Students teach a computer how to play a game

Sylvia da Rosa Zipitría and Andrés Aguirre Dorelo

Institute of Computing, Facultad de Ingeniería, Universidad de la República
{darosa,aaguirre}@fing.edu.uy

Abstract. This paper describes a study into how secondary school students construct knowledge of programming. The study consists of three classroom sessions. In the first session the students play a simple video game called LumberJack. Then, they are asked to describe the rules of *they themselves* playing the game as an algorithm in natural language. In the second session, the students are asked to design *an automata* for a program that plays the game. In the third and final session, the students write *programs* that play the game and execute them in the programming language called TurtleBots.
The aim of the study is to help learners establish a correspondence between the algorithm and the elements relevant to the execution of the program. The results obtained in this study offers significant insights which contribute to the development of didactic guidelines for the introduction of programming to novice learners. These results are presented and analysed in section 4.

Keywords: Learning to program · Novice learners · Piaget's theory.

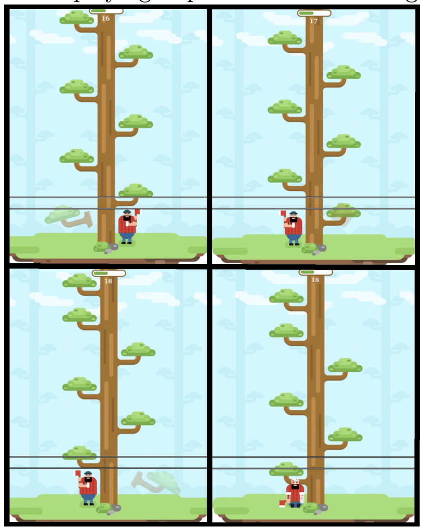
1 Introduction

Over the years we have investigated the construction of knowledge of algorithms and data structures by novice learners with the main purpose of developing of insights conducive to students learning how to write program texts [8, 9, 11, 10]. However, if the key to education research in computer science is programming, then knowledge about the text is necessary, but insufficient for learning how to program. This is due to the dual nature of computer programs [14].

Therefore, the study described in this paper focuses on students' awareness of the relationship between knowledge of algorithms and data structures (the text) and knowledge of the program as an object executed by a physical device.

In this paper we make two claims. First, we claim that the process of gaining awareness of said relationship plays a fundamental role in learning to program. This claim builds on a specific perspective of the didactics of programming, grounded on a philosophical understanding of the notion of program, as a text (an algorithm and data structures) and as an object executed by a machine [12]. Our second claim is that for the development of didactic guidelines for the introduction of programming to novice learners, each step involved in the process of knowledge construction must be considered. We argue that the point of

Fig. 1. Game playing sequence with row 2 highlighted



departure for teaching formal knowledge must always be at the level of knowledge that the student has already constructed. This claim is based on our theoretical framework, based on Piaget's theory of Genetic Epistemology, briefly described in section 1.1. In section 2 we formulate an instance of Piaget's general law of cognition [3].

Average school students between 13 and 15, of an ordinary public High School in Uruguay, participated in this study. They were asked to play a simple video game called Lumber Jack (https://tbot.xyz/lumber). We consider that playing this game is an easily understandable activity to be programmed by the students. The game is described below.

The game consists of helping the woodcutter, Jack, to cut a large tree, as shown in figure 1. As Jack hits the tree with the ax, the tree descends a fixed unit. Jack must prevent the branches of the tree from touching his head, if this happens, then the game ends.

The player can move Jack to the left or to the right by pressing two arrow buttons on the screen or the keyboard keys. Each time Jack moves he gives an ax blow on the side of the tree where he has been positioned himself. The player must choose where to position Jack to avoid being hit by the branches of the tree as it descends. It is always possible to dodge the branches that appear since the combination of having branches on the left and on the right is never given.

1.1 Main theoretical principles

We have adopted Piaget's theory -Genetic Epistemology- as our theoretical framework. This theory explains the construction of knowledge and offers a model that can be used in all domains and at all levels of development [4].

