The construction of knowledge of basic algorithms and data structures by novice learners

Sylvia da Rosa

Instituto de Computación - Facultad de Ingeniería - Universidad de la República
darosa@fing.edu.uy

Abstract. Piaget's theory offers a model for explaining the construction of knowledge that can be used in all domains and at all levels of development, based on establishing certain parallels between general mechanisms leading from one form of knowledge to another, both in psychogenesis and in the historical evolution of ideas and theories. The most important notion of these mechanisms is the triad of stages, called by Piaget the *intra, inter* and *trans* stages. The main goal of our work is to build an instance of that model for research about the knowledge of basic algorithms and data structures constructed by novice students. This paper presents some aspects of our work, focusing on the passage from conceptual knowledge (intra-inter stages) to formalized knowledge (trans stage).

Keywords: genetic epistemology, constructing knowledge, conceptualization, formalization

1 Introduction

The central points of Piaget's theory -Genetic Epistemology- have been to study the construction of knowledge as a process and *to explain how the transition is made from a lower level of knowledge to a level that is judged to be higher* (Piaget, 1977).

The supporting information comes mainly from two sources: first, from empirical studies of the construction of knowledge by subjects from birth to adolescence (giving rise to Piaget's genetic psychology)(Piaget, 1975, 1964; Piaget & coll., 1963; Piaget, 1978b), and second, from a critical analysis of the history of sciences, elaborated by Piaget and García to investigate the origin and development of scientific ideas, concepts and theories. In (Piaget & García, 1980) the authors present a synthesis of Piaget's epistemological theory and a new perspective on his explanations about constructing knowledge. They investigate the possible analogy between the mechanisms of psycho-genetic development concerning the evolution of intelligence in children, and sociogenetic development concerning the evolution of the leading ideas and theories in some domains of science. Throughout the chapters the authors present striking examples of this analogy in relation to the history of geometry, algebra, mechanical and physical knowledge in general. The main idea of their synthesis consists of establishing certain parallels between general mechanisms leading from one form of knowledge to another - both in psychogenesis and in the historical evolution of ideas and theories -, where the most important notion of these mechanisms is the triad of stages, called by the authors the *intra, inter* and *trans* stages. The triad explains the process of knowledge construction by means of the passage from a first stage focused on isolated objects or elements (intra stage), to another that takes into account the relationships between objects and their transformations (inter stage), leading to the construction of a *système d'ensemble*, that is, general structures involving both generalized elements and their transformations (trans stage), integrating the constructions of the previous stages as particular cases.

Piaget's theory offers a model for explaining the construction of knowledge that can be used in all domains and at all levels of development. The main goal of our work is to build an instance of that model for research on the knowledge of basic algorithms and data structures constructed by novice students. Over the years we have investigated the intra-inter-trans stages in the construction of knowledge of algorithms and data structures and in previous papers (da Rosa, 2010, 2007, 2005, 2004; da Rosa & Chmiel, 2012; da Rosa, 2003) we have described our research about this passage:

- from an intra stage, in which the knowledge is instrumental, that is, in the plane of actions (the students pursue a result but are unaware of how they achieve it)
- to an inter stage giving rise to conceptual knowledge, that is in the plane of thought (the students give accurate descriptions of how they did it and why they succeeded, being aware of the coordination of their actions and the transformation of objects).

A summary of this research is included in Section 3.1. The goal of this paper is to describe our research about the passage from earlier stages above to

– a trans stage of formal knowledge (where the students are able to express algorithms in given formalisms and modify their knowledge to solve similar problems).

Regarding the methodology of our research, the passage from intra to inter stage is investigated by means of conducting individual interviews, in the sense of Piaget's studies of genetic psychology (da Rosa, 2010, 2007, 2005, 2004; da Rosa & Chmiel, 2012; da Rosa, 2003). The passage to the trans stage is investigated by means of conducting instructional episodes where students work in groups and some formalism is introduced (mathematical language, pseudo code and/or programming languages). In this part, our methodology follows Piaget's studies about the role of social relations and formal education in knowledge construction (Youniss & Damon, 1997; Ferreiro, 1996)[1]. The goal is to help students in establishing correspondences between the concepts that they have previously constructed and expressions of the formalism in order to obtain formal descriptions of their solutions. The dialectic process of this construction is explained in Section 3.2 and constitutes the main goal of this paper, and includes brief descriptions included in previous work. Our investigations have been conducted with students entering university or enrolled in the final year pre-university. That means that they have no (or very little) experience with programming (in Uruguay Computer Science is not part of High School curriculum). All the research episodes were recordered and/or filmed and students wrote out some of their responses.

Finally, we offer some comments about the motivation and the questions behind our research. Several researchers consider programming as a powerful and essential subject not only in computer science studies but in other studies as well (Dowek, 2005, 2013; Wing, 2000; Peyton Jones, 2013a; Bradshaw & Woollard, 2012; Peyton Jones, 2013b; R.Page & Gamboa, 2013). At the same time it is seen as a difficult topic both to teach and to learn, and studies in the didactics of informatics have become necessary (Holmboe, McIver, & E.George, 2001; Saeli, 2012; Hubwieser, 2013; Nickerson, 2013; Ambrosio, 2014).

