Jl. of Computers in Mathematics and Science Teaching (2005) **24**(3), 255-272

Reducing Abstraction When Learning Graph Theory

ORIT HAZZAN
Technion-Israel Institute of Technology
Israel
oritha@techunix.technion.ac.il

IRIT HADAR
The University of Haifa
Israel
hadari@is.haifa.ac.il

This article presents research on students' understanding of basic concepts in Graph Theory. Students' understanding is analyzed through the lens of the theoretical framework of reducing abstraction (Hazzan, 1999). As it turns out, in spite of the relative simplicity of the concepts that are introduced in the introductory part of a traditional Graph Theory course, some students exhibit ways of thinking that indicate reduction of the level of abstraction. The importance of this study is derived from the importance of graph algorithms in any Computer Science curriculum and the centrality of the concept of abstraction in Computer Science education.

INTRODUCTION

Graph Theory is one of the basic courses in any computer science curriculum. It "has long become recognized as one of the more useful mathematical subjects for the computer science student to master" (Even, 1979, p. V). In this article the focus is placed on the introductory part of Graph Theory that addresses the following topics: basic definitions, different kinds of graphs and their properties, graph traversing, shortest paths, trees (binary

trees, directed trees, spanning trees, ordered and positional trees) and flow in networks.

The above mentioned concepts are usually learned in an introductory Graph Theory course by mathematics and computer science undergraduate students. Depending on the student population, different aspects of the subject are emphasized. For example, in mathematics programs the theoretical aspects of Graph Theory may be emphasized whereas in computer science undergraduate programs it may be the algorithmic nature of Graph Theory that is emphasized. Naturally, a mixed approach can be adopted as well. This mixed approach is applied in the course that is addressed in this article.

This article analyzes undergraduate students' understanding of the above basic concepts of Graph Theory. The analysis is performed through the lens of reducing abstraction. The theme of reducing abstraction is a theoretical framework that has been used up to now for explaining student conception of different undergraduate mathematics and computer science topics, such as abstract algebra, computability and data structures topics (Hazzan 1999, 2003A, 2003B). This framework suggests several mental processes, which can be interpreted as a reduction of the level of abstraction, that students employ unconsciously while trying to cope with problem solving situations. In this article we illustrate the application of this framework with respect to students' conception of Graph Theory concepts by looking at two ways by which abstraction is reduced: process perception and specification. This theoretical framework of reducing abstraction is further elaborated in the section in which the data is analyzed.

The first section describes the educational research that has been done so far on the learning and teaching of Graph Theory. The second section describes the background of our research. The third section examines the conception of Graph Theory through the theoretical framework of reducing abstraction. First, within the framework of the process-object duality we illustrate student comprehension of concepts in Graph Theory as processes (as opposed to objects). Second, we illustrate student tendency to specify the concept with which they think either by considering a particular case when the situation they are faced with requires the consideration of a set of objects, or by relying on the visual aspect of graphs. The last section concludes with some pedagogical remarks and suggestions for future research.

RESEARCH ABOUT THE TEACHING AND LEARNING OF GRAPH THEORY

According to our literature review, the community of computer science education research gives little attention to the analysis of students' under-

standing of concepts in Graph Theory. As our literature review reveals, most of the literature concerning the learning and teaching of Graph Theory deals with the development of visualization tools that aim at supporting the learning process (cf. for example Khuri and Holzapfel, 2001; Hansen *et al*., 2003; Hamilton-Taylor and Kraemer, 2002; Baker *et al*., 1999).

One research work that does analyze students' understanding of Graph Theory concepts is described in Dagdilelis and Satratzemi, 1998. In this article, the authors divided students' difficulties in Graph Theory algorithms into three categories. The first category addresses difficulties related to the recognition of the details of a given algorithm. More specifically, even in cases in which students exhibit an understanding of how an algorithm works in general and are able to follow a schematic representation of its application on a graph, they may face difficulties in understanding the details of the algorithm[1]. The second category addresses students' difficulties in understanding the meaning of the intermediate stages and the results of an algorithm. The third category looks into the difficulties caused by the complexity of the programming languages used for the implementation of graph algorithms. The authors note that common programming languages are not suitably designed for the implementation of graph algorithms, and are not suited to the simplicity and shortness that characterize some of the graph algorithms. In other words, while the execution of some Graph Theory algorithms is trivial when done by hand, their execution becomes complex when expressed in terms of a programming language.

