Assurance For Increasingly Autonomous (IA) Safety Critical Systems

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA

Introduction

- Increasing Autonomy (IA) is US airplane language for systems that employ machine learning (ML) and advanced/General AI (GAI) for flight assistance short of full autonomy
- Like driver assistance in cars below Level 5
- Cars and planes have different challenges, but also similarities
- I'll mostly use airplane examples because that's what I know
- Typical scenario for IA airplanes is single-pilot operation
  - e.g., Long flights with two pilots: one can sleep
  - While the other flies with assistance from "the box"
  - "The box" has to be more like a human copilot than conventional flight management or autopilot
  - So there's more to it than just automation

Basic Challenges

- Integration and autonomy
- Crew Resource Management
- Never Give Up
- Unconventional implementations (ML etc.)

I will focus on the last of these but I want to touch on the first three because they also have large impact on the structure of safety-critical flight systems and on their assurance

And they are consequences of IA

(Recall early history of Airbus A320)

Integration and Autonomy (Do More)

- If the IA box is like a copilot, it has to do the things that human pilots do
- Not just simple control, and sequencing tasks like A/P, FMS
- But things like: radio communications, interpreting weather data and making route adjustments, pilot monitoring (PM) tasks, shared tasks (flaps, gear), ground taxi, communication with cabin-crew (emergency evacuation)
- Currently, automation just does local things, and the pilot integrates them all to accomplish safe flight
- An IA system must be able to do the integration
- And have overall situation assessment
- Overall, it needs to do a lot more that current systems
- Same in cars

  (was just brakes and engine, now driver assistance)

Crew Resource Management (CRM)

- Since UA 173 Portland crash in 1978
- At all times, and especially in emergencies, tasks must be shared appropriately, clear coordination, listen to all opinions
- And someone must always be flying the plane
  - "I'll hold it straight and level while you trouble shoot"
  - "You've shut down the wrong engine" (cf. social distance)
- The box needs to participate in this
- Field of Explainable AI (EAI) contributes here, but...
- EAI typically assumes human is neutral, just needs to hear reasons, but in emergencies, human often fixed on wrong idea
  - cf. AI 855, Mumbai 1978
- So the box needs a theory of mind (model of other's beliefs)
  - Does fault diagnosis on it to find effective explanation
- Sometimes the human is right! So box needs to take advice
  - cf. QF 32, Singapore 2010

Never Give Up (NGU)

- Current automation gives up when things get difficult
- Dumps a difficult situation in the pilot's lap, without warning
- Human pilots do a structured handover:
  - "your airplane," "my airplane"
- Should do this at least, but then cannot give up
- So the standard automation must now cope with real difficulties
  - Inconsistencies, authority limits, unforeseen situations
- In the case of AF 447, there was no truly safe way to fly
  - Human pilots are told to maintain pitch and thrust
  - Automation could do this, or better (cf. UA 232 Sioux City)
- But it is outside standard certification concepts
  - Must not become a getout
  - Nor a trap (inadvertent activation)
- Maybe a notion of ethics for the worst case (cf. trolley problems)

Unconventional Implementations

- Machine learning, neural nets, GAI etc.
- No explicit requirements (just training data),
  opaque implementation
- Why this matters: you cannot guarantee safety critical systems by testing alone
  - Nor even by extensive prior experience
  - The required reliabilities are just too great
- AC 25.1309: "No catastrophic failure condition in the entire operational life of all airplanes of one type"
- Operational life is about 10^9 hours, we can test 10^5
- Suppose 10^5 hours without failure, probability of another 10^5?
  - About 50%, probability of 10^9? Negligible!
  - Even high-fidelity simulations won't get us there
- Need some prior belief: that's what assurance gives us

What Assurance Does (Step 1)

- Extreme scrutiny of development, artifacts, code provides confidence software is fault-free
- Can express this confidence as a subjective probability that the software is fault-free or nonfaulty: p_{nf}
  - Frequentist interpretation possible
  - There's also quasi fault-free (any faults have tiny pfd)
- Define $p_{F|f}$ as the probability that it Fails, if faulty
- Then probability $p_{srv}(n)$ of surviving n independent demands (e.g., flight hours) without failure is given by

$$p_{srv}(n) = p_{nf} + (1 - p_{nf}) \times (1 - p_{F|f})^n \quad (1)$$

  A suitably large n can represent "entire operational life of all airplanes of one type"
- First term gives lower bound for $p_{srv}(n)$, independent of n

