Formal Verification

Ana Borges, Quim Casals, Juan Conejero,
Mireia González, Eduardo Hermo and Joost J. Joosten

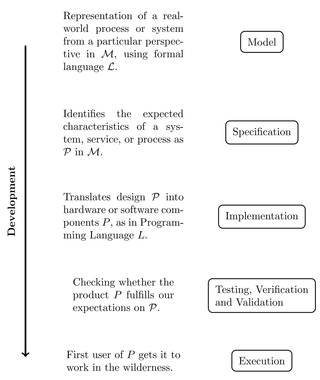
Formal Analysis

Consists of mathematical methods applied to the better understanding of the program's behaviour.

It usually involves three types of activities, namely system modeling, formal specifications and formal verification.

- When modeling one seeks to develop an abstract representation of our program.
- The specification defines the properties the program has to satisfy, sometimes characterizing the full abstract model.
- Finally, the verification ensures that our model complies with the specification.

Birth of P



Formal Verification

We intend to prove or disprove the correctness of the program with respect to our specification, through a model of our implementation. There are three kinds of methods:

- *Model Checking* explores all possible behaviours of a model of P in formal language L. If the property does not hold, it generates a counterexample. Model Checking achieves highly efficient and fully automated verification, but it is limited by the scale and complexity of the model.
- *Theorem Proving* models the system P and the specification $\mathcal{P}$ as logic formulae and proves through deductive methods that, in $\mathcal{M}$, P entails $\mathcal{P}$. Using inductive methods to describe the behavior and attributes of the program can solve the "state explosion" problem but cannot be automated at present.
- *Static Analysis* models the semantics of L and determines some properties of P through abstract interpretation.

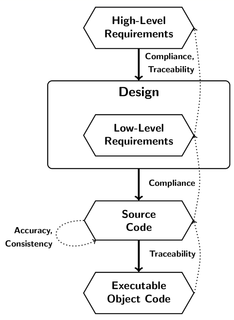
An Example

Airborne is a safety-critical software, so requiring the compliance of DO-178c standards, using formal methods as part of the validation process. The paper[1] shows the process of verifying such software with formal methods following the DO-333 standard from DO-178c. The process consisted of four stages:

- Formal analysis of requirements.
- Compliance from source code to requirements.
- Static analysis of source code.
- Formal analysis of the executable.

They verified 1049 properties, finding 16 errors from 54 potentials errors.

DO-333 (simplified)



FV Public Certification

In Formal Vindications we believe algorithmic law deserves a high reliability, requiring formal methods for its validation.

Our team is developing a Public Certification Method in order to certify formally verified software. Certified software must include the following 4 basic elements written in a formal language (our company's choice is Coq):

- A formal specification.
- Its implementation.
- A mathematical proof that the implementation fulfills the specification.
- The automatically generated executable, along with the method used for extraction.

As the certified software will be used by humans and a complete and sound translation of natural language is not attainable, a 5th element is customary.

- An interpretation, containing three different abstraction levels of language.

References

[1] Zongyu Cao, Wanyou Lv, Yanhong Huang, Jianqi Shi and Qin Li
Formal Analysis and Verification of Airborne Software Based on DO-333

Proyecto RTC-2017-6740-7 financiado por MCIN/AEI/10.13039/501100011033 y por FEDER.

Universitat de Barcelona

Formal Vindications

FVTM — FV Time Manager powered by Formally Verified Time Library

Guretruck

Generalitat de Catalunya

Ministerio de Ciencia e Innovación

Unión Europea — Fondo Europeo de Desarrollo Regional — "Una manera de hacer Europa"

Agencia Estatal de Investigación

Bosch i Gimpera — Universitat de Barcelona

Doctorats Industrials — talent alliance!