We find it interesting to explore student conception of Graph Theory for several reasons. First, the basic definitions in Graph Theory are quite simple. One reason for this is that, unlike groups (introduced to students in abstract algebra courses) or recursive languages (learned in computability courses) for example, *graphs* can be easily visualized. However, as it turns out and illustrated in this paper, it may be the relative simplicity of the introductory part of Graph Theory that enables us to identify subtle observations in student conception of the relevant concepts. Second, *algorithms* play a central role in this course. In a similar way to the concept of function, algorithms have a dual nature. On the one hand, algorithms can be executed and can be treated as processes; on the other hand, algorithms can be viewed as objects with properties, such as complexity. As illustrated in this paper, this dual perspective yields some interesting results. Finally, as is mentioned above, up until now almost no research has been done on the topic by the computer science education community. In this respect, we hope that this article will be of value to the computer science educators who teach Graph Theory courses.

RESEARCH BACKGROUND

This section presents the course in which the data was gathered for our research and the research tools we used.

The Course

The data presented in this article were collected in a "Selected Algorithms in Graph Theory" course taken by 17 undergraduate students who studied for their high school teaching certificate in mathematics or computer science. The course was taught by the two authors on the basis of weekly meetings (each meeting for the duration of three hours). The main topics learned in the course are listed in the Introduction.

In order to help students in their mental construction of the relevant concepts, a significant portion of the course is based on active learning (McConnell, 1996; Jackowitz, Plishka & Sidbury, 1990; Flaningam & Warriner, 1987; Cote, 1987; Clements & Battista, 1990). Accordingly, the course is based on activities, discussions, and intuitive explanations before formal proofs are introduced. During active learning sessions students are requested to explore algorithms using the computer (with well-selected Java applets) and/or pen and paper. Sometimes, based on their exploration of such applet, they are asked to formulate the algorithm; on other occasions they are asked to use an applet in order to explore subtle details of an algorithm after the algorithm has been presented to them. Yet, in other cases algorithms and proofs are explored prior to the presentation. For example, before the proof of Cayley's theorem[2] is introduced, students are asked to suggest possible correspondences between words and trees. When such an exploration is conducted prior to the presentation of the correspondence on which the proof is based (even though it does not lead to the formulation of the required correspondence), it does give students a meaningful mental basis on which they can proceed to construct their knowledge.

This approach is generally based on the constructivist perspective. Constructivism is a cognitive theory that examines the nature of learning processes. According to this approach, learners construct new knowledge by rearranging and refining their existing knowledge (cf. Davis, Maher and Nodding, 1990; Smith, diSessa and Roschelle, 1993). More specifically, the constructivism approach suggests that new knowledge is constructed *gradually,* based on the learner's existing mental structures and on feedback that the learner receives from the learning environments. In this process, men-

tal structures are developed in steps, each step elaborating on the preceding ones. Of course, there may also be regressions and blind alleys. This process is closely related to the Piagetian mechanisms of assimilation and accommodation (Piaget, 1977).

One way to support such gradual mental constructions is to provide learners with a suitable learning environment in which they can be *active*. The above ways by which we applied active learning may provide learners with such an environment.

In order to guide students not to rely on the visualized or on the algorithmic aspect of graphs, we emphasize the analysis of graphs as objects (Dubinsky, 1991; Sfard, 1991) through their properties. Thus, for example, tasks that students are asked to solve treat algorithms both as processes and as objects and also address the interrelation between these two perspectives. Among different kinds of activities that can highlight this dual perspective, one kind of tasks asks to give examples of graphs that meet specific properties (see Figure 1 for an example of a "Give an example" task).

Task:

Construct three graphs, as is described in what follows, one for each section.

Construct a directed graph with at least 7 vertices and positive arc weights. Mark 2 vertices - *s* and *t* - so that Dijkstra's algorithm, when is executed on this graph in order to find the shortest path from *s* to *t*:

a. upon termination yields $\lambda(v) = \infty$ for exactly two vertices.

b. performs exactly two iterations.

c. changes $\lambda(t)$ three times.

