Using Declarative Programming in an Introductory Computer Science Course for High School Students

Maritza Reyes,[1] Cynthia Perez,[2] Rocky Upchurch,[3] Timothy Yuen,[4] and Yuanlin Zhang[5]
[1]Department of Psychology, University of Texas at Austin, USA
[2]Department of Psychology, Texas Tech University, USA
[3]New Deal High School, Lubbock, Texas, USA
[4]Department of Interdisciplinary Learning and Teaching, University of Texas at San Antonio, USA
[5]Department of Computer Science, Texas Tech University, USA
[1]maritza_reyes@utexas.edu, [2]cynthia.perez@ttu.edu, [3]RockyU@ndisd.net, [4]Timothy.Yuen@utsa.edu, [5]y.zhang@ttu.edu

Abstract

Though not often taught at the K-12 level, declarative programming is a viable paradigm for teaching computer science due to its importance in artificial intelligence and in helping student explore and understand problem spaces. This paper discusses the authors' design and implementation of a declarative programming course for high school students during a 4-week summer session.

Background and Rationale

Computer science education researchers have long debated imperative versus object-oriented approaches to teaching introductory computer science courses, but the conversation often leaves out declarative programming (DP) approaches. The last two decades have seen significant advances in theory and application of declarative programming including answer set programming (Gelfond and Kahl 2014). Answer set programming (ASP) has its roots in Logic Programming and nonmonotonic reasoning and becomes a dominant language in the knowledge representation community because it offers elegant and effective solutions to classic artificial intelligence problems, such as frame problems, ramification problems, and many challenging knowledge intensive application problems. Thanks to its simplicity and clarity in both informal and formal semantics, ASP provides natural modeling of many knowledge intensive problems.

Given its importance in artificial intelligence and general problem solving, the authors assert the need for students to learn and have a basic understanding of declarative programming. The purpose of this paper is to discuss the rationale for and the design of an introductory computer science course based on the declarative programming paradigm. This paper describes how the authors implemented this as a four-week summer course for high school students taking part in a summer enrichment program.

Course Description

Computer Science 4 (CS4) is a four-week course on declarative programming using the answer set programming approach (Gelfond and Kahl 2014). The course described in this paper was offered during the summer of 2015. The course met Monday through Friday from 1:00pm to 1:50pm, and was taught in a computer lab.

This course was offered as part of the TexPREP program hosted at Texas Tech University. TexPREP is a math, science, and engineering summer enrichment program for 6th - 12th graders. Each summer, students take courses in math, science, and engineering typically taught by local K-12 teachers and university students, and occasionally by university faculty.

Students taking this course were in their fourth year of the TexPREP program, and had taken two CS courses (CS2 and CS3) prior to CS4. CS2 taught Scratch and CS3 taught Alice. Both courses covered the foundations of programming up to and including conditionals, recursions, and loops within their respective programming environments. Note there is no CS1 in TexPREP.

Course Objectives. The overall objective of this course was to make students capable of solving problems using a declarative approach. Specifically, by the end of this course, students should be able to: 1) identify the knowledge needed to specify a problem at hand, 2) identify the objects (and their categories), the relations among objects and how one relation is defined by other relations in terms of the knowledge identified in the first objective, and 3) write ASP programs to represent the objects, relations and definitions identified in the second objective.

Course topics included defining sorts (categories of objects), declaring relations, defining a relation using other relations (possibly in a recursive manner), solving problems in terms of answering relationship ties among members of a family, coloring a map, and a Sudoku problem.

Development Environment. The specific ASP language selected for this course was SPARC (Balai, Gelfond, and Zhang 2013) mainly due to the explicit modeling methodology underlying this language. The programming environment we used for writing and reasoning with SPARC programs is ASPIDE, a graphical user interface based integrated development environment.

Students. There were originally 17 students registered for this course; however, one dropped out mid-way through the course. Sixteen students consented to having their demographic information and coursework released as part of a larger study. Of these students, there were 9 males and 7 females. The average age was 15. Most students were in 10th grade (11 students) with three 11th graders, and two 12th graders.

Instructional Staff. The course was taught by the last co-author, a computer science professor at the hosting university, along with three teaching assistants (the first three co-authors) who were given a one week training on ASP. The assistants would help with any questions or concerns that the students had.

