AI for verification and verification for AI

Overview and discussion

KIT, February 16th 2026

Lecture Overview

- **Verification for AI**
  - Shielding
  - Neural-network verification
- **AI for Verification**
  - Imitation learning of theorem proving
  - Reinforcement learning of theorem proving
  - Formal Methods in the Era of Large Language Models

Machine-Learning Primer

Supervised Learning as an Optimization Problem

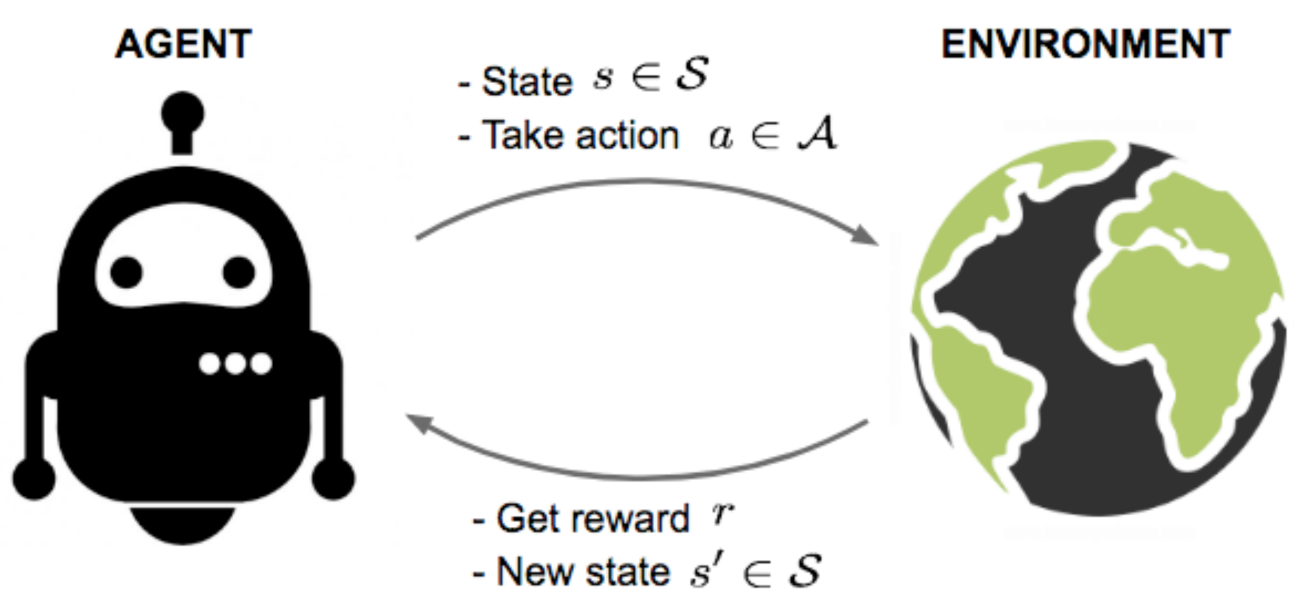
Problem: learning an unknown function from many input/output examples.

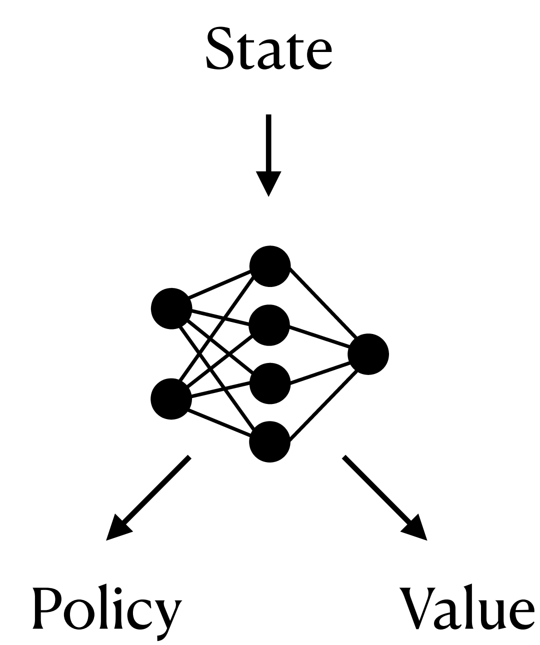
$$L(W) = \frac{1}{|D|} \sum_{x,y \in D} (y - f_W(x))^2$$