In Piaget's theory, human knowledge is considered essentially active, that is, knowing means acting on objects and reality, and constructing a system of transformations that can be carried out on or with them [4]. The more general problem of the whole epistemic development lies in determining the role of experience and operational structures of the individual in the development of knowledge, and in examining the instruments by which knowledge has been acquired **before** their formalisation. This problem was deeply studied by Piaget in his experiments about genetic psychology. From these he formulated a *general law of cognition* [3, 6], governing the relationship between know-how and conceptualisation, generated in the interaction between the subject and the objects that he/she has to deal with to solve problems or perform tasks. It is a dialectic relationship, in which sometimes the action guides the thought, and sometimes the thought guides the actions.

Piaget represented the general law of cognition by the following diagram

$$\mathrm{C} \leftarrow \mathrm{P} \rightarrow \mathrm{C'}$$

where P represents the periphery, that is to say, the more immediate and exterior reaction of the subject confronting the objects to solve a problem or perform a task. This reaction is associated to pursuing a goal and achieving results, without awareness neither of actions nor of the reasons for success or failure. The arrows represent the internal mechanism of the thinking process, by which the subject becomes aware of the coordination of his/her actions (C in the diagram), the modifications that these impose to objects, as well as of their intrinsic properties (C' in the diagram). The process of the grasp of consciousness described by the general law of cognition constitutes a first step towards the construction of concepts.

Piaget also describes the cognitive instrument enabling these processes, which he calls *reflective abstraction* and *constructive generalisation* [3, 5].

The problem about the construction of computer science concepts is an instance of the ones deeply studied by Piaget. The general law of cognition regulates the construction of knowledge about (basic) algorithms (C) and data structures (C') from problems that students are asked to solve (P) (for instance sorting, counting, searching elements) [7].

However, in the case that *the object on which knowledge is to be constructed is a program*, some challenges inherent to the relevance of the machine that executes it appear. These challenges took us to instantiate the law to the case of knowledge about programs, as described in the next section.

2 Instantiating the general law of cognition

The construction of knowledge about algorithms and data structures is a process regulated by the general law of cognition. Over the years we have investigated the construction of knowledge by novice learners of algorithms and data structures. Our research methodology is based on applying Piaget's general law of cognition to make students solve problems (for instance sorting, counting, searching elements [7–10]) and reflect about the method they employ and the reasons for their success (or failure), as a first step towards the conceptualisation of algorithms and data structures.

We developed an instance of Piaget's general law of cognition as we identified the need to describe cases where the subject must instruct an action to a computer. The thought processes and methods involved in such cases differ from those in which the subject instructs another subject, or performs the action themselves.

Our instance of Piaget's law was developed to take into account the specificities of the subject instructing a computer to solve the problem of playing a video game, in this case the Lumber Jack one. In such a case, the aim are for students; on the one hand, to express the rules of *they themselves* playing the game as an algorithm in natural language, on the other hand, to design *an automata* for a program that plays the game, and finally to write and execute *a program* that plays the game.

As Simon Papert says in [13] (page 28) referring to the programming of a turtle automata, *Programming the turtle starts by making one reflect on how one does oneself what one would like the Turtle to do.* In this case, programming an automata that plays a game starts by making the student reflect on how he/she does herself what he/she would like the automata to do.

To programming an automata solving a problem, the learners have to establish a causal relationship between the algorithm (he/she acting on objects), and the elements relevant to the execution of the program (the computer acting on states). Not only they have to be able to write the algorithm (the text), but also they have to be able to understand the conditions that make *the computer* run the program.

The generalisation of Papert's words above can be described as: programming an automata starts by making one reflect on

$$\underbrace{how\ one\ does\ oneself}_{what\ one\ would\ like\ the\ automata\ to\ do}$$

The causal relationship between the first row and the second row is the key of the knowledge of *a machine executing a program.* It is indicated with the brace in above description.