In contrast to several proposals to help students in the learning of programming involving the use of some programming language or computer tool (Gomes Anabela, 2007; Moström, 2011; Linda Mannilaa, 2007; Budd, 2006; E.George, 2000; Tina Götschi, 2003), our approach is based on observations of situations in day-to-day life in which people successfully use methods to solve problems or perform tasks such as games, ordering of objects, different kinds of searches, mathematics problems, etc. In such situations an action or a sequence of actions is repeated until a special state is reached, which can be solved easily by a straight-forward action. People's descriptions include phrases like "I do the same until ... " and "now I know how to do it", referring to cases where they use the same method and arrive at the easy-to-solve special state respectively. These descriptions are related to programming in the sense that repeating actions until a special case is reached, is formalized by recursive or iterative program instructions.

These observations lead us to formulate questions such as "does there exist any connection between the 'knowing how to' (instrumental knowledge) revealed by people solving problems and formal algorithms? If there is, what is the nature of this connection and what is the role of the instrumental knowledge in the learning process? How is this instrumental knowledge generated and how can it be transformed into conceptual knowledge? How can the algorithms that the students learn to use taken into account in the learning of programming? Will the answer to these questions help in improving the teaching and learning of programming and how should this be done?" The approach of our research arises from the above observations and questions and from studying the theory of Jean Piaget that explains the construction of knowledge and the evolution of cognitive instruments from the interaction of the subject (his/her methods) with the objects (data structures).

The following sections of this paper include: the main theoretical principles of our research (Section 2), how these are applied (Section 3), some related work (Section 4) and conclusions and further work (Section 5).

2 Main theoretical principles

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 (Piaget, 1977). 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 formalization. This problem was deeply studied by Piaget in his experiments about genetic psychology. From these he formulated a *general law of cognition* (Piaget, 1964, 1978b), governing the relationship between know-how and conceptualization, 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

$$C \leftarrow P \rightarrow C'$$

[1] Piaget's ideas about social construction are integral to his epistemological theory but less known than those about child's construction of logical thought.

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 the *reflective abstraction* and *constructive generalization*. Reflective abstraction is described as a two-fold process (Piaget, 1964): in the first place, it is a projection (transposition) to the plane of thought of the relations established in the plane of actions. Second, it is a reconstruction of these relations in the plane of thought adding a new element: the understanding of conditions and motivations. The motor of this process is called by Piaget the search of reasons of success (or failure). On the other hand, facing new problems presenting variations and similarities with the old ones causes a desequilibrium of students' cognitive structures which have to be transformed in order to attain a new equilibrium, making possible the construction of appropriate knowledge to solve the new situation. Once a particular method is understood, students' reasoning attempts to generalize what has been successfully constructed to all the situations, by means of inductive generalization where deductions or predictions are extracted from observations of the new objects. A process of inferences and reflections about the subject's actions or operations by means of constructive generalization gives raise to new methods (Piaget, 1978a, 1975; Jacques Montangero, 1997) and opens possibilities for constructing structures characteristic of the trans stage.

The table below summarizes the main points of the theory related to our methodology of research.

Table 1. A model of applying Piaget's theory

Methodology	Goals	Cognitive tools	Stages
individual interviews	actions → operations search of reasons detaching concrete cases	reflective abstraction automatization	intra-inter
instructional instances work groups similar problems	formal description operations → structures	reflective abstraction inductive and constructive generalization	inter-trans

3 Applying the theory

Our previous work focused on the first part of our studies, in which we conducted individual interviews applying the general law of cognition described above, in the manner of (Piaget, 1964), accounting for the passage from intra to inter stage. In Section 3.1 we include a summary of these investigations. The main goal of this paper is to describe the second part of our studies, where instructional instances were conducted and the students worked in groups. The main point is to illustrate, on the one hand, the way we introduced a formalism and encouraged the students to represent their descriptions of algorithms in it, and on the other hand, how students attempted to solve new problems presenting similarities and differences to those already solved, applying previously constructed concepts. This is presented in Section 3.2 using as an example, the study of the construction of knowledge about sorting algorithms.

3.1 Summary of investigations conducted through individual interviews

This section includes a summary of our previous work (da Rosa, 2010, 2007, 2005, 2002, 2004; da Rosa & Chmiel, 2012). The problems that students had to solve were instances of some of the problems studied in basic programming courses (sorting, searching, counting) and the objects were instances of data structures (a paper-dictionary, numbered cards, words). All students succeeded in solving the problem in the plane of actions and the questions were aimed to obtain accurate descriptions of what they did and why it worked, as a first step towards conceptualization. Further, the students were encouraged to derive a general solution for the problem (detached from concrete cases) by means of teaching to a robot (played by the teacher) to do the task.

In the example used here, a bag containing an undetermined amount of numbered cards was given out to the students. These numbers were not necessarily consecutive and were not repeated. Students were asked to take cards from the bag, one by one, and to order them in an upward sequence on the table. A set of questions was elaborated for the interviews which were posed once students have solved the problem in the plane of actions. Throughout the interview, it was possible to pose new questions depending on the answers provided by the students. The interviews pursued three goals:

– to conduct a process in which the students reflected about how they solved the problem. By means of reflective abstraction their actions were transformed into actions-in-the-plane-of-thought (concepts). This process was the source of knowledge of the repeating part of the algorithm. For the case of sorting the cards the actions were: pick up a card, compare numbers, insert the card in the right place, repeat the actions, finish.
– to apply Piaget's ideas about the role of "searching for the reasons of success" in the conceptualization: the constant motor driving the subject to complete or to replace the observables of facts, by deductive or operative inferences is the search of *reasons* for the obtained result (Piaget, 1964, 1978b). We applied this principle by making the students comprehend that the reasons of success lie both in their actions and in the modifications of the objects. For the case of sorting the cards, in each repetition, a card was inserted in a partially sorted row and the number of cards in the bag decreased (until the bag was empty). This process was the source of knowledge of the base case, the invariants of the algorithm and its relationship with data structures.
– to help students to go from particular cases towards a general algorithm. We found that introducing *automatization* is of great help because the students have to strive to give general descriptions to a robot (played by the teacher) which otherwise does not understand the instructions. A set of primitive operations was given and the students had to design a list of instructions to make the robot do the task.