$$f_W(x) = W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x))$$

$$W_{t+1} \leftarrow W_t + \lambda \nabla_{W_t} L$$

Reinforcement Learning





AI for Verification

```
init → [(ctrl; plant)*] safe
```

Shielding

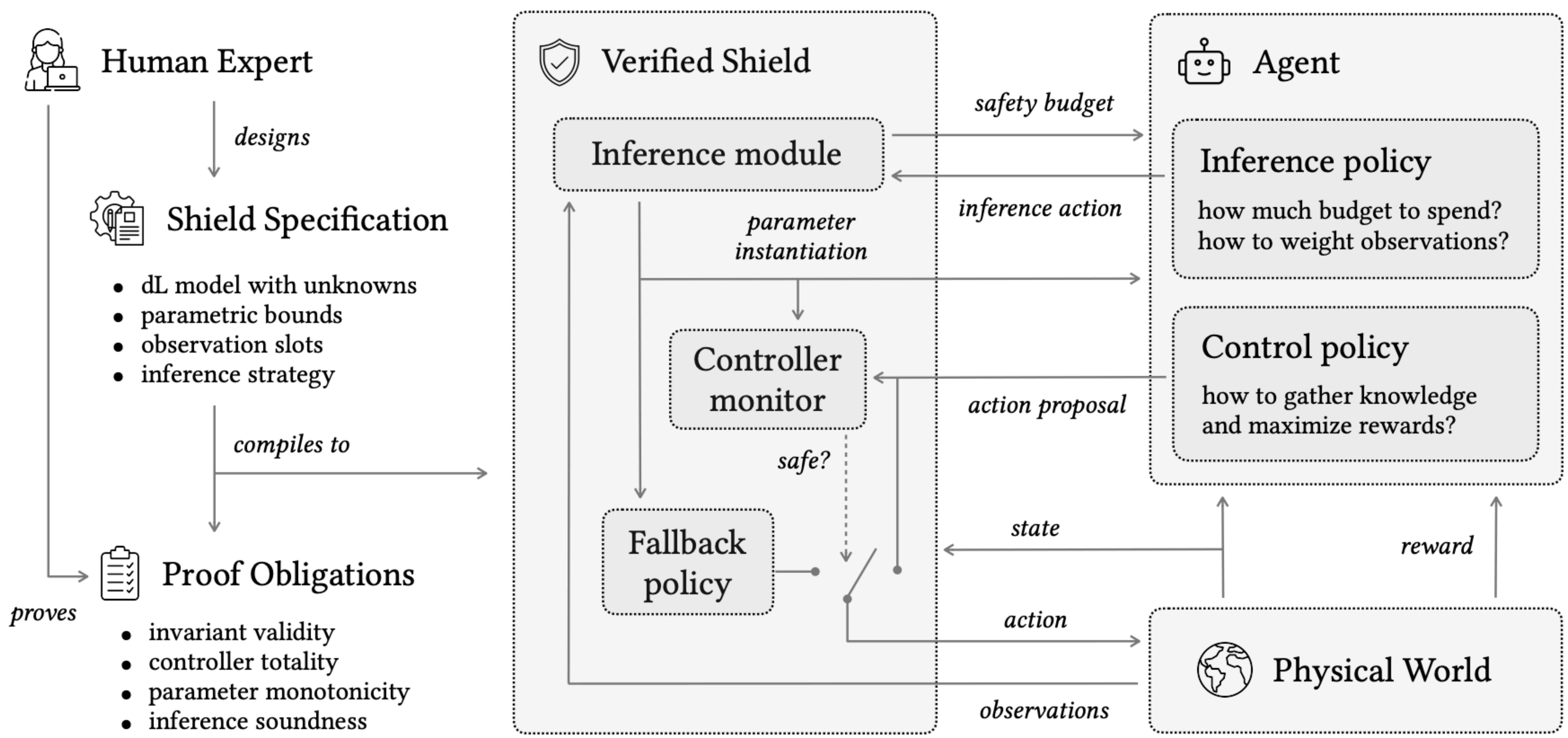
Approach: do not trust the neural-network, sandbox it!

```
init → [{
   {?safeAccel; accel
    ∪ brake};
   t:=0; {pos’=vel,vel’=acc}
}*](pos < stopSign)
```

