Rank: A Tool to Check Program Termination and Computational Complexity (Extended Abstract)

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 that strictly decreases at each program step. In a previous paper [1], we proposed an algorithm to compute multidimensional affine ranking functions for flowcharts of arbitrary structure. Our method, although greedy, is provably complete for the class of rankings we consider. The ranking functions we generate can also be used 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 abstract aims at presenting RANK, the tool that implements 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** (WCCC) 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 Piplib (http://www.piplib.org/). Some tricks must be applied to avoid a complexity blow-up. For example, by construction, many useless variables are introduced (as a result of the application of Farkas lemma). Some of them can be eliminated by Fourier-Motzkin elimination.

The non-termination part consists in detecting infinite loops with ILP as well. Usually, the ILPs involved are reasonably 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 integer points in a polyhedron. This is done thanks to Polylib (http://icps.u-strasbg.fr/polylib). The result is a set of *Ehrhart polynomials*, guarded by affine predicates on program parameters.

RANK is available through a web demonstrator at the url:

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

It has been tested successfully on examples collected from the literature and provided on the web page too.

Discussion. All in all, RANK is bounded by (i) the cost of ILP and (ii) the precision of state invariants. Surprisingly, (ii) was accurate 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, such as the non-negativity of a square. The impact of (i) can be reduced by analyzing the program in a modular fashion. The program can be split into slices, which can be analyzed separately. Such an approach can explore the trade-off between the size of the slices (impacting (i)), and the precision of the invariants (impacting (ii)).

[1] C. Alias *et al*. Multi-Dimensional Rankings, Program Termination, and Complexity Bounds of Flowchart Programs. In *SAS'10*, pages 117–133, 2010.