By way of analogy with Piaget's law we describe this relationship in the following diagram

$$\underbrace{C \leftarrow P \rightarrow C'}_{newC \;\longleftarrow\; newP \;\longrightarrow\; newC'}$$

where *newP* is characterised by a periphery centred on the actions of the subject and the objects he/she acts on. The centres *newC* and *newC'* represent awareness of what happens inside the computer.

The diagram describes the situation in which the subject reflecting on his/her role as problem solver becomes aware of how to do to make the computer solve the problem. According to Piaget, we identify that the construction of knowledge of methods (algorithms) and objects (data structures) occurs in the interaction between C, P and C'. Likewise, we claim that the construction of knowledge of the execution of a program takes place in the internal mechanisms of the thinking process; marked by the arrows between newC, newP and newC'. In other words, the general law of cognition remains applicable to the thinking process represented by the arrows; in both lines of the diagram pictured above.

2.1 A pilot study

The application of our research methodology in the case of this study is based on getting the students to reflect about the method they employ to successfully play the game and the reasons for their success, as a first step towards the conceptualisation of the rules of the game as an algorithm. In order to achieve such a reflection, we conducted activities consisting in several questions and exercises. To designing those questions and exercises we conducted a pilot study to detect aspects that could eventually be improved. In the pilot study the players (not the students participating of the study) were asked to play the game for a while and then to describe how they play in natural language in their own words. They described actions and objects related to themselves; as exemplified by the quotes below:

1. *I try to play on the phone ... it is uncomfortable ... I use a notebook as support to improve my posture. I get frustrated when I do not succeed and I start again; paying attention to any mistakes in order to correct them. When I start I look up the tree to anticipate movement ... I change the position of my fingers; with index fingers it's better.*
2. *I go slowly when I see a branch and I go faster if there is no branch. I try to prevent. I go slower when it approaches.*
3. *Jack is cutting the trunk; moving to the right and left depending on where the branches appear. When the branch is on the right side Jack runs to the left and when the branch appears on the left side Jack moves to the right.*

The most notable observation at this stage is that the players did not notice that Jack's movement depends on the actions of the player (for instance, at the third quote the player describes Jack's movements as independent from his/her own actions of pressing the keys/buttons.) In other words, there is a lack of awareness of the causal relationship between what the player does and what Jack does.

Keeping in mind our task of inducing students' reflection on how he/she does herself what he/she would like the program to do, we set out to design questions aimed to direct students' attention away from newP (e.g. the position of their fingers or how they feel) to newC and newC' (Jack's positions, branches states at the row above Jack and what has to be done for not losing). By this we mean, to be aware of the causal relationship between their own actions (perceive the branches, press the keys/buttons) and the events in the computer (Jack's positions, the descending branches, the key/buttons events).

The result of the pilot study is the list of questions **Q1 - Q4** of section 3.1.

3 Describing the study

The main objective of the study is to help learners establish a correspondence between the algorithm and the elements relevant to the execution of the program. To achieve this main objective the following tasks are proposed to students:

1. play the game and learn how it works
2. reflect on the rules and express them as inference rules in natural language
3. represent the rules in the previous point as a state machine (an automata)
4. program the automata in TurtleBots and make the program play the game

The strategy described by tasks 1 to 4 and the results of the study are based and validated within our theoretical framework, as explained in section 4.

These tasks were grouped into three classroom sessions conducted over several months. In order of recalling the main concepts, each session began by handing out students' previous work with feedback. We also used activities, not included here for space reasons, with the same purpose of recalling concepts.

Points 1 and 2 in the list above make the first session; the second session includes point 3. Finally, point 4 takes place during the third session of the study. We describe the three classroom sessions in the following sections. The activities listed above are included in the descriptions but not explicitly numbered as listed. The description of each classroom session also includes the motivation behind the chosen formalisms.

