Model-Centered Assurance for Safe Autonomy

John Rushby

Based on joint work with Susmit Jha and N. Shankar

http://www.csl.sri.com/users/rushby/abstracts/safecomp20

Computer Science Laboratory
SRI International
Menlo Park, California, USA
Intelligent Vehicle Dependability

- My focus: theory/speculation about assurance for autonomous systems
- Quintessential example: self-driving cars, Level 5, a little on Level 3/3+
- Autonomy architecture invariably has two components
  Perception: use sensors to build a model of the local world
  Action: use the model to calculate safe and effective behavior
- Both components may use Artificial Intelligence (AI) software, including Machine Learning (ML)
  - Difficult to anticipate behavior in all circumstances; often fails
  - Perception/action errors due to flawed ML and AI are vastly more significant source of failure than classical software and hardware faults
- Yet we want assurance:
  - Confidence in claims about the system within some context (ODD)
  - e.g., safety of self-driving on freeways
  - To very high levels (100 times better than human; $10^{-9}$ and beyond)
Assurance and Predictability

• Assurance requires predictable system-level behavior
  ○ But here we are using unpredictable components
  ○ Within an unpredictable environment

• Predictable behavior need not be deterministic
  ○ But requires some predicates to hold, given assumptions

• Components may be unpredictable, if larger architecture ensures predictability
  ○ e.g., unpredictable action component is guarded by a predictable monitor
    ★ Calculating effective behavior may require AI
    ★ But can check or monitor or guard its safety with assured conventional software
    ★ Given a model of the world
    ★ So the model becomes the focus of our attention

• Action and monitor components might use different models/sensors

• Both models need to be accurate, or at least safely approximate

• For simplicity, we speak of just the model
Architectures for Action Generation Monitor/Guardian/Checker

Can protect against flawed action generation, but utterly dependent on model(s)
Safely Approximate Models

• Safety is defined in human-focused (i.e., naturalistic) terms
• Therefore model needs to be naturalistic
  ○ i.e., defined on the variables of some human-defined framework
  ○ Not on the latent variables of a learned classifier
• For cars, typical model has detected objects list (what it is, size, velocity, intent)
  ○ Plus occupancy grid (bird's eye view of road and object layout)
  ○ And these can be probabilistic
• A good model the most valuable asset in the architecture
  ○ Needs to be cultivated with care
• A good model need not be perfectly accurate
• Requirement:
  if behavior looks safe in the model, then it must be safe in the real world
• Reverse need not be true
• So model can be conservative: safely approximate
(Un)Predictable (In)Accuracy of Models

- Models are built by machine learning
- Despite astonishing performance, accuracy of ML methods is not predictable, nor fault-free

**Evidence:** observed failures
  - Real-world testing (reveals large fat tails)
  - Adversarial examples (minor input changes produce big & bad effects)
  - Training data is often low quality, poorly labeled (garbage in, garbage out)

**ML explanation:** there's no understanding of the world
  - Debate on memorization vs generalization in deep learning
  - It's just curve fitting (Pearl)
  - Will always be anomalies adjacent to correct behavior (Shamir et al)

**Deeper explanation:** traditional perception is anti-causal (i.e., backwards)
  - The world causes the impressions that our sensors observe
  - Sensors try to infer the world from sense impressions: anti-causal
  - Unpredictable because different worlds can generate same sense impressions

- So instead, try to reason causally: generative models
Understanding Generative vs Anti-Causal Model Construction

- **Thought experiment**: reconstruct sharp image of face from degraded surveillance photo
- **Old way**: attempt to undo degradation—image sharpening, deconvolution etc.
  - Hopeless, utterly anti-causal
- **Machine Learning**: DNNs work by memorizing training examples, then make prediction on new examples by fitting them to training ones (Pedro Domingos)
  - So train on lots of sharp/degraded image pairs
    - New degraded image will be matched to training examples
    - And output will be a combination of the sharp images that generated them
  - Not bad, but anti-causal, vulnerable to anomalies, unpredictable

- **Generative Models**: start with candidate sharp image (use learning as above to find it)
  - Predict degraded image (a causal simulation/calculation)
  - Compare to actual, use error to revise sharp image
  - Repeat until prediction error is small
  - Works well (implemented as GANs), works causally
Dealing with (Un)Predictable (In)Accuracy of Models

- **Massive training** to reduce unpredictability ("collecting miles") requires **infeasible effort**
  - Billions of training and test miles (RAND and others)

