20th International School on Trustworthy Software
Shanghai, China, July 9 - 12

Trustworthy AI through Neural Certificates, Runtime Monitoring and Multi-Agent Reasoning

Thomas A. Henzinger
Institute of Science and Technology Austria

Djordje Zikelic
Singapore Management University

Lecture 2 and Lecture 3

Learner-verifier framework
for (stochastic) control

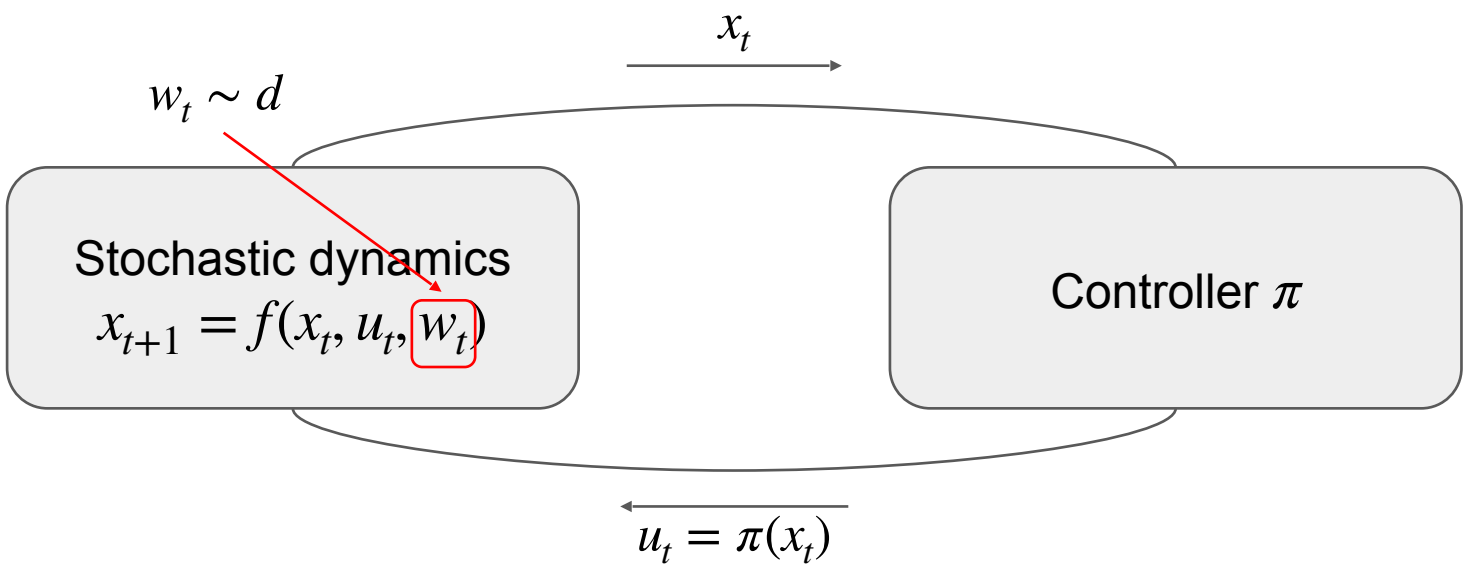
Lecture 2

Theory of certificates
for (stochastic) control

Lecture 3

Learner-verifier
automation

Recap: Stochastic dynamical system (SDS)



Recap: Stochastic control problem

Definition

Given: An SDS (X, U, W, f, d), initial region $X_0 \subseteq X$, target region $X_T \subseteq X$, unsafe region $X_U \subseteq X$, probability threshold $p \in [0,1]$

Task: Compute a controller $\pi : X \to U$ such that $\mathbb{P}_{x_0}[\text{ReachAvoid}(X_T, X_U)] \geq p$ for every $x_0 \in X_0$.

<u>Notation</u>: ReachAvoid(X_T, X_U) is the set of all trajectories that satisfy the reach-avoidance specification.

Recap: Reach-avoid supermartingales (RASMs)

Definition: Reach-avoid supermartingale

Consider an SDS (X, U, W, f, d) with initial region $X_0 \subseteq X$, and let $\pi : X \to U$ be a controller.
A RASM with respect to X_T, X_U and p is a (measurable) function $V : X \to \mathbb{R}$, which satisfies:

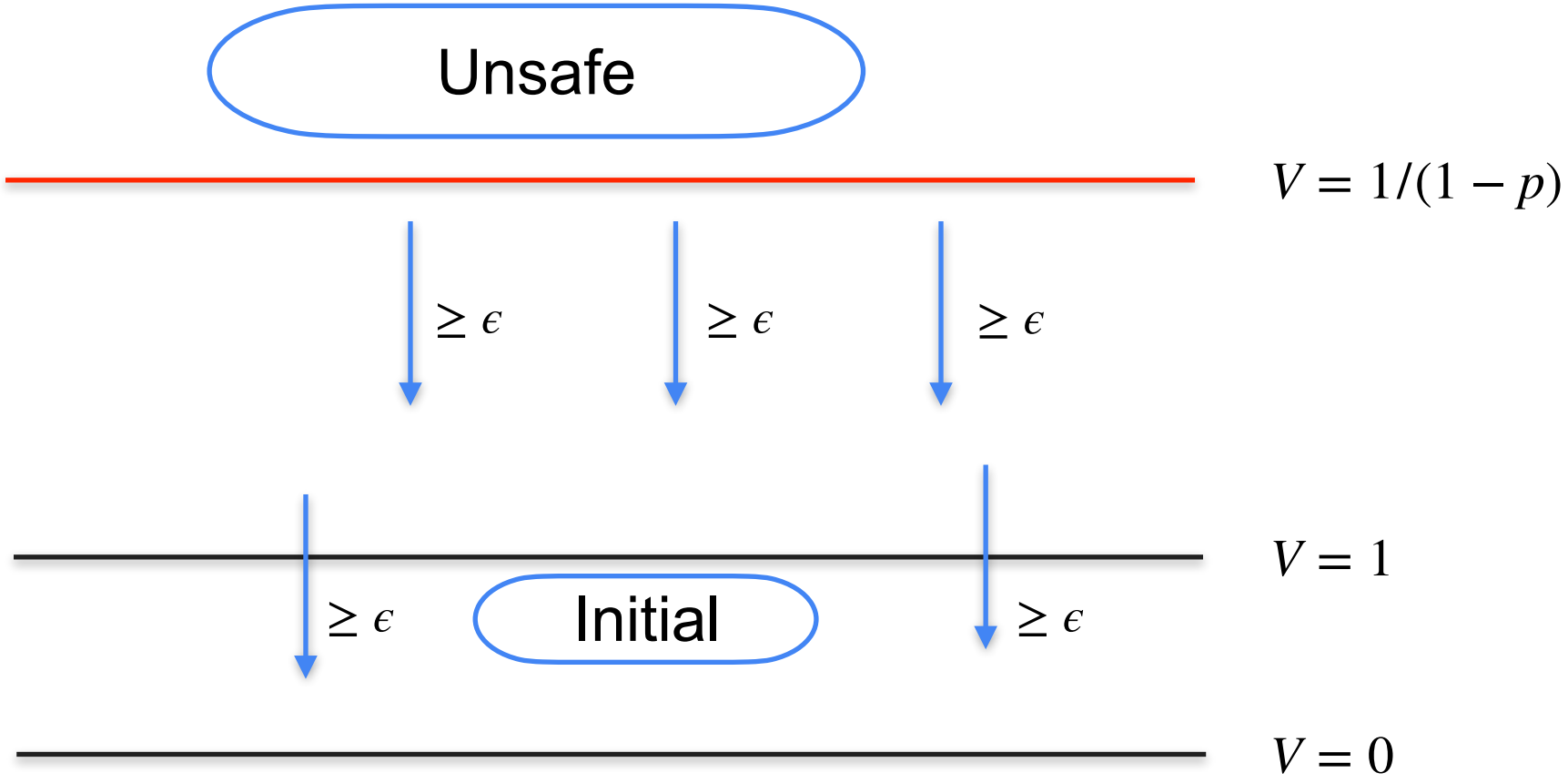
(1) **Non-negativity**. $V(x) \geq 0$ for each state $x \in X$.

(2) **Strict expected decrease**. There exists $\epsilon > 0$ such that $V(x) \geq \epsilon + \mathbb{E}_{w \sim d}[V(f(x, \pi(x), w)]$
for all $x \in X \backslash X_T$ at which $V(x) \leq 1/(1-p)$

(3) **Initial condition**. $V(x) \leq 1$ for each initial state $x \in X_0$.

(4) **Unsafe condition**. $V(x) \geq 1/(1-p)$ for each unsafe state $x \in X_U$.