In the following a summary of results from students' interviews for the case of sorting the cards is presented.

Towards to know how
The first descriptions of the students about how they sorted the cards clearly demonstrate that their thought was in the periphery (P in the diagram of the general law of cognition in Section 2), in other words, they were concentrated on the result: asked about **how** they did, they answered **what** they did (*"I sorted the cards in increasing order ..."*). The goal of the questions was that students explicitly mentioned the actions composing the method, as accurately as possible. For instance, almost all students said something like *"I compared the card I picked up from the bag with all the cards on the table ..."* that actually corresponded to the case in which the picked card was greater than all the cards in the row. Otherwise, they compared just to find the right place for the picked card[2]. One of the goals of the questions was to help the students become better able to describe accurately how they managed to insert a card into a partially ordered row, realizing that they compared only until a certain result of the comparison occurred.

Towards to know why
Further questions were related to the search of reasons for success: on the one hand, the existence of a base case (or several) (in this example, the bag became empty) and on the other hand, the invariant (or several), (in this example, all partial rows were sorted). The questions posed to students were like the following *"Is it always possible to construct a row in this way, in a finite time? In other words, do we always finish the task with a ordered row on the table? Why?"*

According to the theory the answers can be classified as:

– the reasons lie with the objects: *"... because they are numbers"* or *"... because of the order of numbers"*,
– the reasons lie with the actions: *"... because of my systematic actions"* or *"... because I do always the same"*.

To make students realize that the reasons for success lie *both in their actions and in their modifications of objects*, these kinds of question were posed:

– for the first type of answer:
"If you are asked to sort cards with letters in alphabetic order, should you change your method? What about sorting objects of different sizes?"
– for the second type of answer:
"Imagine you are asked to take one card at a time from the bag, and to set them on the table, one after the other. Would you have a sorted row? Observe that what you are doing is also systematic."

Although students answered correctly, none of them gave a satisfactory explanation of the existence of a base case as the reason for success. In previous work (da Rosa, 2010, 2007, 2005) we found, as well as other authors (Haberman & Averbuch, 2002; Velazquez, 2000), that it is significantly difficult to comprehend the *base case* (or base cases). To help students with this difficulty in an effective way, *they themselves have to experience the need for the existence of a base case (or more)*. To do that, we asked the students to use another method by which the bag was never emptied: *"Take from the bag one card at a time, and write down the numbers of the cards you took - in upwards sequence - on a piece of paper. Toss the cards back into the bag. Do you think that you will be able to finish by using this method?"*

This strategy was effective: after using this method, all the students immediately became aware of the sequence of states of the bag, *each time getting smaller* until it became an empty bag, as a reason for success.

[2] This is the source of a common programming mistake in which students use a for loop to access an element of a structure in cases where a while loop is more adequate.

Observe that by means of reflective abstraction the students reconstructed what they do in the plane of actions in the plane of thought, where the relationships become enriched by the comprehension of conditions and motivations (how and why). To finish, each student was asked to write down both the problem and the algorithm, step by step. Taking those descriptions as a starting point, they were asked to teach a robot to do the task, as described in the following section.

A general sorting algotithm: the role of automatization

In (Piaget & coll., 1963) Benjamin Matalon published a chapter entitled *Recherches sur le nombre quelconque*, in which he analyzed the relation between the generic element concept and the reasoning by induction, which requires a proof that P(n) $\rightarrow$ P (n+1) for a generic number n and a given property P. Matalon worked with the structure of natural numbers, stating that it is necessary to abstract away all the particular properties of n, except the property of it being a number, that is, an element belonging to the series of natural numbers.

Matalon addressed the problem of making the leap from particular cases to general ones and introduced variables for their reference. For example, he explained that Fermat made his arithmetic demonstrations using a particular number, but treating it as a generic number, for example, the number 17. If none of the specific properties of the number 17 were involved in the demonstration, then the demonstration could be considered valid for all numbers. Matalon added that in geometry, when a property is to be proven and the statement is "given a generic triangle" a particular triangle is drawn, avoiding right triangles, equilateral triangles or isosceles triangles, and not involving any particular properties of the triangle in the demonstration of the property. Among other things, Matalon concluded that to construct the concept of the "generic" element, it would be necessary to perform a generic action, that is, the repeated action to build a generic element. We applied these results interpreting *the generic action* that Matalon mentioned as the automatic version of an algorithm, that is to say, *a program.*

In this section we describe the process of going from a correct description of an instance of the method, to a general algorithm as the first step of program construction. Here is an example of a starting description: *"To order the numbers that are in the bag on the table, I did the following: First, I took a card and placed it on the table, then I took another card from the bag. If the second card I took is of a higher value than the card on the table, I placed it on the right side of the card already on the table, while if it is of a lower value, I placed it on the left side. Then I continued the process in the same way, I took more cards and ordered them with the cards that were already ordered on the table as reference. For example, if I have these numbers: -2, 0, 5 and 8, and I pick number 1, I place it between 0 and 5, since 1 is less than 5 and greater than 0."* Starting from that we asked the students to give oral instructions to a robot, played by a teacher, who tried to construct the ordered row by following the instructions. Our goal was to confirm the role of *automatization* to help students to detach themselves from particular cases, according to our interpretation of Matalon's results.

