Responsible AI: Deep Learning Model Verification

Progress Review - 3

Candidate: Guanqin Zhang
Main Supervisor: Yulei Sui,
Co-Supervisor: Dilum Bandara, Shiping Chen

University of New South Wales, Sydney, Australia
CSIRO, Data61

Review Agenda

1. **Part I:** Introduction of Neural Network Verification
2. **Part II:** Progress Review Summary
3. **Part III:** Main Contribution of Works During PhD Candidature (leading as first-authored paper)
   - A Tale of Two Cities: Data and Configuration Variances in Robust Deep Learning (IEEE Computer Society)
   - Efficient Neural Network Verification via Order Leading Exploration of Branch-and-Bound Trees (ECOOP 2025) - CORE A
   - Adaptive Branch-and-Bound Tree Exploration for Neural Network Verification (DATE 2025) - CORE B
   - Efficient Incremental Verification of Neural Networks Guided by Counterexample Potentiality (OOPSLA 2025) - CORE A
4. **Part IV:** Conclusion

Introduction Neural Network Verification

Progress Review - 3: Part I

Candidate: Guanqin Zhang
Main Supervisor: Yulei Sui,
Co-Supervisor: Dilum Bandara, Shiping Chen

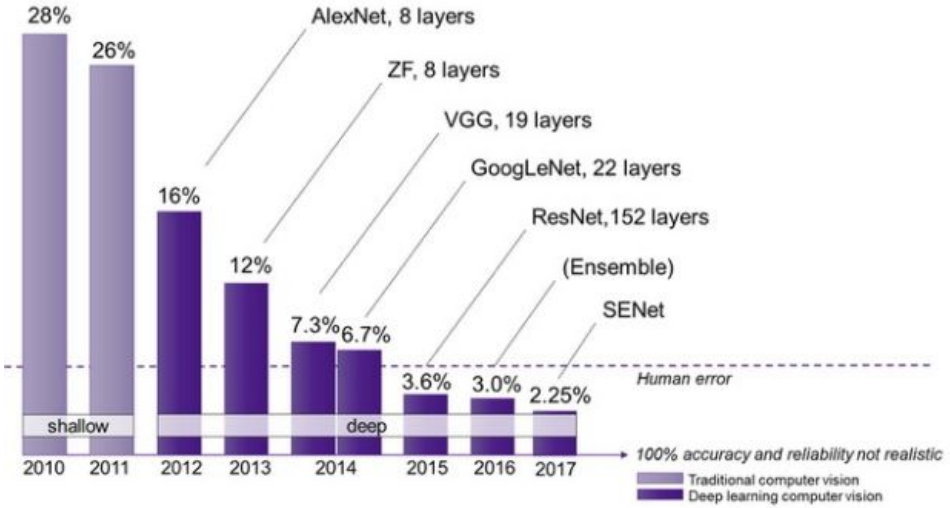
University of New South Wales, Sydney, Australia
CSIRO, Data61

Part I: Introduction of Neural Network Verification

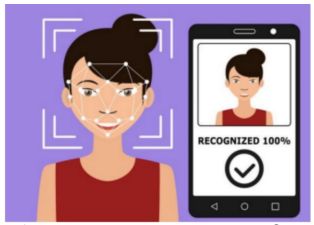
1. Background
2. Local Robustness
3. Verification in Practice
4. Branch & Bound

Modern AI Models are Large and Complex

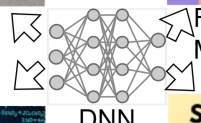
Significant Advancement of Deep Neural Networks



Autonomous Driving Vehicles



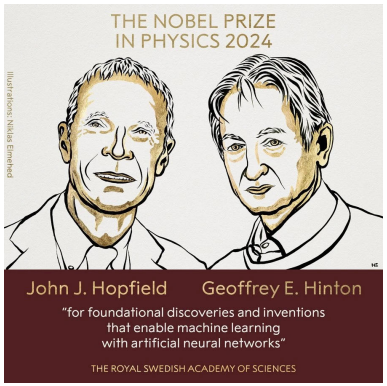
Facial Recognition of Mobile Payment System



DNN

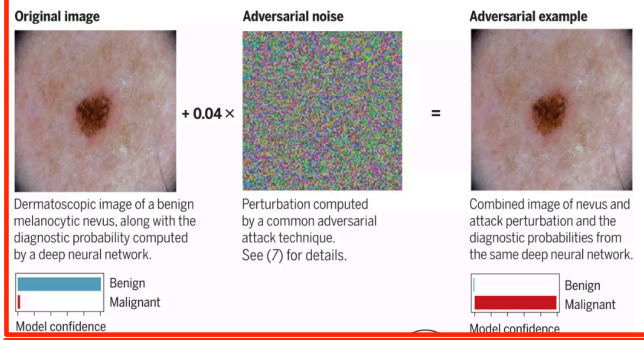
Finance Prediction

Speech Recognition



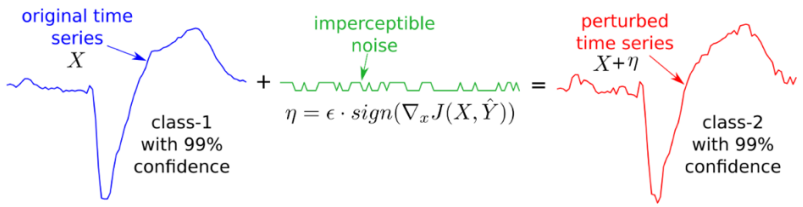
Robustness Verification is Needed in all Hands