Recap: Intuition behind RASMs



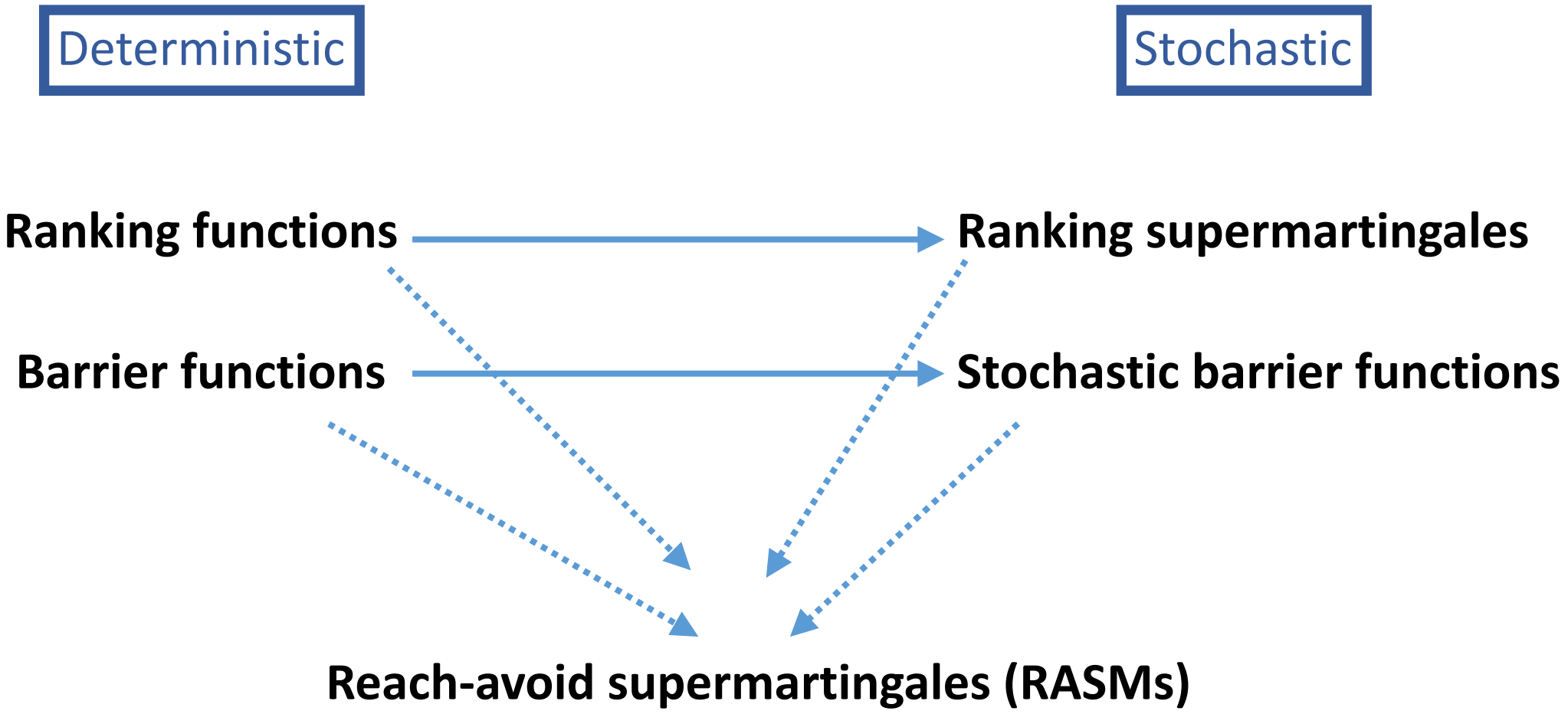
Recap: Soundness of RASM certificates

Theorem

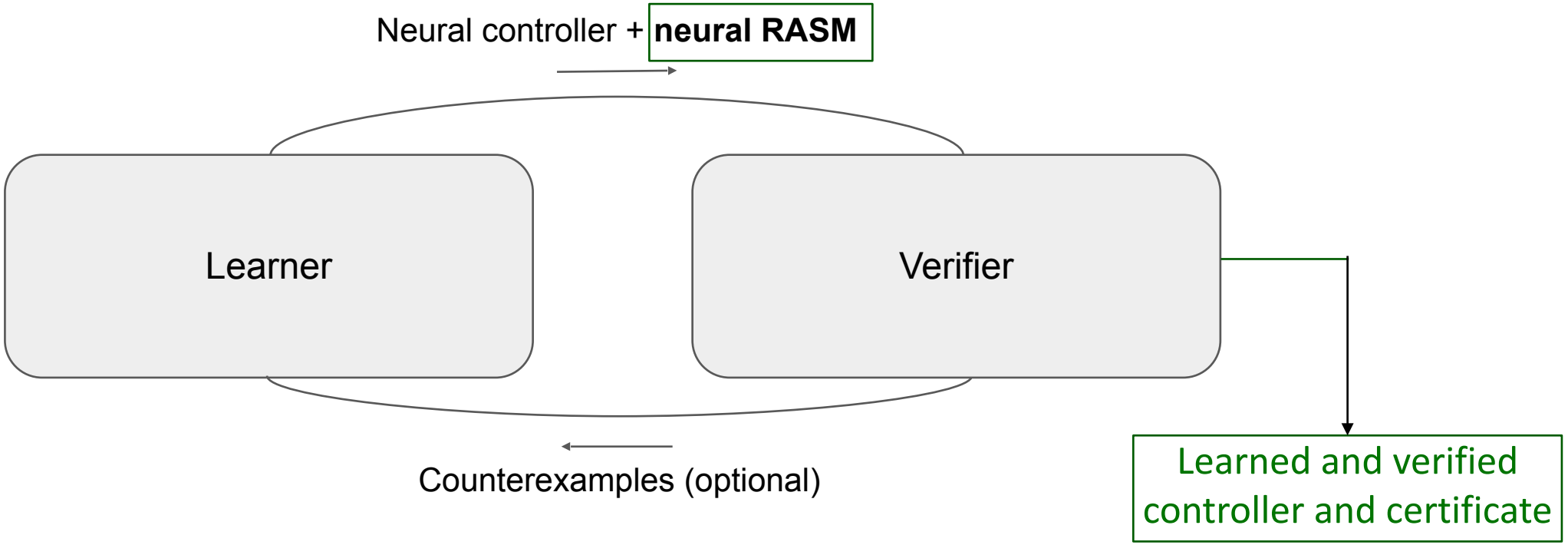
Consider an SDS (X, U, W, f, d) with initial region $X_0 \subseteq X$, and let $\pi : X \to U$ be a controller. Suppose that there exists a RASM with respect to X_T, X_U and p. Then, we have that

$$\mathbb{P}_{x_0}[\text{ReachAvoid}(X_T, X_U)] \geq p \text{ for every } x_0 \in X_0.$$

Overview of discrete-time certificate functions



Learner-verifier framework for controller synthesis



Assumptions needed for automation

In what follows, we impose the following assumptions:

- The dynamics function f of the SDS is **Lipschitz continuous** with Lipschitz constant L_f
- The state space $X \subseteq \mathbb{R}^m$ of the SDS is **bounded**
- The probability distribution d has **bounded support**

Definition

Let (X, d_X) and (Y, d_Y) be metric spaces. A function $f : X \to Y$ is said to be Lipschitz continuous, if there exists a constant $C > 0$ such that, for every $x_1, x_2 \in X$, we have $d_Y(f(x_1), f(x_2)) \leq C \cdot d_X(x_1, x_2)$.

We say that each such C is a Lipschitz constant of f.

Plan for today

1. Learner module for neural controllers and RASMs

2. Verifier module for neural controllers and RASMs

3. Neural controllers for more general specifications

Plan for today

1. **Learner module for neural controllers and RASMs**

2. Verifier module for neural controllers and RASMs

3. Neural controllers for more general specifications

Learning neural controllers and RASMs

Controller = function of type states $X \rightarrow U$ actions

RASM = function of type states $X \rightarrow \mathbb{R}$ real values

Learn two feed-forward neural networks:

- Controller network has $\dim(X) = n$ inputs and $\dim(U) = m$ outputs
- RASM network has $\dim(X) = n$ inputs and 1 output

Learner module

Supervised learning task, loss function encodes the defining conditions of RASMs

(Non-negativity imposed by default, by applying ReLU/softplus on neural RASM output)