Course Activities. The activities of the course mainly included lecture and assignments. Each week consisted of four lectures and one lab assignment, which was based off of the topics and activities done during the lectures. Lecture notes were provided to the students so they could follow along with the lecture. In the lecture, the instructor introduced the main concept, shows examples and asks students to apply what was learned to new examples, and demonstrate the programs. Then, students will implement the same example programs on their own on computers. Assignments dealt with stating factual relations, recursive definition of relations, and solving map coloring, and Sudoku problems.

Labs. There were three graded lab problems through which course topics and activities centered on. *Family Problem*: Students were asked to identify interesting queries about relationships (e.g., mother, mom, ancestor) of a family, create specific information about a family, and write a SPARC program based on which the identified queries can be answered automatically by the SPARC system. *Map Coloring Problem*: Given a map of US states, the students were asked to write a SPARC program to answer whether this map can be colored with red, green or blue such that no neighboring states are allowed to have the same color. *Sudoku Problem*: This problem is based on the well known number logic game. Students were asked to write a SPARC program which found all the missing numbers in a Sudoku problem.

Evaluation and Feedback

To examine the impacts of declarative programming on computer science learning more rigorously, the authors are currently analyzing data from artifacts, surveys, and interviews collected from students taking this course in Summer 2015. This upcoming study will show how the declarative programming paradigm guides students in computational thinking, problem solving, and interests in computer science. Preliminary results with regards to achievement and interests in this CS course are presented here.

Learning. Students had performed well in the lab assignments and in class. Each lab was graded by two graders, and the average class score on the three graded labs were: Lab 2: 93.2%, Lab 3: 92.2%, and Lab 4: 93.2%. These averages only include grades for labs that were submitted on time.

Students were given an open-ended questionnaire at the end of the course. One question asked students to summarize what they learned. The students reported that they were able to solve problems using the methodologies and tools of the declarative programming paradigm. Students also reported that they learned specifically how to solve problems through programming with the SPARC language. This preliminary finding suggests that novice CS students are able to operate within the declarative programming paradigm.

Interest. According to course evaluations, most students have enjoyed this course. In order to evaluate how CS4 impacted students' interest in CS careers, the authors administered the STEM Career Interest Survey (Kier et al. 2014). This survey was adapted for this course by focusing all 11 survey questions in computer science. Participants were asked to rate their level of agreement between 1-Strongly Disagree to 5-Strong Agree with each statement related to interest in CS. The average overall pre-test score was 3.62 and the average overall post-test score was 3.52. Though there was moderate interest in pursuing CS as a career, there was a slight decline after the course. These survey results could be due to the fact that this course was the third CS course these students have taken. Therefore, their interests in CS would have stabilized by this time. Another reason could be that four weeks was insufficient time to spark further interest in declarative programming. Further, as indicated by an open-ended questionnaire on their career interests, the group of participants had a diverse set of career aspirations and reported some issues with the way the course was organized, which may account for the decrease. Further iterations of this course may draw in more relevant examples that are relevant to both students' everyday lives and interests.

Discussion and Conclusions

The authors believe that the declarative programming paradigm is as important as imperative and object-oriented approaches to computer science. A precise specification and understanding of a problem is critical in problem solving and lays in the core of computer science. Declarative programming emphasizes problem specification instead of how a problem is solved, and offers a natural and simple representation for the specification of numerous problems. It allows the students to work on more interesting and challenging problems such as Sudoku problem in their early stage of learning programming. Programming becomes a formalization of the "familiar" thinking (in terms of specification and understanding of problem) that people need when solving intellectually interesting problems.

Acknowledgment

We thank Michael Gelfond for sharing with us his teaching materials and philosophy. This work is partially supported by NSF grant IIS-1018031 and CNS-1359359.

References

Balai, E.; Gelfond, M.; and Zhang, Y. 2013. Towards answer set programming with sorts. In *Logic Programming and Nonmonotonic Reasoning*. Springer. 135–147.

Gelfond, M., and Kahl, Y. 2014. *Knowledge representation, reasoning, and the design of intelligent agents: The answer-set programming approach*. Cambridge University Press.

Kier, M. W.; Blanchard, M. R.; Osborne, J. W.; and Albert, J. L. 2014. The development of the stem career interest survey (stem-cis). *Research in Science Education* 44(3):461–481.