Correctness Issue: Misclassification

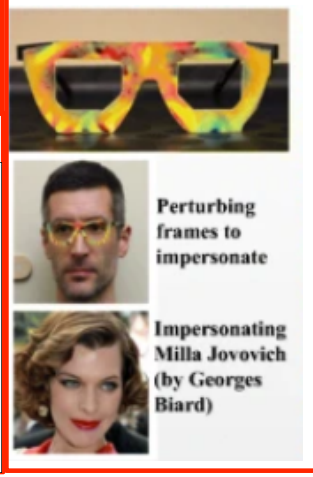


Security Issue: DeepFake

Robustness Issue: fragile to the imperceptible noise



Privacy Issue: Imprecise facial recognition

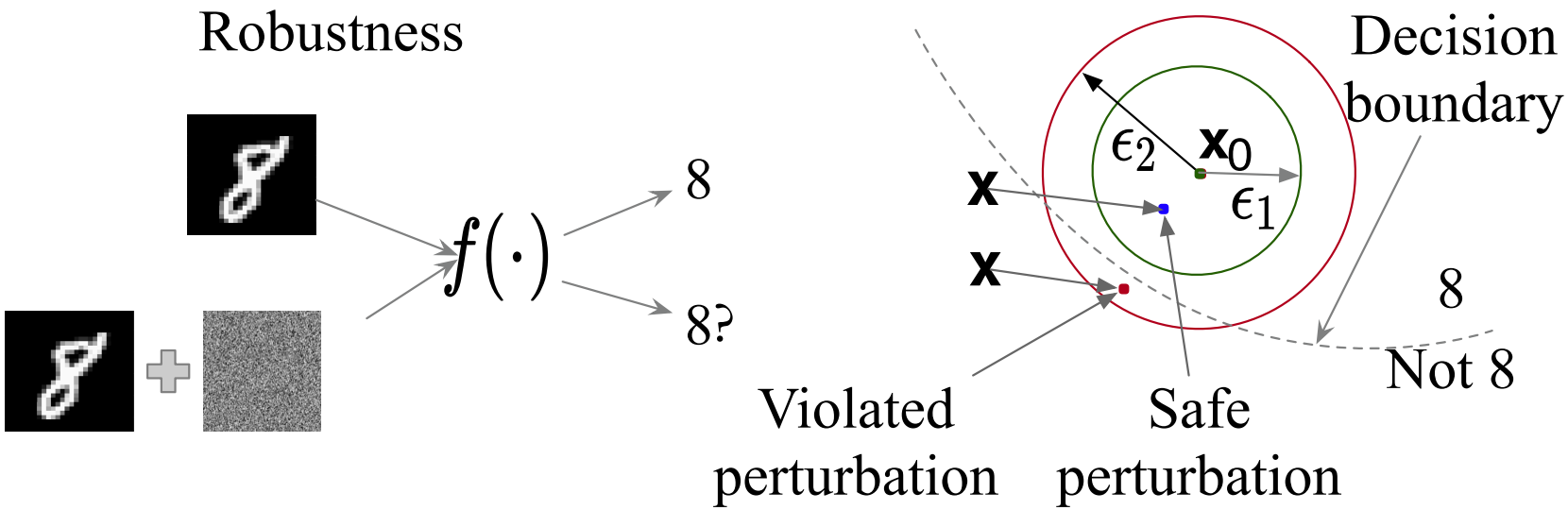


Safety Issue: incorrect recognition

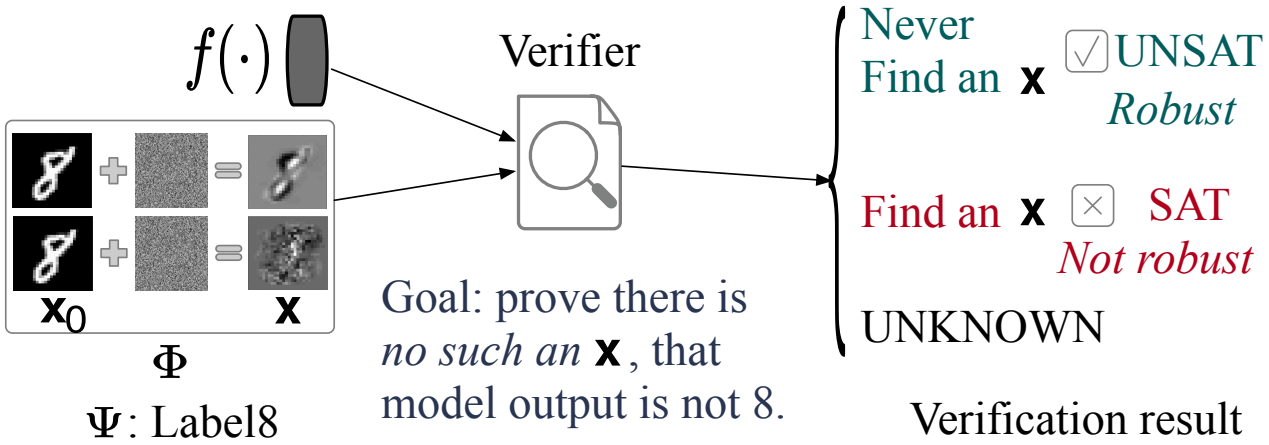
Small perturbations in the input space can **fool** the neural networks to yield **incorrect output**.

