

Formal Methods for the Analysis of Critical Control Systems Models: Combining Non-Linear and Linear Analyses

Adrien Champion
Rockwell Collins France

Rémi Delmas
ONERA

Michael Dierkes
Rockwell Collins France

Pierre-loic Garoche
ONERA

Romain Jobredeaux
Georgia Tech Univ and Univ of Toulouse

Pierre Roux
ONERA

ABSTRACT

Critical control systems are often built as a combination of a control core with safety mechanisms allowing to recover from failures. For example a PID controller used with triplicated inputs. Typically those systems would be designed at the model level in a synchronous language like Lustre or Simulink, and their code automatically generated from those models.

In previous SAE symposium, we addressed the formal analysis of such systems - focusing on the safety parts - using a combination of formal techniques, ie. k-induction and abstract interpretation.

The approach developed here extends the analysis of the system to the control core. We present a new analysis framework combining the analysis of open-loop stable controller with those safety constructs. We introduce the basic analysis approaches: abstract interpretation synthesizing quadratic invariants and backward analysis based on quantifier elimination. Then we apply it on a simple but representative example that no other available state-of-the-art technique is able to analyze.

This contribution is another step towards early use of formal methods for critical embedded software such as the ones of the aerospace industry.

CITATION: Champion, A., Delmas, R., Dierkes, M., Garoche, P. et al., "Formal Methods for the Analysis of Critical Control Systems Models: Combining Non-Linear and Linear Analyses," *SAE Int. J. Aerosp.* 6(1):2013, doi: 10.4271/2013-01-2109.

ADDRESSING THE NEED OF CONTROL-COMMAND SOFTWARE FOCUSED ANALYSES TO ANSWER V&V AND CERTIFICATION NEEDS

The aerospace industry is notoriously faced with highly critical issues. The safety of its system should be guaranteed even if the cost of such safety is huge. In [FBG[±]], figures of the development costs of the Boeing 777 are detailed. In these, software accounts for a third of all development costs. In this third, 70% consists in verification costs while only 30% are devoted to software development. Other aircraft manufacturers have similar figures.

The software specific certification regulatory document, ie. the recently updated DO 178-C, characterizes different levels of criticality from level A - the most critical - to level E - the less one. Depending on the identified level, verification and validation activities are more or less intensive and therefore costly. This certification document has recently been updated and it also provides a formal methods supplement, identified as RTCA DO 333. This supplement explicitly enables the use of formal methods for critical embedded software.

Among the various systems of an aircraft, and their associated software, one of the most critical is the control-command system of the aircraft. Addressing specifically this issue of verifying such software seems to be a pertinent goal: proposing new ways to validate it could both increasing the truth we have in the released software and reduce the cost of V & V by providing more automatic (and exhaustive) analysis means.

Those reactive system could be seen as the composition of two parts. A first one is the computation core itself achieving the main objective of the software: controlling the aircraft by receiving inputs from sensors and moving the aircraft through actuators. A second part tries to handle any possible failure of this core system. This safety architecture is mainly based on redundancy and information fusion. Those two parts are usually defined using a model based development.

The approach of control system modeling as proposed by the MathWorks with MATLAB Simulink, by Esterel Technologies with the SCADE language or by the academic community with the Lustre language, is extensively used for reactive systems design and often allows the automatic generation of the embedded code. However, despite the existence of a few formal verification tools supporting these languages, few system builders actually rely on formal approaches to demonstrate safety properties of their software products.

Our goal is to support the verification and validation of such systems for all their specification, at the various stages of their development. This paper focus on a representative running example: a controller for a physical device together

with triplication of its inputs and voters. We show how we propose to analyze such systems by composing new generation formal methods analyses.

The paper is structured as follow: a first section recalls the state of the art of formal methods in that context; section 4 briefly describes our proposal based on combination of analyses; section 5 presents in details our running example; sections 6 and 7 introduce our two new automatic analyses; section 8 illustrates the use of our techniques on the example; while section 9 concludes.

STATE OF THE ART OF FORMAL METHODS

Since the early 60s, researchers have proposed multiple theoretical frameworks to analyze systems and programs. These techniques, based on formal foundations - mainly discrete mathematics, algebra - allow the exhaustive study of all the behaviors of some categories of systems, as opposed to the test or simulation methodologies which only cover the system traces identified by the run tests.

We briefly focus here on relevant techniques for the analysis of functional specification of control systems. Those techniques are currently used in the aerospace industry, at different level of TRL¹, from early academic work driven by industry needs to actual transferred technologies[SFF04].

Deductive Methods

The basic ideas behind deductive methods where introduced in the late 60s by Floyd and Hoare [Flo67, Hoa69, Dijkstra]. This approach was later extended by Dijkstra and became known as the weakest precondition technique.

