JACQUES CARETTE, McMaster University, Department of Computing and Software
Algorithm families, or how to write less code that does more

Program families are well-known in software engineering, and generic algorithms are pervasive in computer algebra. Nevertheless, in computer algebra libraries, one finds an inordinate amount of code duplication, even at the level of named algorithms. Some techniques are already in use to deal with this issue, with C++ template programming in common use in Linbox and post-facto extensions in Aldor.

By combining ideas from program families and generic algorithms, a careful study of the variations of well-known algorithms allows us to re-unite these offshoots back into a coherent family, where all members find their place. This talk will mostly concentrate on showing how many dimensions exist in simple families of algorithms, with LU decomposition being the guiding example. While previous work has clearly documented between 4 and 6 dimensions, we have discovered at least 19 separate "dimensions of variation" in that same family. As time permits, we will show that our chosen implementation method (higher-order modules of code generators in MetaOCaml) possesses some technical advantages over other solutions.

This is joint work with Oleg Kiselyov.