Verification for Local Robustness

Given a model $f(\cdot)$, an input $\mathbf{x}_0$, and a perturbation radius ϵ, the model is robust for all perturbed $\mathbf{x}$:

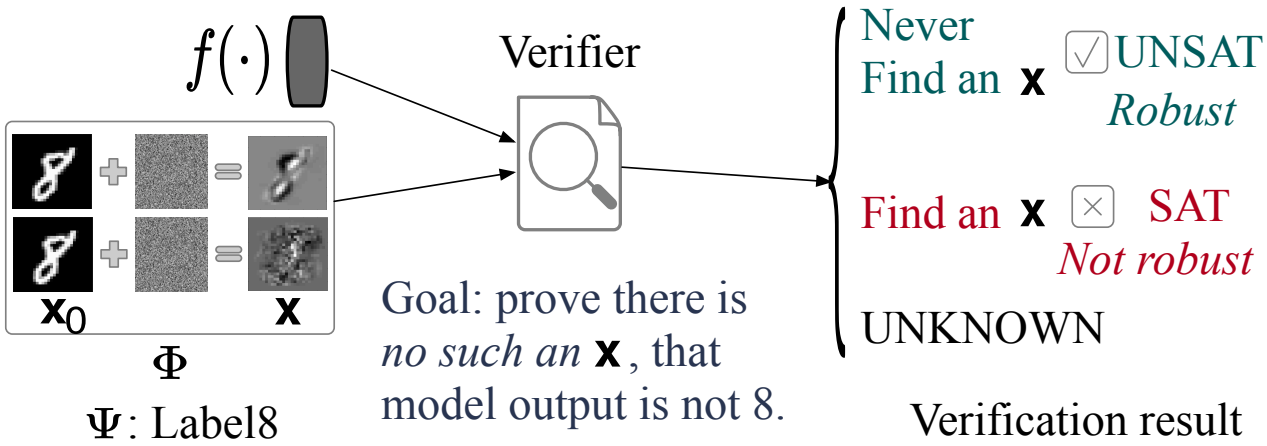


Verification Procedure



- Verification aims to certify "specification guarantees" on model behaviours by rigorously proving the absence of any specification violations.

Verification Procedure



- Verification aims to certify "specification guarantees" on model behaviours by rigorously proving the absence of any specification violations.
- Given an input $\mathbf{x} \in \mathcal{I}$ of a network $f(\cdot)$ to be verified, and a specification consisting of an input property Φ and an output property Ψ, the verification is to check $\mathbf{x} \models \Phi \implies f(\mathbf{x}) \models \Psi$.

Neural Network Testing v.s. Verification

Testing evaluates DNN behaviour, while **verification** proves formal guarantees

- **Methodology**: Testing uses **empirical** methods, while verification employs **formal** techniques.

Neural Network Testing v.s. Verification

Testing evaluates DNN behaviour, while **verification** proves formal guarantees

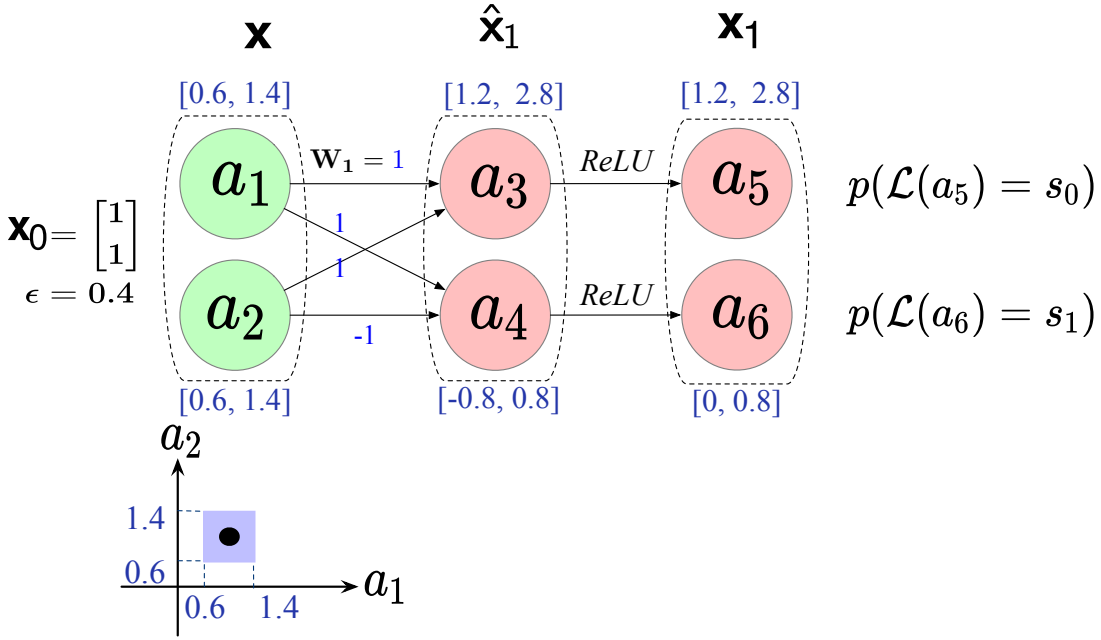
- **Methodology**: Testing uses **empirical** methods, while verification employs **formal** techniques.
- **Completeness**: Testing uncovers issues via **case by case**, but doesn't **guarantee** the absence of errors, whereas verification aims to prove correctness.