The students read the instructions and the robot acted, until the sentence *Then I continued the process ...* which the robot was not able to follow. Further questions were posed to encourage the students to give more precise descriptions, until they came to a description similar to *"What we are going to do is compare the card we want to insert with each card on the row, starting from the first card. When we find a card of higher value, then we insert our card before it. We do that until there are no more cards in the bag"*. That instruction the robot was able to perform (the robot was assumed to know how to compare the numbers of the cards).

The type of knowledge briefly described above is involved in the construction of concepts **before** formal knowledge, that is, no formalism intervenes (except for natural language). The introduction of a formal language was done in a group class as described in the next section.

3.2 Working groups: thematized knowledge or formalization

The main goal of this section is to describe our investigation about the process of formalizing constructed knowledge. We interpret this as the means to put into correspondence mental constructions (concepts) with some universal system of symbols or formal language.

Often, formal definitions or descriptions are presented to the students without taking into consideration their non-formal knowledge, that is, with no connection with what students already know about the subject (da Rosa, 2004). By contrast, in our approach the formalism introduced by the teacher is considered as a new object that students need to interact with in a process governed by the general law of cognition. As pointed out by Piaget (Piaget & García, 1980), the process of transiting the stages of the triad (intra-inter-trans) is of a dialectic nature, that is, the construction of formalized knowledge traverses its own stages. That means that an interaction between the students and formal representation of the objects (in our example rows, bag, cards) and their methods (inserting, comparing, deciding) has to take place. Our starting point is what students have said (and written) to teach a robot: *"What we are going to do is compare the card we want to insert with each card on the row, starting from the first card. When we find a card of higher value, then we insert our card before it. We do that until there are no more cards in the bag"*. Our goal is that the students succeed in transforming that description into an algorithm using given pseudo code and primitives.

We adopted a notation suitable for expressing rows as lists, similar to that used in functional programming (empty row as [ ], non-empty row as first:tail (where first is the first element of the list, tail is the rest (a list) and

: is the constructor function for lists), or with elements between brackets separated by commas (S.Thompson, 1999)). From the interviews we have learnt that the description of the action to place a card in a given non-empty row is relevant. Consequently, we decided to work on this part of the algorithm first. The main point here is that students understand that if we call this action **insert t in row** then if the value of t is not lower than the value of the first card in the row, then the first card remains the same and with t and the tail it is necessary to carry out the same action, that is to say, the result is **first : insert t in tail**. In the case that the picked card is greater than all the cards in the row, it is placed at the end, which means that it is inserted in an empty row (the tail). The students are asked to fill some tables as in Table 2 below. Observe that in this way, understanding the repetition of actions and the new objects in each repetition is straightforward.

Table 2. Inserting a card in a non-empty row

row	picked card	comparison	first of current insertion	tail	result
[-2,0,1,4]	3	-2 < 3	-2	[0,1,4]	-2 : (insert 3 in [0,1,4])
		0 < 3	0	[1,4]	-2 : 0 : (insert 3 in [1,4])
		1 < 3	1	[4]	-2 : 0 :1 : (insert 3 in [4])
		4 > 3	3	[4]	-2 : 0 : 1 : 3 : 4 : [ ]

Students were given a list of primitive actions that the robot (played by one of the students of the group) understands. In this first step towards formalization, the repetition of actions is modeled by "go to" instructions:

- decision: if something do a else do b (a and b are actions)
- compare cards: t < first, first < t
- insert card t in a row: t:[ ] or t:first:tail or first: (insert t in tail)
- update (a variable becomes a new object): row ← tail
- sequence of actions: {action$_1$, action$_2$, ..., action$_n$}
- go to an action of the sequence: go to n (n is a natural number)
- a = b (the robot is able to verify if an object a is equal to an object b)

The students were allowed to go back to making the actions with the bag and the cards as many times as they needed. They attempted several lists of instructions, until the robot student succeeded, in approximately 15 minutes. The following primitives were added to the list and students are asked to modify and complete the instructions for the robot to be able to order the cards from the bag on the table:

- pick a card from the bag
- decide if the bag is empty
- finalize

In 40 minutes aproximately, all groups produced a list similar to the following:

```
0 pick a card t from the bag
1 if row = [] then {t: [], go to 2}
  else if t < first then {row <- t:first:tail, go to 2}
       else first : {row <- tail, go to 1}
2 if the bag is empty then finalize
  else go to 0
```

The final stage of the study included introducing an iteractive sentence in the manner of Pascal. We handed out to each group of students, a collection of small pieces of paper with *'if then else'* and *'while'* sentences, with suggested indentation in order to guide them. We explained the semantics of the sentences and the references to objects in each iteration (the rows on the table are H and H1, and the picked card and first are T and T1 respectively).

The interesting point to observe here is that students had to be able to construct *another description of the same algorithm* (the "while" sentences instead of the "go to" description). This means that the students needed to find a solution to a situation that had some similarities and differences with the one that they had already solved, by means of the cognitive instrument of *generalization*, both phases of which Piaget described (Piaget, 1978a): inductive generalization and constructive generalization. In the first phase, the individual transfers to new objects what has been previously constructed, without taking into account the transformations of the knowledge required for the conditions of the new situation. In this case, students tried to use the relation between cards ($t < first$) and the termination condition (*if the bag is empty*) of the "go to" version in the "while" version. (Observe that in the "while" version, these are *"T is greater than T1"* (that is, $t > first$) and *"the bag is not empty"* respectively).

