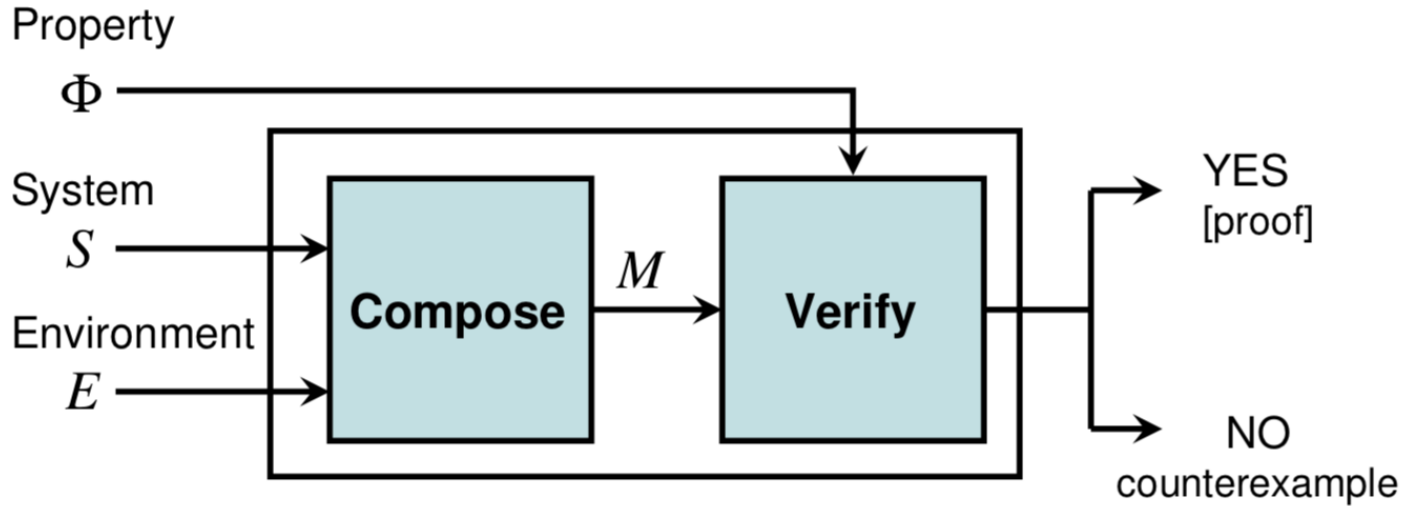


Problem and Principles for AI verification

Paper of Sanjit A. Seshia, Dorsa Sadigh and S.
Shankar Sastry

Presented by Nick Feng

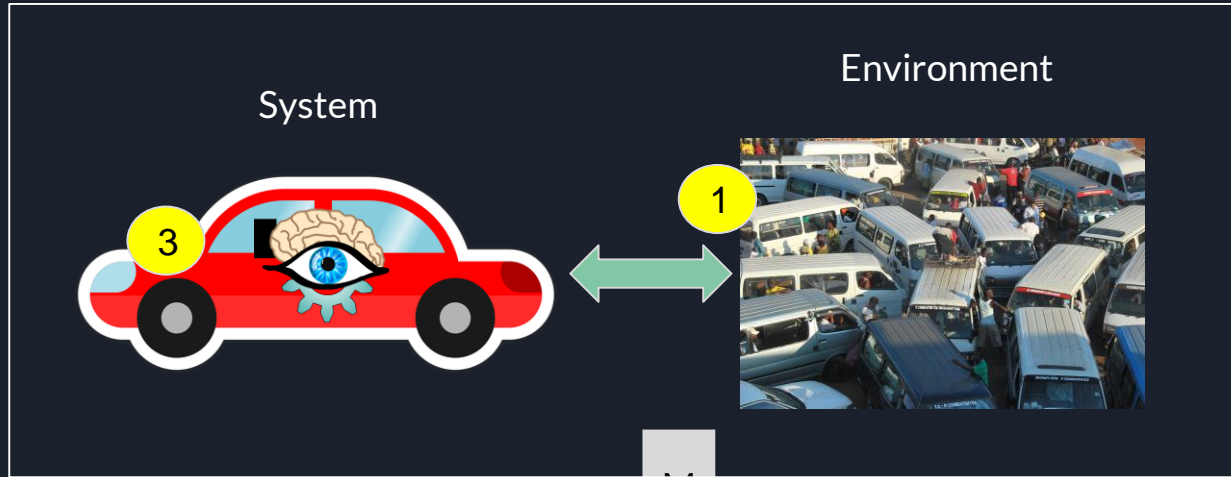
Formal Verification



Formal Verification of Autonomous Vehicle



Design



2

Property : **Safely** arrive at destination?