Figure 1. Example of a "Give an example" task

Prior to the course design, we conducted a cognitive analysis of the course content. The aim of this analysis was to select the appropriate teaching methods, to set the research tools and to determine (at least tentatively) the theoretical framework within which students' understanding is analyzed. That analysis yielded that the framework of reducing abstraction (Hazzan, 1999) may be an appropriate framework for the analysis of student conception of Graph Theory concepts. The theme of reducing abstraction refers to problem solving situations in which students are unable to cope with mental

manipulations of the concepts appearing in a given problem. In order to cope with such situations, students unconsciously reduce the level of abstraction of the concepts involved. Further details of this theoretical framework are described in the section in which the research results are described.

Similar reasons to those which explain our interest in exploring students' understanding of concepts in Graph Theory enlighten the choice in this theoretical framework as well. First, one aspect of the theme of reducing abstraction deals with the process-object duality (Dubinsky, 1991; Sfard, 1991). As has been mentioned before this duality also exists in Graph Theory. Second, another aspect of the theme of reducing abstraction deals with abstraction as it is reflected in the complexity of the thought concept. Since Graph Theory deals with objects (graphs) that can be decomposed (into vertices and nodes) on the one hand, and may be grouped into sets of objects (for example, the set of binary trees) on the other hand, we found it interesting to observe how students move between the levels of abstraction that this perspective inspires. The research findings are based on these two aspects.

Research Tools

The research tools were mainly qualitative, though some quantitative data was gathered and analyzed as well. However, the quantitative analysis was limited since only 17 students participated in the course.

In order to gain a wide and varied picture on which to base our observations, the data were collected by using a variety of tools. Some of these sources were planned (observations in our class, analysis of students' homework assignments and tests); others were incidental (like occasional talks with students during office hours). Although not all of these sources are explicitly presented in the present report, they did help determine the directions and shape the ideas in the different stages of the study. In what follows we briefly explain each of the research tools.

Homework assignments: Keeping in mind that we may analyze our data through the lens of reducing abstraction, among the other questions we presented the students with two specific kinds of questions:

- **Processes – Object Duality:** In these exercises students are requested to carry out some or all the following activities: to execute an algorithm on different graphs (cf. Figure 2, Example 1), to analyze properties of algorithms, to compare algorithms, or to compose

algorithms (cf. Figure 2, Example 2). In other exercises, students are asked to give an example of graphs that meet specific properties (cf. Figure 2, Example 3).

- **Specification – Generalization Duality:** These exercises are formulated in general terms (cf. Figure 2, Example 4). They help us observe whether students remain in the general level, or whether, alternatively, they answer in terms of specific instances that belong to the family of objects presented in the question.

Example 1:

Execute DFS algorithm on a given graph, and find the resulting DFS tree.

Example 2:

Given an undirected graph *G(V, E)* with a weight function on the edges:

W: $E \rightarrow R^+$. For each path in the graph we define:

l(p) – the path length, that is, the number of edges of *p*.

w(p) – the path weight, that is, the sum of weight of the edges of the path.

For a pair of vertices $s,t \in V$ a shortest lightest path from *s* to *t* is a path *p* that satisfies the following: for all path *q* from *s* to *t*: *l(p)<l(q)* or (*l(p)=l(q)* and *w(p)<w(q)*).

a. Define an algorithm that finds a shortest lightest path from a given vertex *s* to any vertex *v* in the graph.

b. Construct a graph with at least 6 vertices and execute the algorithm you defined in (a). Outline all the algorithm stages.

Example 3:

Construct a graph with Euler's path that does not have a Hamilton Path.

Example 4:

For even *n*, write a frequency formula so that Huffman coding of *n* letters with these frequencies creates deterministically a tree of the following structure:

Figure 2. Examples of homework assignments

Tests: Two exams were given. The first test was based on standard questions and dealt with the following topics: basic definitions, Euler path, Hamilton path, order of functions, Breadth First Search (BFS), Dijksta algorithm, binary trees, binary search trees. The second exam was based on True/False questions (for which students were requested to explain their choice) and focused on the following topics: spanning trees, Cayley's theorem, directed trees (König theorem, Wang tiling problem), Depth First Search (DFS), ordered and positional trees and flow in networks.