Further instances in which the robot-student tried to follow the while version of the algorithm, resulted in correct positions for each piece of paper, as shown in Figure 1 below (in Spanish, an English version is included in Appendix A). That means that, because of constructive generalization, students understood the new conditions giving rise to structures of the trans stage, thus opening the possibility of studying new elements, such as other sorting algorithms and more formal representations, as well. This constitutes the focus for further work.

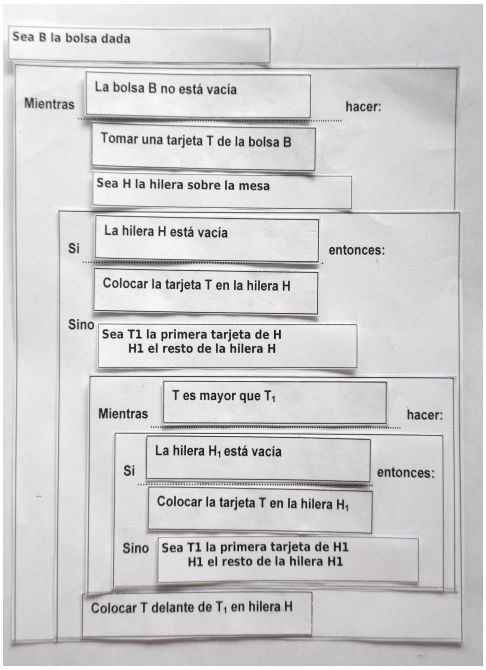

Fig. 1. A version of the insertion sort algorithm

4 Related Work

Although the amount of research in Computer Science Education (CSE) has grown significantly, in many countries the didactics of informatics is not considered as part of Computer Science (CS), including Uruguay where our work develops. The training of teachers of CS, for example, emphasizes content issues (which must be treated seriously) and neglects issues of Pedagogical Content Knowledge (Saeli, 2012; Hubwieser, 2013) of the concepts of CS. Few teachers of CS can answer with solid theoretical foundations, questions as to what, for whom, why and how to teach CS. Mathematics education is in this sense a model not only for having introduced the notion of specific didactics, that is, education of a discipline is part of that discipline, but also for the importance which it gives to the theoretical foundation in research. We share the concern of some authors (Holmboe et al., 2001; Zendler & Spannagel, 2008; Winslöw, 2005) about the need to conduct a significant amount of educational research in order for CSE to accomplish the same level as mathematics education.

Related works can be analyzed from several points of views, some of which we have included in previous work (for instance: learning of algorithms by novices, relationship between algorithms and data structures, recursion and induction, discrete mathematics and programming). Here we include some related work that shares Piaget's theory as a theoretical framework with us. On the one hand, the question of how students learn is deeply investigated in the didactics of mathematics by mathematic academics, giving rise to theories and methodologies based on Piaget's work (Dubinsky, 1996; Sánchez-Matamoros, García, & Llinares, 2006; Clark, 1997; Sadovsky, 2000). Among these, the Theory of Situations (Sadovsky, 2000) is closely related to our work as explained in Section 4.1. On the other hand, Section 4.2 includes recently developed neo-Piagetian work, research that considers advances in neuroscience and influences constructivist theories. Finally, we add a few words about related work referring to the content of our research (basic algorithms and data structures).

4.1 Brousseau's theory and our work

Brousseau's theory of didactical situations is one of the ways of thinking about mathematics as a subject for teaching. The theoretical bases are described in the following quotation (taken from the English translation of Brousseaus book "Theory of didactical situations in mathematics" (Brousseau, 1997) in (Winslöw, 2005)):

"Mathematicians don't communicate their results in the form in which they discover them; they reorganize them, they give them as general a form as possible. Mathematicians perform a "didactical practice" which consists of putting knowledge into a communicable, decontextualized, depersonalized, detemporalized form.
The teacher first undertakes the opposite action; a recontextualization and a repersonalization of knowledge. She looks for situations which can give meaning to the knowledge to be taught. But when the student has responded to the proposed situation (...) she will, with the assistance of the teacher, have to redepersonalize and redecontextualize the knowledge which she has produced so that she can see that it has a universal character, and that it is re-usable cultural knowledge."

In the 60s, when Brousseau was studying mathematics, he studied cognitive psychology with one of Jean Piaget's collaborators, Pierre Greco. Not surprisingly, Brousseau applies the central hypothesis of genetic epistemology by Jean Piaget as a model framework for the production of knowledge. He argues that mathematical knowledge is constituted essentially from recognizing, addressing and resolving problems, which are generated in turn by other problems.

The constructivist concept leads Brousseau to postulate that individuals learn as a result of their adaptation to an environment that is a factor of contradictions, difficulties and imbalances. He suggests a model to teach and learn based on two basic types of interactions:

a) the interaction of the student with a problem that offers resistance and feedback on the mathematical knowledge used, and, b) the interaction of the teacher with students, in terms of their interaction with mathematical problems.