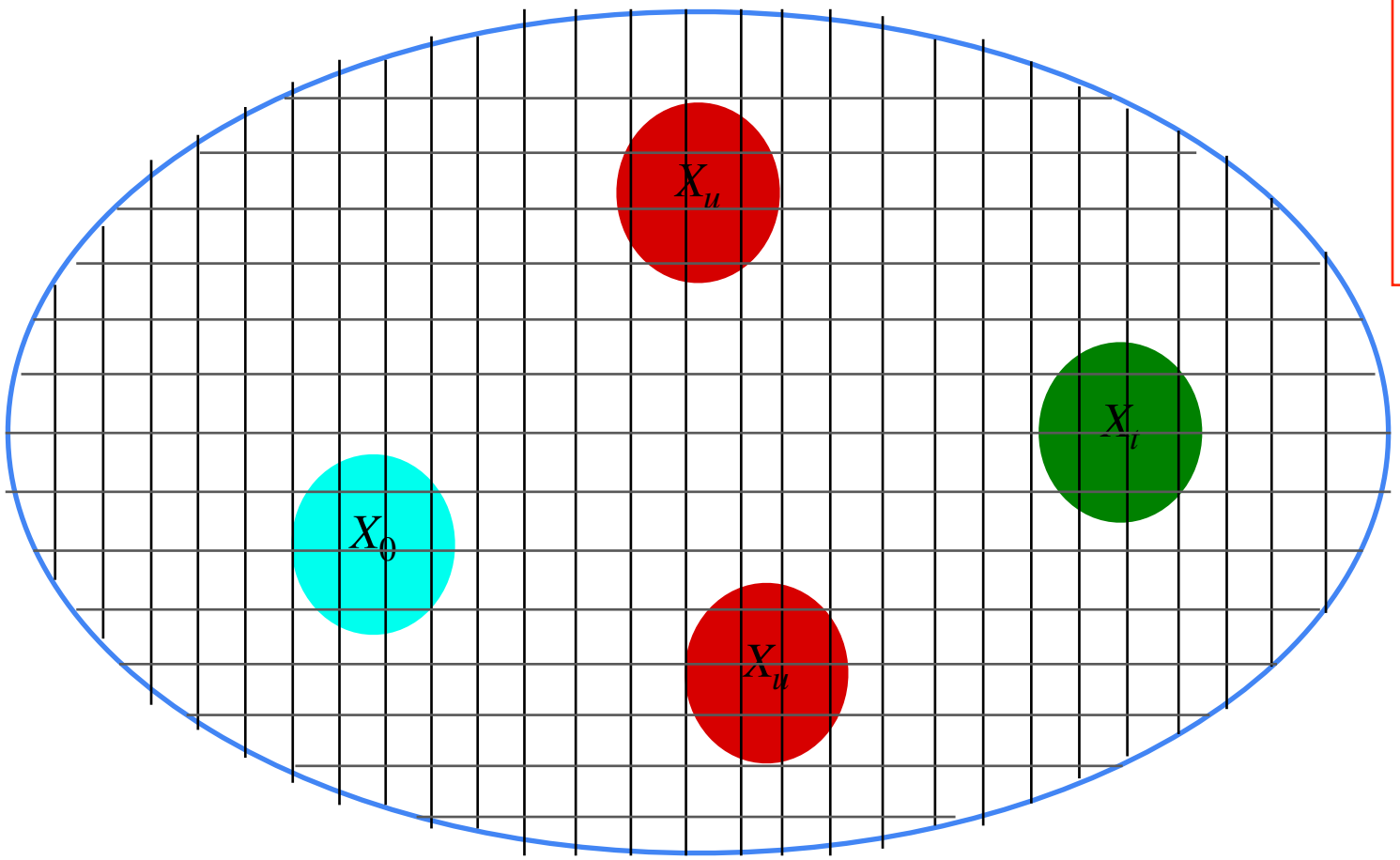
$$\mathscr{L}(\theta, \nu) = \boxed{\mathscr{L}_{\text{Init}}(\nu) + \mathscr{L}_{\text{Unsafe}}(\nu) + \mathscr{L}_{\text{Decrease}}(\theta, \nu)}$$

Intuition: The loss function empirically encodes all RASM defining conditions. Hence, it guides the learner to learn a neural controller that admits a neural RASM and thus guaranteeing reach-avoidance with the desired probability.

<u>Notation</u>: Neural controller π_θ, neural RASM V_ν

θ, ν are vectors of neural network parameters (weights and biases)

Training set: Discretization



$\tilde{X}$ = hyperrectangular discretization of X

$C_{\text{init}} = X_0 \cap \tilde{X}$

$C_{\text{unsafe}} = X_U \cap \tilde{X}$

$C_{\text{decrease}} = \tilde{X} \setminus (X_T \cup X_U)$

Loss function

$$\mathscr{L}(\theta,\nu) = \boxed{\mathscr{L}_{\text{Init}}(\nu) + \mathscr{L}_{\text{Unsafe}}(\nu) + \mathscr{L}_{\text{Decrease}}(\theta,\nu)}$$

Empirically enforce RASM defining conditions

$$\mathscr{L}_{\text{Init}}(\nu) = \max_{\mathbf{x}\in C_{\text{init}}} \left\{ V_\nu(\mathbf{x}) - 1, 0 \right\}$$

$$\mathscr{L}_{\text{Unsafe}}(\nu) = \max_{\mathbf{x}\in C_{\text{unsafe}}} \left\{ \frac{1}{1-p} - V_\nu(\mathbf{x}), 0 \right\}$$

Expected value estimate

ϵ (see later)

$$\mathscr{L}_{\text{Decrease}}(\theta,\nu) = \frac{1}{|C_{\text{decrease}}|} \cdot \sum_{\mathbf{x}\in C_{\text{decrease}}} \left(\max \left\{ \sum_{\omega_1,\ldots,\omega_N \sim \mathcal{N}} \frac{V_\nu\big(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega_i)\big)}{N} - V_\nu(\mathbf{x}) + \tau \cdot K,\ 0 \right\} \right)$$

Plan for today

1. Learner module for neural controllers and RASMs

2. **Verifier module for neural controllers and RASMs**

3. Neural controllers for more general specifications

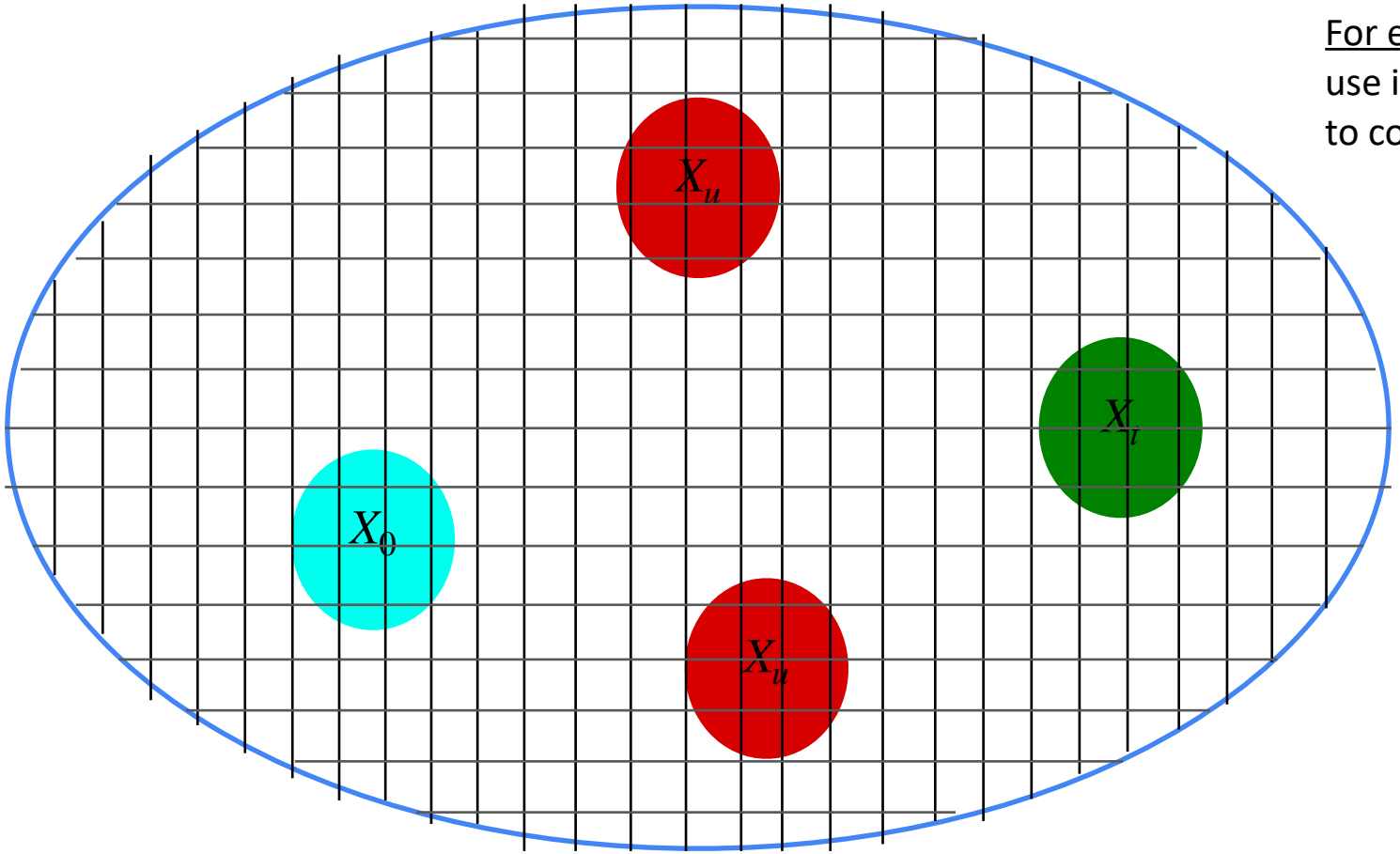
Verifier module

Need to **formally verify** that the learned RASM satisfies:

- Strict expected decrease condition;
- Initial condition;
- Safety conditions.

