Synthesizing Programs with Constraint Solvers

Rastislav Bodik and Emina Torlak

University of California, Berkeley

Abstract. Classical synthesis derives programs from a specification. We show an alternative approach where programs are obtained through search in a space of candidate programs. Searching for a program that meets a specification frees us from having to develop a sufficiently complete set of derivation rules, a task that is more challenging than merely describing the syntactic shape of the desired program. To make the search for a program efficient, we exploit symbolic constraint solving, lifted to synthesis from the setting of program verification.

We start by describing the interface to the synthesizer, which the programmer uses to specify the space of candidate programs $\mathcal{P}$ as well as the desired correctness condition ϕ. The space $\mathcal{P}$ is defined by a program template whose missing expressions are described with a grammar. The correctness condition is a multi-modal specification, given as a combination of assertions, input / output pairs, and traces.

Next, we describe several algorithms for solving the synthesis problem $\exists P \, \forall x \; \phi(x, P(x))$. The key idea is to reduce the problem from 2QBF to SAT by sampling the space of inputs, which eliminates the universal quantification over x.

Finally, we show how to encode the resulting SAT problem in relational logic, and how this encoding can be used to solve a range of related problems that arise in synthesis, from verification to program state repair. We will conclude with open problems on constraint-based synthesis.