Neural Network Testing v.s. Verification

Testing evaluates DNN behaviour, while **verification** proves formal guarantees

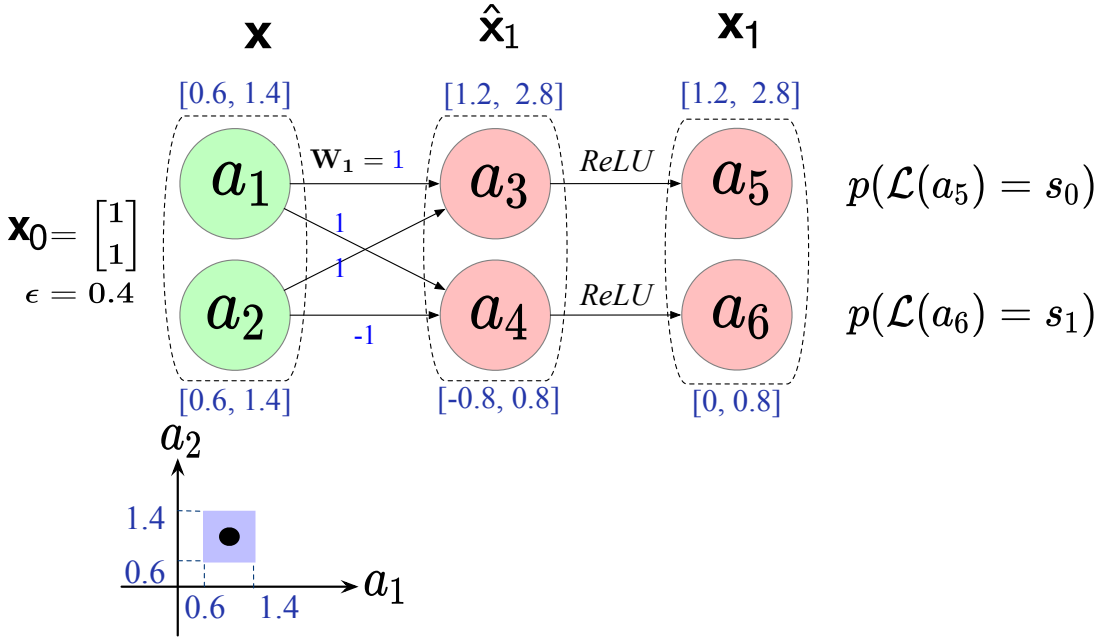
- **Methodology**: Testing uses **empirical** methods, while verification employs **formal** techniques.
- **Completeness**: Testing uncovers issues via **case by case**, but doesn't **guarantee** the absence of errors, whereas verification aims to prove correctness.
- **Scalability**: Testing is more scalable, while verification can be computationally intensive since it aims to approximate **all** possible scenarios to provide a correctness guarantee.

Verification in Practice



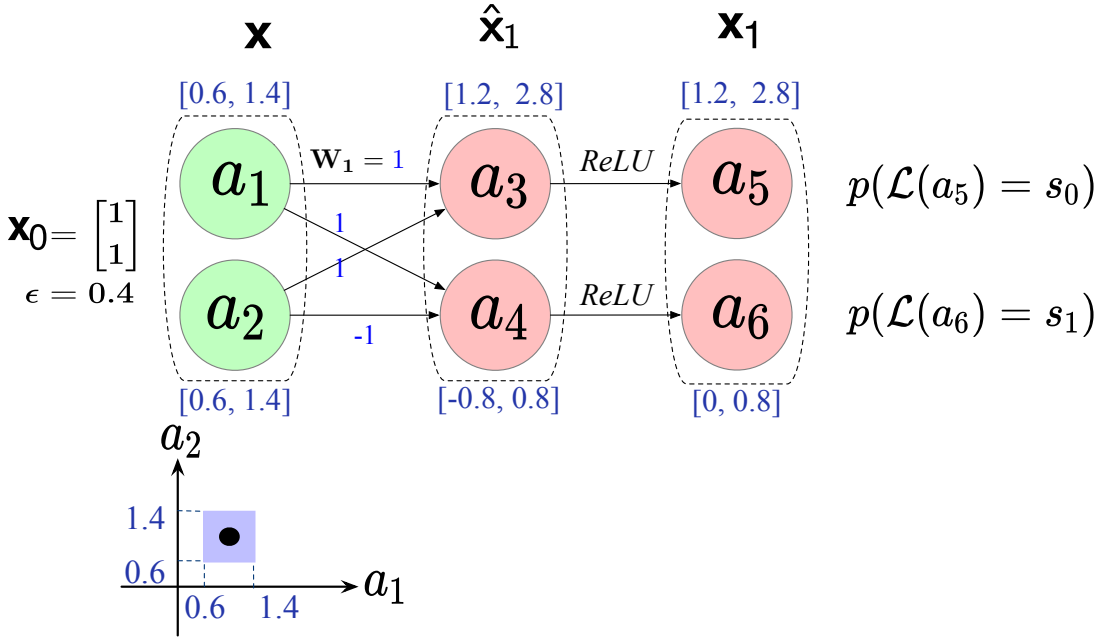
- Perturbed space is $C_{in} = 0.6 \leq a_1 \leq 1.4 \wedge 0.6 \leq a_2 \leq 1.4$