From this model, he postulates that an "environment" is required, with a didactical purpose: he defines an *adidactical situation* as a situation in which the student is allowed to use some knowledge to solve a problem (the environment) "without appealing to didactical reasoning [and] in the absence of any intentional direction [from the teacher]" (Sadovsky, 2000). The concept of an environment includes an initial mathematical problem which the individual copes with, and a set of mathematical relations, which are modified in the course of the subject's actions during his production of knowledge, thus transforming his cognitive reality. The interaction of the teacher with the student is related to the student's interaction with the environment and it is called *didactical situation* by Brousseau.

Brousseau's model has points in common with ours: on the one hand, adidactical situations correspond in our model to instances in which the teacher does *not* intervene and students interact with an instance of a problem (Brousseau's environment) or discuss in groups. The didactical situations, on the other hand, correspond to the instructional instances, where the interventions of the teacher are based on the knowledge constructed by the students through their interaction with the environment (the problem).

A significant difference arises, however. While Brousseau's model is used in classrooms to the learning of mathematics, we have not had enough institutional support to do the same. One of the most important consequences of that is the difficulty of training the teachers.

4.2 Neo-Piagetian theory and our work

Several teaching investigations regarding basic algorithms have been performed within the frameworks of constructivism and mental model studies (E.George, 2000; Tamar, Dalit, & Paz, 1999; Ben-Ari, 2001). The central matters of these studies are implicitly related to tenets of the epistemological theory of Jean Piaget. Recent studies following neo-Piagetian theory consider the advances in neuroscience and whether these can influence constructivist learning theories. Neo-Piagetian theory is deemed as a suitable theoretical framework for some authors (Lister, 2011; Murphy, 2012; Falkner, Falkner, & Vivian, 2013; Gluga, 2012). Neo-Piagetian theory of cognitive development is a branch of cognitive psychology. It is influenced by the information-processing paradigm, the idea of linguistic theorists regarding the specific domain of cognition as well as the advances in neuroscience. Although several authors focus on different considerations, encompassed under the term "neo-Piagetian theories", they all have a common starting point, which are Piaget's ideas from his earliest studies regarding children's progress in cognitive development by stages (genetic psychology). Neo-Piagetian theory extends and complements the studies referred to. Genetic and cognitive psychology have almost always been related. Close collaborators of Piaget have continued and extended research on genetic psychology, reinforcing its bonds with cognitive psychology (Inhelder & Cellérier, 1992).

On the other hand, sustained by historical investigations, Piaget and García improved some aspects of Piaget's theory (García, 2000, 1997; Piaget & García, 1980). They synthesized results from psycho-genetic studies *and science history*. Our work follows these results, including the application of a general mechanism, from the analysis of objects (intra-stage or instrumental knowledge) to the analysis of relations or transformations between objects (inter-stage or conceptual knowledge). This leads to the analysis of the construction of structures (trans-stage or thematized knowledge), a characteristic of the development of scientific theories and school subjects. In (Sánchez-Matamoros et al., 2006) the authors present a similar study in mathematics education, for the concept of the derivative of a function.

Finally, some words in relation to the content of our work (basic algorithms and structures). In (Zendler, 2013; Zendler & Spannagel, 2008; Zendler, McClung, & Klaudt, 2012) the authors indicate as examples of content concepts central to CS, *problem and algorithm* among others, and as central process concepts, *categorizing and classifying*. In our approach we consider that the source of knowledge for these concepts are basic actions that the students are familiar with, such as defining correspondences, sorting, searching and comparing. We explain the process by which students are able to transform this knowledge into conceptual and formal (thematized) knowledge, as well as the means by which teachers support this task.

5 Summary and further work

Piaget's epistemological theory explains the construction of knowledge giving detailed descriptions of the processes, mechanisms and cognitive instruments intervening. The theory is suitable for most of scientific domains and all levels of knowledge. In this paper we present an application of that theory to investigate the construction of Computer Science concepts by students entering University or in the last years of High School. We have built this application over several years of study of learning different basic algorithms and corresponding data structures. Our approach can be taken as a model for doing research in Computer Science Education and also for designing instructional guidelines.

The questions of our research are related to ways of helping students to use their cognitive resources to attain higher levels of knowledge. One of the most important theoretical ideas about knowledge construction is the categorization of different stages, called by Piaget and Garcia in "Psychogenesis and the History of Sciences" as the triad intra-inter-trans. Accordingly, our research on the construction of concepts focuses on how students transform their instrumental knowledge (intra stage) into conceptualized knowledge (inter stage) and how this becomes an academic formalization (trans stage). The latter involves on the one hand, the construction of correspondences between concepts and expressions of some formal language as a universal system of symbols and on the other hand, the application of knowledge by students to solve new problems with differences and similarities to those already solved. Describing this process has been the goal of this paper, while the construction of knowledge before any formalization (except the natural language) is only briefly presented here as it has been described in previous papers, including excerpts from interviews.

At all stages of our research, results described by Piaget and collaborators in their work are indicated. The main points on which our strategies were effective are:

- students succeed in conceptualizing their know-how, by means of reflecting about how they solve a problem and why the solution works, especially when they themselves experience the need for a base case (or several). The evidence is given by students' correct descriptions in natural language of their algorithmic solutions and of the reasons for their success (Section 3.1)
- how to surmount one of the main difficulties, that is, that students' thought detaches from particular cases and makes the leap to general solutions, by means of introducing automatization (Section 3.1).
- introducing a formal language as a new object about which the students have to construct knowledge using the same principles as any other object. That means starting the interaction in the plane of action, allowing students to work with particular cases of formal expressions to facilitate the transition to a general algorithm in the formal language (Section 3.2).
- to face students with problems that encourage them to apply their knowledge taking into account necessary transformations. This is a first step to the construction of structures of the trans stage and the beginning of the comprehension of algorithms as classes of methods (sorting, searching, counting, etc) (Section 3.2 and further work).

