A Logical Approach to Program Verification

Tudor Jebelean

Research Institute for Symbolic Computation
Johannes Kepler University, Linz, Austria

We approach the generation of the verification conditions for the total correctness of both functional as well as imperative programs in a purely logical manner: The conditions must ensure the existance and uniqueness of the function implemented by the program.

We assume that the program is based on an existing logical *object theory* which contains the properties of the objects used in the program (integers, rationals, tuples, etc.). When a new function is implemented, then its specification (input and output conditions) are added to the theory, together with the respective (new!) function symbol. Note that, since the program is using formulae and terms from this theory, no translation of these is necessary.

Functional programs are just syntactic versions of the logical formulae which constitute the implicit (because of recursion) definition of the function implemented by the program. *Imperative programs* are meta-terms containing the usual imperative programming constructs. We use a natural and simple transformation of imperative programs into functional programs, thus defining their semantics. This transformation as well as the generation of the verification conditions are fully formalized in predicat logic as meta-level functions. They use *forward symbolic execution*: the values of the current variables are computed symbolically, and the branching statements generate various path conditions.

There are three kinds of verification conditions: for *coherence*, for *functional* correctness, and for *termination*. The coherence conditions insure that actual arguments of each function call fulfill the input condition of that function. The functional conditions insure that the output of the program fulfills the output condition of the function. The termination conditions are certain induction principles built according to the structure of the recursive calls (respectively of the iterative loops). The termination conditions, together with the coherence conditions, allow to construct the proof (at object level!) of the formula stating the existence and uniqueness of the function implemented by the program. Together with the functional conditions, they also allow to prove the total correctness.

Our meta-model does not include a special abstract environment for defining the behaviour of programs, but everything is based on the logical approach described above. We can prove at meta-level that the necessity and sufficience of the conditions. This facilitates a future automated approach to the correctness proof of the verification conditions generator, and possibly a self-reflective approach (since the generator is also a program).