An Explicit Method for Teaching Generalization to Pre-Service Teachers Using Computer Programming

Cynthia L Stenger Janet T Jenkins James A Jerkins Jessica E Stovall
University of North Alabama

As colleagues in a Mathematics/Computer Science department, we found that many of our undergraduates were not able to participate successfully in the full range of STEM course offerings. In response to this need, we developed a strategy for explicit instruction in mathematical generalization. Our instructional design is grounded in a theory of mathematical learning that uses computer programming to induce students to build the mental frameworks needed for understanding a math concept. The design includes writing mini programs to explore a mathematical concept, finding general expressions in the code, making conjectures about the relationships among general expressions, and writing logical arguments for the conjectures. We share results from a study of 18 undergraduate math/secondary education majors. Our results indicate most pre-service teachers showed improvement in their level of abstraction over the concept of direct variation.

Key words: Generalization, Computer programming, Pre-Service teachers

Introduction

Abstraction and generalization are critical skills for navigation through the computer science and mathematics curriculum. Any effort to improve instruction should take into consideration how students learn. It is widely believed that teaching proof writing as a solitary activity may not provide students the building blocks to become proficient in reasoning about mathematical concepts. Stylianides claims that the four essential components of ‘reasoning-and-proving’ are ‘identifying patterns, making conjectures, providing non-proof arguments and providing proofs’ (Stylianides, 2008). Jenkins, et al., developed an explicit approach to teaching abstraction and generalization that considers the mental processes by which abstract concepts are acquired and utilized (2012). In developing the instructional treatment, close attention was paid to the theory of learning called APOS theory (Dubinsky, 1984). APOS is an acronym that stands for Action, Process, Object, and Schema. Each level denotes a cognitive classification of the learner's conception. APOS theory is an outgrowth of Piaget's theory of Reflective Abstraction (Piaget, 1971). As a constructivist theory, the basic tenet of APOS theory is that an individual's understanding of a mathematical topic develops through reflecting on problems and their solutions in a social context and constructing or reconstructing certain mental structures, then organizing these mental structures into schemas to use in dealing with problem situations. Specifically, in APOS theory, the process of reflective abstraction is the key to cognitive construction of logico-mathematical concepts (Dubinsky & McDonald, 2001, Beth & Piaget, 1966).

Programming as a Vehicle for Building Abstraction in the Mind of the Learner

Researchers in APOS theory have long employed computer programming as a means to teach undergraduate mathematics. In numerous studies, spanning several countries, and applied to a spectrum of mathematical topics, APOS theory has been applied to the use of computer experiences to encourage the construction of mental processes that lead to mathematical concepts (Asiala et al., 1998, Weller et al., 2008). The computer treatments have consistently yielded an
increase in likelihood that students acquired the desired concepts. It is a commonly held belief among this substantial group of researchers that computer constructions are an intermediary between concrete objects and abstract entities (Dubinsky, 1997, Asiala et al., 1998).

**Instructional Treatment: Building Mental Structures with Computer Programming**

Applying this theoretical framework, an instructional treatment was developed using computer programs to push students to build the mental frameworks for abstraction and generalization. This instructional treatment consists of four stages as shown in Figure 1. In the first stage essential characteristics (ESS) of a problem are identified. Next, mini programs (PROG) are written to explore the essential characteristics. General expressions (GEN) are found in the programs. Participants are taught to write these generalizations as mathematical statements. Further exploration with computer programs (PROG) leads to more generalizations, and general expressions (GEN) are collected as participants conjecture about relationships between concepts. Participants are taught to write convincing arguments (CA) for some of the conjectures, using the general expressions (GEN).

![Figure 1. The four stages of the instructional treatment](image)

**Methodology**

The participants were 18 undergraduate math/secondary education majors who were enrolled in a mathematical methods class. At this regional state university preservice teachers earn a bachelor of science in education and, in addition, complete all of the coursework for a mathematics major. This course is offered the fall semester before their internship.