A piece of code is annotated with its specification as a Hoare Triple (precondition, code, postcondition). Then, a set of predicate transformations is applied to transform the postcondition into the weakest precondition (wp) of the code that entails the postcondition after the code execution. It only remains to prove that precondition $\Rightarrow$ wp(code, postcondition) to prove that the postcondition is always satisfied.

This approach is currently used at Airbus[SFF04] as a replacement for some of the unit tests, covering all possible behaviors of the code. Existing tools often target specifications of imperative code. The main issues with imperative code being to handle complex memory models and properties over structures in the heap, manipulated by algorithms.

This approach is not currently supported at model level. However associating a Hoare triple to each of the model blocks or to most of them, is a needed step to later perform formal analysis of this specification.

¹Technology Readiness Levels as defined by NASA; see http://esto.nasa.gov/files/trl_definitions.pdf

Abstract Interpretation

Another method, abstract interpretation, is very efficient to discover numerical invariants over programs.

The basic principle of this static analysis technique is to automatically compute an over-approximation of the set of all behaviors of the program (i.e. its semantics). This over-approximation is computed thanks to *abstract domains*. The role of abstract domains is twofold:

1. they characterize the nature of the overapproximation which is performed,
2. they are equipped with a set of functions, the *abstract primitives*, which allow to compute the abstract semantics.

Each abstract domain is associated with a given trade off between precision (depending on the kind of properties that it can infer) and efficiency (related to the computational complexity of its abstract primitives).

Many different abstract domains have been defined in the literature. Each of them is able to infer a particular class of properties over a given set $v_1, \dots, v_d$ of variables involved in the analyzed program. Some abstract domains allow to infer numerical properties. Among the most well known, we find:

- the interval abstract domain[CC77], which infers invariants of the form $a \leq v_i \leq b$, $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$ for every variable v_i ,
- the abstract domain of zones [Min01a], which expresses properties of the form $v_i - v_j \leq c$,
- the domain of octagons[Min01b], which provides invariants of the form $\pm v_i \pm v_j \leq c$
- the Karr domain[Kar76, MOS04], which infers affine equality invariants $\sum_i \alpha_i v_i = \beta$,
- the domain of closed convex polyhedra[CH78], which provides affine inequality properties $\sum_i \alpha_i v_i \leq \beta$,
- the domains based on tropical polyhedra, which express max- and/or min-invariants[AGG08, All09].

Other kinds of abstract domains allow to infer symbolic invariants on the program, such as memory shape or pointer aliasing properties (see for instance[Ste96, SRW98, BCC[±]07]).

A recent success story of abstract interpretation is the complete analysis of the flight control systems of the Airbus A380[KWN[±]10], guaranteeing that no runtime error (out of bounds memory accesses or arithmetic exceptions) can occur. As for deductive methods, this static analysis technique targets the imperative, embedded C code, and not the models from which this code was derived.

One of the domains used to perform this analysis is specifically focused on second order linear filters. This domain is able to precisely over-approximate their set of reachable states. We present in this paper another similar abstraction that outperforms it in terms of expressivity.

Satisfiability Modulo Theory Decision Procedures

Satisfiability Modulo Theories solvers are decision procedures for logical theories in which some atoms belong to certain decidable first order theories such as linear real/integer arithmetic, the theory of bit-vectors, the theory of arrays (with read over write axioms), etc. Roughly speaking, these procedures are usually built by extending boolean satisfiability procedures with a combination of dedicated background theory solvers[Tin10]. There are two main paradigms for combining theories: The Nelson-Oppen combination procedure[NO79] and Shostak like algorithms[Sho84, RS01, CCKL08].

SMT-solvers are used as backend reasoning engines in a wide range of formal verification applications, such as weakest precondition calculus, bounded model checking, k-induction, test case generation, etc. Recently, the SMT-lib initiative (<http://www.smtlib.org/>) has gathered major SMT-solver developers around a standardized formula and solver command language. The SMT-lib 2.0 standard introduced specific features easing the implementation of incremental verification approaches like BMC or k-induction.

Quantifier Elimination

Quantifier elimination is a fundamental topic in mathematics, which enjoys many applications in formal verification, controller synthesis, preimage computation, automatic program abstraction, etc. [DDH[±]06] [ST11] [Mon10a].

Assuming a first order formula over boolean, real or integer variables $\exists x, F(x; y_1, \dots, y_n)$, where F is quantifierfree, quantifier elimination allows to generate a new formula $G(y_1, \dots, y_n)$ equisatisfiable to $\exists x, F(x, y_1, \dots, y_n)$ by *eliminating* the quantified variable x from F . Slightly rephrased, quantifier elimination generates a condition G on variables $y_1, \dots, y_n$ which, when satisfied, entails the existence of an x such that $F(x, y_1, \dots, y_n)$ is also satisfied.