Safe Reinforcement Learning via Formal Methods, Fulton and Platzer, AAAI 2018

Adaptive Shielding

- **Problem:** what if the environment is not fully known offline?
- **Solution:** write and verify **parametric safety proofs** offline, and estimate parameters online, allowing increasingly permissive shielding.



Adaptive Shielding via Parametric Safety Proofs, Feng et al., OOPSLA 2025

Example 1: estimating scalar parameters

CONSTANT A, B, T, σ

UNKNOWN θ, φ

ASSUME $A > 0,\ B > 0,\ T > 0,\ \sigma > 0,\ \theta > 0$

BOUND $\underline{\theta} : \underline{\theta} \leq \theta,\ \bar{\theta} : \bar{\theta} \geq \theta,\ \bar{\varphi} : \bar{\varphi} \geq \varphi$

CONTROLLER

$$u := * \ ; \ ?(-B \leq u \leq A) \ ; \ ?(x + vT + (\bar{\theta}u + \bar{\varphi})T^2/2 + (v + (\bar{\theta}u + \bar{\varphi})T)^2/2(\underline{\theta}B - \bar{\varphi}) \leq e)$$

PLANT $t := 0 \ ; \{x' = v, v' = \theta u + \varphi, t' = 1 \ \& \ t \leq T \land v \geq 0\}$

SAFE $x \leq e$

INVARIANT $(\underline{\theta}B - \bar{\varphi} > 0) \land (x + v^2/2(\underline{\theta}B - \bar{\varphi}) \leq e)$

NOISE $\eta \sim \mathcal{N}(0, \sigma^2)$

OBSERVE $\omega = \theta u + \varphi - \eta$

INFER

$$\underline{\theta}, \bar{\theta} := \text{AGGREGATE } i, j : (\omega_j - \omega_i)/(u_j - u_i) \text{ AND } (\eta_j - \eta_i)/(u_j - u_i) \text{ WHEN } u_j > u_i \ ;$$

$$\bar{\varphi} := \text{AGGREGATE } i : \omega_i - \bar{\theta}u_i \text{ AND } \eta_i \text{ WHEN } u_i \leq 0$$

Example 2: handling functional unknowns with local bounds

CONSTANT A, B, F, k, σ

UNKNOWN $f(*)$

ASSUME

$$A > 0,\ B > 0,\ T > 0,\ k > 0,\ \sigma > 0,\ F < B,\ A + F > 0,$$
$$(\forall x\ -A \le f(x) \le F),\ (\forall x\, \forall y\, |f(x) - f(y)| \le k|x - y|)$$

BOUND $\bar{f} : f(x) \le \bar{f}$

CONTROLLER

$$y := \min(y, \bar{f})\ ;$$
$$((a := -B) \cup$$
$$(?(x + vT + \tfrac{1}{2}(A+F)T^2 + \mathsf{Bdist}_{v+(A+F)T}(B - \min(F,\ y + k(vT + \tfrac{1}{2}(A+F)T^2) +$$
$$k \cdot \mathsf{Bdist}_{v+(A+F)T}(B - F))) \le e)\ ;\ a := A)$$