Class observations: While the first author taught the course, the second author participated in all the lessons and documented discussions, students' questions, difficulties, interesting statements, and interactions that took place during the lessons.

REDUCING ABSTRACTION IN LEARNING GRAPH THEORY

Hazzan (1999) identifies several mental processes by which students reduce the level of abstraction. In this article we analyze students' perception of Graph Theory concepts by looking at the following two ways by which abstraction is reduced: process perception and specification. In the following two sub-sections these ways by which abstraction level is reduced are explained and an illustration is given of how the level of abstraction is reduced in the context of Graph Theory.

Process-object duality

In this sub-section the idea of reducing abstraction is reviewed based on the process-object duality suggested by some theories of concept development in mathematics education (Beth & Piaget, 1966; Dubinsky, 1991; Sfard, 1991, 1992). Theories that discuss this duality distinguish mainly between *process conception* and *object conception* of mathematical notions. Dubinsky (1991) captures the passage from the first conception to the second one as a "conversion of a (dynamic) process into a (static) object". *Process* conception implies that one regards a mathematical concept "as a potential rather than an actual entity, which comes into existence upon request in a sequence of actions." (Sfard, 1991, p. 4). When one conceives of a mathematical notion as an *object*, this notion is captured as one "solid" entity. Thus, it is possible to examine it from various points of view, to ana-

lyze its relationships to other mathematical notions, and to apply operations on it.

According to these theories, when a mathematical concept is learned, its conception as a process precedes – and is less abstract than – its conception as an object (Sfard, 1991, p. 10). Thus, process conception of a mathematical concept can be interpreted as being on a lower level of abstraction than its conception as an object; that is, abstraction level is reduced.

This way of reducing abstraction is reflected in the study presented in this article with respect to Graph Theory by student tendency to demonstrate process conception in their answers rather than object conception. Figure 3 presents examples that illustrate this tendency.

The next two observations elaborate how process conception is reflected in students' tendency to present an overly complicated entity in order to illustrate their claims.

Observation 1 – Presenting an overly complicated *object*: According to this observation, when students can answer a question by presenting an object, the object that they sometimes choose to present is too complicated for the relevant purpose. Figure 4 presents examples in which students present overly complicated objects. This phenomenon is explained by students' process-conception of the concepts involved. More specifically, students' process-perception of the concepts involved leads them to base their answer on a process rather than on the properties of the object under discussion. Naturally, a process is demonstrated in a better way when it is executed on a complicated object. We suggest that students' process-conception leads them to present a complicated object, on which the execution of an algorithm is not trivial and includes more than one iteration (usually at least 3-4 iterations).

Observation 2 – Presenting an overly complicated algorithm execution: All the algorithms learned in the course are iterative. Our data shows that when students demonstrate how an algorithm works, they tend to illustrate this by more than one iteration. Figure 5 presents an example for such an overly complicated execution of an algorithm. One way to interpret this phenomenon is by claiming that the students only desire to prove that they know how the algorithm works. However, this interpretation actually strengthens our suggestion that by this behavior students, in fact, exhibit a process conception of the relevant algorithm. We suggest that had the students conceived of the said algorithm as an object (through its properties) they would have understood that its demonstration by one iteration is sufficient in order to illustrate its nature.

Example 1 (class observation):

In one of the first lessons the following lemma was introduced: In a directed graph, the sum on d_{in} is equal to the sum of d_{out}. One of the students explained it in the following way: "This is because each arc that leaves a node has to enter a different node". This answer reflects a process conception of the argument that is: The arc leaves a node and while moving along a path should find another node to enter. This explanation was repeated by the instructor from an object perspective, emphasizing the properties of the concept arc: "Each arc has two edges. One is in, one is out".

Example 2 (second exam):

True or false:

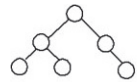
Each graph has a unique DFS tree.