3.1 First classroom session

The first classroom session took place on March 23 of 2017. The class taking part consisted of 25 students and the session was divided into two sessions of 45 minutes each with a break of 5 minutes. As a first task, the students were asked to play the game for approximately 15 minutes. All 25 students were able to play the game successfully. Based on the results of the pilot study, our assumption is that the players would be unaware that their game playing caused the computer to follow certain instructions of the algorithm of the game. Therefore, the first objective of the session was for the students to become aware of this relationship and conceptualise the rules of the player's algorithm; successfully expressing them in natural language in the form of inferences like *if ... then ...*.

In order to facilitate the process we described the game emphasising the critical moment before Jack changes side of the tree by displaying figure 1 on the whiteboard (see page 2) and using a table similar to table 1 to represent the critical situation of the game (as Jack may be hit) and the decisions of the player (pressing of buttons to avoid Jack's fall).

Having played the game the players were asked to individually write down the answers to the following questions:

Q1 Before starting to play, what are the possible positions that Jack can be in, in relation to the tree?

Table 1. Table used in question 3

line	Jack	branch	button
1	Left	No	Left
2	Left	Yes	Right
3	Right	No	Left
4	Left	No	?
5	Left	Yes	Right
6	Right	Yes	Left
7	Left	No	Left
8	?	Yes	Right
9	Right	No	Right
10	Right	Yes	?
11	Hit	X	X

Q2 How do you decide which buttons to press?
Q3 Could you complete the cells that have the symbol '?'in the table 1?
Q4 Could you explain in your own words when success and failure occur?

Examples of students' answers In the analysis we organised the answers according to three levels; level 1, intermediate level and level 2. A description of each level is included after the examples.

- Student 20 (level 1): Q4: *You succeed when you press the correct buttons, for example: Jack is on the left side and the branch is almost on top of him. You have to go to the right so that the branch does not hit you and you die.* Q3: only line 1 has been filled in incorrectly.
- Student 14 (level 1): Q2: *We press right when there is a branch on the left and if one is on the right, we move to the left.* Q3: all the lines are correct. There is no inference in Q4 in this case. However, it is often the case that students omit an answer if they feel they have already answered the question elsewhere.
- Student 7 (intermediate level): Q2: *Depending on the position of the branches, the key we are going to press is: branch on the right, we press the arrow on the left so that Jack moves to the left.* Q3: No answer.
- Student 19 (intermediate level): Q2: *You can use left or right buttons; the one you use will depend on where Jack is.* Q4: *Success occurs when, for example, Jack is on the left, there is no branch and we press the left button, or when Jack is on the right, there is a branch and we press the left button.* Q3: line 10 has been filled in incorrectly.
- Student 18 (intermediate level): Q4: *Success occurs when we press the right button: if Jack is on the left side and there is a branch, you press the other button.* Q3: line 10 has been filled in incorrectly.
- Student 4 (level 2): Q2: *I decide to press the keyboard which is easier.* Q3: all lines incorrectly filled.

– Student 3 (level 2): Q2: *I decide to go to the opposite side in order to cut the tree.* Q3: only line 4 is correct.

Analysis of students' responses and their classification in three levels For analysing students answers we use the following criteria: In the students' answers to questions 2 and 4 (Q2 and Q4) we were looking for explicit inferences similar to "if Jack was on such a side and there is a branch on that side, then the pressed button was such ... ". In the students' answers to question 3 (Q3) we were looking for implicit inferences, because the correct answer has to be deduced from the lines above and/or below of table 1 (used in question 3, see page 7). The students find that on lines 4 and 8, the correct answer can be deduced by looking only at the "Jack" and "Button" columns.

In contrast, line 10 of the table is a challenge because the correct answer has to be deduced from the failure in the line below. In general students answered correctly in lines 4 and 8 (in this case; success) and wrongly in line 10 (in this case; failure). As a result we classify the thinking of the students into three distinct levels:

– Level 1 is formed by the answers in which inferences appear in the students' answers to Q2 and/or Q4 and the student has managed to fill line 10 in the table 1 correctly.
– The intermediate level is formed by two possible scenarios:
  • inferences appear in the students' answers to Q2 and/or Q4. However, in this case the students did not manage to fill in line 10 correctly. Or:
  • there are no inferences, but the table 1 has been filled in correctly in its totality; which indicates an implicit inference.
– Level 2 is formed by cases where there are no inferences to Q2 and Q4, and the table 1 has been filled in with the wrong answers, particularly in line 10.