PLANT $t := 0\ ; \{x' = v, v' = a + f(x), y' = kv, t' = 1\ \&\ t \le T \wedge v \ge 0\}$

SAFE $x \le e$

INVARIANT $(v \ge 0) \wedge (y \ge f(x)) \wedge (x + \mathsf{Bdist}_v(B - \min(F,\ y + k \cdot \mathsf{Bdist}_v(B - F))) \le e)$

NOISE $\eta \sim \mathcal{N}(0, \sigma^2)$

OBSERVE $\omega = f(x) - \eta$

INFER $\bar{f} := F\ ;\ \bar{f} :=$ BEST $i : \bar{f}_i + k|x - x_i|\ ;\ \bar{f} :=$ AGGREGATE $i : \omega_i + k|x - x_i|$ AND η_i

Example 2: handling functional unknowns with local bounds

NOISE $\eta \sim \mathcal{N}(0, \sigma^2)$

OBSERVE $\omega = f(x) - \eta$

INFER $\bar{f} := F$; $\bar{f} :=$ BEST $i : \bar{f}_i + k|x - x_i|$; $\bar{f} :=$ AGGREGATE $i : \omega_i + k|x - x_i|$ AND η_i

Proof obligations:

$$\mathsf{Init} \wedge \mathsf{Inv} \wedge \bar{f} = F \rightarrow f(x) \leq \bar{f}$$

$$\mathsf{Init} \wedge \mathsf{Inv} \wedge \bar{f} = \bar{f}_i + k|x - x_i| \wedge f(x_i) \leq \bar{f}_i \rightarrow f(x) \leq \bar{f}$$

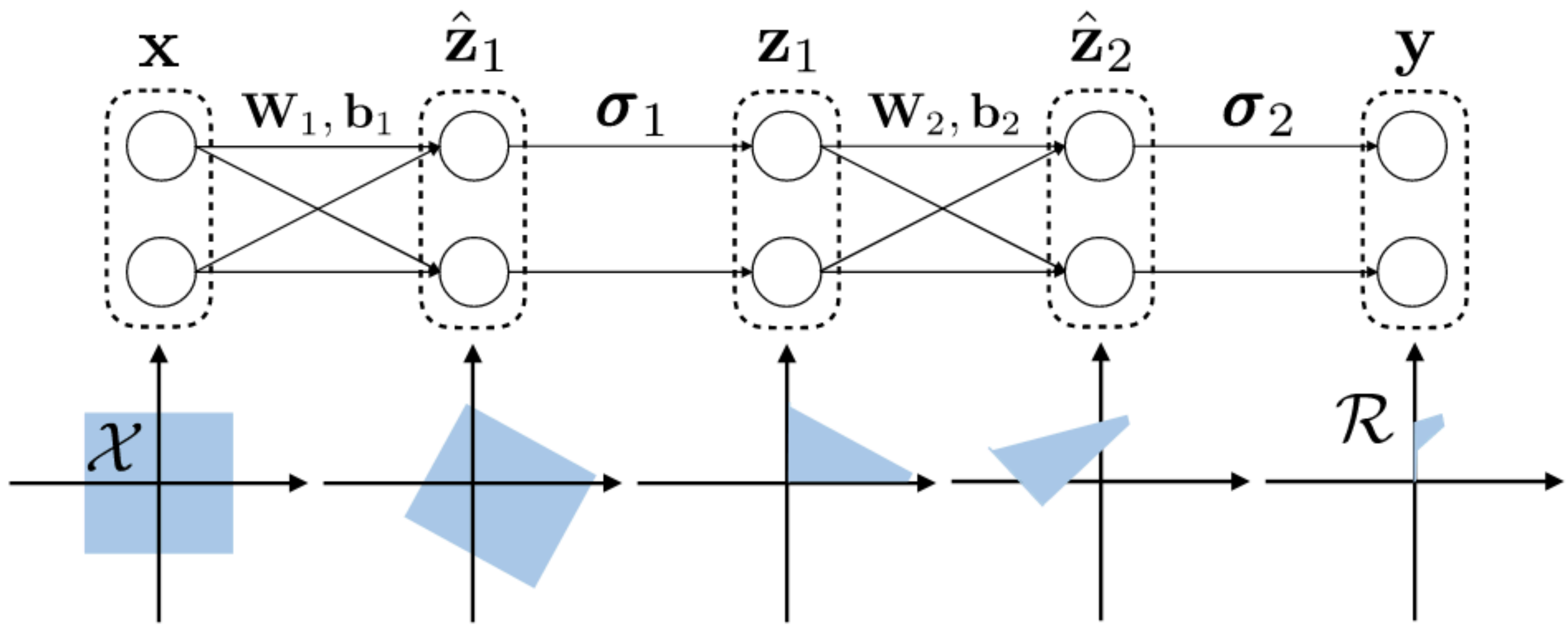
$$\mathsf{Init} \wedge \mathsf{Inv} \wedge \bar{f} = \omega_i + k|x - x_i| + \eta_i \wedge \omega_i = f(x_i) - \eta_i \rightarrow f(x) \leq \bar{f}.$$