Only 3 students found it sufficient to present only a counter example that violates this statement. The other 13 students felt a need to execute the DFS algorithm on a specific tree, in addition to the presentation of the counter example itself, in order to illustrate that two different trees may be obtained. Ten out of these 13 students executed it correctly; 2 students executed it incorrectly; 1 student suggested that the vertex from which we start the DFS is the only factor that determines whether we get different trees.

Example 3 (second exam):

True or false:

Based on the proof of the theorem that deals with the number of binary positional trees, the following sequence of parenthesis ((()) ()) (()()) corresponds to the following binary positional tree:



Out of the 16 students who took the exam, not a single student based his or her argument merely on the number of parenthesis, an argument that immediately yields that the statement is false. Such an explanation would reveal an object conception since it would have emphasized properties of concepts. As it turns out, solutions which reflect process conception were more predominant. Twelve students worked correctly and built the corresponding tree, illustrating that it is not the presented tree. Three out of these 12 students mentioned additionally that the number of parenthesis actually strengthens their argument. However, this explanation was not viewed by them as a sufficient argument and they felt the need to specifically construct the corresponding tree. The other 4 students gave a wrong answer.

Figure 3. Examples of answers reflecting process conception

Example 1 (first exam):

Draw a graph that contains a Hamilton path, but does not contain an Euler path.

Although 10 students presented simple examples (few vertices and simple graph structures), 7 students out of the 17 who took the exam felt the need to present an overly large or complex example. Figure (i) and (ii) show two examples of such graphs, presented by two students:

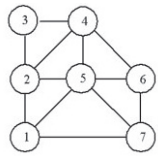


Figure (i)

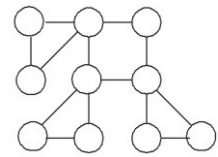


Figure (ii)

Clearly, simpler graphs could be presented to illustrate this claim. In Figure (i) the student explains: "In this graph Hemilton path exists and it passes through arcs: e_{12}, e_{23}, e_{34}, e_{45}, e_{56}, e_{67}. But, there is no Euler path because there are more than 2 vertices each having an odd degree; actually there are 4 vertices with an odd degree."

Example 2 (second exam):

True or false:

There exists a graph and specific executions of Prim algorithm and DFS algorithm, so that the execution of Prim and DFS algorithms on that graph create the same tree.

Out of the 16 students who took the second exam only 2 students gave a simple example which consisted of a graph with 2 nodes with an arc connecting them.

Example 3 (second exam):

True or false:

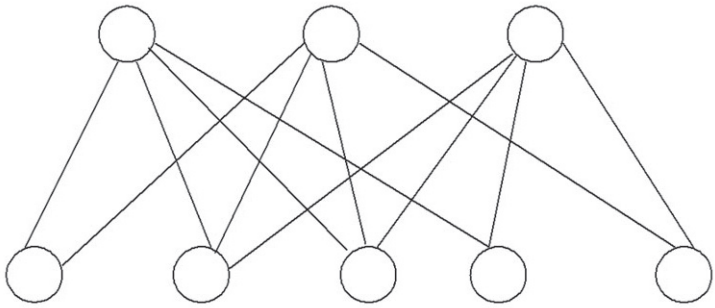
If the appearance probability of two letters in a given text is equal, then their Hufmann coding consists of the same number of letters.

Though it is sufficient to illustrate the fact that the statement is incorrect with three letters that are coded by $\sum = \{0, 1\}$, only 5 students out of the 16 students gave an example with three letters. Half of the students present a four/five letter-based example, and 3 students presented an example with more than five letters. Two of these three students presented an example with six letters, and 1 student presented an example with thirteen letters.

Figure 4. Examples of "overly complicated objects"

Example (first exam):

Execute Euler algorithm on the following graph. List all stages.



Although it was possible and quite simple to execute the algorithm in a single iteration, only 1 student out of 17 students presented this solution, and even he added a lengthy explanation to "excuse" his demonstration. All the other students executed more than one iteration (mostly 2-3 interactions).

