CMACS — NSF meeting, Pittsburgh, 2010/3/4 – 5

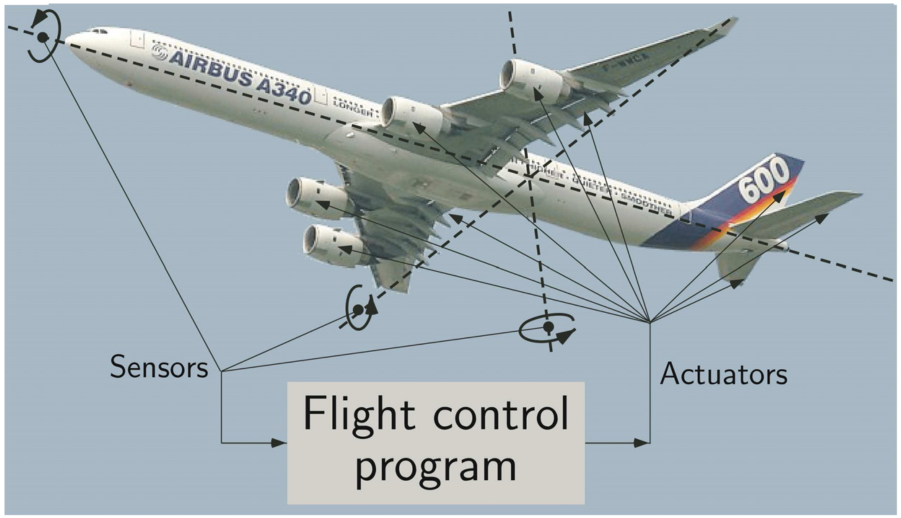
Challenge Problems in Aerospace Software Verification

Patrick Cousot

March 4, 2010

Motivation

Computer controlled systems



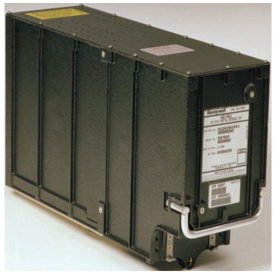
Example of bug report

"The Australian Transport Safety Bureau (ATSB) found that the main probable cause of this incident was a *latent software error* which allowed the ADIRU to use data from a failed accelerometer"

`http://www.atsb.gov.au/newsroom/2008/release/2008_43.aspx`,
`http://en.wikipedia.org/wiki/Qantas_Flight_72`
ADIRU = Air Data Inertial Reference Unit (provides air speed, altitude & position)

- The initial effects of the fault were:
  - false stall and overspeed warnings
  - loss of attitude information on the Captain's primary flight display
  - several Electronic Centralised Aircraft Monitoring (ECAM) system warnings
- About two minutes later, ADIRU #1, which was providing data to the captain's primary flight display, provided very high (and false) indications for the aircraft's angle of attack, leading to:
  - the flight control computers commanding a nose-down aircraft movement, which resulted in the aircraft pitching down to a maximum of about 8.5 degrees,
  - the triggering of a Flight Control Primary Computer pitch fault.
- On 15 January 2009 the EASA issued an Emergency Airworthiness Directive to address the above A330 and A340 Northrop-Grumman ADIRU problem of incorrectly responding to a defective inertial reference.

- A memo leaked from Airbus on the Flight 447 (from Rio de Janeiro, Brazil, to Paris, that crashed into the Atlantic Ocean on 1 June 2009) suggests that there was no evidence that Honeywell manufactured ADIRU malfunction was similar to the failure of the Northrop Grumman manufactured ADIRU in Qantas flight incidents

An Air Data Inertial Reference Unit (ADIRU) is a key component of the integrated Air Data Inertial Reference System (ADIRS), that supplies air data (airspeed, angle of attack and altitude) and inertial reference (position and attitude) information to the pilots' Electronic Flight Instrument System displays as well as other systems on the aircraft such as the engines, autopilot, flight control and landing gear systems

Examples of breakthroughs in MC for avionics

Example of successful application

- ADGS- 2100 Adaptive Display and Guidance System Window Manager [1]

- Five components analyzed independently, 9.8×10^{9} to 1.5×10^{37} states, boolean, enumeration type and small integer types
- 563 properties checked, 98 errors found