**Procedure Description**

The instruction took place over a three day period. Each class session was 75 minutes. The format for the lessons included writing mini-programs using Python to explore concepts and make conjectures about relationships among concepts. For example, participants were asked to produce a table with columns consisting of distance, rate and time, then insert additional columns to show what happens when the rate is doubled while time is unchanged. From program output, they observe that distance doubled as rate doubled. They were taught to find the general
expression for this relationship in their code and then to write it in mathematical language. After sufficient time exploring the relationship between increasing rate and resulting distance, they were led to make conjectures about relationships between distance, rate and time, in general. For example, they might conjecture that when the rate was multiplied by $k$, then the resulting distance was $k$ times the original distance or “If $r_2=kr_1$, then $d_2=kd_1$.” This was followed by instruction constructing convincing arguments for some of the conjectures. The proof writing activity was designed to push students to progress to the next level of cognition by affording them the opportunity to apply their conceptual knowledge in a different setting.

**Data Collection/Analysis**

All participants were pre-tested and post-tested to determine their level of abstraction for the concept being explored. Throughout the lessons, response sheets were also collected. This allowed the conceptual knowledge to be evaluated at multiple points to determine how mental frameworks were being built. Based on this data, each participant was assigned scores representing their level of abstraction over the concept before and after each lesson.

APOS analysis was used to assign level of abstraction demonstrated by a participant. Scores ranged from 0 to 3 representing No abstraction, Action, Process, or Object. Each entry was scored by at least three trained data analysts with a rubric developed for that particular concept and based on responses elicited on the pre- and post-tests and the response worksheets. Triangulated scores were tested for inter-rater reliability using Randolph’s Kappa and assigned to each participant for each lesson, before and after instruction.

**Results**

Table 1 lists the item text associated with each scored response on the pre-test, response sheet, and post-test. Table 2 shows student scores for responses on the pre-test, participant response sheets, and post-test. Participant U0065 was not viable because they did not participate on the second day of the study. Twelve of the eighteen participants improved at least one level of abstraction in the direct variation lesson.

<table>
<thead>
<tr>
<th>Response Item</th>
<th>Item Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-test 2,3</td>
<td>What happens to distance when time is fixed and you triple the rate? Write a convincing argument.</td>
</tr>
<tr>
<td>Computer 3,4</td>
<td>What happens to distance when time is fixed and you double the rate? Write a convincing argument.</td>
</tr>
<tr>
<td>Math 7,8</td>
<td>What happens to distance when time is fixed and rate is cut in half? Write a convincing argument.</td>
</tr>
<tr>
<td>Math 9,10</td>
<td>What happens to the distance when time is fixed and rate decreases? Write a convincing argument.</td>
</tr>
<tr>
<td>Post-test 2,3</td>
<td>What happens to distance when time is fixed and you triple the rate? Write a convincing argument.</td>
</tr>
</tbody>
</table>

Table 1. Question text on pre-test, response sheet, and post-test
Table 2. Scored participant responses on pre-test, response sheet, and post-test

The following snips from participant U0068 are representative of how the students’ convincing arguments improved over the course of the instruction.

In the pretest, participant U0068 was able to recall the correct formulas to express the relationship between distance, rate, and time. They had a correct “intuition” concerning the result, “distance gets larger also” (Figure 2). After the computer programming instruction,
participant U0068 could state the effect of the increase specifically, rather than saying “the distance gets larger”. When asked to give a general expression for the relationship, they referenced their computer code. In addition, they wrote a description of the output of the code and described the relationship in terms of the columns of data produced as output (Figure 3). The mathematics portion of the lesson taught students to find general expressions in their code and write them in mathematical notation or symbology.

![Figure 3. Intermediate work after the computer programming instruction for participant U0068](image)

In the post-test, participant U0068 gave a convincing argument that used general expressions to express the relationship between distance, rate and time (Figure 4).

![Figure 4. Post test participant U0068](image)

In the post-test, participant U0068 gave a convincing argument that used general expressions to express the relationship between distance, rate and time (Figure 4).

**Conclusion**

In this study we have described an explicit method for teaching generalization. We have reported results from a pre-service teachers in a mathematical methods course. We have shown that most participants’ level of abstraction or ability to apply generalizations for direct variation increased. This is a strong indication that generalization can be taught explicitly. It suggests that further research into computer programming as an effective tool for teaching mathematical thinking is warranted.
References


