# Work in progress: Courses dedicated to the development of logical and algorithmic thinking

Orna Muller, and Amir Rubinstein
*ORT Braude College* of Engineering, Israel, *ornamu@braude.ac.il*, *amirrub@cs.technion.ac.il*

*Abstract* - **Undergraduate students often start their academic course of studies with inadequate learning and thinking skills. Our college has a policy of setting high standards, while supporting students' learning in a variety of ways. In this paper we present two distinctive courses designed to aid students develop logical and algorithmic thinking, essential for coping with software engineering studies. The courses are taught independently from but in parallel to Introduction to Computer Science and Math courses of first semester. Courses elaborate on algorithmic thinking, logical reasoning and argumentation while explicating abstract ideas which are often hidden in a loaded curriculum of the disciplinary courses. At the same time, connections are made to the application of the abstract ideas in the disciplinary courses. Feedback from participants in the two courses demonstrates an increased awareness and appreciation of abstract ideas beyond mathematical and programming knowledge, improved problem-solving skills and deeper understanding of concepts and principles.**

*Index Terms* - Algorithmic problem-solving skills, Argumentation, Logical reasoning.

## MOTIVATION

Computer science and software engineering curricula require high algorithmic and logical reasoning skills. Poor algorithmic thinking results in difficulties in formulating programmed solutions to algorithmic problems, in figuring an idea for a solution, in recognizing similarities among problems and in identifying familiar subtasks in a compound problem [1]-[2]. Insufficient logical thinking abilities result in poor and illogical argumentation processes, disconnected, imprecise and ambiguous statements, and incorrectly used terms [3]-[4]. Both algorithmic and logical thinking are essential for handling abstractions in a precise manner, which is a fundamental skill in computer science and software engineering [5].

According to knowledge construction theories, instructional design has a significant effect on the development of the aforementioned thinking skills. However, introductory math and programming courses frequently emphasize mastering specific math content and a programming language, while problem-solving processes are dealt with in an unstructured and implicit manner.

## LEARNING AND THINKING SKILLS COURSES

Several projects for the promotion of teaching and learning skills are operated in our college. One project is a mandatory Learning and Thinking Skills (LTS) courses for all freshman students. LTS aims at improving students' skills to cope with highly demanding engineering studies, thereby decreasing failures, and lowering first year drop-out rate. Students have to choose one of a variety of courses, two of which are discussed here.

## COURSES' RATIONALE AND DESIGN

We propose that algorithmic and logical thinking needs to be introduced separately from math and computer science courses, but in parallel to them. Limitations of transfer of skills from one context to another are recognized [6]-[7]; therefore, Introduction to Computer Science (CS1) and Calculus and Discrete Mathematics are taught in the same semester as the LTS courses. Consequently, connections and the implementation of ideas are constantly made.

Courses' planning includes a process of mapping and defining the skills that each course intends to develop and the instruction methodologies most suitable for supporting the acquisition of those skills.

The *Algorithmic Problem-Solving Skills* (APSS) course introduces generic "expert" solutions to recurring algorithmic problems (such as: Searching for an Item or Finding an Extreme Value in a List), and a well-planned selection of problems in different contexts are analyzed and solved (in pseudo-code). Issues such as generalization and abstraction, analogical reasoning, problem decomposition and recursive thinking are discussed and elaborated.

APSS course development was motivated by positive results of previous research on teaching similar contents to highly capable high-school students, indicating an improvement in students' ability to develop correct, efficient, and elegant algorithms [8]-[9].

Course setting is mostly based on workshop activities, such as comparing different solutions to a problem, categorizing problems, composing analogical problems to a given one, abstracting a prototyped solution out of several analogical problems, identifying and naming subtasks, and efficiency and elegance of algorithms. The APSS course emphasizes reflection on thinking and problem-solving processes, analyzing common difficulties and verbalization of ideas.

The *Logical Thinking and Argumentation* (LTAA) course has two main goals: 1) to develop logical and precise thinking, and 2) to enhance students' skills in formal argumentation techniques. The primary theme of proof and disproof is always in the spot-light. Among the topics introduced are basic propositional and predicate logics (with emphasis on ambiguity of statements, evaluating their truth or falsity, etc.), and different proof techniques of existential and universal statements (direct, indirect, by contradiction, constructive, non-constructive, induction, etc.).

The course aims at exposing the existing reasoning schemas students have, acknowledging explicitly some of the differences between mathematical logic and the logic used in everyday life, and then allowing incorporation of new schemas. Topics are taught not too formally, subtle issues are discussed to enhance students' awareness of them, and raising difficulties and examples for discussion is encouraged (for a comprehensive discussion of the motivation for such a course see [3]).

## METHODOLOGY

The LTAA is taught in college for the second time, and the APSS is taught for the fourth time. Reflective written questionnaires were presented to the students at the end of each course (twice in the APSS and once in the LTAA course). The questionnaires which consist of open questions, allow us to learn about students' attitudes regarding the influence of the course on their problem-solving skills, and changes in their perception of problem solving in general.

