AIAA Infotech@Aerospace 2010
20 - 22 April 2010, Atlanta, Georgia

AIAA 2010-3385

# Static Analysis and Verification of Aerospace Software by Abstract Interpretation

# (Abstract)

Julien Bertrane*
*École normale supérieure, Paris*

Patrick Cousot*,†
*Courant Institute of Mathematical Sciences, NYU, New York & École normale supérieure, Paris*

Radhia Cousot*
*École normale supérieure & CNRS, Paris*

Jérôme Feret*
*École normale supérieure & INRIA, Paris*

Laurent Mauborgne*,‡
*École normale supérieure, Paris & IMDEA Software, Madrid*

Antoine Miné*
*École normale supérieure & CNRS, Paris*

Xavier Rival*
*École normale supérieure & INRIA, Paris*

The validation of software checks informally (e.g., by code reviews or tests) the conformance of the software executions to a specification. More rigorously, the verification of software proves formally the conformance of the software semantics (that is, the set of all possible executions in all possible environments) to a specification. It is of course difficult to design a sound semantics, to get a rigorous description of all execution environments, to derive an automatically exploitable specification from informal natural language requirements, and to completely automatize the formal conformance proof (which is undecidable). In model-based design, the software is often generated automatically from the model so that the certification of the software requires the validation or verification of the model plus that of the translation into an executable software (through compiler verification or translation validation). Moreover, the model is often considered to be the specification, so there is no specification of the specification, hence no other possible conformance check. These difficulties show that fully automatic rigorous verification of complex software is very challenging and perfection is impossible.

We present abstract interpretation[1] and show how its principles can be successfully applied to cope with the above-mentioned difficulties inherent to formal verification.

- First, semantics and execution environments can be precisely formalized at different levels of abstraction, so as to correspond to a pertinent level of description as required for the formal verification.
- Second, semantics and execution environments can be over-approximated, since it is always sound to consider, in the verification process, more executions and environments than actually occurring in real executions of the software. It is crucial for soundness, however, to never omit any of them, even rare events. For example, floating-point operations incur rounding (to nearest, towards 0, plus or minus infinity) and, in the absence of precise knowledge of the execution environment, one must consider the

*École normale supérieure, Département d'informatique, 45 rue d'Ulm, 75230 Paris cedex 05, First.Last@ens.fr.
†Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street New York, N.Y. 10012-1185, pcousot@cs.nyu.edu.
‡Fundación IMDEA Software, Facultad de Informática (UPM), Campus Montegancedo, 28660-Boadilla del Monte, Madrid, Spain.