Formal verification of an UAV autopilot

JDD @ DISC

B. Pollien[1], C. Garion[1], G. Hattenberger[2], P. Roux[3], X. Thirioux[1]

October 21, 2021

[1]ISAE-SUPAERO, [2]ENAC and [3]ONERA

Context

The developement of a system can be dived in 3 steps:

1. **Specification** of the functionnal needs and constraints.
2. **Implementation** of the system.
3. **Verification** that the implementation respects the specification.

Verification methods:

- Code reviews
- Tests,
- Formal methods.

Context

Formal methods

- Verification techniques based on mathematical models
- Provides stronger guarantees but with some cost
- Recommended in avionics with DO-178C and DO-333 standards
- Examples: abstract interpretation, deductive methods, model-checking

The goals of my PhD

- Define verification processes that use formal methods,
- Apply these methods to a drone autopilot: Paparazzi.

Presentation of Paparazzi

Paparazzi is an autopilot for micro-drones

- Developed at ENAC since 2003,
- Open-Source under GPL license.

Complete drone control system:

- Offers the control software part,
- Also offers some designs of hardware components,
- Supports for ground and aerial vehicles,
- Supports for simultaneous control of several drones.

Paparazzi as a formal method subject of study?

Paparazzi is a good candidate for testing if formal methods are usable/efficient:

- the code is written:
    - without verification purpose,
    - by good C programmers,
    - with use of classic C idioms (pointers etc).
- the code base is consequent ($\sim$ 350k loc).

First results

Analysis of a mathematical library of Paparazzi:

- Using Frama-C,
- Checking for the absence of runtime errors,
- Verification of some functional properties,
- Without modifying the code.

Frama-C

frama C

Software Analyzers

Frama-C is a C code analysis tool

- Developed by CEA and Inria,
- Modular, which supports different analysis methods
  ex: static analysis with EVA or dynamic analysis with E-ACSL.

Verification process of a C program using Frama-C:

1. Code specification with **ACSL** (*ANSI C Specification Language*),
2. Generation of the abstract syntax tree of the analyzed code,
3. Analysis of the tree by the plugins
   $\Longrightarrow$ Verify if the specification is respected.

Note: We used RTE, WP and EVA plugins.

Formal verification with Frama-C

The **goals** is to determine the minimum contracts for the functions of the library in order to guarantee the absence of runtime errors and some functional properties:

- Runtime errors: Dereferencing an invalid pointer, division by 0, overflows, non finite float value, ...
- Functional properties: Offer guarantees on the behavior or the result of a function.

$\Longrightarrow$ Approximately 3,500 lines of annotation.

gitlab.isae-supaero.fr/b.pollien/paparazzi-frama-c

Current works: certified flight plan generator

Paparazzi Flight Plan generator:

- Input: XML describing a flight plan.
- Ouput: Embedded C code.

Goals: Rewrite the OCaml generator in Coq

- Add new features
- Verification of the preservation of the semantics

Summary

Formal verification of an UAV autopilot

Study case: Paparazzi

Work done:

- Technical report: Formal verification for autopilots: preliminary state of the start
- Verification of some parts of Paparazzi mathematical library
  Publications: AFADL 2021, FMICS 2021

Current work:

- Developement of a certified flight plan generator

Thank you