Certified code generation for geometric predicates

- **Theme:** Computer arithmetics, code certification, geometric computing
- **Laboratory:** Centre Inria Nancy Grand-Est
- **Team:** ALICE http://alice.loria.fr
- **Advisor:**
  Bruno Lévy - Bruno.Levy@inria.fr
- **Director of the lab:**
  Sylvain Petitjean - Sylvain.Petitjean@inria.fr
- **Context:**
  This master's project deals with the robustness of geometric programs w.r.t. numerical degeneracies: Some classical tasks in geometric modeling, including computing the Delaunay triangulation, computing the intersection between surfacic and/or volumetric meshes, and meshing using Voronoi diagrams are examples involving such geometric programs. A geometric programs take as an input a set of objects (points, lines, segments, triangles ...) and return a combinatorial structure (e.g. a mesh) that inter-connects the elements of the input. "Geometric predicates" are central components of geometric algorithms. They are functions that take as input a small set of geometric objects and return a binary e.g., left/right, above/below (or ternary, e.g. left/on/right) answer. Using standard floating point arithmetics for these predicate sometimes lead to inacurate - and more importantly unconsistent - results. Several techniques replace standard floating points with arbitrary-precision number types, thus overcoming the lack of precision encountered with floating points (see e.g. my survey in [1]).
- **Subject:**
  The implementation of such arbitrary precision arithmetics is quite involved, and error-prone. The goal of this project is to contribute to the on-going development of a compiler, that transforms the mathematical expression of a predicate into a C++ program that evaluates the predicates. In particular, it would be interesting to strengthen our existing "Predicate Construction Kit" compiler with some means of certification, to prove that the generated code computes the exact sign of the mathematical expression. The Master student will study different means of proving the bounds for the computed expression, including using several existing Coq packages (Gasper) or Haskell.
- **References:**
  [1] http://www.loria.fr/~levy/PCK/PCK_paper.pdf
  http://www.loria.fr/~levy/PCK/PCK_slides.pdf

- **Required Skills:**
  Ideally, the student will have a good knowledge of both computer arithmetics and software certification techniques. A taste for efficient computer implementation is also appreciated.