 to –0.7), there are six positive

Table 3

Descriptive statistics of the questionnaires applied to the students of five classes (2005–2009) and eleven groups.

#	Variables	Mean	SD	SE of mean	Variance
1	Degree of difficulty of the problem	2.57	0.96	0.12	0.93
2	Interaction among the members of the tutorial groups	4.26	0.83	0.10	0.69
3	Student knowledge of the subject before solving the problem	2.17	0.80	0.10	0.65
4	Student knowledge of the subject after solving the problem	3.75	0.77	0.09	0.60
5	Speed in solving the problem	3.10	0.73	0.09	0.53
6	Level of response demanded	2.64	0.79	0.09	0.62
7	Participation of tutorial group members in solving the problem	3.69	0.88	0.10	0.77
8	Personal satisfaction after solving the problem	4.14	0.77	0.09	0.60

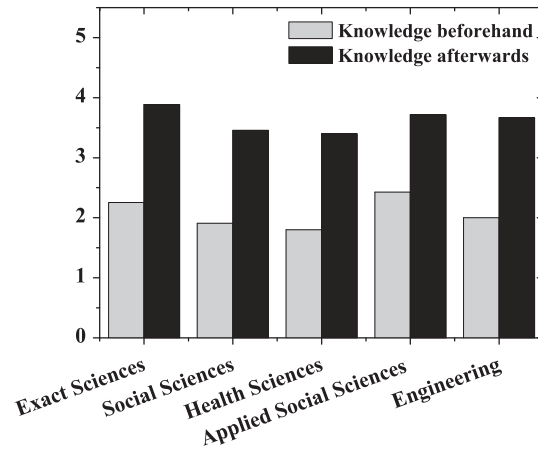


Fig. 11. Relationship between the knowledge before and knowledge after solving the problem by the student formation of knowledge areas.

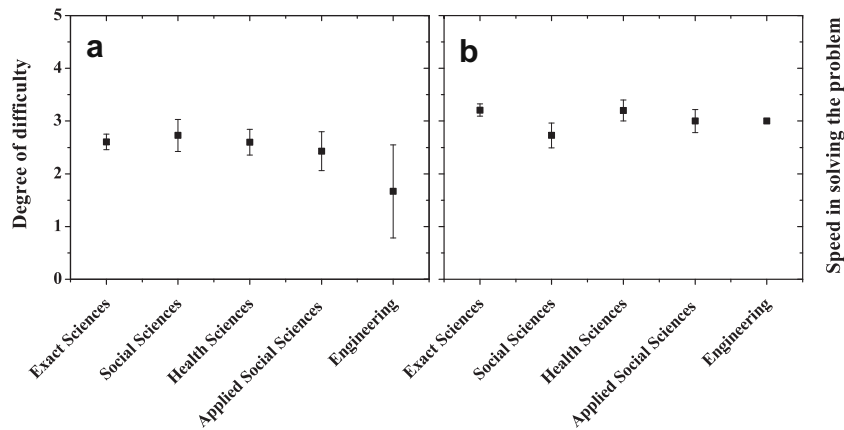


Fig. 12. Relationship between student formation knowledge areas: (a) degree of difficulty and (b) speed in solving the problem.

Table 4
Pearson correlations and ANOVA.

Variables	Variables (r) (F value; prob > F)							
	1	2	3	4	5	6	7	8
1								
2	-0.40 0.57; 0.50							
3	0.65 2.22; 0.23	-0.91 13.71 ^a ; 0.03						
4	0.10 0.032; 0.87	0.60 1.72; 0.28	-0.42 0.63; 0.49					
5	0.68 2.57; 0.21	0.18 0.10; 0.77	0.06 0.01; 0.92	0.78 4.70; 0.12				
6	0.96 40.055 ^a ; 0.01	-0.30 0.30; 0.62	0.62 1.89; 0.26	0.27 0.24; 0.65	0.75 3.90; 0.14			
7	0.19 0.11; 0.76	0.71 3.06; 0.18	-0.47 0.84; 0.43	0.96 34.86 ^a ; 0.01	0.80 5.19; 0.11	0.34 0.39; 0.58		
8	-0.31 0.31; 0.61	0.97 51.91 ^a ; 0.01	-0.85 7.54 ^a ; 0.07	0.77 4.43; 0.13	0.36 0.44; 0.55	-0.18 0.10; 0.78	0.84 7.21 ^a ; 0.07	

(1) Degree of difficulty of the problem; (2) Interaction among the members of the tutorial groups; (3) Student knowledge of the subject before solving the problem; (4) Student knowledge of the subject after solving the problem; (5) Speed in solving the problem; (6) Level of response demanded; (7) Participation of tutorial group members in solving the problem; (8) Personal satisfaction after solving the problem.

^a Correlation is significant at the 0.05 level.