Figure 5. Example of "overly complicated algorithm execution"

We conclude this section by suggesting that the complexity of the object the students work with is determined by their conception of the object at hand. That is, a student who conceives of a concept as an object may present a simple graph or a simple execution of an algorithm as far as the example possesses the relevant properties. A student who holds a process conception may tend to present a complicated object and/or a complicated execution of the algorithm so that a more convincing process (that is, a process that contains more than one iteration) is illustrated.

Specification

This section explored two ways by which students reduce the level of abstraction. The common attribute to these two ways is students' reduction of the complexity of the concept of thought by focusing on particular cases. In the first case students consider a specific object instead of dealing with the whole set of objects described to them. In the second case students over emphasize the visual aspect of an object and address a particular planar representation of an object they are requested to work with.

Observation 1 – Considering a specific case: This observation illustrates how students reduce abstraction level by replacing a set of objects with one of its elements. In other words, abstraction is viewed here through the lens of the complexity attributed to concepts. The working assumption is that the more compound an entity is, the more abstract it is. It does not imply automatically, of course, that it should be more difficult to think in terms of compound objects. In this respect, this section focuses on students working with a less compound object than those with which they were asked to work. Figure 6 presents an example which illustrates this phenomenon.

Example:

True or false:

Every tree can represent Hufmann coding.

Five of the students who answered correctly False presented a wrong explanation because they considered only a binary code. In other words, their consideration of {0, 1} as the only possible alphabet led them to think in terms of a specific set of trees instead of all possible trees.

Here are 2 examples of reasons presented by students, that indicate their relying on a binary code:

- "The Huffman coding algorithm is built in such a way that only a full binary tree can represent it."
- "The Huffman tree has external nodes which represent letters, and internal nodes which are probabilities. These nodes have to be the sum of two probabilities."

Figure 6. Example of treating a specific case

Considering a specific case may be a positive and helpful heuristic, as recommended by Polya (1973): "Specialization is passing from the consideration of a given set of objects to that of a smaller set, or of just one object, contained in the given set. Specialization is often useful in the solution of problems." (p. 190). The role of such specialization is to lead towards a general solution. However, there are cases in which students do use this heuristic, presenting an answer based on an analysis of a specific case, and do *not* return to the general case. Sometimes they do not go back to the general case simply because they are unable to do so – the mental structures needed to deal with the general case have not yet been constructed. This observation is coherent with what Young has pointed out: "Students whose only knowl-

edge of mathematical objects is through their concrete representations will be limited in their ability to assimilate new relationships which transcend the properties of those particular models" (Young, 1982, p. 130 in Hart, 1994).

Observation 2 – Considering a particular planar graph representation: This observation illustrates how students over emphasize the visual aspect of Graph Theory, and base their answers on a specific planar representation of graph's vertices and arcs. Indeed, visualization can be interpreted as a way by which abstraction is reduced. For example, Zazkis, Dubinsky and Dautermann (1996) contrast visualization with several concepts. Among other contrasts they mention "visualization as spatial versus abstraction". Indeed, sometimes visualization is used to construct that abstraction gradually bottom up (Harel, 1989).

However, it seems that in the case of Graph Theory reduction of the level of abstraction by means of visualization demands special attention. This is because Graph Theory deals with concepts that can be easily visualized, unlike group theory for example that most of whose concepts do not have a visual aspect. Figure 7 presents two examples in which the reduction of the level of abstraction by visualization yields either a wrong answer (Example 1) or the addition of redundant details (Example 2).

CONCLUSION

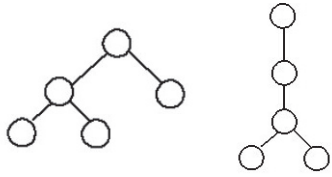
Understanding of concepts in high level of abstraction is very important for problems solving. In this article we present an examination of students' understanding of concepts in Graph Theory through the lens of abstraction. Specifically, we address two mental processes by which students reduce the level of abstraction. Within each we identify two particular ways by which these mental processes are expressed. With respect to the first way – process conception – we illustrate how students present an overly complicated object and an overly complicated algorithm execution; with respect to the second way – specification – we illustrate how students work with a specific case and how they consider a particular planar graph representation.