What Assurance Does (Step 2)

- If assurance gives us the confidence to assess, say, $p_{nf} > 0.9$
- Then it looks like we are there
- But suppose we do this for 10 airplane types
  - Can expect 1 of them to have faults
  - So the second term needs to be well above zero
  - Want confidence in this, despite exponential decay
- Confidence could come from prior failure-free operation
- Calculating overall $p_{srv}(n)$ is a problem in Bayesian inference
  - We have assessed a value for p_{nf}
  - Have observed some number r of failure-free demands
  - Want to predict prob. of $n - r$ future failure-free demands
- Need a prior distribution for $p_{F|f}$
  - Difficult to obtain, and difficult to justify for certification
  - However, there is a provably worst-case distribution

What Assurance Does (Step 3)

- So can make predictions that are guaranteed conservative, given only p_{nf}, r, and n
  - For values of p_{nf} above 0.9
  - The second term in (1) is well above zero
  - Provided $r > \frac{n}{10}$
- So it looks like we need to fly 10^8 hours to certify 10^9
- Maybe not!
- Entering service, we have only a few planes, need confidence for only, say, first six months of operation, so a small n
- Flight tests are enough for this
- Next six months, have more planes, but can base prediction on first six months (or ground the fleet, fix things, like 787)
- Theory due to Strigini, Povyakalo, Littlewood, Zhao at City U

What Assurance Does (Summary)

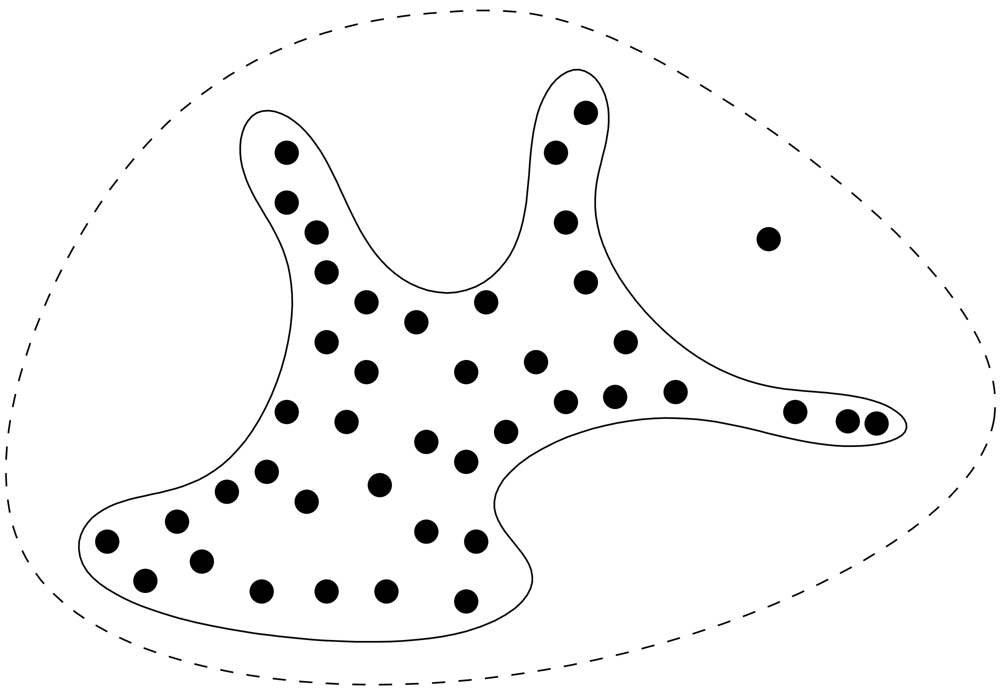
- We want confidence that failures are (very) rare
- Cannot get it by looking at failures alone
- Also need confidence there are no faults
- That's what assurance is about
- But to do it, you need requirements, visible design, development artifacts, etc.
- None of these are present in ML: just the training data
- Could rely on that
- Or look for a different approach
- I'll sketch ideas for both