M

Verify

Yes, Proof

4

No, CEX



Problems for AI verifications:

1. Environment modeling
2. Formal specification
3. Model system that learns
4. Computation engine for training, testing and verification.
5. Correct-by-construction system.

Problem 1: Environment Modeling

- We want to model environment to capture (typically over-approximate) its behaviors and its interaction with the system.

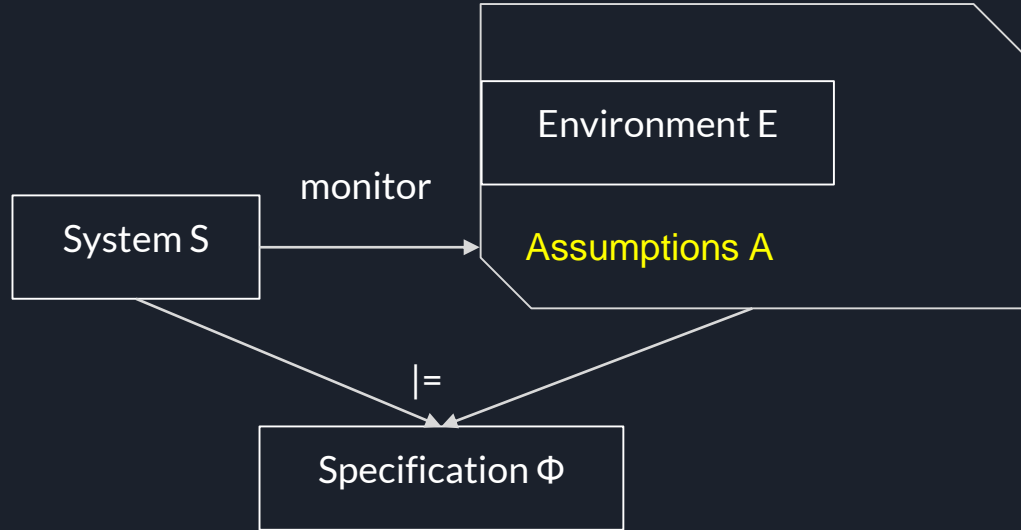


We want methods of environment modeling to provide provable guarantees on the system's behavior even when there's **high uncertainty** about the environment.

Strategy 1: Introspect Environmental Modeling

Come up with a strong enough A about E for Φ with 2 requirements.

1. S monitor A during runtime.
 $\text{state}(S) \wedge \text{observed}(E) \rightarrow A$
1. A can be explained to human.