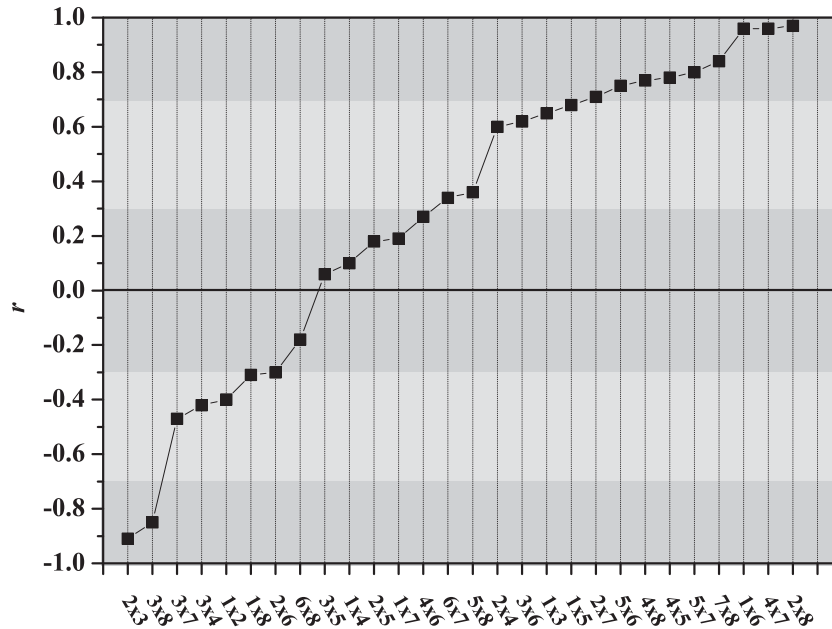


Fig. 13. Pearson Correlation in ascending exponent r . The vertical range representing the size of a correlation.

correlations and five negative correlations; and in the small size range (0.1–0.3; –0.1 to –0.3), there are five positive correlations and one negative correlation.

The results show that the strategy of the problem implementation is very effective. For example, the similarity of opinion can be explained by the high correlation between *Interaction among the members of the tutorial groups* × *Personal satisfaction after solving the problem* (Fig. 13 and Table 5) one of the principles and goals of PBL method.

5.2. Some reflections

The multidisciplinary nature of this problem acted on the one hand as a stimulus to the exercise in the interconnection of various concepts that are generally discussed separately, and on the other, as a training/introduction to the interdisciplinarity among three fields of knowledge (1) computational logic of the Turing Machine, (2) the conceptualization of fractals and (3) the regular cellular automata.

At the moment the problem was read out, some reactions of students could be observed. One of the reactions most noted was anxiety coming from fear of the “unknown”, since the unknown was not only the newly presented content (e.g. cellular automata, the fractals theory, amount others), but also computer programming using only such elementary operations as “reading”, “recording”, “move left” and “move right”. These manifestations can be seen in one of the arguments presented in the final report of the problem:

“The problem created some fear, dread for the group, which initially did not know how to solve it. Several readings were made selecting the keywords that could guide the work. An initial doubt the group had was to question the veracity of the representation of the shell design [shown in Fig. 6 of this Article] by Rule 90 of a Cellular Automaton, but then we saw we could really take it as an absolute truth and start to draw a matrix representation of Rule 90.”

During classes/meetings with the students we observed an increase in student motivation because of the challenge to connect apparently dissimilar themes and obtain a result, i.e. an algorithm for the problem given them, using so few resources. Thus, we observed that interdisciplinarity allied to an ignorance of the language, while causing a “certain fear”, became motivator elements to confront the challenge set them.

The difference between the code in any programming language or pseudo-structured programming language and the code built up from the TM became increasingly evident when the students realized on a practical level the limitations in operations that the TM offers.

Table 5
The size of correlation between the variables.

Size of Correlation	Positive	Negative
Large (0.7–1.0; –0.7 to –1.0)	2 × 7; 5 × 6; 4 × 8; 4 × 5; 5 × 7; 7 × 8; 1 × 6; 4 × 7; 2 × 8	2 × 3; 3 × 8
Medium (0.3–0.7; –0.3 to –0.7)	6 × 7; 5 × 8; 2 × 4; 3 × 6; 1 × 3; 1 × 5	3 × 7; 3 × 4; 1 × 2; 1 × 8; 2 × 6
Small (0.1–0.3; –0.1 to –0.3)	3 × 5; 1 × 4; 2 × 5; 1 × 7; 4 × 6	6 × 8

“An algorithm was created that could facilitate the process, however (...) the algorithm was based on a structured language and we had some difficulties in implementing it. (...) We used about 16 h to implement it.”

The 16 h needed to complete the implementation of the TM to the proposed problem, commented on by the students, refers to the time spent used in encoding taking into account the structured programming and the TM programming.

6. Concluding remarks

We observed that the theoretical foundations that have underpinned the development of computers over the last few decades are of great use in teaching computer programming. Therefore, the reasoning based on logical-mathematical abstract devices such as Turing or Post machines was shown to be effective, in that it allowed the representation of a considerable degree of complexity in a simple manner.

