Adaptable Semantic Code Search to Support Developer Behavior
Dr. Kathryn Stolee, Assistant Professor, North Carolina State University

Semantic code search is an approach to finding code based on behavior rather than syntax. I have developed an approach to semantic code search that represents code snippets as constraints and, given a query in the form of one or more input/output examples, uses a constraint solver (e.g., SMT) to match code snippets to the query. The end results is code for reuse that behaves as specified.

The particular flavor of code search I developed, semantic code search for reuse, has been applied in several languages: Yahoo Pipes (TOSEM 14), SQL (TOSEM 14), Java (JSS 16, TOSEM 14), and C (ASE 2015). It has been successful in 1) finding code from a database of code snippets that is as relevant as a Google query, and 2) finding code to patch faulty programs. The former was part of my dissertation work while the latter is part of a recently-funded NSF proposal on semantic search driven program repair. In my CAREER proposal, I want to bring code search beyond reuse, beyond concept localization, and toward tasks that developers use search for, but that are not well supported.

In short: I propose a broader vision of semantic code search that adapts to the needs of the programmer (context for search, abilities/experience, precision of query format).

Abstractions and Synthesis: A skilled programmer can easily adapt nearly correct code. A beginner programmer may need more help, like synthesis to create the needed code, even if it takes longer. Thus, we propose to develop techniques for **code abstractions that approximate behavior** and find "close enough" or "easily adaptable" solutions. We also propose to **synthesize examples based on desired behavior** to create solutions when none exist, essentially operationalizing test-driven development.

Query Formats: Keywords are the *modus operandi* for code search today, but are specific to written language and common terms are often overloaded. When it comes to source code, keywords lack the structure and specificity needed to reuse code, such as data types. Query formats also need to be adaptive to context. Keywords may not be precise enough when finding code to reuse, but might work for an example of how to use an API. Searching to identify when an error was introduced may involve specifying a line of code and searching commit history. I propose to develop **query formats that are more precise than keyword**, lightweight, and adaptable to a developer's context, taking advantage of information on hand.

Similarity Metrics: The more metadata we store about source code, the longer a search takes. The larger code gets, the longer a search takes. **Similarity metrics for code search can help organize code snippets, compare/contrast similar examples, find errors, and - if language agnostic - support identification of similar code between languages**. As a complement to similarity metrics, we also propose approaches to navigating the *dissimilarity* between code snippets. Developers frequently search for examples (FSE 2015), and other researchers have explored the makings of a good example. Many examples are returned by search engines, making differentiating them an important task. This differentiation could be provided through the generation of differentiating inputs that maximally divide the space of potential examples (ICSE NIER 2017), leading to code that behaves as desired and requires minimal decision points to identify.

Education: Code search that identifies multiple semantically identical implementations of reusable code could be valuable in education, especially for newcomers. Characterizing the dissimilarities between a developer's code and an oracle (as in a programming class) would be useful as well; test cases provide a sampling, but I propose to characterize semantic differences more precisely. The techniques developed in this grant will be piloted on entry-level programming classes at NCSU.