(Non-negativity satisfied by default, by applying ReLU/softplus on neural RASM output)

Verifier module



<u>For each discretization cell:</u>
use interval arithmetic abstract interpretation (IAAI) [1]
to compute bounds on the RASM over each cell

[1] Gowal et al. *On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models.* arXiv 2018

Abstract interpretation for neural networks

- Abstract interpretation = theory of sound over-approximation of program semantics
- Recent years have seen much interest in abstract interpretation for neural networks, **still an active research area**
- Many abstract domains have been considered (polyhedra, zonotopes, intervals, etc.)
- We will use interval abstract domain

Interval arithmetic abstract interpretation for NNs

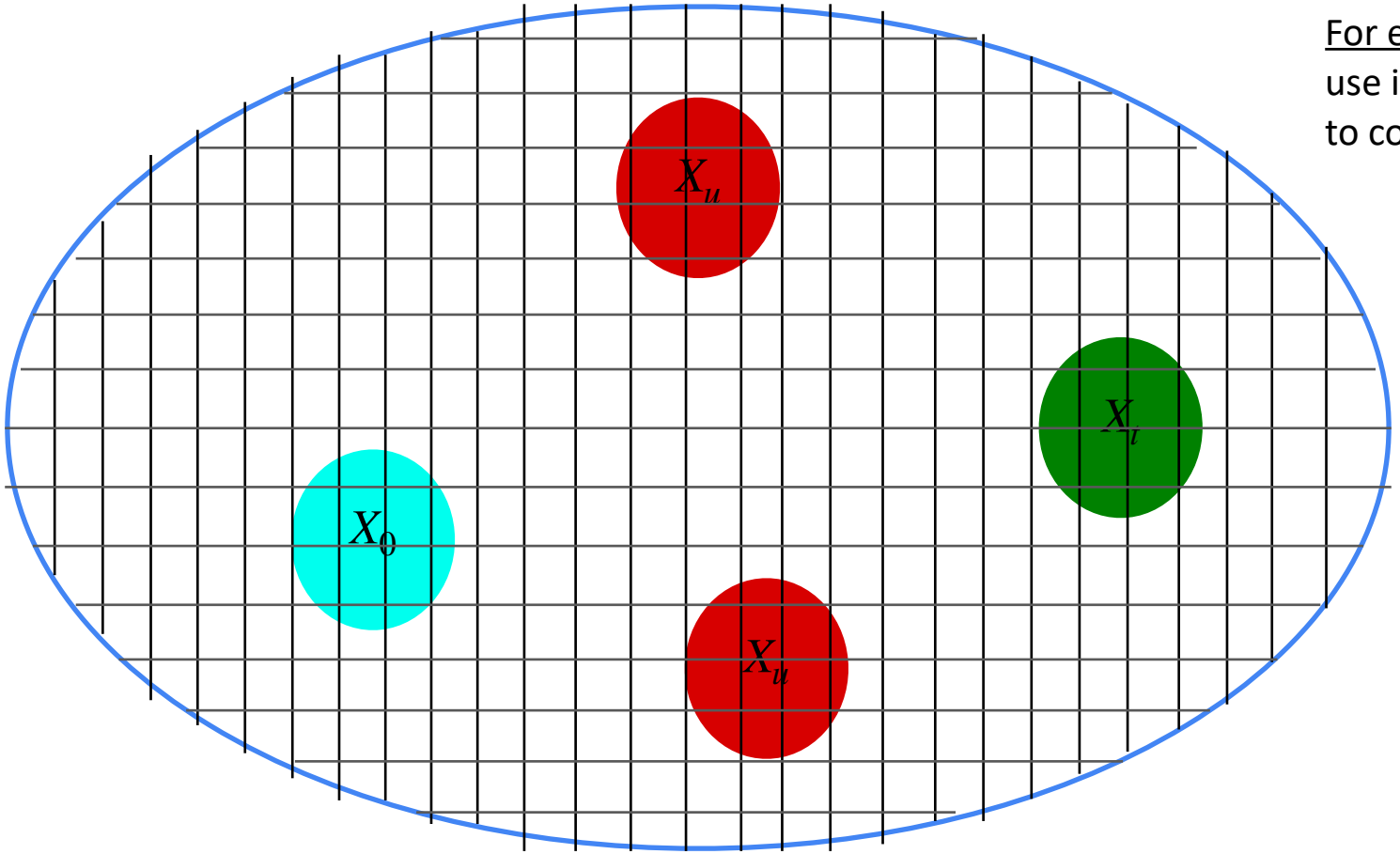
- Main idea: Given a set of input neuron values, propagate interval bounds layer-by-layer to over-approximate the set of all possible values of each neuron in the NN

- How to do this:
  - For two consecutive layers, first apply the linear transformation to interval bounds
  - Then, compute interval over-approximation of the activation function

- To automate the second step, we use **ReLU activation functions** in neural RASMs [1]

[1] Gowal et al. *On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models.* arXiv 2018

Verifier module



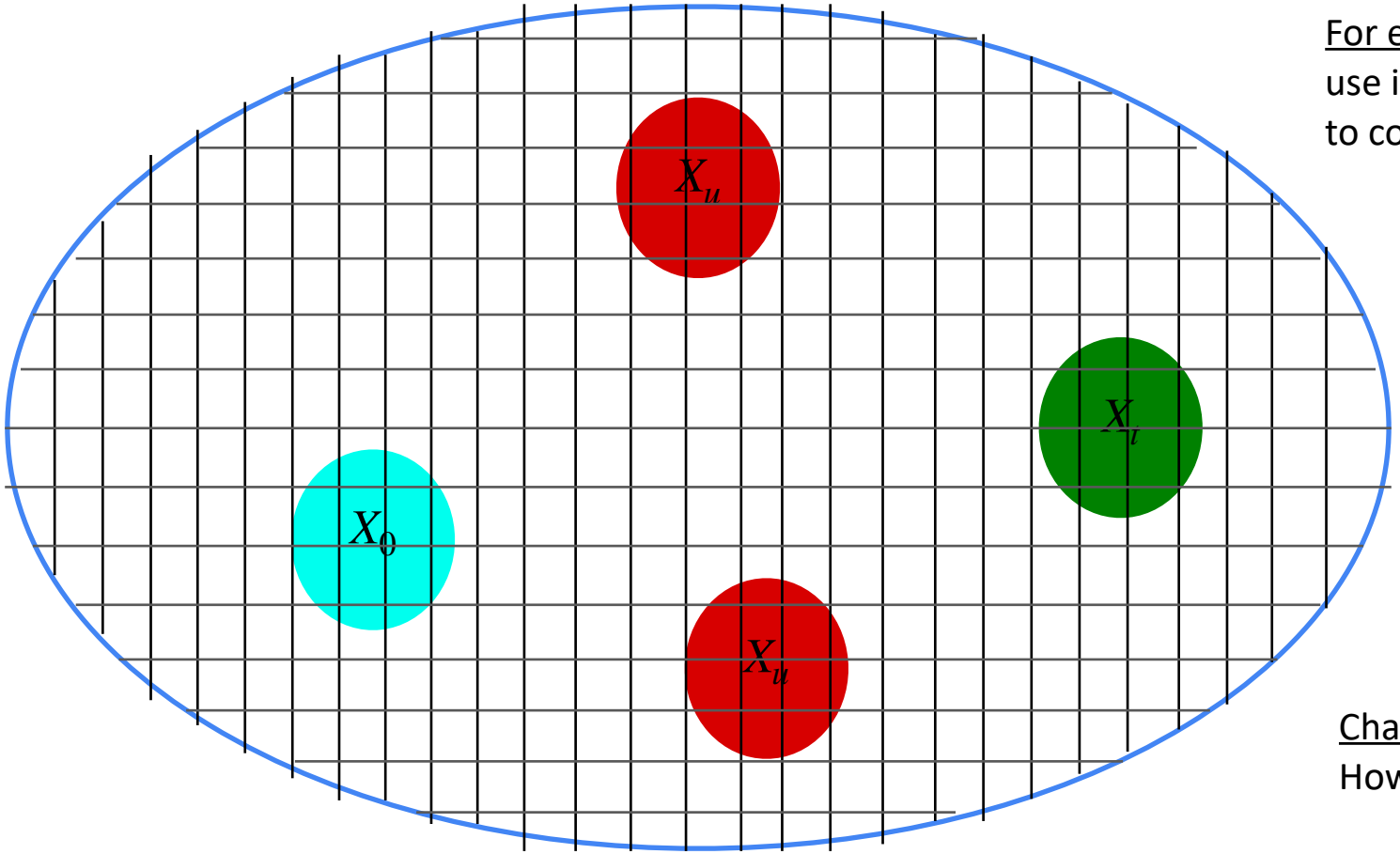
<u>For each discretization cell:</u>
use interval arithmetic abstract interpretation (IAAI) [1] to compute bounds on the RASM over each cell

<u>Check Initial and Safety conditions:</u>

- For Initial condition, we check for each cell that intersects X_0 if **upper** bound ≤ 1
- For Safety condition, we check for each cell that intersects X_U if **lower** bound $\geq 1/(1-p)$

[1] Gowal et al. *On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models.* arXiv 2018

Verifier module



For each discretization cell:
use interval arithmetic abstract interpretation (IAAI) [1] to compute bounds on the RASM over each cell

Check Initial and Safety conditions:

- For Initial condition, we check for each cell that intersects X_0 if **upper** bound ≤ 1
- For Safety condition, we check for each cell that intersects X_U if **lower** bound $\geq 1/(1-p)$

Challenge:
How to verify the Strict expected decrease condition?