Table 2 shows the distribution of the answers of the students according to these three levels. The students are referenced by numbers from 1 to 25 and NA means "no answer".

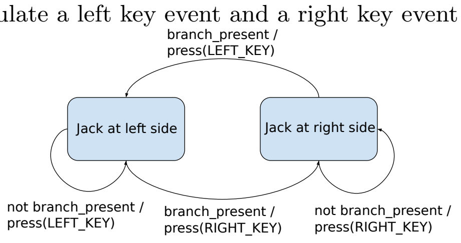
Table 2. Classification by level

level	**student**	**total**
level 1	14, 16, 20, 21	4
intermediate	6, 7, 8, 13, 17, 18, 19, 22	8
level 2	1, 2, 3, 4, 5, 9, 11, 12, 15, 23, 24	11
NA	10, 25	2

3.2 Second classroom session

We now turn to the second classroom session in which the students face the challenge of designing an automata like the one in the figure 2. This challenge refers to the third point in the list of activities (see section 3, page 6).

Fig. 2. Finite state machine that represents the winning behaviour of Jack as an automata. The input predicate, branch_present, represents the state of the row above Jack, is true when there is a branch and false otherwise. The function press() used as output in the machine, takes as parameter the constants LEFT_KEY and RIGHT_KEY, to simulate a left key event and a right key event respectively.



The second activity of the study was carried out on May 9 of 2017 with the same class of 25 students.

Following Piaget's general law of cognition, our premise is that: having participated in the first session, players now possess a higher level of knowledge of the program/game than that of a novice learner who simply plays the game. The qualitative difference resides in the experience of the novice learner who gains knowledge about the game by playing, and the players whose gaming experience has been followed by a process of reflection as a result of the activities in the first classroom session. This fact is, indeed, revealed by the answers that players provide during the first classroom session.

Therefore, the aim of activity three (see section 3, page 6) is to ground the knowledge constructed during the first session by expressing the rules of the game as a state machine. To do so, we begin the second classroom session by introducing the concepts of automata and of state machine as diagrams that provide a visual representation that encompasses all the elements of an automata. We use the example of programming instructions to help a blind person, connected to devices equipped with sensors, to cross the street where there is a traffic light. For space reasons this part is not included here.

For homework, the students were asked to design an automata for the Lumber Jack game, based on the work they had done in the first classroom session, and on the problem of the traffic light. A few days later, we received all of the students' homework: of the 18 returned answers only four contained mistakes, all of which were of minor significance. These mistakes were: three failed to indicate one of the transitions in the diagram, and one failed to indicate the direction of the transitions as the arrows were replaced by lines.

It is worth mentioning at this point that the state machine formalism is generally introduced in tertiary or university education, and is often considered complex by the students. By including the state machine formalism in the study we were able to determine whether this formalism in itself implies a level of difficulty that is inappropriate for students aged 13 to 15. In other words; whether novice learners would be able to convert an algorithm expressed in natural language, into a diagram with representations of elements of implementation (states, inputs, outputs) (see figure 2 page 9).

Keeping this in mind, the students' work on the state machine diagram was kept fairly simple. Even so, we found it to be remarkable that out of all of the activities in the study, it was in this particular task of designing the automata using the state machine formalism that the students achieved the best results.