Three lines of research are relevant as further work: on the one hand, to elaborate pedagogical proposals and didactic guidelines based on our model, which can be useful to teachers in supporting their researches and practices, and a way of getting institutional support as well. On the other hand, to study the construction of elements of the trans stage, that is to say, of classes of algorithms and formal representations. This includes for instance, other sorting algorithms, that could be compared with each other, and recursive or iterative implementations. Finally to complement this research, we need to investigate the historical evolution of the concepts of iteration, induction and recursion, that are the higher forms of themathized (formalized) knowledge of ways of solving problems by repeating actions. According to the methodological analysis of Piaget's theory, a critical historical analysis of those concepts teaches about the construction of knowledge and can therefore cast light on the learning process.

6 Acknowledgements

I thank Ben du Boulay for correcting the English. The comments of the referees are gratefully acknowledged.

References

Ambrosio, A. (2014). Exploring Core Cognitive Skills of Computational Thinking. *Proceedings of the 25th Annual Psychology of Programming Interest Group Workshop, PPIG, University of Sussex, UK*.

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching, Vol. 20, Issue. 1, pp 45-73*.

Bradshaw, P., & Woollard, J. (2012). Computing at School: An Emergent Community of Practice for a Re-Emergent Subject. *In: International Conference on ICT in Education*.

Brousseau, G. (1997). *Theory of didactical situations in mathematics*. Dordrecht: Kluwer.

Budd, T. A. (2006). An Active Learning Approach to Teaching the Data Structure Course. *ACM SIGCSE'06, Houston, Texas, USA*.

Clark, J. (1997). Constructing a Schema: The Case of the Chain Rule? *Journal of Mathematical Behavior ISSN 0364-0213*.

da Rosa, S. (2002). The Role of Discrete Mathematics and Programming in Education. *Workshop: Functional and Declarative Programming in Education*.

da Rosa, S. (2003). Psychogenesis of the Concept of Recursion. *CIESC 2003, Congreso Iberoamericano de Educacion Superior en Computacion, La Paz, Bolivia*.

da Rosa, S. (2004). Designing Algorithms in High School Mathematics. *Lecture Notes in Computer Science, vol. 3294, Springer-Verlag*.

da Rosa, S. (2005). *The learning of recursive algorithms and their functional formalization.* Website. (www.fing.edu.uy/darosa)

da Rosa, S. (2007). The Learning of Recursive Algorithms from a Psychogenetic Perspective. *Proceedings of the 19th Annual Psychology of Programming Interest Group Workshop, Joensuu, Finland*, 201–215.

da Rosa, S. (2010). The Construction of the Concept of Binary Search Algorithm. *Proceedings of the 22th Annual Psychology of Programming Interest Group Workshop, Madrid, Spain*, 100–111.

da Rosa, S., & Chmiel, A. (2012). A Study about Students' Knowledge of Inductive Structures. *Proceedings of the 24th Annual Psychology of Programming Interest Group Workshop, London, UK*.

Dowek, G. (2005). Quelle informatique enseigner au lycée? *Bulletin de l'APMEP, nr. 480*. Website.

Dowek, G. (2013). L'enseignement de l'informatique en France, Il est urgent de ne plus attendre. Website. (Rapport de l'Académie des Sciences: www.academie-sciences.fr/activite/rapport/rads_0513.pdf)

Dubinsky, E. (1996). A Framework for Research and Curriculum Development in Undergraduate Mathematics Education. *CBMS Issues in Mathematics Education, 6, 1-32*.

E.George, C. (2000). Experiences with Novices: The importance of Graphical Representations in Supporting Mental Models. *In A.F.Blackwell and E.Bilotta (Eds). Proc. PPIG 12, pp 33-44*.

Falkner, K., Falkner, N., & Vivian, R. (2013). Neo-Piagetian Forms of Reasoning in Software Development Process Construction. *First International Conference on Learning and Teaching in Computing and Engineering (LATICE)*.

Ferreiro, E. (1996). *The acquisition of cultural objects: the case of written language.* Prospects, Vol 26, Issue 1.

García, R. (1997). *La Epistemología Genética y la Ciencia Contemporánea.* Barcelona, Gedisa, 325 pp. ISBN 84-7432-645-1.

García, R. (2000). *El Conocimiento en Construcción.* Barcelona: Gedisa, 252 pp. ISBN 9 788474-328110.

Gluga, R. (2012). On the Reliability of Classifying Programming Tasks Using a Neo-Piagetian Theory of Cognitive Development. *Proc. of the International Workshop on Computing Education Research, ICER'12. ACM 2012*.

Gomes Anabela, M. A. J. (2007). Learning to program - difficulties and solutions. *International Conference on Engineering Education ICEE 2007*.

Haberman, B., & Averbuch, H. (2002). The Case of Base Cases: Why are They so Difficult to Recognize? Student Difficulties with Recursion. *Proceedings of the 7th annual conference on Innovation and technology in computer science education, ITiCSE'02. ACM 2002*, 84–88.

Holmboe, C., McIver, L., & E.George, C. (2001). Research Agenda for Computer Science Education. *In G.Kadoda (Ed). Proc. PPIG 13, pp 207-223*.