[1] Gowal et al. *On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models.* arXiv 2018

Strict expected decrease condition

Need: $\mathbb{E}_{\omega \sim d}\left[V_\nu\left(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega)\right)\right] \leq V_\nu(\mathbf{x}) - \epsilon$ for all $x \in X \backslash X_T$ at which $V(x) \leq 1/(1-p)$

Challenge: We cannot use IA-AI to soundly approximate the <u>expected value of a neural network</u>

Solution: Reduce to checking a stricter condition, but at finitely many points

$$\mathbb{E}_{\omega \sim d}\left[V_\nu\left(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega)\right)\right] < V_\nu(\mathbf{x}) - \tau \cdot K$$

Lipschitz error term

Exploiting Lipschitz continuity

<u>Notation</u>: τ = **mesh** of the discretization grid

(For each state x of a grid cell with center x_c, we have $||x - x_c||_1 \leq \tau$)

L_f, L_π, L_V = Lipschitz constants of L_f, π_θ and V_ν

(We assume that L_f is given, whereas L_π and L_V can be computed as in e.g. [2]))

[2] Szegedy et al. *Intriguing Properties of Neural Networks.* ICLR 2014

Exploiting Lipschitz continuity

For each state x of a grid cell with center x_c,

$$\begin{aligned}
\mathbb{E}_{\omega\sim d}\Big[V_\nu\Big(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega)\Big)\Big] &\leq \mathbb{E}_{\omega\sim d}\Big[V_\nu\Big(f(\mathbf{x_c}, \pi_\theta(\mathbf{x_c}), \omega)\Big) + ||f(x, \pi(x), \omega) - f(x_c, \pi(x_c), \omega)||_1 \cdot L_V\Big] \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V_\nu\Big(f(\mathbf{x_c}, \pi_\theta(\mathbf{x_c}), \omega)\Big) + ||(x, \pi(x), \omega) - (x_c, \pi(x_c), \omega)||_1 \cdot L_F \cdot L_V\Big] \\
&= \mathbb{E}_{\omega\sim d}\Big[V_\nu\Big(f(\mathbf{x_c}, \pi_\theta(\mathbf{x_c}), \omega)\Big)\Big] + ||(x, \pi(x)) - (x_c, \pi(x_c))||_1 \cdot L_F \cdot L_V \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V_\nu\Big(f(\mathbf{x_c}, \pi_\theta(\mathbf{x_c}), \omega)\Big)\Big] + ||x - x_c||_1 \cdot (1 + L_\pi) \cdot L_F \cdot L_V \\
&\leq \mathbb{E}_{\omega\sim d}\Big[V_\nu\Big(f(\mathbf{x_c}, \pi_\theta(\mathbf{x_c}), \omega)\Big)\Big] + \tau \cdot (1 + L_\pi) \cdot L_F \cdot L_V
\end{aligned}$$

Exploiting Lipschitz continuity

On the other hand, for each state x of a grid cell with center x_c,

$$V(x) \geq V(x_c) - ||x - x_c||_1 \cdot L_V \geq V(x_c) - \tau \cdot L_V$$

Hence, combining with the inequality in the previous slide,

$$V(x) - \mathbb{E}_{\omega \sim d}\left[V_\nu\left(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega)\right)\right] \geq V(x_c) - \mathbb{E}_{\omega \sim d}\left[V_\nu\left(f(\mathbf{x_c}, \pi_\theta(\mathbf{x_c}), \omega)\right)\right]$$

$$-\tau \cdot L_V - \tau \cdot (1 + L_\pi) \cdot L_F \cdot L_V$$

To check the Strict expected decrease condition over the whole cell, suffices to show

$$V(x_c) > \mathbb{E}_{\omega \sim d}\left[V_\nu\left(f(\mathbf{x_c}, \pi_\theta(\mathbf{x_c}), \omega)\right)\right] + \tau \cdot K$$

Strict expected decrease condition (continued)

S_{cells} = set of grid cells that intersect $X \backslash X_T$ and whose IA-AI lower bound is $\leq 1/(1-p)$

Need: For each x being a center of some grid cell in S_{cells}, we now need to check

$$\mathbb{E}_{\omega \sim d}\left[V_\nu\left(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega)\right)\right] < V_\nu(\mathbf{x}) - \tau \cdot K$$

Note that this condition now needs to be checked at **finitely many points** defined by set S_{cells}

Expected value bound computation

Goal: Compute upper bound on $\mathbb{E}_{\omega\sim d}\left[V_\nu\left(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega)\right)\right]$ for a <u>fixed</u> $\mathbf{x} \in X$

Problem: V is a neural network, so no closed form solution in general

Solution:

1. cells = hyperrectangular discretization of the support of stochastic disturbance distribution d
2. By the definition of Lebesgue integration (measure theoretical definition of an expectation), the expected value is formally bounded by

$$\sum_{C\in cells} \text{vol}(C) \cdot \inf_{\mathbf{x}\in C} V_\nu(\mathbf{x}) \leq \mathbb{E}_{\omega\sim d}\left[V_\nu\left(f(\mathbf{x}, \pi_\theta(\mathbf{x}), \omega)\right)\right] \leq \sum_{C\in cells} \text{vol}(C) \cdot \sup_{\mathbf{x}\in C} V_\nu(\mathbf{x})$$

3. For each $C \in$ cells, compute an upper bound on $\sup_{\mathbf{x}\in C} V_\nu(\mathbf{x})$ by IA-AI

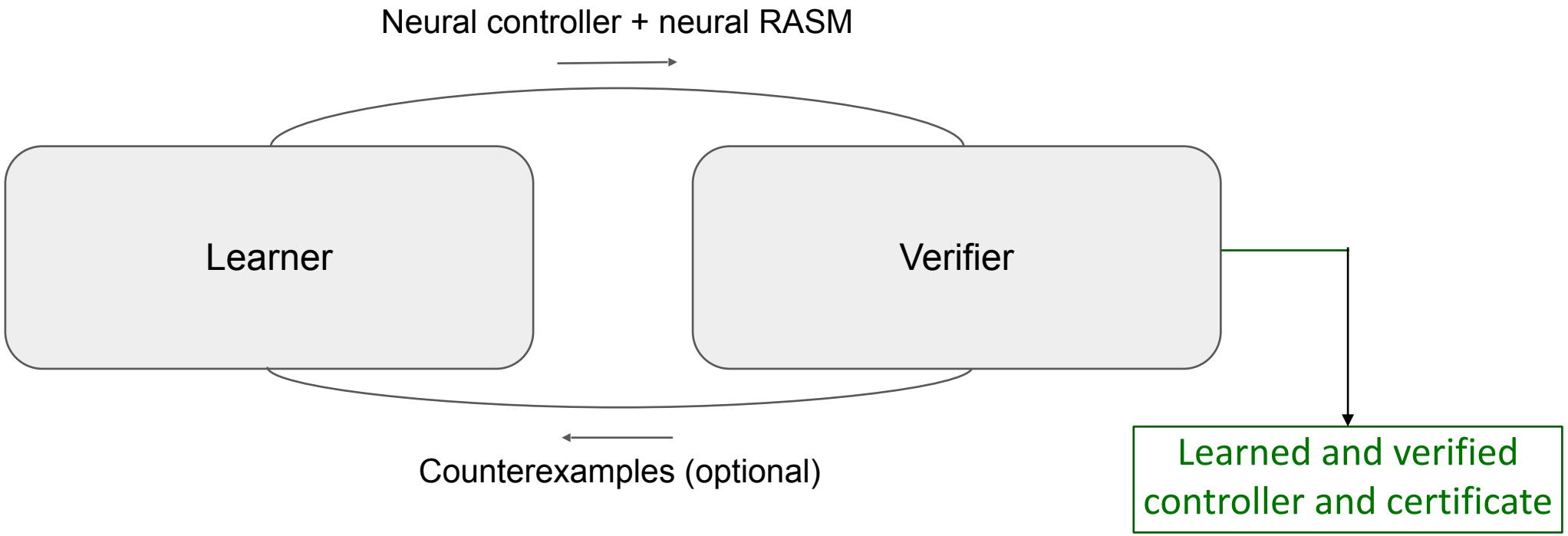
Verifier module - conclusion

Formal verification of the defining conditions of RASMs:

- Strict expected decrease (via discretization + Lipschitz continuity + IA-AI);
- Initial (via IA-AI);
- Safety (via IA-AI).