[1] S.P. Miller, M.W. Whalen, and D. Cofer, Software Model Checking Takes Off, CACM, 53(2), Feb. 2010, pp. 58 — 64

Example of promising application

- Effector Blender (EB) logic of an Operational flight program (OFP) for an Unmaned Aerial Vehicle (UAV) (1)
- 2000 basic Simulink blocks for generating the actuator commands for the 6 UAV control surfaces
- Specification: commands within dynamically computed limits
- Floats ➥ fixed-point ➥ integers (for SMT-solver): unsound
- OFP is too large model to include aircraft model
- Even OFP alone need to be decomposed into subsystems

(1) S.P. Miller, M.W. Whalen, and D. Cofer, Software Model Checking Takes Off, CACM, 53(2), Feb. 2010, pp. 58 — 64

MC characteristics

- Works on models (e.g. translated to Lustre – from which C programs can be generated)
- Check user-provided safety and liveness specifications (via very expressive temporal logics)
- Universal representations of properties (set enumeration/BDDs/predicates) and models (transition systems)
- Fully automatic
- Counter-examples are provided for specification violations

Limits of MC tools

- Models (e.g. reals in Scade) not programs (e.g. floats)
- Requires a specification of the model (often as complex as the model, limited expressibility)
- Combinatorial state explosion :
  - Large models have to "be broken down into ... components analyzed individually"
  - Either sound by exhaustive verification with restriction to booleans, enumerated types, small integers, etc
  - Or, unsound bug-finding with partial exploration,

Examples of breakthroughs in AI for aerospace

AI breakthroughs

- Beyond research, static analyzers start being deployed in industrial production, e.g.:
  - Astrée [2]: absence of runtime errors in synchronous control/command programs



  - AiT [3]: worst case execution time analysis
  - Clousot [4]: static contract checking for .NET

[2] http://www.absint.de/astree/
[3] http://www.absint.de/ait/
[4] http://research.microsoft.com/apps/pubs/default.aspx?id=70614

Example of successful application [5]

- Verification of the absence of RTE in the electric flight control C code of the A380

- ≈ 1000,000 LOCS, 48h (15h on a quadriprocessor), no false alarm
- Example of non-linear domain-specific abstraction: numerical filters, cumulation of rounding errors for floats, etc

[5] http://www.astree.ens.fr/

Example of promising application

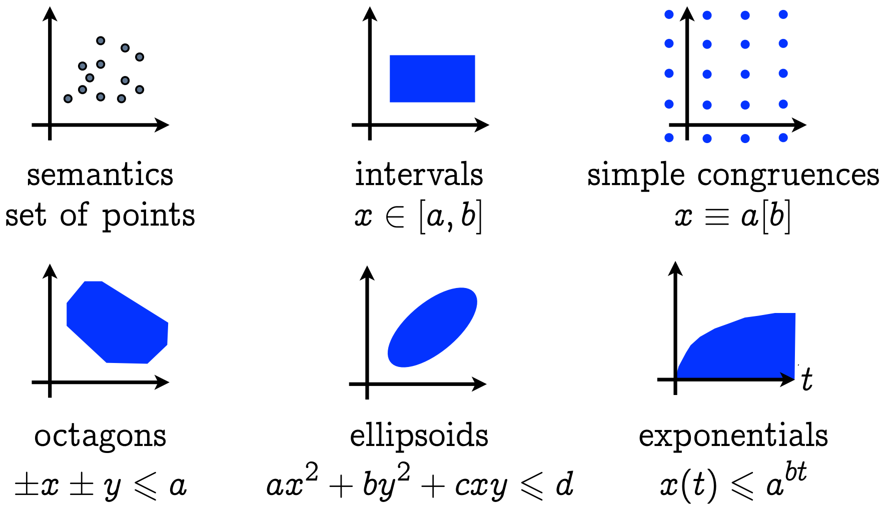
- Analysis of a parallel program (kernel of an actual C application on ARINC 653, leaving out SCADE and error recovery)

- 5 concurrent communicating processes, 100,000 LOCS, no decomposition needed, absence of RTE
- about 70 false alarms (essentially due to the total absence of input specifications or to the imprecision of the interference analysis)