Training Data: Trust but Verify

- We could choose to believe that our ML system generalizes correctly from the training data
  - This is arguable, but let's go with it
- Next, need some measure that the training data is adequately comprehensive (i.e., no missing scenarios)
  - Don't really know how to do this, but let's go with it
- Can be "comfortable" provided current inputs are "close" to examples seen in training data (i.e., not a missing scenario)
- And we are not facing adversarial inputs
- Can use a second, trustworthy ML system for these

Checking We've Seen This Before

- Use unsupervised learning to construct compact representation of the set of inputs seen in training data
- There are related techniques in control, learn "moded" representation, guaranteed sound



- Similarly for adversarial inputs: want space to be smooth
- Also, want smooth evolution in time
  - stop sign, stop sign, stop sign, birdcage, stop sign

Another Approach

- Observe the idea just presented is a kind of runtime monitor
- I've no evidence that it works, plan to try it
- But lets consider another kind of runtime monitor
- Idea is you have
  - An operational system responsible for doing things
  - And a second, monitor system, that checks behavior is "safe" according to high level safety requirements (not the local requirements of the (sub)system concerned)
  - Take some alternative safe action if monitor trips
- Theory says reliability of resulting compound system is product of reliability of operational system and p_{nf} of monitor
- Monitor can be simple, has explicit requirements
  - So p_{nf} could be high
- Aha! (Theory due to Littlewood and me, others at City U)

Pervasive Monitoring

- Code up the rules for safe flight, driving etc.
  - FAA “Aviation Handbooks & Manuals”
  - California driving code, UK Highway code etc.
- Could be a collaborative effort across each industry
- Possibly with regulatory approval like DO-178C, ISO 26262 etc.
- Need a suitable logic
  - Clear and easy to write, and easy to read
  - Decent automation, small distance from rules to code
  - Answerset programming?
- Could have general sections: rules of the air
  - And specialized: GenAv, big jets, 777-300 etc.
- Speculate that much of it is (de)composable
  - Cruise, approach, landing gear, radios, collision avoidance etc.
- But beware the experience of expert systems 20 years ago

Feasibility of High-Assurance Pervasive monitoring

- Checking is much easier than doing
- We have requirements, for one thing
  - E.g., when should wheels be up/down
- But still need to do situation/state assessment
  - And it needs to be unequivocal (cf. EK 521 crash)
  - And integrated (e.g., 87/101 sign cannot be 105 mph 'cos. . .)
- Might use the same sensors, but different/simpler/no ML
  - E.g., lane-keeping in cars: have to find the lane
  - Monitor just makes sure no obstacles, nothing coming at you
- Consider fatal self-driving car crashes (Level 2 used as Level 4)
  - Tesla May 2017: didn't see a truck crossing its path
  - Tesla March 2018: swerved(?) into median
  - Uber March 2018: didn't see lady crossing with a bike
- Pervasive monitors would surely have prevented these
- False alarms are a challenge: danger as well as nuisance

Summary

- Challenge is not just ML and GAI systems themselves
- But the architecture and HCI changes they require/enable
  - Do more, NGU, CRM
- Specific problem with ML and GAI is not just (un)predictability and opacity of systems themselves
  - Those might be be controlled by monitoring inputs against training data, and for smooth evolution
- But lack of requirements
  - Critical failures are judged wrt. safety requirements
- Cannot achieve confidence in safety-critical systems by observing failures: too few of them, want none
  - Need assurance for absence of faults
- So monitor the safety requirements: that's pervasive monitoring

Summary: Pervasive Monitoring

- Monitor the safety requirements
  - Need suitable logic and automation
  - Several small simple independent monitors (speculation)
  - Industry and regulatory collaboration to construct definitive safety requirements in logical form
  - Update following any incidents
- There's a plausible statistical theory that it can work
- But needs research and practical investigation
- Not just requirements specification and monitoring
- But system architecture for trustworthy situation assessment
  - Shared sensors, independent interpretation?
- Introspection suggests it's how humans work
- Let's try it!