InSynth: A System for Code Completion using Types and Weights

Tihomir Gvero[1], Viktor Kuncak[1], Ivan Kuraj[2], Ruzica Piskac[3]

[1] EPFL, Switzerland
[2] MIT, USA
[3] Yale, USA
[1]{tihomir.gvero,viktor.kuncak}@epfl.ch
[2] ivanko@csail.mit.edu
[2] ruzica.piskac@yale.edu

Abstract: Developing modern software typically involves composing functionality from existing libraries. This task is difficult because libraries may expose many methods to the developer. To help developers in such scenarios, we present a technique that synthesizes and suggests valid expressions of a given type at a given program point. As the basis of our technique we use type inhabitation for lambda calculus terms in long normal form. We introduce a succinct representation for type judgements that merges types into equivalence classes to reduce the search space, then reconstructs any desired number of solutions on demand. Furthermore, we introduce a method to rank solutions based on weights derived from a corpus of code. We implemented the algorithm and deployed it as a plugin for the Eclipse IDE for Scala. We show that the techniques we incorporated greatly increase the effectiveness of the approach. Our evaluation benchmarks are code examples from programming practice; we make them available for future comparisons.

The results presented here were published as a research paper at the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '13), Seattle, WA, USA, June 16-19, 2013. [GKKP13] This extended abstact also contains some excerpts from the original paper.

InSynth - A System Overview

Libraries are one of the biggest assets for today's software developers. Useful libraries often evolve into complex application programming interfaces (APIs) with a large number of classes and methods. It can be difficult for a developer to start using such APIs productively, even for simple tasks. Existing Integrated Development Environments (IDEs) help developers to use APIs by providing code completion functionality. For example, an IDE can offer a list of applicable members to a given receiver object, extracted by finding the declared type of the object.

InSynth [GKP11, GKKP13] is a tool that automatically generates code snippets, similarly as in the case of auto-completion of code. By invoking InSynth, the user asks our tool to suggest a list of suitable code fragments for the given program point, as illustrated in

Fig. 1. InSynth displays a ranked list of suggested code snippets for that program point and the user can chose the best solution. To return meaningful snippets, and thus help a programmer, our tool synthesizes and suggests valid expressions of a given type in a short time (less than 0.5 sec).

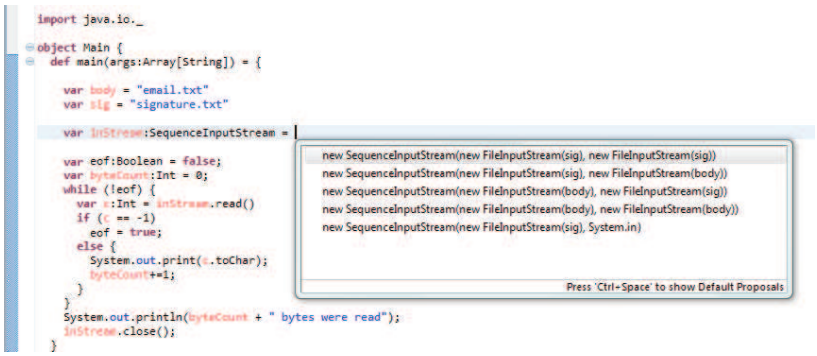

Figure 1: InSynth suggesting five highest-ranked well-typed expressions synthesized from declarations visible at a given program point

Finding a code snippet of the right type is closely related to the type inhabitation problem, where we ask whether for a type environment and some calculus, there exists an inhabitant of a given type. As providing the correct answer fast is a very important requirement, we introduce a succinct representation for type judgments that merges types into equivalence classes to reduce the search space.

Furthermore, it is not enough to provide a code fragment of the correct type – we should also be able to guess what the user had in mind. For this purpose we introduce a ranking of solutions based on a weight function. Moreover, the weight function is used to direct the search for type inhabitants. The weight function is defined on all the symbols appearing in the program, and it is based on the proximity to the program point at which InSynth was invoked. Additionally, the weight is also partially derived from a corpus of code.

We have found our system to be useful for synthesizing code fragments for common programming tasks, and we believe it is a good platform for exploring software synthesis techniques.

References

[GKKP13] Tihomir Gvero, Viktor Kuncak, Ivan Kuraj, and Ruzica Piskac. Complete completion using types and weights. In *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '13, Seattle, WA, USA, June 16-19, 2013*, pages 27–38, 2013.

[GKP11] Tihomir Gvero, Viktor Kuncak, and Ruzica Piskac. Interactive Synthesis of Code Snippets. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, pages 418–423, 2011.