Closed-Loop Network Verification

```
init → [(
    u := NN(x);
    x := plant(x, u);
)*] safe
```

Reachability analysis of neural networks

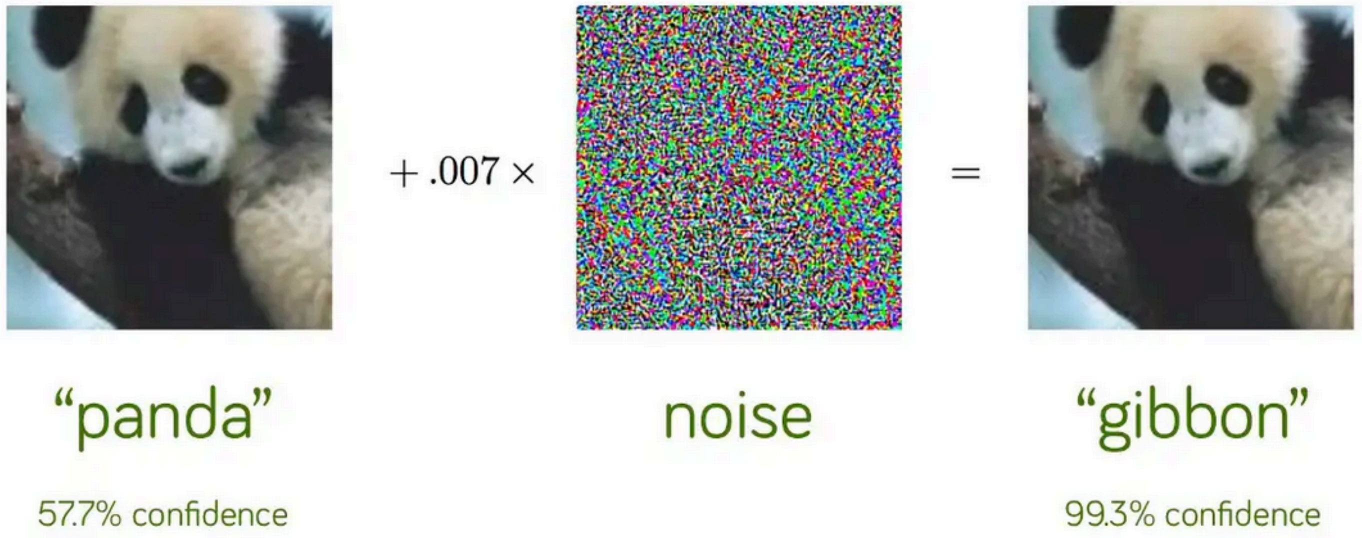
$\mathrm{NN}(x) = W_3 \cdot \sigma(W_2 \cdot \sigma(W_1 \cdot x))$



Survey: *Algorithms for Verifying Deep Neural Networks*, Liu et al.