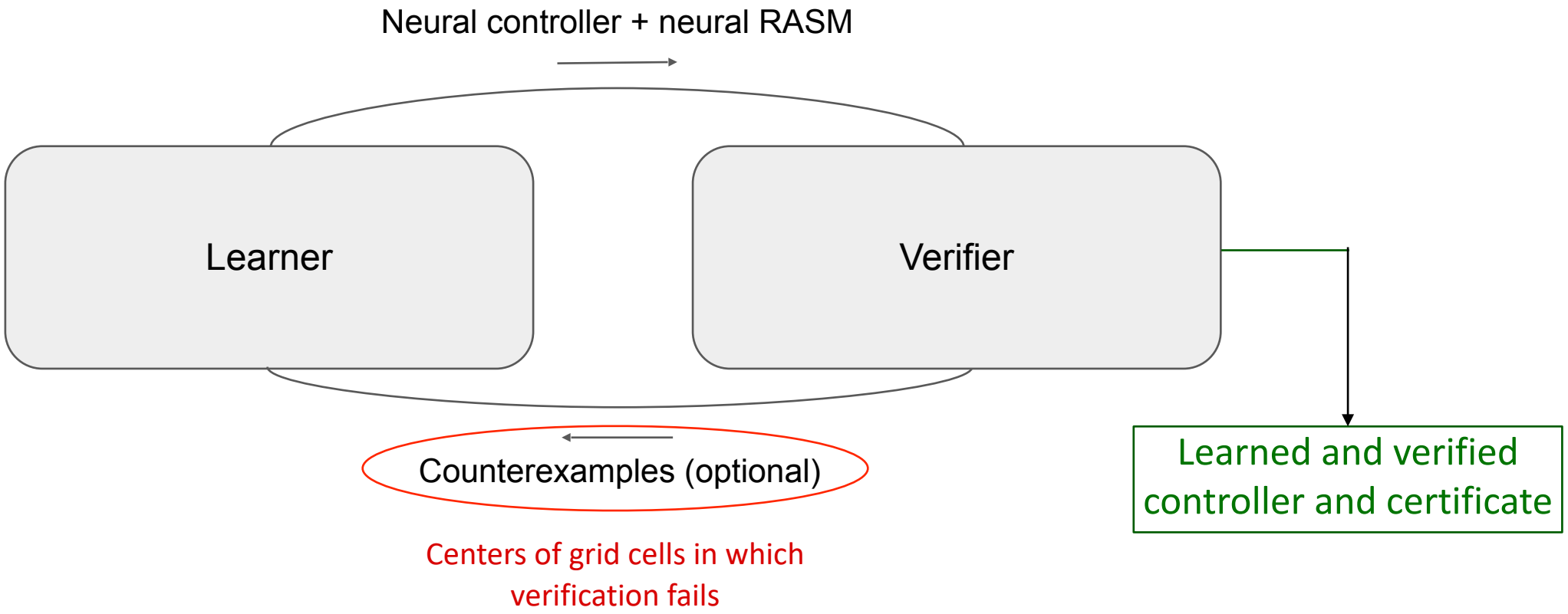
(Non-negativity satisfied by default, by applying ReLU/softplus on neural RASM output)

Learner-verifier framework for controller synthesis



Theorem (Soundness). If the algorithm outputs a neural controller π, then V is a valid RASM and reach-avoidance is satisfied with probability at least p.

Learner-verifier framework for controller synthesis



Theorem (Soundness). If the algorithm outputs a neural controller π, then V is a valid RASM and reach-avoidance is satisfied with probability at least p.

Verifier guides the learner

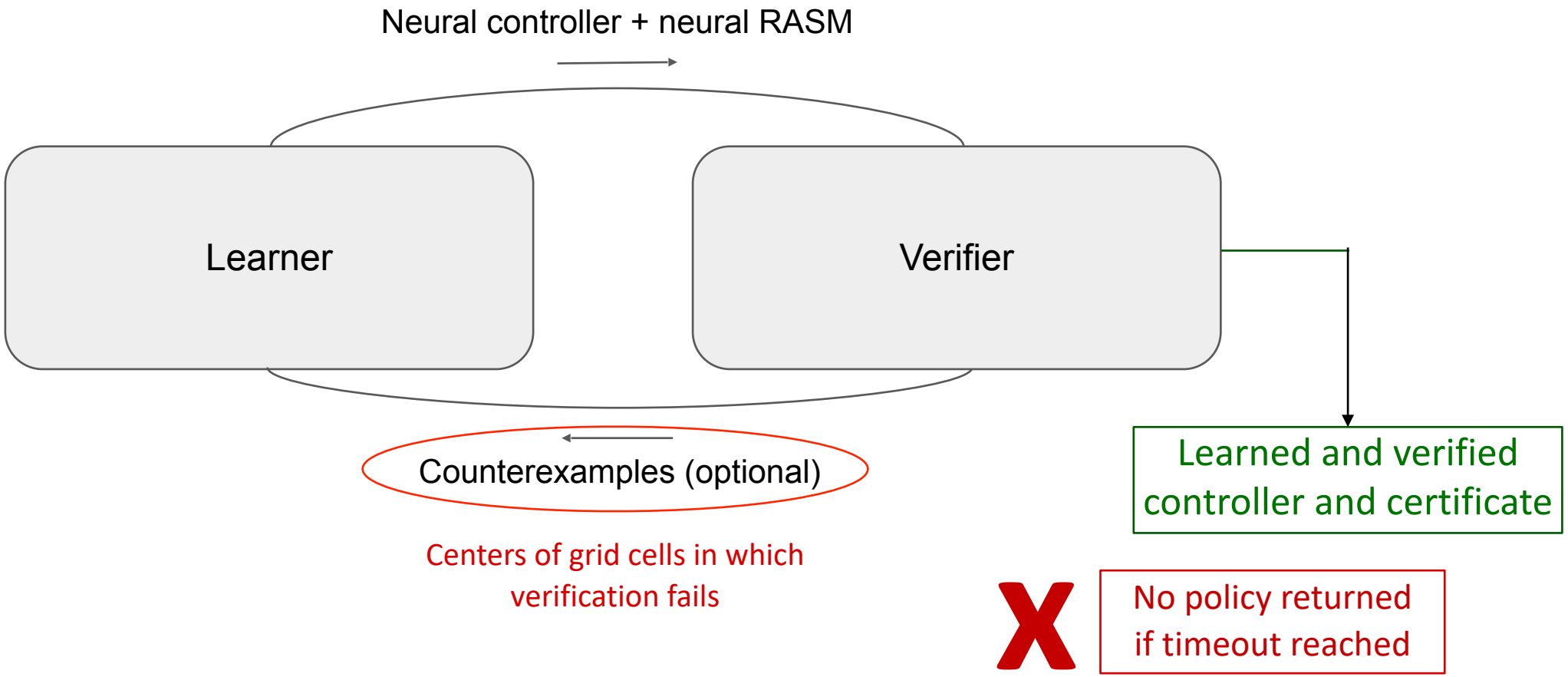
1. **Counterexample guided inductive synthesis (CEGIS)**