In 1948, Tarski [Tar51] proved the decidability of the first order theory of real closed fields by exhibiting a (intractable) quantifier elimination procedure. In 1975, Collins [Col75] published the quantifier elimination method by Cylindrical Algebraic Decomposition, followed in 1997 by Weispfenning with the method of Virtual Substitution [Wei97]. These general QE methods allow to handle polynomial formulas over real variables and have extremely high computational complexities, limiting their practical applications to systems with very few state variables.

A large body of research studies highly efficient QE algorithms for linear fragments of integer and real theories. The most recent advances for linear QE combine state of the art SMT-solving with polyhedral projection [Mon10b, PBM12]. It is these linear SMT-based methods we are studying and using in our formal verification framework.

Deployment of Formal Methods in the Industry

It took some time for these techniques to mature enough to be transferred in the industry practices, but they become more and more available to R&D engineers or advanced users. As mentioned in the previous section, new certification standards (DO-178C, ISO26262, EN50128) highly recommend formal methods, and furthermore, formal methods are more and more required by companies themselves (especially in the railway industry) from their system providers. Formal methods can thus be truly considered as a key technological advantage on the critical systems market. Thanks to long term research efforts, formal methods have matured up to the point they were found, by industrials, to be helpful in dealing with the difficulties arising from highly complex system designs, and enabled system providers to meet the requirements of certification which are to:

- provide evidence of the system safety, and
- master the overall product life-cycle.

OUR PROPOSAL: A FRAMEWORK TO COMBINE FORMAL METHODS TOOLS RESULTS

In [CDGR11] we introduced our analysis framework that combines different formal verification approaches, namely abstract interpretation and k-induction, to analyze safety properties of control systems. The illustrative example was then a piece of the safety architecture, a typical counter as used in the alarms logic of such critical embedded systems.

The current paper elaborates on that framework to focus on parts of the software that contain more numerical computations and seem clearly out of reach of current existing formal methods.

The core framework is the same: multiple solvers are run in parallel on the same system model with the same set of proof objectives. The framework provides means to communicate between tools. Formal methods could exchange lemmas, ie real invariants, tentative invariants, counterexamples and also analysis parameters (see [KGTW12, HZJO[±]12, GKT13, CDD12] for more details). For example the counter-example of a k-induction process for a given proof objective could later drive the analysis by abstract interpretation of the same system.

This framework targets data flow descriptions of the system, as often used in the aerospace industry with Matlab Simulink or Esterel Scade. The system is describe in the synchronous language Lustre and translated by the framework to each tool input language.

This paper introduces two extensions:

- a new abstraction for our abstract interpreter engine that is suited for expressing stability of linear or quasilinear systems² that are stable in open-loop analysis.
- a new model-checking approach based on underlying SMT solvers that computes a backward analysis, over-approximating set of co-reachable states from a given proof objective.

Both extensions require none to minimal parametrization and showed great results on hard problems.

RUNNING EXAMPLE: A COUPLED MASS SYSTEM LINEAR CONTROLLER BEHIND TRIPLICATED INPUTS WITH SATURATIONS

Throughout this article, we consider the control of the coupled mass system³ shown in [figure 1](#). Such coupling can be used to model numerous physical phenomena such as vibration propagation patterns in fluids and flexible structures, among other applications.

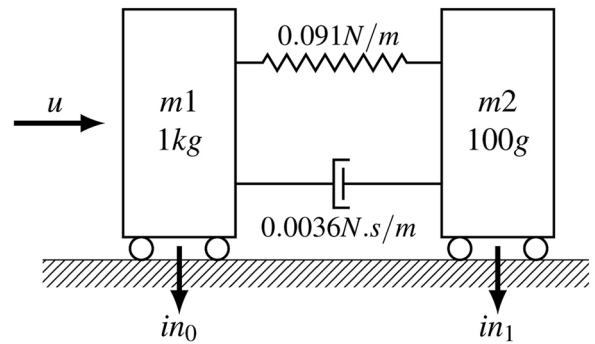


Figure 1. Coupled mass system - running example.

Controller Design

For the design of a controller for the system, we assume the output variables are x_1 and x_2 , the displacement of mass 1 and mass 2 with respect to their equilibrium position.

The continuous model for the plant is as follows:

$$\begin{aligned} \dot{x}_p &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ -K/m & -B/m & K/M & B/M \\ 0 & 0 & 0 & 1 \\ K/M & B/M & -K/M & -B/M \end{bmatrix} x_p \\ y_p &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x_p + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1/M \end{bmatrix} u_p \end{aligned}$$

²We denote by quasi-linear systems, systems that are defined by a linear core with saturations.

³This system is extracted from [Row04].