Example 1 (second exam):

True or false:

Each tree has a single DFS tree

Different planar arrangements of the vertices that resulted from two DFS traverses led students to assume that these arrangements are different DFS trees. Five students presented the wrong answer, basing their argument on the different planar arrangement of the graph's vertices and edges. For example, these two (non-directional) trees were suggested by a student as essentially different trees:



Example 2 (first exam):

For each of the following attributes, state at least one object to which the attribute may relate:

(a) Degree

(b) Even sum of the degrees

(c) The number of vertices which have an odd degree is even

(d) Weight

(e) Complexity

During the test the students kept asking the following questions with respect to the above task: "Should we give an example?"; "Is it possible that the same object appears twice as an answer?"; "Should we define?"; "Should we draw?". It seems that students could not accept a one-or-two-words answer – only the name of a concept – as a legitimate answer. Consequently, since the name of a concept is not conceived to be a sufficient answer, another explanation is needed. Accordingly, in 10 (out of 17) cases an additional visualized or literal explanation was presented. For example, while referring to the first concept "degree", a student drew a small graph in which one node was connected to three other nodes, and wrote: "The degree of node A is 3".

Figure 7. Examples of reducing the level of abstraction by visualization

We find this research important from a pedagogical point of view as well as from a theoretical point of view. From the pedagogical perspective the importance of this study is expressed by the fact that it may increase instructors' awareness of students' ways of thinking when learning Graph Theory in general, and to students' ways of reducing abstraction in this context in particular. Such awareness may clarify some of the difficulties that students are faced with when learning first concepts in Graph Theory. Indeed, this is a great challenge, which can possibly be achieved, for example, by consistently increasing students' awareness to the level of abstraction on which a specific discussion takes place, training them to think in terms of different levels of abstraction and to move between levels of abstraction, all according to the problem they face.

Theoretically, we illustrate the application of the theme of reducing abstraction for the analysis of students' understanding of concepts in Graph Theory. By doing so we expand the applicability scope of this theoretical framework. It is interesting to note that a third way by which students reduce the level of abstraction – relationships between the thinker and the object of thought – that is expressed in the analysis of students' understanding of other fields, is not expressed in our research. This interpretation suggests that whether something is abstract or concrete (or on the continuum between these two poles) is not an inherent property of the thing, "but rather a *property of a person's relationship to an object*" (Wilensky, 1991. p. 198). In other words, for each concept and for each person we may observe a different level of abstraction that reflects previous experiential connection between the two. The closer a person is to an object and the more connections s/he has formed to it, the more concrete (and the less abstract) s/he feels about it. Based on this perspective, some students' mental processes can be attributed to their tendency to make an unfamiliar idea more familiar or, in other words, to make the abstract more concrete. The fact that this way by which students reduce the level of abstraction is not expressed in our research may be explained by the fact that the introductory part of Graph Theory deals with relatively simple objects. Accordingly, it does not make much sense to rely on other objects in one's attempt to comprehend Graph Theory concepts.

In the future we intend to expand our work and to analyze student conception of concepts in Graph Theory with larger groups and with more in depth interviews. We believe that the findings presented in this article form a basis based on which this continuation research can be developed.

References

Baker, R. S., Biolen, M., Goodrich, M. T., Tamassia, R. and Stibel, B. A (1999). Testers and visualizers for teaching data structures, *ACM SIGCSE Bulletin* **31**(1), pp. 261-265.

Beth, E. W. and Piaget, J. (1966). *Mathematical Epistemology and Psychology*, D. Reidel Publishing Company.

Clements, D. H. and Battista, M. T. (1990). Constructivist learning and teaching, *Arithmetic Teacher* **38**(1), pp. 34-35.

Cote, V. (1987). Teaching oral communication in computer science, *SIGCSE Bulletin* **19**(2), pp. 58-60.

Dagdilelis, V. and Satratzemi, M. (1998). DIDAGRAPH: Software for teaching graph theory algorithms, *Proceedings of ITiCSE'98, ACM press,* pp. 64-67.

Davis, R. B., Maher, C. A. and Noddings, N. (1990, eds.). Constructivist views on the teaching and learning of mathematics*, Journal for Research in Mathematics Education,* Monograph Number 4*,* The National Council of Teachers of Mathematics, Inc.