   — counterexamples cell centers are added to training sets used by the learner

2. **Adaptive grid refinement**

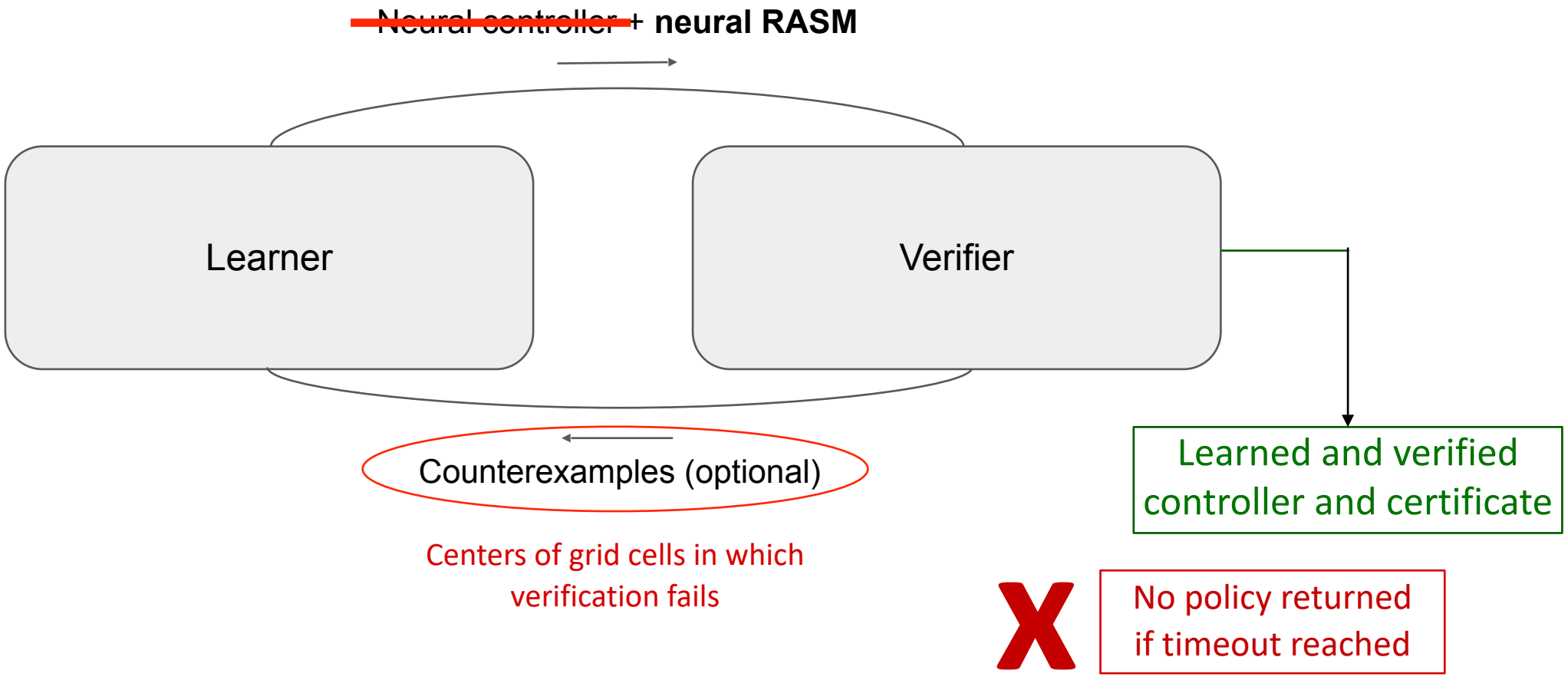
   — grid cells that contain counterexamples are refined

Learner-verifier framework for controller synthesis



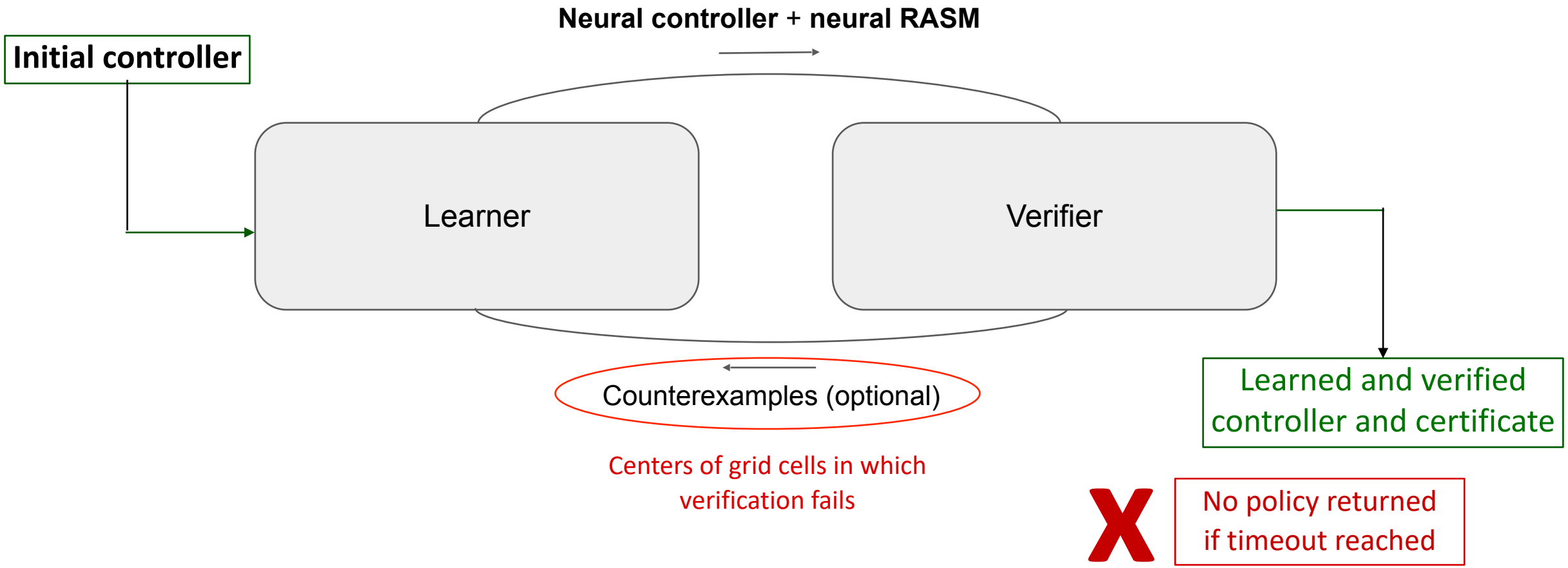
Theorem (Soundness). If the algorithm outputs a neural controller π, then V is a valid RASM and reach-avoidance is satisfied with probability at least p.

Formal verification of neural controllers



Theorem (Soundness). If the algorithm outputs a neural controller π, then V is a valid RASM and reach-avoidance is satisfied with probability at least p.

Formal repair of neural controllers



Theorem (Soundness). If the algorithm outputs a neural controller π, then V is a valid RASM and reach-avoidance is satisfied with probability at least p.

Some practical considerations

1. **Neural controller initialization [1]**

   — good controller initialization is extremely important for the success of learner-verifier algorithm

2. **Lipschitz regularization in the loss function [2]**

   — additional loss term that helps keep neural network Lipschitz constants small

[1] Zikelic, Lechner, Chatterjee, Henzinger. *Learning Stabilizing Policies in Stochastic Control Systems.* SRML@ICLR 2022
[2] Lechner, Zikelic, Chatterjee, Henzinger. *Stability Verification in Stochastic Control Systems via Neural Network Supermartingales.* AAAI 2022

Experimental evaluation

- Three stochastic environments
  - Linear 2D system with non-linear control bounds
  - Inverted pendulum with additive stochastic noise
  - Collision avoidance
- Initialize neural network policy using 100 iterations of PPO
  - Policy network: [128,128] hidden dimension with ReLU activation
- Run prototype implementation of our algorithm
  - RASM network: [128,128] hidden dimension with ReLU activation

Experiment: Solving reach-avoid tasks

Environment	Reach-avoidance probability
2D system	93.3%
Inverted pendulum	92.1%
Collision avoidance	90.4%

Other learner-verifier frameworks

Y.-C. Chang, N. Roohi, S. Gao
A. Abate, M. Giacobbe, et al.
S. Sankaranarayanan et al.

Continuous-time, deterministic

Certificates are Lyapunov functions and control barrier functions

Closed-form/symbolic reasoning
Verification reduced to SMT-solving

[AAAI'22,AAAI'23]

Discrete-time, **stochastic**

Certificates are **supermartingales**

Discretization, Lipschitz continuity
Verification via **abstract interpretation**

Plan for today

1. Learner module for neural controllers and RASMs

2. Verifier module for neural controllers and RASMs

3. **Neural controllers for more general specifications**

Extension to more general specifications

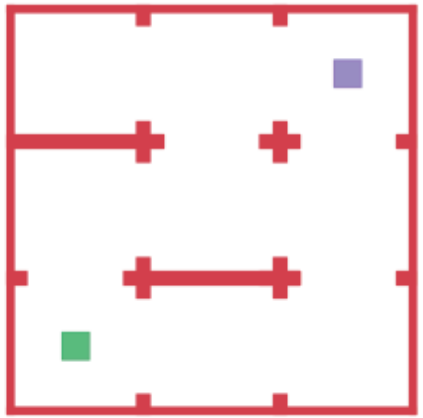
1. **Compositional reasoning** about reach-avoid specifications

2. **Stability** (a.k.a. co-Büchi or reach-and-stay) - see reference at the end

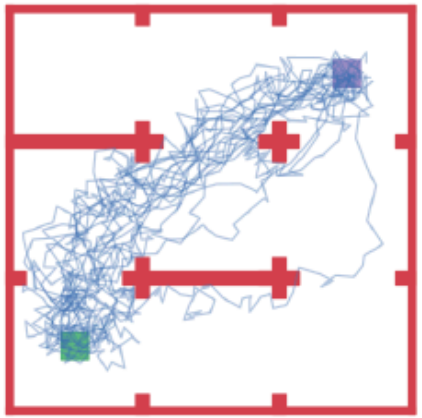
Compositional policy learning

- RL algorithms **struggle** with long-horizon tasks and complex logical specifications
- Hence, we will not attempt to learn a single policy for the complex specification
- Compositional policy learning:
  1. Decompose complex logical specifications into simpler subtasks
  2. Solve subtasks
  3. Compose subtask policies into a global policy
- We can use what we learned so far to reason about compositions of reach-avoid specifications

