Algorithmic Program Synthesis with Partial Programs and Decision Procedures

Rastislav Bodik

University of California, Berkeley

Abstract. Program synthesizer can derive programs that are efficient, even surprising, but it must be first "programmed" with human insights about the domain and its implementation tricks. In deductive synthesis, the insights are captured by domain theories, often elusive and always requiring formal expertise. To bring synthesis to everyday programmers, we have been exploring algorithmic synthesis, which is to deductive synthesis what model checking is to deductive verification: Rather than deducing a program with a theorem prover, algorithmic synthesis systematically finds the program in a space of candidate implementations. If we help programmers turn their insights into descriptions of candidates, we have a chance for a practical synthesizer.

I will show how sketches-partial programs that syntactically define the candidate space-allow programmers to express their insight while eliding tedious code fragments. These fragments are filled in by CEGIS, our counterexample-guided inductive synthesis algorithm that exploits recent advances in automated decision procedures. I will also show how these decision procedures allow us to implement an oracle that helps the programmer refine and formalize his insight about a problem. Finally, I will describe the linguistic support for synthesis in our SKETCH language and show how we synthesized complex implementations of ciphers, scientific codes, and concurrent lock-free data-structures.

J. Palsberg and Z. Su (Eds.): SAS 2009, LNCS 5673, p. 1, 2009.