- **Runtime checking** for unfavorable cases can help
  - E.g., detect when input is far from training data
  - Or influential parts of input do not coincide with decision
    - e.g., pixels that affect cat vs. dog do not coincide with face
  - Update the model conservatively when these trigger
    - These yield some improvement, but not to the levels required (beyond $10^{-9}$)

- Need to address the basic problem: **anti-causal** inference

- **So turn things around and reason causally from model to sensors**

- We use the model to **generate/predict** sensor input

- **Difference** between predicted and sensed input is **prediction error**

- Use prediction error for **model update and fault detection**
Predictive Processing and Model Update

- **Predictive processing** is the application of generative modeling to model-based control
- Use *(small)* prediction error to adjust the model
- E.g., by refining its parameters
- May have several candidate models and use prediction error to discriminate
  - E.g., detected object might be a bicycle or a pedestrian
  - The alternative models generate different predictions
    - Bicycles tend to go with traffic, pedestrians across it
  - Over time, better model will have smaller prediction errors
- Models can be probabilistic
  - Bayesian framework: predictions are priors, errors give posteriors
  - Whole loop can be mechanized as Variational Bayes
  - Provides iterative model refinement to minimize prediction error
- Like a Kalman Filter, generalized to complex data representations
Predictive Processing and Fault Detection

- **Large** prediction errors register **surprise**; indicate something is wrong
  - **All sources of failure detected by this single indicator**: untrained space, adversarial input, inaccurate model, unexpected evolution of world, etc.

- Prediction error provides **constant feedback** on quality of model and sensor interpretations

- Hence, prediction error is a single organizing principle for operation and assurance
  - Small prediction error: **all is well**, do model update
    - ★ Sound, provided no systematic faults (see later)
  - Large prediction error: **surprise**, deal with it (see later)
    - ★ Also a good trigger for event recorders and SPIs

- **Assurance is itself autonomous!**
Predictive Processing in Practice (skip)

• Use anti-causal methods (and prior knowledge) to construct initial model

• Thereafter use predictive processing to refine and update it

• At what level are the predictions?
  ◦ Pixel/point cloud level is too low
    ★ e.g., color is irrelevant, so need some abstraction
  ◦ Detected object list is an attractive level
    ★ May require some anti-causal interpretation to get there

• Predictions can guide sensors to better/faster interpretations
  ◦ e.g., can localize search for lane markings
Responses to Surprise

There are exactly three ways to respond to surprise (i.e., a large prediction error)

1. Adjust the sensors (or their interpretation/lower level model)
   - e.g., change interpretation algorithm/ML parameters
   - Ignore troublesome sensors for a while
   - Temporarily synthesize fallback sensors: e.g., in fog, cannot see lane markings
     - Use proximity and radar to detect neighboring cars and infer lanes

2. Adjust the model
   - e.g., increase uncertainty
   - Or make more surgical adjustments, rebuild anti-causally

3. Adjust the world
   - e.g., get in shadow of adjacent truck to avoid blinding sun

Or a combination

How to choose? Next slide
Managing Responses to Surprise

Surprising (i.e., large) prediction errors could be due to:

- **Localized sensor fault** (e.g., ML blip, hardware hiccup)
  - Ride it out for a while, using other sensors

- **Major fault** in one (class of) sensor
  - We assume different classes of sensor fail independently
    - e.g., cameras dazzled by sun, radar unfazed
    - Ride it out, using other sensors, increase uncertainty in model

- **Systematic misinterpretation**: must not happen, see later

- **Hardware or other traditional fault**: not our problem,
  Must be resolved by FT platform

- **Real world did not evolve as model expected**
  - Large prediction errors from several (classes of) sensors
  - Need to adjust either the model or the world, but what information to trust?
  - Employ dual-process architecture for just this reason
Dual-Process Architecture (Illustrative Example)

• Suppose we’re on a freeway, camera detects a truck ahead
  ○ Truck has bicycle painted on its rear

• As we get closer, camera changes detected object to bicycle
  ○ Or flickers between truck and bike
  ○ Or says probability $x$ for truck, $y$ for bicycle, and these wobble

• Prior was truck, so large prediction errors: a surprise

• But we are on a freeway, bicycles not allowed

• So object must be a truck

• System needs to apply AI knowledge and reasoning to model
  ○ Here, it is “laws and rules of the road”
  ○ The more you know, the less you need to sense