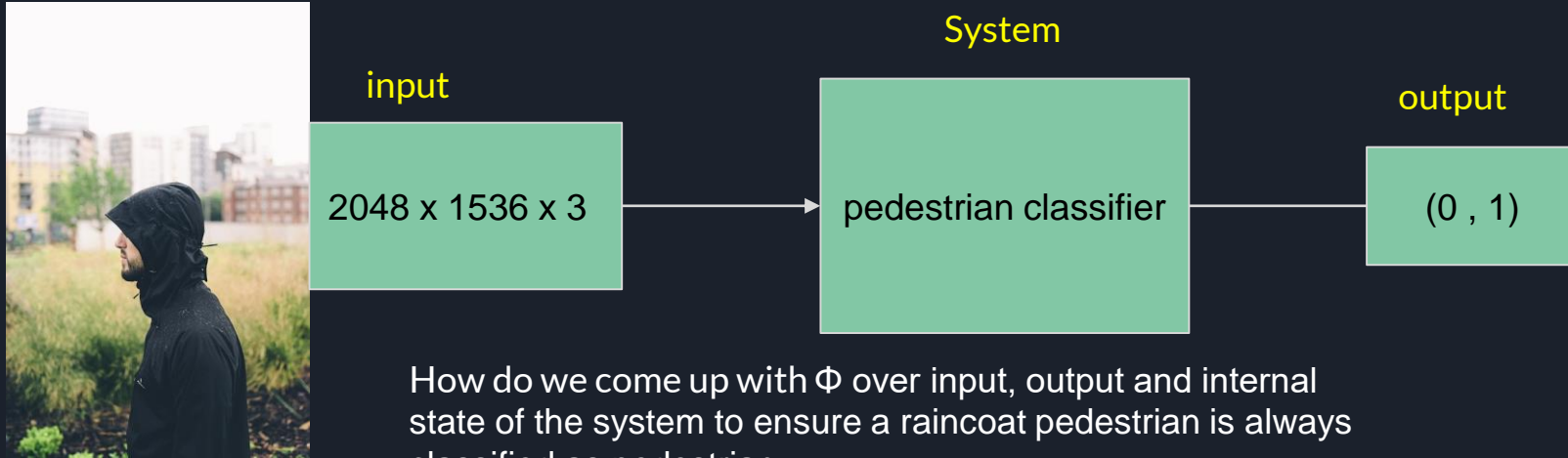
Extract assumption A

- Extract good monitorable assumptions A during:
 - Design process: mining assumptions for synthesis (feasible for simple cases)
 - Verification process (abstraction refinement cycle)
 - **Runtime (online environment monitoring)**

Often, we want to actively gather data about the real or simulated environment to learn and update the environment model.

Problem 2: Formal Specification Φ

Φ , a precise, mathematical statement about the system S



How do we come up with Φ over input, output and internal state of the system to ensure a raincoat pedestrian is always classified as pedestrian.

Training data + objective function + test result VS Φ



Strategy 2: End to End , Quantitative specification.

- Specifying AI/ML component behavior might be unnecessary.

End to End specification: System level specification over entire system including AI\ML components.

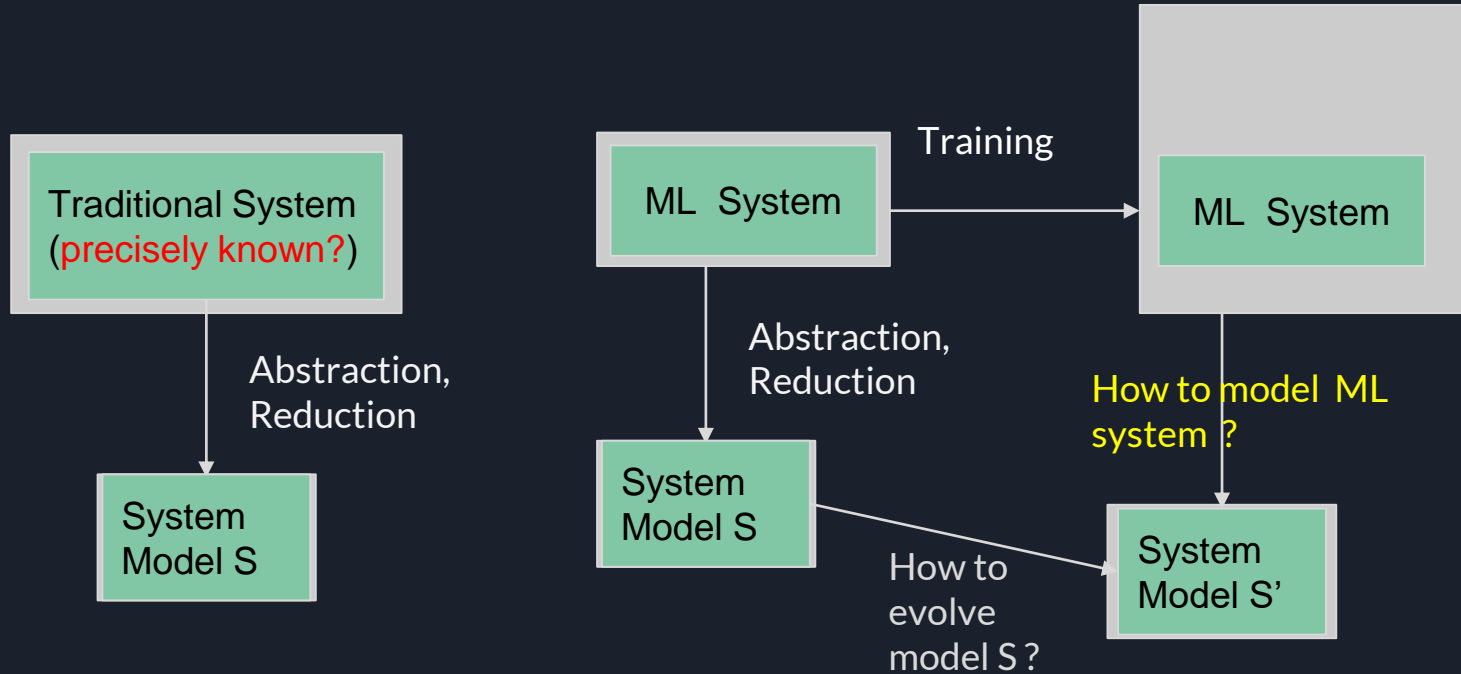
- Specification in **boolean** logic VS objective function.