3.3 Third classroom session

The third and final session was held on August 10 of 2017. The purpose of this session was to complete activities of point 4 of section 3, page 6. The students were asked to implement the game Lumber Jack in TurtleBots. We chose the programming language TurtleBots [2] because students already have some experience with the Scratch language which is based on blocks like TurtleBots. In addition, TurtleBots contains a plugin called Xevents [1] that programmatically generates keyboard and mouse events, and gathers screen information. These features are necessary to implement the actuation and sensing components of the LumberJack game controller. They were given a central axis and a series of scattered blocks on which to work with, as shown in figures 3 and 4 in the Appendix. The task consisted in placing the blocks in the correct places in order to assemble the program and then execute it. Although every student completed the task, they generally did not handle the blocks with ease, and mistakes were made in the assembly of the blocks. In the following section, this and others results are analysed.

4 Conclusions

The study has provided us with new theoretical and practical insights which we will discuss here. Our premise is that the depth of the results of this study are not clearly visible until we locate them within our theoretical framework. We therefore use our theoretical principles in this section to explain, validate and verify our results. More specifically, our results can be explained from theoretical principles about the relationship between non-formal, conceptual and formal knowledge. By non-formal we mean the instrumental knowledge constructed at the level of actions [3]. This is the level in which novice learners can develop a skill simply by playing. Their knowledge of the game may allow them to play it successfully, but it remains at the level of action nonetheless.

The first step towards the conceptual construction consists in the students' reflection about the rules of the game in terms of inferences involving the elements of the program (Jack and branch positions, key/buttons). The reflection players experienced during the first session, implies the transit of the students' thought from a periphery (where it is focused on the goal of the player) towards the awareness of how to do for an automata to play. This awareness implies a conceptual construction and, therefore, a higher level of knowledge.

We argue that the point of departure for teaching formal knowledge (by this we mean conceptual knowledge expressed in some formalism different from natural language) must always be at the level of knowledge that the student has *already constructed*. In other words; any learning process is built stepwise and is governed by the general law of cognition. In the specific case of learning to program, the process is governed by the instance of the law of cognition as we have formulated it on page 4.

Our theoretical argument is verified experimentally by students answers to questions Q1 to Q4 in section 3, page 7. Two main factors are central to the task of answering the questions: first, that the students have already internalised the rules of the game by playing the game for 15 minutes and, thus, have developed gaming skills in terms of their actions. Second, that the first question is formulated directing students' attention away from *they themselves* (newP) towards the elements in the *world model of the game* (newC and newC'). They were then able to express the causal relationship between their own actions and the events in the computer. This factor is a direct consequence of applying our framework of the instantiated law of cognition in designing the questions.

By explaining and theoretically locating the learning process that we have studied empirically, we reaffirm the significance and fundamental role of the learning process when developing teaching guidelines: the key is to respect the process when teaching a class; for the conceptual we start at the non-formal, and for the formal we begin at the conceptual. In accordance to this, we have verified that the activity of introducing the formalism of a state machine for designing the automata, which is usually considered complex, can be carried out without difficulties by novice learners aged 13 to 15.

The fact that the results of the first classroom session revealed greater difficulties also confirms the transition from actions to the construction of conceptual knowledge. The reflective process the students have to experience in order to successfully express themselves about the game in natural language, is a difficult one. For this reason we felt it necessary to dedicate this part of the process a section of analysis (see, page 8).

In terms of the limitations we encountered we would highlight the unexpected difficulties of the students in the implementation of the proposed exercise with TurtleBots. In order to address this limitation we pose the following questions:

Given that the program in TurtleBots introduces repetition and variables to represent the states, is the level of prior knowledge of the students sufficient to model these concepts or is it necessary to work with them more carefully?

Are block languages an adequate tool to implement this type of problem with students aged between 13 and 15? Or would it be more convenient to use a textual programming language, like Python for example?

When considering future work we would direct our research towards answering the questions mentioned above.

Finally, we find necessary to point out that we have not included references to other authors in this paper because the aim of the study is contributing of designing a didactics for programming, based on our model of applying Piaget's theory. References to related work are included in our previous papers, already cited in this paper.

