Avionic Software Verification by Abstract Interpretation

Patrick COUSOT

École normale supérieure, Département d'informatique
45 rue d'Ulm, 75230 Paris cedex 05 (France)
Patrick.Cousot@ens.fr, www.di.ens.fr/~cousot

An flight control surface actuation system in avionics is safety critical and complex since it is placed between the pilot's controls (sidesticks, rudder pedals) and the control surfaces of the aircraft, whose movement it controls and monitors. For reliability and dependability, several redundant software and computers are used but each one must be proved to be correct. With the exponential increase of the power of computers, the flight control software has become much more powerful hence complex. Since the cost of tests increasing more rapidly that the size of programs, formal methods become an attractive complement for program verification.

The difficulties with formal methods are that they need a formal specification, a formal semantics of the programming language and, because of undecidability, have serious limits in the automatic verification that the program semantics satisfies the specification. A theorem prover needs human assistance while model-checking requires finite models which, but for hardware, are generally incomplete.

Static analysis offers an interesting completely automatic alternative in that the specification can be chosen to be implicit. For example the absence of runtime error with not require the user to define a complex specification. Moreover static analysis consider infinite models of computations that can be directly computed from the program text so that the end-user does not need to provide a (finite) model of the program computations and environment. Finally, the reachable states during any program computation are computed approximately through an overapproximation that omits no possible case. So the delicate questions about the program semantics can be solved by considering all possible alternatives.

If this overapproximation copes with the undecidability problem, its inconvenience is that it considers spurious executions, which does not exist in any actual execution, but may be at the origin of false alarms. This approach is sound in that no runtime error will ever be omitted in the diagnostic. It is incomplete, because of the potential false alarms. The whole problem is therefore to choose abstractions of the program semantics that yield few or no alarm. To do so, one can restrict the family of programs to be analyzed so as to adapt the abstraction exactly the domain-specific primitives and algorithmic schema found in this family of programs.

We will present elements of the theory of abstract interpretation on which the soundness of the notion of overapproximation does rely. Then we will introduce the ASTRÉE static analyzer (www.astree.ens.fr), which is specialized for the verification of the absence of runtime errors in control-command programs. The analyzer has general-purpose abstractions (e.g. octagons, control and data partitioning) as well as domain-specific abstractions (e.g. to handle filters or the potential accumulation of rounding errors). Finally, we will report on the successful application of ASTRÉE to the proof of absence of runtime errors in recent avionic flight control software.