Verification in Practice



- Perturbed space is $C_{in} = 0.6 \leq a_1 \leq 1.4 \wedge 0.6 \leq a_2 \leq 1.4$
- Expected output condition is $C_{out} = a_5 > a_6$

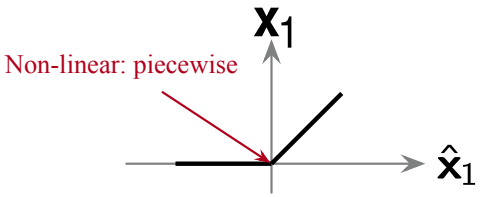
Verification in Practice



- Perturbed space is $C_{in} = 0.6 \leq a_1 \leq 1.4 \wedge 0.6 \leq a_2 \leq 1.4$
- Expected output condition is $C_{out} = a_5 > a_6$
- The neural network is $C_f = \bigwedge_{i=1}^{1} ReLU(\sum_{j=1}^{1} \mathbf{W}\mathbf{x} + b)$.(In our example, $b = 0$)

Why Neural Network Verification is Hard?

ReLU functions is **piece-wise**, i.e., $ReLU(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$



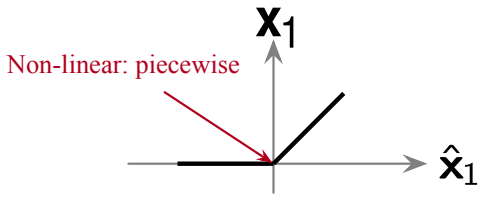
$$\mathbf{x}_1 \equiv ((\hat{\mathbf{x}}_1 \leq 0) \wedge (\mathbf{x}_1 = 0) \vee (\hat{\mathbf{x}}_1 > 0) \wedge (\mathbf{x}_1 = \hat{\mathbf{x}}_1)),$$

bisects in the verification problem.

[6] Katz *et al.* Reluplex: An efficient SMT solver for verifying deep neural networks. CAV 2017

Why Neural Network Verification is Hard?

ReLU functions is **piece-wise**, i.e., $ReLU(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$



$$\mathbf{x}_1 \equiv ((\hat{\mathbf{x}}_1 \leq 0) \wedge (\mathbf{x}_1 = 0) \vee (\hat{\mathbf{x}}_1 > 0) \wedge (\mathbf{x}_1 = \hat{\mathbf{x}}_1)),$$

bisects in the verification problem.

These **disjunction** constraint $C_{in} \wedge C_f \wedge \neg C_{out} =$

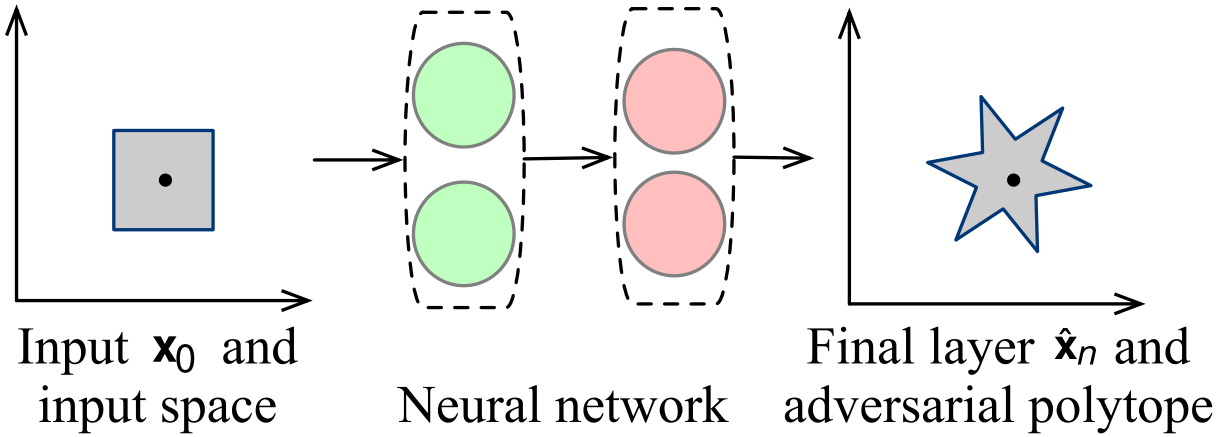
$$\begin{gathered} 0.6 \leq a_1 \leq 1.4 \wedge 0.6 \leq a_2 \leq 1.4 \wedge (a_1 + a_2 = a_3) \wedge (a_1 - a_2 = a_4) \wedge \\ ((a_3 \leq 0) \wedge (a_5 = 0) \vee (a_3 > 0) \wedge (a_5 = a_3) \wedge \\ (a_4 \leq 0) \wedge (a_6 = 0) \vee (a_4 > 0) \wedge (a_6 = a_4) \wedge \\ a_5 \leq a_6 \end{gathered} \tag{1}$$

The SAT solving spends **exponential** time to find all variables assignments that satisfy all disjunct clauses [6]. It is an **NP-hard** problem.