5 Acknowledgements

We would like to thank teacher Néstor Larroca and students of second year of Liceo 2, La Paz, Uruguay, where we developed the study; Bruno Michetti for supporting the activities; and Manuela Cabezas for correcting the English. The comments of the anonymous referees are gratefully acknowledged.

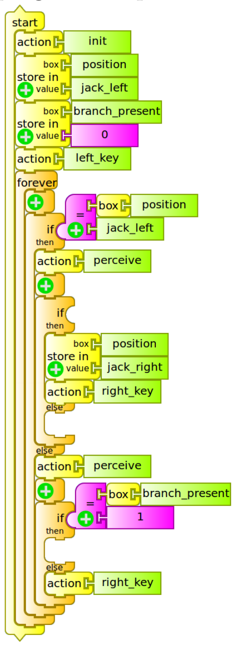
References

1. InCo: https://www.fing.edu.uy/inco/proyectos/butia/mediawiki/index.php/Xevents (2015), accessed: 2018-02-17
2. InCo, Sugarlabs: Turtlebots. https://www.fing.edu.uy/inco/proyectos/butia (2015), accessed: 2017-04-19
3. Piaget, J.: La Prise de Conscience. Presses Universitaires de France (1964)
4. Piaget, J.: Genetic Epistemology, a series of lectures delivered by Piaget at Columbia University, translated by Eleanor Duckworth. Columbia University Press (1977)
5. Piaget, J.: Recherches sur la Généralisation. Presses Universitaires de France (1978)
6. Piaget, J.: Success and Understanding. Harvard University Press (1978)
7. da Rosa, S.: Designing Algorithms in High School Mathematics. Lecture Notes in Computer Science, vol. 3294, Springer-Verlag (2004)
8. da Rosa, S.: The Learning of Recursive Algorithms from a Psychogenetic Perspective. Proceedings of the 19th Annual Psychology of Programming Interest Group Workshop, Joensuu, Finland pp. 201–215 (2007)
9. da Rosa, S.: The Construction of the Concept of Binary Search Algorithm. Proceedings of the 22th Annual Psychology of Programming Interest Group Workshop, Madrid, Spain pp. 100–111 (2010)
10. da Rosa, S.: The construction of knowledge of basic algorithms and data structures by novice learners. Proceedings of the 26th Annual Psychology of Programming Interest Group Workshop, Bournemouth, UK (2015)
11. da Rosa, S., Chmiel, A.: A Study about Students' Knowledge of Inductive Structures. Proceedings of the 24th Annual Psychology of Programming Interest Group Workshop, London, UK (2012)
12. da Rosa, S., Chmiel, A., Gómez, F.: Philosophy of Computer Science and its Effect on Education - Towards the Construction of an Interdisciplinary Group. Special edition of the CLEI Electronic Journal (see http://www.clei.cl/cleiej/), Volume 19 : Number 1 : Paper 5 (2016)
13. Simon Papert: Papert, S. Mindstorms: Children, Computers, and Powerful Ideas. Basic Books (1980)
14. Tedre, M.: The Science of Computing: Shaping a Discipline. CRC Press, ISBN 9781482217698 (2014)

A Appendix

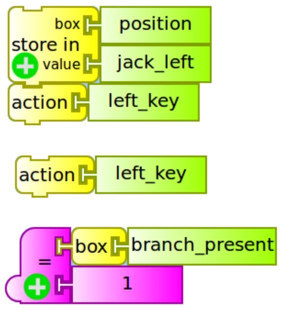
Figures of the third classroom session

Fig. 3. TurtleBots program template for Jack's automata



```
start
action init
store in  box position
          value jack_left
store in  box branch_present
          value 0
action left_key
forever
  if = box position
       jack_left
  then
    action perceive
    if
    then
      store in  box position
                value jack_right
      action right_key
    else
  else
    action perceive
    if = box branch_present
         1
    then
    else
      action right_key
```

Fig. 4. TurtleBots blocks for Jack's automata



```
store in  box position
          value jack_left
action left_key

action left_key

= box branch_present
  1
```