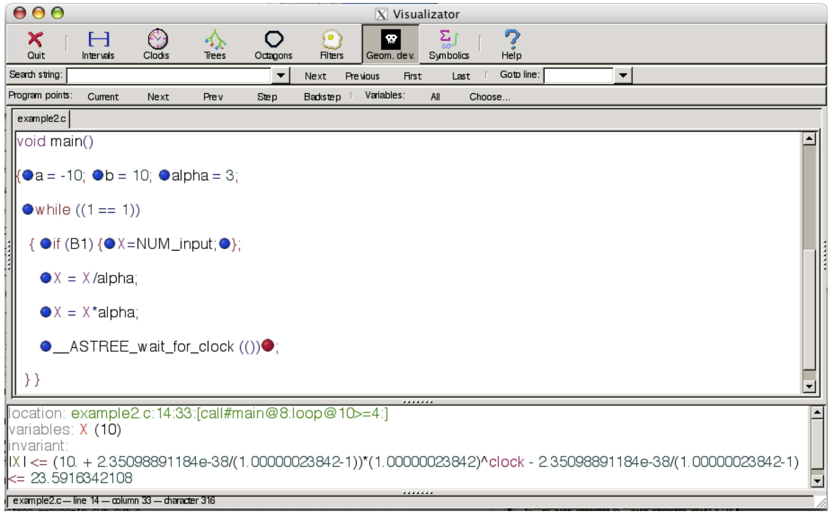
AI characteristics

- Consider a semantic model of programs (in programming or specification languages)
- Formally define the strongest properties of interest in the form of an infinite collecting semantics (e.g. traces, reachable states)
- Systematic design of a sound (weaker) abstract semantics by combining many abstractions of program properties in infinite abstract spaces
- Property inference by effective approximation of the abstract semantics (widening/narrowing)
- Use the abstract semantics to answer questions (e.g. *is the specification satisfied?* or *what is the interval of variation?*)
- Always sound, scales up, but sometimes incomplete (false alarms)
- False alarms can be reduced or even eliminated (= proof) by considering domain specific abstractions

Example of generic abstractions in Astrée



Example of analysis session with Astrée



Random input of the Boolean B1, the float NUM_input in [-10,10] at most 10h at 10ms clock tick. An exponential is used to bound the accumulation of rounding errors over time.

Limits of AI tools

- Semantics of real-life programming languages is hard to define
- Essentially trace-based safety properties (+ termination by reduction to safety) because of infinite systems
- Many tools are built for implicit specifications (e.g. RTE + `assert`), with important exceptions (e.g. Clousot), not mandatory (monitoring (9))
- Indecidability + Soundness ⇨ Incompleteness ⇨ False alarms
- No concrete counter-examples (only in the abstract, hard to concretize for hour-long error scenari)
- Cost/precision efficient balancing is domain specific (no “universal” abstraction for infinite systems)

Combining MC + AI

(MC U AI)*

- (MC U AI)*, would juxtapose MC and AI analyzes on common models and specifications
- e.g.
  - Finite abstractions of model by AI plus MC of finite model (predicate abstraction)
  - State space reduction by AI ahead of MC [8]
  - etc
- Great, but cumulates the limits of both MC & AI

---

[8] Patrick Cousot, Radhia Cousot: Refining Model Checking by Abstract Interpretation. Autom. Softw. Eng. 6(1): 69-95 (1999)

Combining MC + AI

- MC+AI looks like the ideal solution [9]
- But
  - Decidable MC + Undecidable AI = Undecidable!

    i.e. we must be able to analyze infinite models without reduction to a finite model or decidable models (which has fundamental limitations)
  - The key is *induction*: "easy" for safety (widening/narrowing), extremely difficult for liveness (dual narrowing)
  - Can only be a long term goal of the expedition (3/4 years)

---

[9] Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25

Challenges in MC + AI

- Liveness properties for infinite systems is the main difficulty
- A temporal logic is both certainly complex and not often used in its full generality and not expressive enough
- Directions:
  - Which (liveness) properties should we consider first in combining MC + AI?
  - We need examples of discrete/hybrid systems to be verified for this class of properties