Locate this in a separate “higher level” process
  ○ Hence, dual-process architecture
Dual-Process Architecture (ctd. 1)

Level 1
- Model construction
- Predictions
- Prediction errors
- Assured model
- Primary model
- Planned actions
- Primary action function
- Monitor action function
- Override action function
- Actuators

Level 2
- Model refinement

AI/ML sensor interpretation

Sensor

John Rushby; SRI Model-Centered Assurance 15
Dual-Process Architecture (ctd. 2)

- System 1 (lower) does automated model construction
  - Based on predictive processing

- System 2 (upper) does model refinement
  - Based on symbolic methods, rules, reasoning, general AI
  - Intervenes on surprise (persistent, large prediction errors)
  - But its goal is to minimize surprise (next slide)

- Model is like blackboard: repository for all relevant knowledge

- Again: prediction errors provide single organizing principle

- For Level 3/3+, surprise could be the trigger for handover
  - System 2 is implemented by the human driver
Minimizing Surprise: Situation Awareness

- (Automated) System 2 intervenes on surprise
- But it should also anticipate and reduce future surprises
- This is situation awareness
- Explore counterfactuals, hypotheticals, theory of mind in longer range projections from current model
- Plausible contingencies added to model with suitable probabilities
- E.g., hypotheticals due to occluded vision
  - “If there were a car the other side of that truck, we would not be able to see it”
    - Add car to model with low probability (ghost)
  - Or “the driver of that car may not be able to see us” (due to an obstruction)
    - Increase probability car will pull out (adjust intent)
These make the model more conservative: fewer safe actions
- Evolution of actual world will cause adjustments in probabilities, not surprise
Assurance Argument

- Assurance for monitor is conventional, but relies on model
- So need assurance that model is safely approximate
- System 1 Predictive processing provides constant run-time verification of model, assuming sensor interpretation faults are independent and localized
- System 2 AI provides situation awareness, responds to surprise
  - Generally increases uncertainty in model
  - Makes it more approximate, therefore safer
- Must be no systematic (i.e., nonlocalized) interpretation faults
  - e.g., blind to red cars:
    - predict no red cars, see no red cars
    - so no prediction error...and then collide with red car
  - Develop evidence for assurance by model comparison (next slide)
  - Additionally, can employ diverse model construction (later)
Model Comparison for Evidence of No Systemic Faults

- Construct modeled world in simulation environment
- Calculate sensor interpretation of that world
- And derive and maintain a model of the world by predictive processing
- Compare that to the model you started with
- Repeat millions of times
- Ensure failures are few, do not persist over many frames
- This is not the same as collecting miles
  - We are verifying general behavior
  - Not seeking edge cases
- Whole argument provides prior for assurance by Conservative Bayesian Inference (CBI, Strigini et al)
Prior Art: The Human Brain

- Although our architecture is derived and justified on engineering grounds.

- It happens to be the way the brain works.

- Predictive processing
  - Helmholtz (1867), Rao and Ballard
  - Also known as predictive coding, predictive error minimization
    - Metzinger, Clark, Hohwy
  - Generalization: free energy (Friston)
  - Human brain has multiple models at different levels: lower levels like sensors to upper levels.

- Dual Process model (Systems 1 and 2)
  - Frankish, Evans & Stanovich
  - “Thinking, Fast and Slow” (Kahneman)
General Fault Tolerance

- Need redundancy and fault tolerance for traditional software and hardware faults
- Hermann Kopetz has developed principles and candidate architectures for cars
- Need to situate our architecture within his
- Also need to handle OTA updates safely
- Exploring options with Wilfried Steiner
- One opportunity is for secondary model construction to be deliberately diverse, run on a separate ECU; compare/fuse or...
  Periodically exchange primary and secondary models (detects/masks systematic faults, keeps models aligned)
Conclusions

- Can guard autonomous actions with conventional, assured, checker/monitor software
- But it depends on a safely approximate model of the world
- ML used in construction of that model
- Infeasible to assure ML directly
- So do it indirectly by run-time checking of the generative model
- Best framework for this is a two-level architecture
  - System 1 (lower level): Predictive processing
    - Small prediction errors indicate all is well
    - Assumes sensors fail independently
    - And no systematic ML flaws (assured by testing, mitigated by diversity)
  - System 2 (higher level): Model Refinement
    - Goal is to avoid large prediction errors, generally makes model more conservative
    - And recover when they do occur
    - Uses AI for situation awareness
- Needs experimental validation