Hubwieser, P. (2013). Towards a Conceptualization of Pedagogical Content Knowledge for Computer Science. *ACM 978-1-4503-2243-0/13/08 (ICER 2013)*.

Inhelder, B., & Cellérier, G. (1992). *Le Cheminement des découvertes de l'enfant*. Delachaux & Niestlé.

Jacques Montangero, D. M.-N. (1997). *Piaget, Or, The Advance of Knowledge*. Psychology Press.

Linda Mannilaa, T. S., Mia Peltomäkia. (2007). What about a simple language? Analyzing the difficulties in learning to program. *International Conference on Engineering Education ICEE 2007, pp 211-227*.

Lister, R. (2011). Concrete and Other Neo-Piagetian Forms of Reasoning in the Novice Programmer. *13th Australasian Computer Education Conference (ACE 2011)*.

Moström, J. E. (2011). A Study of Student Problems in Learning to Program. (Department of Computing Science Ume university, SE-901 87 Ume, Sweden, ISBN 978-91-7459-293-1)

Murphy, L. (2012). Ability to 'Explain in Plain English' Linked to Proficiency in Computer-based Programming. *Proc. of the International Workshop on Computing Education Research, ICER'12. ACM 2012*.

Nickerson, H. (2013). The Zones of Proximal Flow: Guiding Students Through a Space of Computational Thinking Skills and Challenges. *ACM 978-1-4503-2243-0/13/08 (ICER 2013)*.

Peyton Jones, S. (2013a). Bringing Computer Science Back into Schools: Lessons from the UK. *SIGCSE'13*.

Peyton Jones, S. (2013b). Computing at school in the UK: from guerrilla to gorilla. *Under review by CACM (v4)*.

Piaget, J. (1964). *La prise de conscience*. Presses Universitaires de France.

Piaget, J. (1975). *L'équilibration des Structures Cognitives, Problème Central du Développement*. Presses Universitaires de France.

Piaget, J. (1977). Genetic Epistemology, a series of lectures delivered by Piaget at Columbia University, translated by Eleanor Duckworth. *Columbia University Press*.

Piaget, J. (1978a). *Recherches sur la Généralisation*. Presses Universitaires de France.

Piaget, J. (1978b). *Success and Understanding*. Harvard University Press.

Piaget, J., & coll. (1963). *La Formation des Raisonnements Recurrentiels*. Presses Universitaires de France.

Piaget, J., & García, R. (1980). *Psychogenesis and the History of Sciences*. Columbia University Press, New York.

R.Page, & Gamboa, R. (2013). How Computers Work: Computational Thinking for Everyone. *Programming in Education 2012 (TFPIE 2012). EPTCS 106*, 1–19.

Sadovsky, P. (2000). La Teoría de Situaciones Didácticas: un marco para pensar y actuar la enseñanza de la Matemática. (http://www.buenosaires.gob.ar/areas/educacion/cepa/teoria_situaciones.pdf)

Saeli, M. (2012). Teaching Programming for Secondary School: a Pedagogical Content Knowledge Based Approach. *Dissertation, Technische Universiteit Eindhoven.*.

Sánchez-Matamoros, G., García, M., & Llinares, S. (2006). El Desarrollo del Esquema de Derivada. *Enseñanza de las Ciencias*, *24*, 85–98.

S.Thompson. (1999). *Haskell. The Craft of Functional Programming, second edition.* Addison-Wesley, ISBN 0-201-34275-8.

Tamar, L., Dalit, L., & Paz, T. (1999). Implementing Constructivist Ideas in a Functional Programming Curriculum for Secondary School Students. *Workshop Functional and Declarative Programming in Education*.

Tina Götschi, V. G., Ian Sanders. (2003). Mental models of recursion. *ACM 1-58113-648-X/03/0002*.

Velazquez, J. (2000). Recursion in Gradual Steps (Is Recursion Really that Difficult?). *Proceedings of the thirty-first SIGCSE technical symposium on Computer science education, SIGCSE'00. ACM 2000*, 310–314.

Wing, J. (2000). Computational thinking. *CACM*, *49*, 33–34.

Winslöw, C. (2005). *Didactics of Mathematics - The French Way. Texts from a Nordic Ph.D.-Course at the University of Copenhagen.* Center for Naturfagenes Didaktik, University of Copenhagen. (http://www.ind.ku.dk/publikationer/inds_skriftserie/2005maj4F/4F.pdf)

Youniss, J., & Damon, W. (1997). *Social construction in Piagets Theory.* In H. Beilin and P. Pufall (comp.), Piagets Theory: Prospects and possibilities. Hillsdale, N.J.: Erlbaum.

Zendler, A. (2013). Semantic categorization of content and process concepts relevant to computer science education. *International Journal of Research Studies in Computing*, *2*, 3–10.

Zendler, A., McClung, O., & Klaudt, D. (2012). Content and process concepts relevant to computer science education: A cross-cultural study. *International Journal of Research Studies in Computing*, *1*, 27–47.

Zendler, A., & Spannagel, C. (2008). Empirical Foundation of Central Concepts for Computer Science Education. *ACM Journal on Educational Resources in Computing*, *8*.

A English version of Figure 1

```
let B be the bag
while B is not empty do
    take a card T from B
    let H be the row on the table
    if H is empty then
       put T in H
    else
       let T1 be the first card of H
           H1 be the tail of H
       while T is greater than T1 do
         if row H1 is empty then
            place T in H1
         else
            let T1 be the first card of H1
                H1 be the tail of H1
       place T in front T1 in H
```