Quantitative specification: logic with **quantitative** semantics or weighted automata.

Is a layer of NN (CNN or DNN) a mini weighted automata?

Is a NN a composition of weighted automatas?

Problem 3: Model system that learns

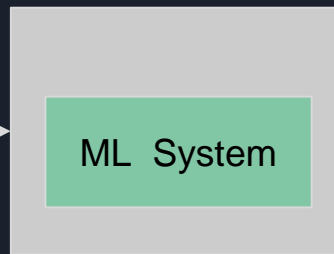


Strategy 3: Formal abstraction for ML



Strategy 3: Formal explanation for ML

Explanation based generalization



Pedestrian, 80%.
Explanation: 80 % due to recognition of human face + raincoat.

New abstract method

Formalism with probabilistic guarantees and uncertainty

guarantees



Pedestrian with probability $p > 80\%$

For all Input with recognizable raincoat and face




Possible strategy 3: Explaining failure or counterexamples

If a system **S** with ML components has failure w.r.t to end-to-end specification Φ with counter example **cex**. We want to explain the failure find all sources of failure.

$$\text{Cex} \models \sim\Phi \wedge \mathbf{S}$$

Analogy: Compute minimal independent support **I** for $\sim\Phi \wedge \mathbf{S}$ to identify the source of failure (extracting minimal unsatisfiable subsets which is well studied).




Problem 4: Computational engine for training, testing and verification.

Effectiveness of formal verification is driven by the advancement of the underlying “computation engine” SAT, SMT.....

For example, symbolic execution(SE) **generates test cases** to explore all feasible program paths within bounded steps. Even for a non-terminating program, SE provides high level of assurance if the bound is sufficiently high.


1. Can we use formal methods approaches to systematically generate training and testing data for ML component?



Problem 4: Computational engine for training, testing and verification.

1. AI based system is typically complex → challenge on verification scalability.
1. System model and environment model can be probabilistic with uncertainty, specification involves quantitative requirement on Robustness and performance → Traditional formal verification “computation engine” is not powerful enough.

New “computation engine” for effective and scalable quantitative verification.




Strategy 4: randomized formal method for training, testing

Perturbation test generation : we want to perturbate test input x by r so that the output $f(x) \approx f(x+r)$.

We want to find perturbation value with constraints:

1. The input after perturbation is realistic. (realistic fog??)
2. The distribution of the generated test case by perturbation must be constrained to ensure convergence to the true concept.

Generated training and test data should be valid and unbiased.



Strategy 4: randomized formal method for training, testing

Control improvisation: improviser generate examples with three constraints:

1. Hard constraint that define space of legal x .
2. Soft constraint that defines acceptance range of similarity to real-world example.
3. A random requirement for constraint on test case distribution.

Control improvisation resembles approximate Model counting and weighted model sampling.

- *Universal sampling legal r under hard constraint while maximizing objective from soft constraints.*



Strategy 4: Quantitative verification.

Formal verification of AI-based system is in general undecidable. So we need to compromise:

1. Find traceable but realistic problem
2. Settle for incomplete or unsound.

Falsification address verification as an optimization problem:

p quantifies how much a trace satisfy a property.

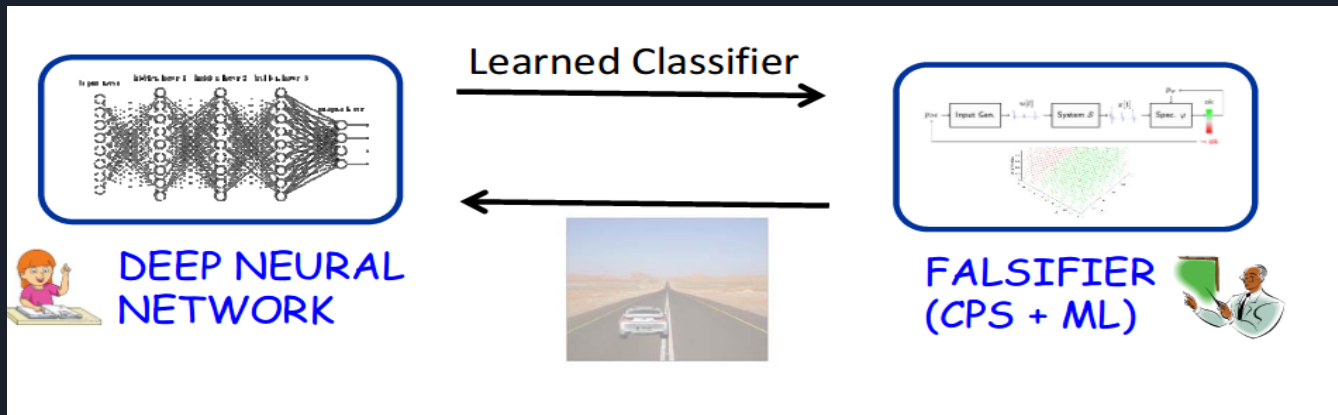
By minimizing p , and if we find a trace with $p < 0$, then we have found a bug.

