Programming with Enumerators: Constructive Approach to Challenge the Status Quo in Automated Data Structure Generation

Ivan Kuraj
CSAIL, MIT, USA

ABSTRACT

Automated generation of data structures is useful for tasks such as software testing, which explores large spaces of valid test inputs. Efficient exhaustive generation of structures with complex properties remains a significant challenge. Such properties are not only defined with types, e.g. being a tree, but also specific constraints, e.g. the tree is balanced. Prior approaches that rely on constraint solving are expressive for generating complex structures but incur significant performance penalties, while those that rely on explicit construction lack the necessary expressiveness.

Recent results affirm that achieving best of both worlds, namely both expressiveness and efficiency, lies in constraint-based generation, thus refuting common wisdom about its inferior performance due to inevitable backtracking [1]. The hypothesis that constructive approaches—which treat generators as building blocks and achieve generation of complex structures with composition—can support an equally expressive definitions while being more efficient, could effectively refute such refutations.

We show that complex data structures, such as red-black trees and B trees, and even graphs, can be generated exhaustively and orders of magnitude more efficiently, by constructing enumerators within a specific algebra [2]. Our enumerator algebra allows constructing and composing enumerators of any given set of values, while embedding constraints into the construction itself to allow decomposing data structure properties into simpler ones that ultimately avoid backtracking. Enumerators uniquely number all defined structures to make them directly indexable with numbers. Generation of structures performs indexing, which uses numbering schemes defined by mathematical bijections to remain efficient for any enumerator composition within the algebra.

BODY

We define enumerators that assign numbers to arbitrary values. Their composition efficiently enumerates big sets of complex data structures.

REFERENCES

[1] K. Dewey, L. Nichols, and B. Hardekopf. Automated data structure generation: Refuting common wisdom. In *Proceedings of the 37th International Conference on Software Engineering*, ICSE '15, pages 32–43, Piscataway, NJ, USA, 2015. IEEE Press.

[2] I. Kuraj, V. Kuncak, and D. Jackson. Programming with enumerable sets of structures. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA '15, pages 37–56, New York, NY, USA, 2015. ACM.

Volume 4 of Tiny Transactions on Computer Science