PROPOSAL: From Program Synthesis to Verification

Program synthesis is the task of generating a program that meets a required specification. Program verification is the task of validating program correctness with respect to a given specification. Both are longstanding problems in computer science, although there has been extensive work on program verification and comparatively less on program synthesis until recently. Over the past several years, certain verification techniques have been adopted to create programs, e.g., applying symbolic execution to synthesize program repairs, suggesting the possibility that these two problems may be "two sides of the same coin". Formalizing such a connection is valuable in both theory and practice; it allows comparisons between the complexities and underlying structures of the two problems, and it raises the possibility of additional cross-fertilization between two fields that are usually treated separately.

The proposed research will explore and exploit such formal connections in program synthesis and verification to build novel and practical program analysis techniques and tools. Preliminary work has constructively proved that the template-based formulation of synthesis, which generates a program in a pre-specified form, and the reachability problem in verification, which checks if a program can reach a specified location, are linear-time equivalent. To demonstrate that the equivalence enables ideas, optimizations and tools developed for one problem to be applied to the other, this work has developed an automatic program repair (a formulation of synthesis) approach using existing reachability techniques and tools. The algorithm transforms a buggy program and its required specification into a specific program containing a location reachable only when the original program can be repaired. The transformed program is then used as input to an off-the-shelf test input generation tool to find test values that can reach the desired location—these test values correspond exactly to repairs for the original program. Preliminary study suggests that this approach has higher success rates than many other standard repair techniques.

Intellectual Merit: Researchers have long hypothesized about the relation between program synthesis and verification and proposed template-based synthesis and program repair approaches using techniques or tools often used to verify programs such as constraint solving or model checking. This work shows that it is not just a coincident that these synthesis works can exploit verification techniques, but that every template-based synthesis problem can be reduced to the reachability formulation in verification. More importantly, the constructively nature of the proof allows for an efficient algorithm to do such the reduction, which results in a new and effective automatic program repair algorithm using existing off-the-shelf reachability tools.

The proposed research will extend this preliminary work and explore other connections in verification and synthesis, with focus on developing practical and scalable tools to analyze real-world programs and defects. This research is highly innovative, different, and will provide a solid foundation for other verification and synthesis work. Although this line of research is highly exploratory in nature, it will also open doors to new areas of research in program languages (PL) and software engineering (SE).

Broader Impacts: This project will link different research fields and thus will allow for the transfer of ideas, techniques, and results among them. We will disseminate our findings through both publication and by developing and sharing implementations of our techniques. These will be available to other researchers and practicing engineers, and we will integrate material from this work into our graduate and undergraduate PL and SE courses. We will actively involve and recruit students belonging to underrepresented groups into our research activity, classes, and the newly created SE program at the University of Nebraska.

Key Words: Program synthesis; program verification; program reachability; automatic program repair; automatic invariant generation; reduction proof; equivalence