Problem 5: Correct-by-design system

We want to design a ML component that satisfy property Φ .

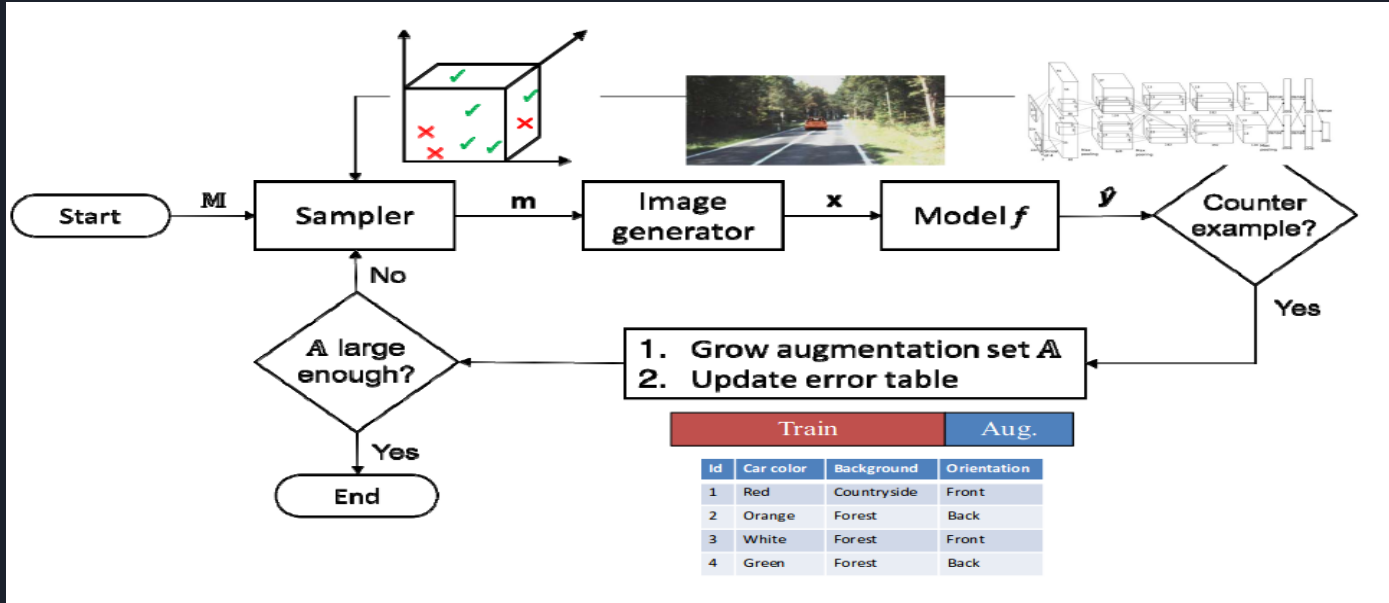
1. Synthesize a learning model.
2. Synthesize a suitable training set.

Formal inductive synthesis. Oracle (CEX) guided learning!



“Counterexample-Guided Data Augmentation”, T. Dreossi, S. Ghosh, X. Yue, K. Keutzer, A. Sangiovanni-Vincentelli, S. A. Seshia, IJCAI2018.

Strategy 5: Counterexample-guided data augmentation.



“Counterexample-Guided Data Augmentation”, T. Dreossi, S. Ghosh, X. Yue, K. Keutzer, A. Sangiovanni-Vincentelli, S. A. Seshia, IJCAI2018.



Discussion questions

Is a formal method a good starting point for verification on AI based system?
How can one argue the effectiveness of formal methods when there are quantitative specifications, probability models and uncertain.

Do we come up specification for AI based system? Or does the system learn specifications from data?

Should we verify system based on the “designed specification” and models?
Or should we try to explain the learning process and understand the “learnt” specifications.