Altogether 85 students answered the questionnaire that was administered at the end of the APSS course. Students were asked to reflect on their experience and give their opinion regarding: (a) the main skills they acquired in the course; (b) the content and structure of the course; and (c) the desirable relationship between this course and the CS1 course (either to keep the courses separated, to take one before the other, or to unite the courses). Analysis of answers revealed several main categories of repeated ideas and comments.

20 students either answered the questionnaire given at the end of the LTAA course, or submitted a final paper in which they were requested to present the course rationale in their eyes, and give personal feedback. The issues students were asked to reflect on were: (a) the effect LTAA on their difficulties in other courses; (b) the content and structure of the course.

## STUDENTS' PERCEPTIONS AND FEEDBACK

Preliminary analysis of students' answers in APSS discloses an increased confidence in approaching and analyzing novel and compound problems. Students mentioned skills they have acquired related to analogical reasoning, such as, the ability to see structural similarities between problems.

Students' answers reflect on their ability to distinguish between the two major levels of problem solving: the abstract level (*i.e*. problem analysis and solution design) and the concrete level (*i.e.* writing code and running programs), and the importance of "planning before implementing". Students expressed their realization of the importance of learning theoretical and abstract aspects of problem solving beyond programming language.

Almost all students, even those with previous experience in programming, mentioned the contribution of the course to broadening their repertoire of algorithmic ideas, especially ideas that lead to clear, straightforward, elegant and efficient algorithm. Many students ascribed a major contribution to the course design that apparently promoted their assimilation of ideas; for example, students have mentioned dividing the course into themes and being exposed to several examples for each type of problem/theme as mostly helpful in utilizing ideas later on, in new contexts.

Since LTAA is being taught in our college for the second time, we have relatively few impressions at the moment. In general, students report improvement in their confidence in the rationality of logical argumentations. Many of them feel that they are more aware today of vague statements made by themselves or others, and therefore can make an effort to avoid them when dealing with argumentation of abstract entities. Some also find proofs they encounter "more familiar" and less threatening because their structure now seems logical to them.

Another feedback mentioned throughout the semester by many was that in this course there was time and legitimacy to expose, argue about and iron out subtle logical issues, which caused difficulties in other first year courses they took (*e.g.*: the fact that in some universal statements the "for all" quantifier is omitted, the right way to disprove an 'if *a* then *b*' statement).

This feedback implies that an efficient introductory unit that explicates the principles of logical reasoning can bridge between natural, informal everyday verbal expression, and problems encountered in science and engineering.

## FUTURE WORK

In order to evaluate the effect of the courses, further research plans include a comparison between the quality of solutions to problems in written exams and interviews, of students who have participated in one or the two courses, and of students who did not participate in any of them. Aspects such as the following will be inspected: correctness and efficiency of solutions, design of solutions, preciseness and clearness in expression of ideas and avoidance of common logical obstacles.

Because of the growing interest in this type of courses in other engineering departments in our institution, a deeper insight on the potential contribution of the courses may help in designing similar courses suited for other disciplines.

## REFERENCES

[1] Deek, F.P., & McHugh, J. 2000. "Problem-solving methodologies and the development of critical thinking skills." *Journal of Computer Science Education*, 14(1-2), 6-12.

[2] Robins, A., Rountree, J., & Rountree, N. 2003. "Learning and teaching programming: a review and discussion." *Computer Science Education*, 13(2), 137-172.

[3] Epp, Susanna S. December 2003. "The role of logic in teaching proof." *American Mathematical Monthly*, *110* (10), 886-899.

[4] Selden A., Selden J. 2003. "Validations of proofs considered as texts: Can undergraduates tell whether an argument proves a theorem?" *Journal for Research in Mathematics Education*, 34(1) 4-36.

[5] Devlin, K. 2003. "Why universities require computer science students to take math." *Communications of the ACM*, September 2003, 37-39

[6] Bassok, M. 2003. "Analogical transfer in problem solving." In Davidson, J.E. and Sternberg, R.J. (Eds.), The Psychology of Problem Solving. Cambridge University Press.

[7] Mayer, R.E., & Wittrock, M.C. 1996. "Problem-solving transfer." In D.C. Berliner & R.C. Calfee (Eds.), *Handbook of Educational Psychology*, New York: Macmillan.

[8] Muller, O. 2005. "Pattern oriented instruction and the enhancement of analogical reasoning." *Proceedings of the 1st International Computing Education Research Workshop* (ICER), Seattle, Washington, 57-67.

[9] Muller, O. & Haberman, B. 2008. "Supporting abstraction processes in problem solving through pattern-oriented-instruction." *Computer Science Education*, 18(3), 187-212.

## AUTHOR INFORMATION

**Orna Miller**, Head of Teaching Studies department and Teaching & Learning Center at Ort Braude College of Engineering, Israel *ornamu@braude.ac.il*

**Amir Rubinstein**, Lecturer at Ort Braude College of Engineering, and PhD candidate at Tel Aviv University, Israel, *amirrub@cs.technion.ac.il*