Writing computer programs can be performed in three principle ways: Generating code based on informal specifications of the desired behavior of the program, automated program construction from formal specifications, or constructing programs by generalization over small sets of input/output examples. Research in the artificial intelligence domain of inductive programming (IP) addresses the third approach. From a technical perspective, IP systems can relieve software developers from routine programming tasks by exploiting test cases for automatic generation of code (for clear cut subproblems). From a cognitive perspective, IP research investigates the intellectual abilities and skills of human programmers who typically are highly successful in coding programs from input/output examples. Our IP system IGOR (Kitzelmann and Schmid, 2006; Kitzelmann, 2007) induces recursive functional programs from small sets of the first input/output examples with respect to the input data type. We conducted an empirical study to investigate whether inexperienced programmers can produce the examples necessary for IGOR and whether generating examples is less prone to errors than directly producing the program code. Participants were 30 first year computer science students who had participated in a course where recursive functional programming with Scheme was one of the major topics. Each student received a booklet with four programming tasks (“length” of a list, “reverse” a list, is a list of “oddlength”, and “flatten” a nested list). Each student had to give the smallest set of input/output examples for two of the tasks and to write the Scheme code for the other two tasks (sequence of tasks was balanced over subjects and giving examples vs coding was balanced over tasks). Nearly all students failed in giving examples as well as in producing correct program code for the most complex problem “flatten”. For the other three tasks we got clear evidence that while students are able to give the first input/output examples they mostly fail in producing correct code. Besides this general result, an analysis of typical errors provided some insights in difficulties of recursive programming. The results show that IP systems cannot only support professional programmers but they also give more power to programming novices who are enabled to produce correct code by providing input/output examples.
References