Open-Loop Neural Network Verification

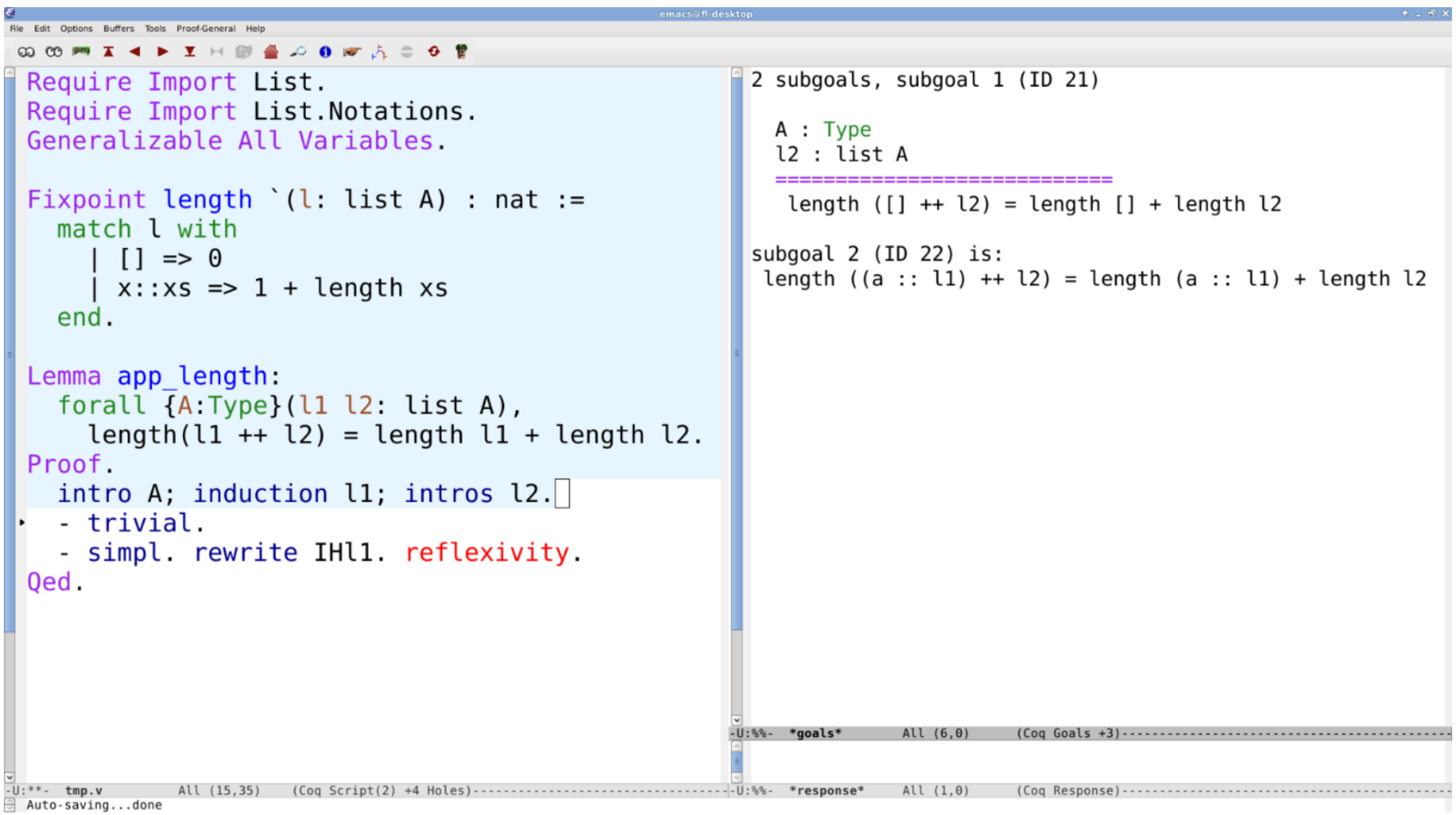
Example: robustness verification



$$\forall (x, y) \in D,\ \forall x',\ \|x - x'\|_\infty < \epsilon \ \longrightarrow F(x') = y$$