Examples of challenges in aerospace

Challenge I: abstraction inference

- Control/command programs are generated from mathematical models
- The controller mathematical model is "hidden" in the control program (dt ➥ clock tick)
- By abstracting a simplified mathematical model from the program, we could
  - Study mathematically this simplified model
  - Which yields program-specific numerical abstractions for the program analysis
  - Hopefully, more precise than generic abstractions

A trivial example (for intuition)

```
x=1;
while (x<10) {
   x = x+3;
}
```

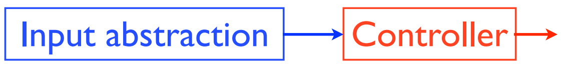
- A loop iteration in dt (as in synchronous control)
- A simple abstract interpretation yields

$$\frac{dx}{dt} = 3$$

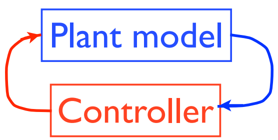
- so $x(t) = 3t + x0$ ($x0 = 1$ by the initial condition) i the appropriate abstraction
- A simple abstract interpretation shows $x \in [1,10]$
- Proving e.g. termination (preserved by abstraction)

Challenges in avionics (II): closing the loop

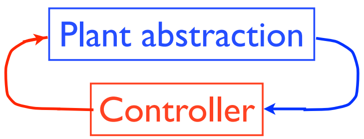
- Analysis of control/command applications (synchronous or asynchronous)
- Current state:



- Ideally:



- Useful approximation:



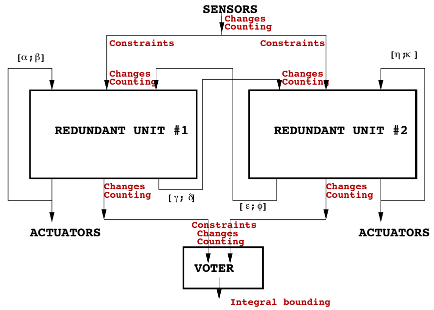
Challenge (II): reactive properties

- Abstraction of plant models:
  - To derive a sound abstraction of the plant behavior (output/input relation from differential equations), on one step
  - To be used in the analysis of reactive properties
  - We can start by thinking of plant models as differential equations / hybrid systems
- Directions:
  - We need to have plant models to abstract
  - We need to know which reactive properties are of interest?

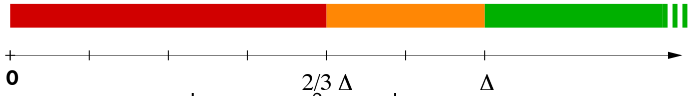
Challenge (III): imperfect synchrony

- The quasi-synchronism model may be wrong so simulation is unsound
- Imperfectly synchone (hardware) systems are very hard to test
- Automatic formal methods are the "only" alternative
- Directions:
  - Examples of imperfectly synchone systems are extremely rare in the academic world
  - Which are the main properties of interest?

Analysis of an imperfectly clocked system [10]



Specification : no alarm raised with a normal input



input stability $< \Delta$: counter-example	Between $\frac{2}{3} \times \Delta$ and Δ : ?	input stability $> \Delta$: the analyzer proves the specification

[10] Julien Bertrane: Proving the Properties of Communicating Imperfectly-Clocked Synchronous Systems. SAS 2006: 370-386

Challenge (IV): security analysis

- The computer network designed to give passengers in-flight internet access, is connected to the plane's control, navigation and communication systems, an FAA report reveals [11].

- Firewalls are extremely vulnerable
- Beyond internet, sound security analysis is also a challenge in avionics

[11] http://www.wired.com/politics/security/news/2008/01/dreamliner_security, citing http://cryptome.info/faa010208.htm

Conclusion

Conclusion

- Extending the scope of automatically verifiable properties for large infinite systems (liveness, security, quantitative, probabilistic, etc) is a grand challenge for FM
- Scaling up beyond synchrony eg to
  - Imperfect synchrony
  - Parallel programs

  is a big challenge, including in aerospace.
- Closing the discrete controller + continuous plant loop is a big challenge in the verification of complex control/ command systems