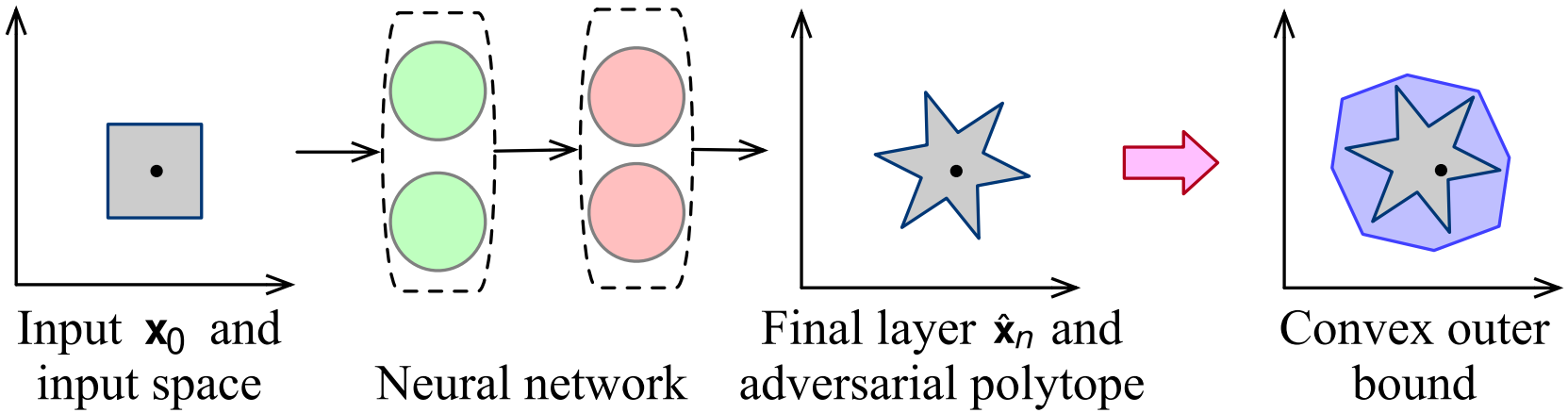
[6] Katz *et al.* Reluplex: An efficient SMT solver for verifying deep neural networks. CAV 2017

Why Neural Network Verification is Hard?

- With **higher** dimensions and **deeper** layers, the **composited** nonlinear activation functions (*ReLU*, *Sigmod*, *tanh*) , operate on **non-convex polytope** hyperspace (**non-convex** NN function $f(\cdot)$).
- The verification process is too computationally costly for non-convex polytope. It needs to approximate tractable bounds to form a provable convex solution space.



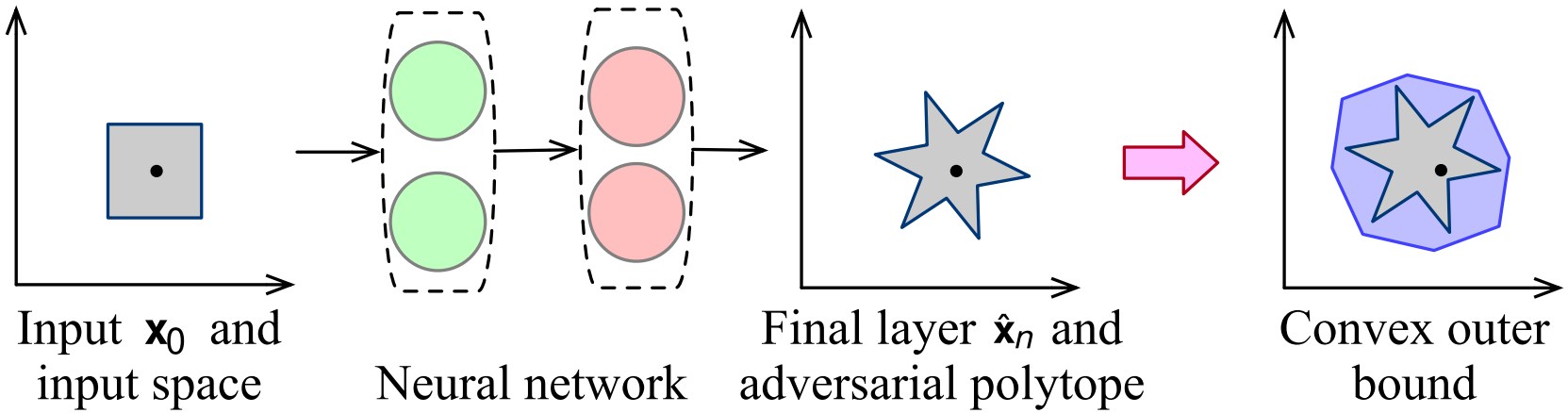
Constraint Extraction: Convex Relaxation