Dubinsky, E. (1991). Reflective abstraction in advanced mathematical thinking. In Tall, D. (ed.). *Advanced Mathematical Thinking*, Kluwer Academic press.

Even, S. (1979). *Graph Algorithms*, Computer Science Press.

Flaningam, D. L., & Warriner, S. (1987). Another way to teach computer science through writing, *SIGCSE Bulletin* **19**(3), pp. 15-16.

Hansen, S., Tuinstra, K., Pisani, J. and McCann, L. I. (2003). Graph Magic: A Visual Graph Package for Students, *Computer Science Education* **13**(1), pp. 53-66.

Hamilton-Taylor, A. G., and Kraemer, E. (2002). Ska: Supporting algorithm and data structure discussion. *ACM SIGCSE Bulletin,* **34** (1), pp. 58-62.

Harel, G. (1989). Learning and teaching linear algebra: Difficulties and an alternative approach to visualizing concepts and processes, *Focus on Learning Problems in Mathematics* **11**(2), pp. 139-148.

Hart, E. W. (1994). A conceptual analysis of the proof-writing performance of expert and novice students in elementary group theory. In Kaput, J. and Dubinsky, E. (Eds), *MAA Notes* 33.

Hazzan, O. (1999). Reducing abstraction level when learning abstract algebra concepts*, Educational Studies in Mathematics* **40**(1), pp. 71-90.

Hazzan, O. (2003A). Reducing abstraction when learning computability theory, *Journal of Computers in Mathematics and Science Teaching (JCMST)* **22**(2), pp. 95-117.

Hazzan, O. (2003B). How students attempt to reduce abstraction in the learning of mathematics and in the learning of computer science, *Computer Science Education* **13**(2), pp. 95-122.

Jackowitz, P. M., Plishka, R. M., & Sidbury, J. (1990, February). Teaching writing and research skills in the computer science curriculum, *SIGCSE Bulletin* **22**(1), pp. 212-215.

Kuhri, S. and Holzapfel, K. (2001). EVEGA: An educational visualization enviroment for graph algorithms, *Proceedins of ITiCSE'2001, ACM press,* pp. 101-104.

McConnel, J. J. (1996). Active learning and its use in Computer Science, *Proceedings of the SIGCSE/SIGCUE Conference on Integrating Technology into Computer Science Education* (Barcelona, Spain, June 2-5, 1996), also published as *SIGCSE Bulletin* **28**, pp. 52-54.

Piaget, J. (1977). Problems of Equilibration. In Appel, M. H and Goldberg, L. S. (1977). *Topics in Cognitive Development, Volume 1: Equilibration: Theory, Research and Application*, Plenum Press, NY, pp. 3-13.

Polya, G. (1973). *How to Solve it?*, (Second edition), Princeton University Press, Princeton, New Jersey.

Sfard, A. (1991). On the dual nature of mathematical conceptions: Reflections on processes and objects as different sides of the same coin. *Educational Studies in mathematics* **22**, pp. 1-36.

Sfard, A. (1992). Operational origins of mathematical objects and the quandary of reification - The case of function. In Dubinsky, E. and Harel, G. (eds.). *The Concept of Function - Aspects of Epistemology and Pedagogy*, MAA Notes.

Smith, J. P., diSessa, A. A. and Roschelle, J. (1993). Misconceptions reconceived: A constructivist analysis of knowledge in transition, *The Journal of the Learning Sciences* **3**, pp. 115-163.

Wilensky, U. (1991). Abstract meditations on the concrete and concrete implications for mathematical education. In I. Harel, and S. Papert (eds.), *Constructionism*, Ablex Publishing Corporation, Norwood, NJ, pp. 193-203.

Zazkis, R., Dubinsky, E. & Dautermann, J. (1996) Coordinating visual and analytic strategies: A study of students' understanding of the group D_4, *Journal for Research in Mathematics Education* ***27***(4), pp. 435-457.

Notes

[1] For example, students may face difficulties in understanding why Dijkstras algorithm works only for graphs with positive weights.

[2] Cayley's theorem: There are n^{n-2} labeled trees with n vertices.