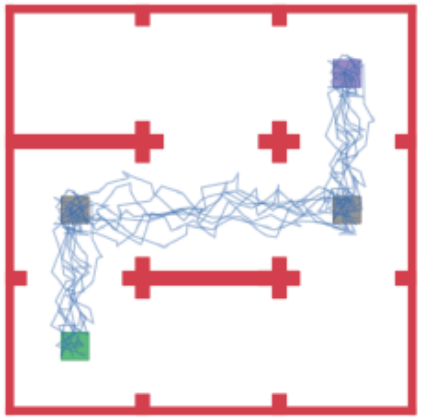
Example: Stochastic 9-rooms environment



Goal: Move from green to purple without hitting a wall



Challenge: End-to-end policy hard to formally verify

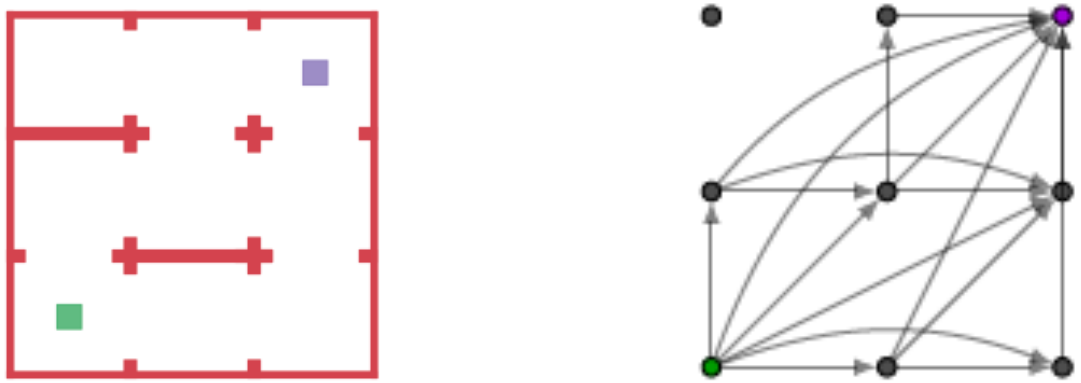


Solution: Decompose the task into simpler subtasks

Compositional policy learning problem

Given: Stochastic dynamical system, SpectRL specification ϕ, probability $p \in [0,1]$

(SpectRL [3,4] = all boolean and sequential compositions of reach-avoid tasks)



Goal: Learn compositional policy that satisfies specification ϕ with probability $\geq p$

(compositional policy = policies for a <u>subset</u> of edges that together solve the task)

[3,4] Jothimurugan, Bansal, Alur, Bastani. NeurIPS 2019, NeurIPS 2022

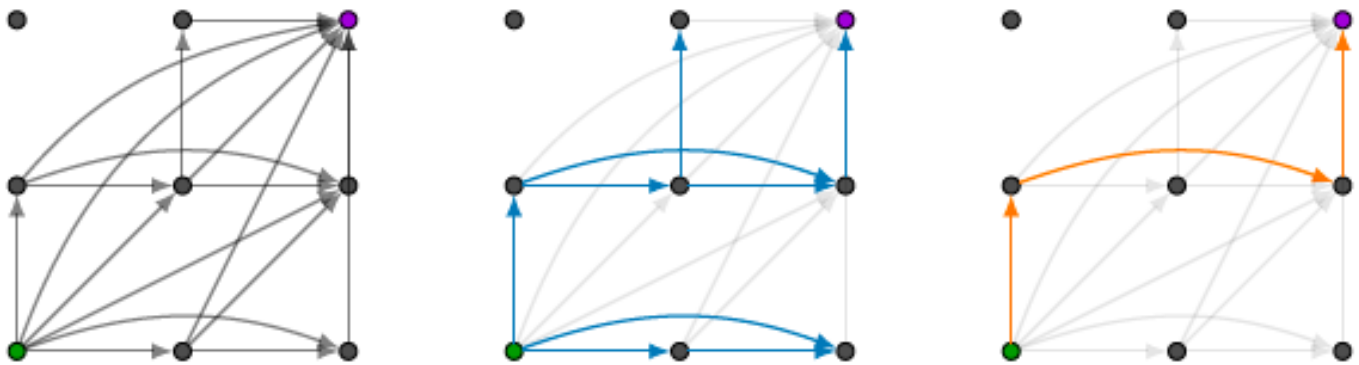
Outline of the approach

1. High-level planning: Decompose the specification into a graph of reach-avoid subtasks

2. Low-level policy learning: Learn neural policy + neural RASM for subtasks

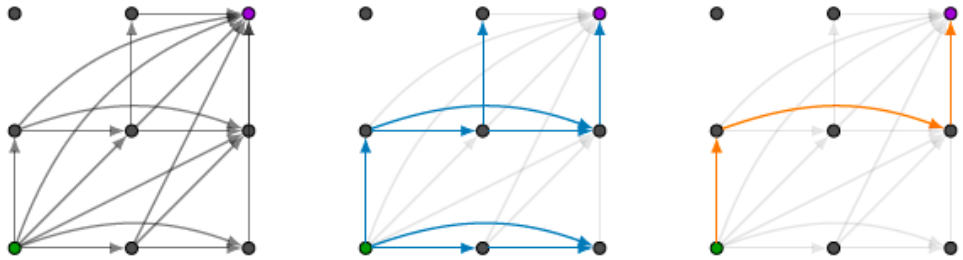
On-demand low-level policy learning

3. Composition: Traverse the graph to compose low-level policies into a global policy,
(while **composing formal guarantees** provided by formal certificates)



Algorithm for compositional policy learning

$s = v_1, v_2, \ldots, v_{|V|} = t \leftarrow$ topological ordering of vertices in G
Prob $\leftarrow$ empty hash map, Prob$[s] \leftarrow 1$
for i = 2, 3, . . . |V| **do**
  Prob$[v_i] \leftarrow 0$
  $N(v_i) \leftarrow \{v \in V \mid (v, v_i) \in E \wedge \text{Prob}[v] \geq p\}$
  for all $v \in N(v_i)$ **do**
    $\pi_{(v,v_i)}$, $p_{(v,v_i)}$ $\leftarrow$ use binary search to find the maximal probability $p_{(v,v_i)}$ for which POLICY+RASM manages to learn a policy $\pi_{(v,v_i)}$ that solves the reach-avoid task associated to edge (v, v_i) with probability at least $p_{(v,v_i)}$
    Prob$[v_i] \leftarrow \max\{\text{Prob}[v_i],\ p_{(v,v_i)} \cdot \text{Prob}[v]\}$
  end for
end for
if Prob$[t] \geq p$ **then**
  $\pi \leftarrow$ global policy obtained by composing edge policies
  return Policy π that guarantees satisfaction of (ϕ, p).
else
  return Policy for the probabilistic specification (ϕ, p) could not be learned.
end if



Summary

1. Learner-verifier framework for synthesizing neural controllers and neural RASMs

2. Also applicable to formal reach-avoidance verification under a <u>given</u> neural controller

3. More general specifications: compositional reasoning about reach-avoid specifications

<u>Future directions</u>: Richer specifications, improved scalability and probability bounds, integration with safe RL approaches, …

Open Positions

SMU
SINGAPORE MANAGEMENT UNIVERSITY

School of
Computing and Information Systems

! Multiple **PhD positions** in computer science (fully funded)

! Multiple **Visiting Research Studentships** (fully funded, 6 months)

Possible topics include (but not limited to)

- Formal methods for probabilistic systems
- Formal methods for trustworthy AI
  (learning-based control, safe RL, NN verification…)
- Program verification (probabilistic and numerical programs)

Apply here (PhD positions): https://computing.smu.edu.sg/phd
Contact me: dzikelic@smu.edu.sg
Personal website: https://djordjezikelic.github.io/

References

Learner-verifier framework for probabilistic reach-avoidance:

[1] D. Zikelic, M. Lechner, T. A. Henzinger, K. Chatterjee. “Learning control policies for stochastic systems with reach-avoid guarantees”. AAAI, 2023

[2] M. Lechner, D. Zikelic, K. Chatterjee, T. A. Henzinger. “Stability verification in stochastic control systems via neural network supermartingales”. AAAI, 2022

[3] K. Chatterjee, T. A. Henzinger, M. Lechner, D. Zikelic. “A learner-verifier framework for neural network controllers and certificates of stochastic systems”. TACAS 2023

Compositional reasoning about reach-avoid specifications:

[4] D. Zikelic, M. Lechner, A. Verma, K. Chatterjee, T. A. Henzinger. “Compositional policy learning in stochastic control systems with formal guarantees”. NeurIPS 2023

Learner-verifier framework for stability (reach-and-stay specifications):

[5] M. Ansaripour, K. Chatterjee, T. A. Henzinger, M. Lechner, D. Zikelic. “Learning provably stabilizing neural controllers for discrete-time stochastic systems”. ATVA 2023