- **Adversarial polytope** is the set of all final-layer activations that can be achieved by applying a norm-bounded perturbation to the input, which is **non-convex**;

[1] Liu, Changliu, *et al.* "Algorithms for verifying deep neural networks." Foundations and Trends in Optimization 2021

Constraint Extraction: Convex Relaxation

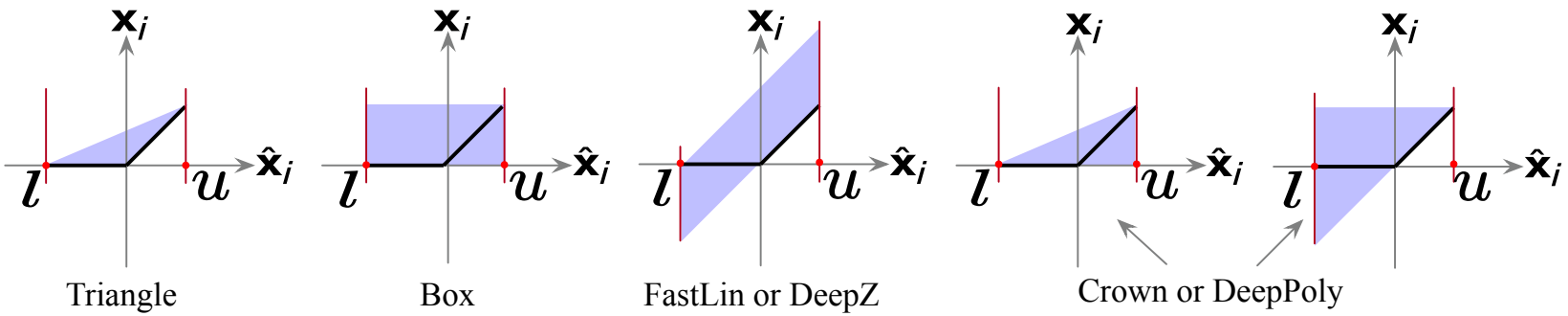


- **Adversarial polytope** is the set of all final-layer activations that can be achieved by applying a norm-bounded perturbation to the input, which is **non-convex**;
- **Convex outer bound** is an over-approximation of the adversarial polytope with linear convexity [1]. Robustness is guaranteed if the classification results do not change (unaffected by perturbations) within the **convex outer bound**

[1] Liu, Changliu, *et al.* "Algorithms for verifying deep neural networks." Foundations and Trends in Optimization 2021

Linear Relaxation (*ReLU*)

Approaches to approximating the *ReLU* to **linear convex** shapes (the blue area) [2]:

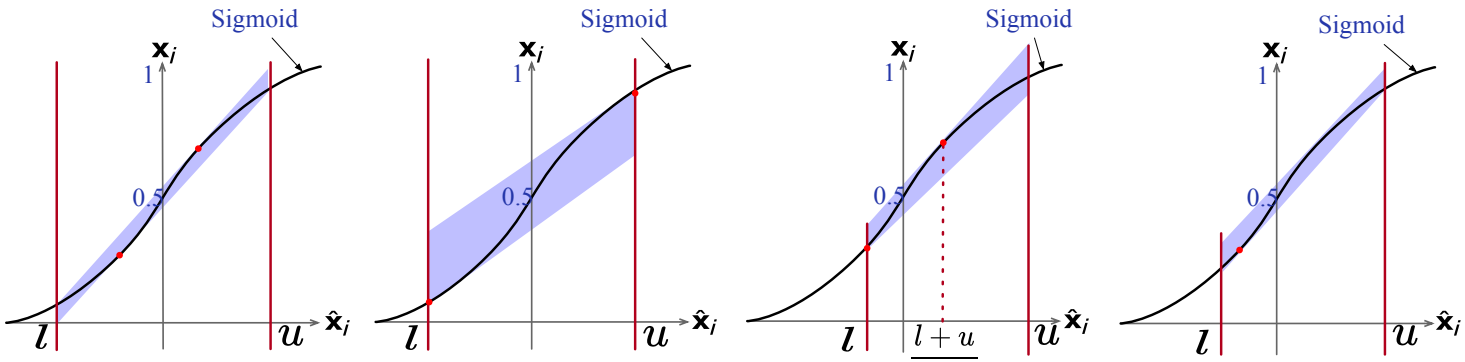


Ehlers *et al.* ATVA 2017 | Gehr *et al.* S&P 2018 | Weng *et al.* ICML 2018 | Crown: Zhang *et al.* NeurIPS 2018 | DeepPoly: Singh *et al.* POPL 2019

[2] Triangle: Ehlers *et al.*; Box: Gehr *et al.*; Zonotope: Fast-Lin and Fast-Lip: Weng *et al.*; DeepPoly: Singh *et al.*; Crown: Zhang *et al.*;

Relaxation (*Sigmoid*)