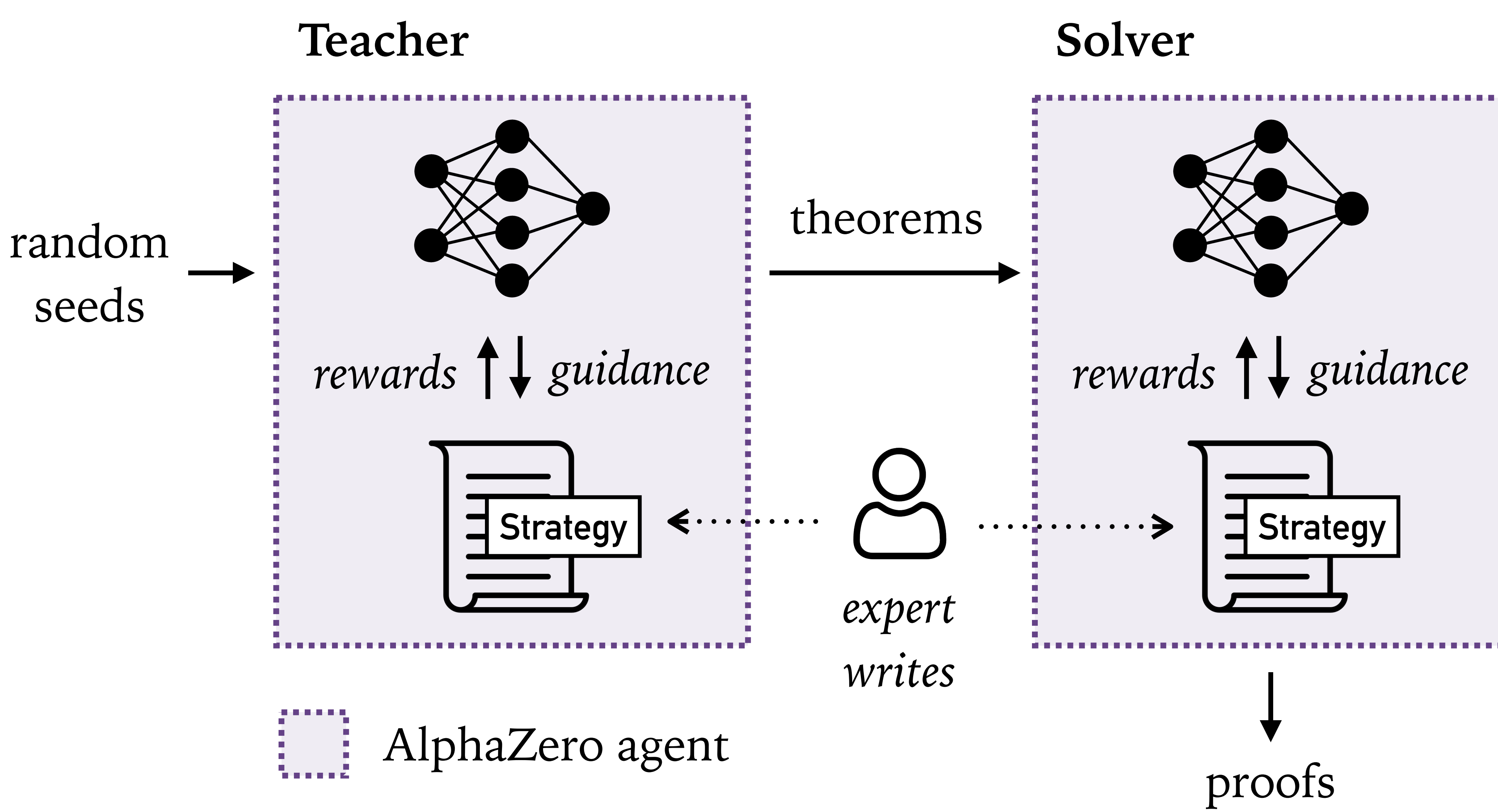
Verification for AI

Can we train an AI to interact with a theorem prover?



