Rank: A Tool to Check Program Termination and Computational Complexity

Christophe Alias*, Alain Darte*, Paul Feautrier*, and Laure Gonnord†
*Compsys, LIP, UMR 5668 CNRS, INRIA, ENS-Lyon, UCB-Lyon
Email: firstname.lastname@ens-lyon.fr
†LIFL, Université Lille I, Email: Laure.Gonnord@lifl.fr

Introduction. Proving the termination of a flowchart program can be done by exhibiting a ranking function, i.e., a function from the program states to a well-founded set, which strictly decreases at each program step. In a previous paper published at SAS'10, we proposed an efficient algorithm to compute multidimensional affine ranking functions on flowcharts of arbitrary structure. Our second contribution was to show how to use the ranking functions we generate to get upper bounds for the computational complexity (number of transitions) of the source program. This estimate is a polynomial, which means that we can handle programs with more than linear complexity. This short abstract aims at presenting our prior work on RANK, the tool implementing our algorithm.

Our tool. RANK starts from a C program, translated as an integer interpreted automaton with state invariants. RANK tries to prove the termination of the program by computing (multidimensional affine) ranking functions. Two cases arise:

- In case of success, RANK computes the **worst-case computational complexity** of the program.
- In case of failure, RANK tries to exhibit an **input that causes non-termination**.

The termination part requires to solve large ILP programs. This is achieved thanks to the Piplib library (http://www.piplib.org/). The dimension of the ILP grows with the square of the program size. By construction, many useless variables are introduced (as a result of the application of Farkas lemma). Some of them can be eliminated by Gaussian elimination, and the remaining with Fourier-Motzkin projection.

The non-termination part consists in detecting infinite loops with ILP as well. Usually, the ILPs involved are reasonnably small and can be solved directly. This feature appears to be very useful, and helps us to debug some of our test programs which were unexpectedly non-terminating (example of "bug": precondition $p, q \geq 0$ missing in $\gcd(p, q)$).

The WCCC part requires counting the number of points into a $\mathbb{Z}$-polyhedron. This is done thanks to the Ehrhart polynomial part of the Polylib library (http://icps.u-strasbg.fr/polylib). The final result is a set of Ehrhart polynomials, guarded by affine predicates on program parameters.

RANK can be tested online at the url:

http://compsys-tools.ens-lyon.fr/rank

RANK has been tested successfully on examples collected from the extent literature, and provided on the web page.

Discussion. All in all, RANK is bounded by (i) the cost of ILP and (ii) the precision of state invariants. Surprisingly, (ii) was precise enough to solve most of our termination problems. Most of the failures come from the approximations made while translating the program to an integer interpreted automaton, when control structures involve non-affine expressions. Some properties could be inferred, as the non-negativity of a square. (i) can be reduced by analyzing the program in a modular fashion. The program can be split into a set of slices, which can be analyzed separately. Such an approach would allow to explore the trade-off between the size of the slices (impacting (i)), and the precision of the invariants (impacting (ii)).