The main idea of this work was to propose a strategy that facilitates the teaching–learning process of computer programming through the implementation of a fractal via Turing Machines. This strategy proved to be challenging for students, since an interdisciplinary problem was presented to the students causing a cognitive analysis more magnified. Throughout the description of the problem here an interdisciplinary environment has been clearly outlined – which can be corroborated by comments from the students. As regards the PBL method, in terms of aspects related to computer programming, students argued in general that:

“According to the constructivist theory individuals are active agents that undertake the building up of their own knowledge, integrating the new information into their mental schemata and representing it in a meaningful way. (...) The work involving Cellular Automata and the Turing Machine itself is very constructive. Bringing it together with the PBL method which is also based on constructivism, the group could build concepts of Turing machines and cellular automata and discover how the rules are implemented.”

One of the students who participated in the discipline made the following comment:

“At first it was scary because it brought a lot of new information, which we had not met yet. There were two different fields of knowledge: the Turing machine itself, and the automaton to be reproduced. However, with the PBL methodology being used, we were able to separate the problems and create a strategy to tackle one at a time, achieving the objective proposed by the teacher. I believe that as an initial discipline of the course this application is a suitable way to introduce the student to the techniques of modeling the problem. Furthermore, the Turing machine represents the beginnings of everything that today we are developing in this course. It is worthwhile to continue using it as a tool for learning.”

So, we hope that the reader dares to use the Turing Machine as a strategy to start the study of computer programming, logic and programming languages, computational complexity, etc. In this paper an experiment was presented that can help the teaching of computer programming, using basic resources such as reading and writing in a linear and contiguous memory for the implementation of a simple rule of a cellular automaton. The study represents a line of research on the teaching of informatics. The goals are: the assimilation and the design of algorithms by beginners together with the preparation by the teachers of more efficient and effective teaching strategies.

Acknowledgments

First of all, we thank to Prof. Durval L. Nogueira *in memoriam*. This work received financial support from CNPq (Brazilian federal grant agency) and from FAPESB (Bahia state grant agency) under the grant project numbers CO-112/2005.

References

- Aho, A. V., Hopcroft, J. E., & Ullman, J. D. (1974). *The design and analysis of computer algorithms*. Reading: Addison-Wesley.
- Bound, D., & Feletti, G. (1998). *The challenge of problem-based learning*. London: Kongan.
- Campello, R. E., & Maculan, N. (1994). *Algoritmos e Heurísticas: Desenvolvimento e Avaliação de Performance*. Rio de Janeiro: Eduff.
- Deslile, R. (1997). *Use problem-based learning in the classroom*. Virginia: ASCD.
- Du, D., & Ker-I, K. (2001). *Problem solving in automata, languages, and complexity*. New York: John Wiley & Sons, Inc.
- Duch, B., Groh, S. E., & Allen, D. E. (2001). *The power of problem-based learning*. Virginia: Stylus Publishing.
- Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2002). *Introdução à Teoria de Autômatos, Linguagens e Computação* (2a. Edição). Rio de Janeiro: Editora Campus.
- Mandelbrot, B. B. (1982). *The fractal geometry of nature*. New York: W.H. Freeman and Company.
- von Neumann, J. (1945). First draft of a report on the EDVAC. Contract No. W-670-ORD-492, Moore School of Electrical Engineering, Philadelphia: University of Penn., 1945. (Reprinted (in part) in pp. 383–392 of Randell, B. (1982). *Origins of digital computers: Selected papers*. Berlin Heidelberg: Springer-Verlag).
- von Neumann, J. (1949). Theory and organization of complicated automata. (Part One). In J. von Neumann, & A. W. Burks (Eds.), *Theory of self-reproducing automata* (pp. 29–87). Urbana: University of Illinois Press, 1966. Based on transcripts of lectures delivered at the University of Illinois, in December.
- von Neumann, J., & Morgenstern, O. (1944). *Theory of games and economic behavior*. Princeton: Princeton University Press.
- Oliveira, P. M. C. (2008). Autômatos Celulares. In H. M. Nussenzveig (Ed.), *Complexidade e Caos* (pp. 83–93). Rio de Janeiro: Editora UFRJ-COPEA.
- Pereira, H. B. B., & Pinto, G. R. P. R. (2004). Problem-based learning method simulation by PBL virtual environment. In *Proceedings of the WWW/Internet 2004-IADIS International Conference, Vol. 1* (pp. 13–20), Madrid.
- Svaiter, B. F. (2008). Complexidade em Computação. In H. M. Nussenzveig (Ed.), *Complexidade e Caos* (pp. 191–198). Rio de Janeiro: Editora UFRJ-COPEA.
- Tenório, R. M. (1991). *Computadores de Papel: Máquinas Abstratas para um Ensino Concreto*. São Paulo, Brazil: Cortez-Autores Associados.
- Wolfram, S. (2002). *A new kind of science*. Champaign: Wolfram Media.