LEARNING TO FIND PROOFS AND THEOREMS BY LEARNING TO REFINE SEARCH STRATEGIES

Jonathan Laurent, André Platzer (NeurIPS 2022)



LOOP INVARIANT SYNTHESIS

- Training data unavailable and hard to generate!
- No pre-existing deep-learning agent capable of generalizing across instances.

```
assume x ⩾ 1
y = 0
while y < 1000 {
    x = x + y
    y = y + 1
}
assert x ⩾ y
```

To prove the final assertion, one must find a loop invariant that:

1. is true before the loop
2. is preserved by the loop body (when the loop guard holds)
3. implies the final assertion (when the loop guard does not hold)

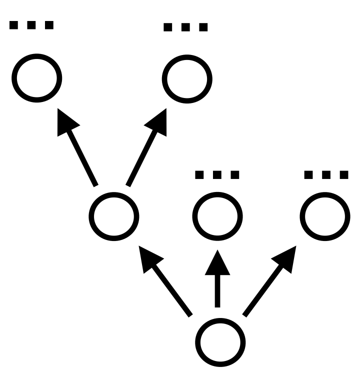
Invariant: $x \geq y \land x \geq 1 \land y \geq 0$

A LANGUAGE FOR EXPRESSING STRATEGIES

We define a strategy language based on **choose** and **event** operators.

Expert strategy

compiled into



MDP amenable to RL and neural-guided search

```
def solver(
    init: Formula, guard: Formula,
    body: Program, post: Formula) -> Formula:
    def prove_inv(inv: Formula) -> List[Formula]:
        assert valid(Implies(init, inv))
        ind = Implies(And(guard, inv), wlp(body, inv))
        event(PROVE_INV_EVENT)
        match abduct(ind):
        case Valid:
            return [inv]
        case [*suggestions]:
            aux = choose(suggestions)
            return [inv] + prove_inv(aux)
inv_cand = choose(abduct(Implies(Not(guard), post)))
inv_conjuncts = prove_inv(inv_cand)
return And(*inv_conjuncts)
```

▲ A **solver strategy** for invariant synthesis

GENERATING TRAINING PROBLEMS

- Generating interesting theorems is harder than proving those!
- **Our approach:** refining **conditional generative strategies** using RL.

```
def teacher(rng: RandGen) → Prog:
    cs = sample_constrs(rng)
    p = generate_prog(cs)
    p = transform(p, rng)
    p = hide_invariants(p)
    return p

def generate_prog(cs: Constrs):
    p = Prog("
        assume init;
        while (guard) {
          invariant inv_lin;
          invariant inv_aux;
          invariant inv_main;
          body; }
        assert post;")
    p = refine_guard(p, cs)
    p = refine_inv(p, cs)
    p = refine_body(p, cs)
    assert valid(inv_preserved(p))
    p = refine_post(p, cs)
    assert valid(inv_post(p))
    p = refine_init(p, cs)
    assert valid(inv_init(p))
    penalize_violations(p, cs)
    return p

def transform(p: Prog, rng: RandGen):
    p = shuffle_formulas(p, rng)
    p = add_useless_init(p, rng)
    ...
    return p
```

▲ Outline of a **teacher strategy** for invariant synthesis

Formal Methods in the Era of

Large Language Models

Language Models are Few-Shot Learners

Tom B. Brown[*] Benjamin Mann[*] Nick Ryder[*] Melanie Subbiah[*]

Jared Kaplan[†] Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei