A Language for Software Variation Research*

Martin Erwig

School of EECS
Oregon State University

Variation occurs in many places in software engineering and takes quite different forms. Software can have different versions, and it can come in different configurations. Software can offer different sets of features, and it can appear in different stages of refactoring without any visible effect in functionality. Traditionally, all these forms of variation have used different representations. While this specialization might have some benefits by facilitating the tailoring to the specific needs of one form of variation, it has also some serious drawbacks. First, different representations prevent or complicate a potential integration of different forms of variation. For example, variation in functionality is currently only poorly supported in most versioning tools by branching. Second, it can be difficult to transfer research results achieved within one representation to other representations. Finally, different representations can lead to duplicated work and a balkanization of variation research efforts.

In this talk I describe the *choice calculus*, a formal representation for software variation that can serve as a common, underlying representation for variation research, playing a similar role that lambda calculus plays in programming language research. I will sketch the syntax and semantics of the choice calculus and present several applications.

At the core of the choice calculus are *choices*, which represent different alternatives that can be selected. Choices are annotated by names, which group choices into *dimensions*. Dimensions provide a structuring and scoping mechanism for choices. Moreover, each dimension introduces the number of alternatives each choice in it must have and tags for selecting those alternatives. The semantics of the choice calculus is defined via repeated elimination of dimensions and their associated choices through the selection of a tag defined by that dimension. The choice calculus obeys a rich set of laws that give rise to a number of normal forms and allow the flexible restructuring of variation representations to adjust to the needs of different applications.

Among the potential applications of the choice calculus are feature modeling, change pattern detection, property preservation, and the development of change IDEs. These are described in the long version of this abstract [1]; more technical details about the choice calculus can found in [2].

References

1. Erwig, M.: A Language for Software Variation. In: ACM SIGPLAN Conf. on Generative Programming and Component Engineering, pp. 3–12 (2010)
2. Erwig, M., Walkingshaw, E.: The Choice Calculus: A Representation for Software Variation. ACM Transactions on Software Engineering and Methodology (to appear, 2011)

* This work is supported by the Air Force Office of Scientific Research under the grant FA9550-09-1-0229 and by the National Science Foundation under the grant CCF-0917092.