- Differential functions, such as $Sigmoid(\mathbf{x}) = \frac{1}{e^{\mathbf{x}}+1}$, are '$\mathcal{S}$-**shaped**' curves, which is convex for values less than a **particular point** (e.g., 0), and concave for values greater than that point.
- They can be approximated to a convex region when we decide on values less than a **particular point**, and bounded with different strategies:



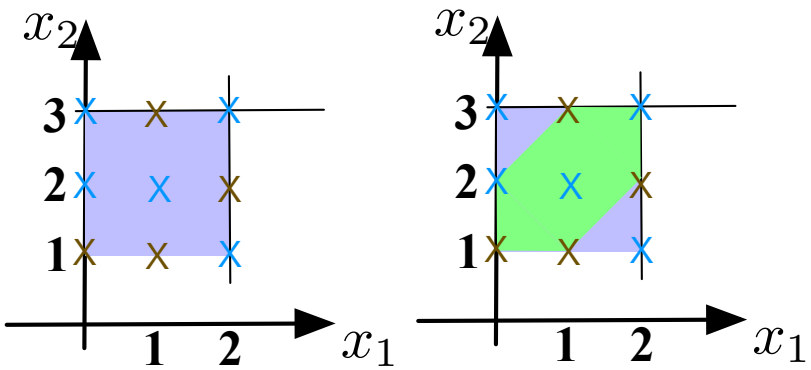
Minimal area and parallel lines — Henriksen *et al.* ECAI 2020

Endpoints — Zhang *et al.* ASE 2022

Minimal area — Henriksen *et al.* ECAI 2020

Parallel lines — Wu *et al.* AAAI 2022

Interval & Zonotop I



Interval **Zonotop**

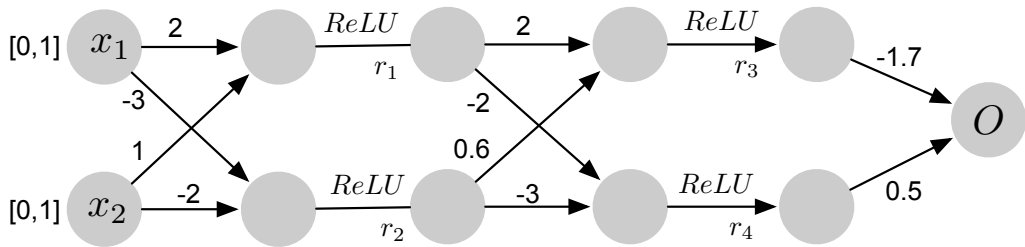
- Brown points are from the concrete domain.
- Blue points are the approximated points from the abstract domain.

- $x_1, x_2 \in \mathbb{Z}$
- Interval space takes 9 integer points, while Zonotope takes 7 points.

A Case Study by BaB-based Sound and Complete DNN Verification

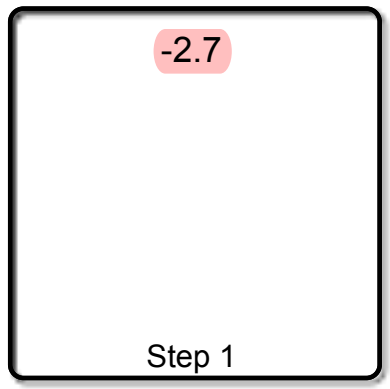
Specification: $\Phi \wedge \Psi$ Input: $\Phi = x_1 \in [0,1] \wedge x_2 \in [0,1]$ Output: $\Psi = (O \geq 0)$

Network: N



To calculate:
Minimize(O) -> LP($\Phi \wedge \Psi \wedge N$)

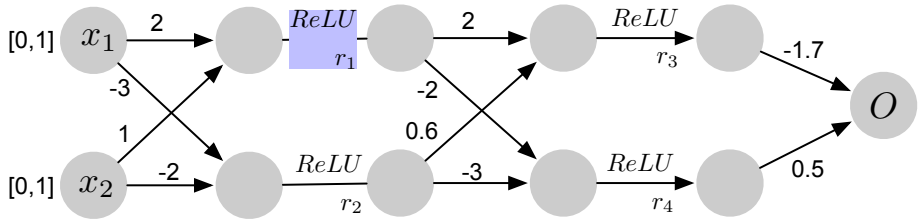
Sound but incomplete

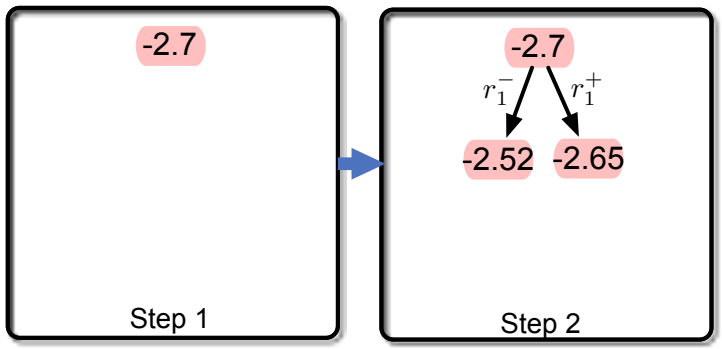


A Case Study by BaB-based Sound and Complete DNN Verification

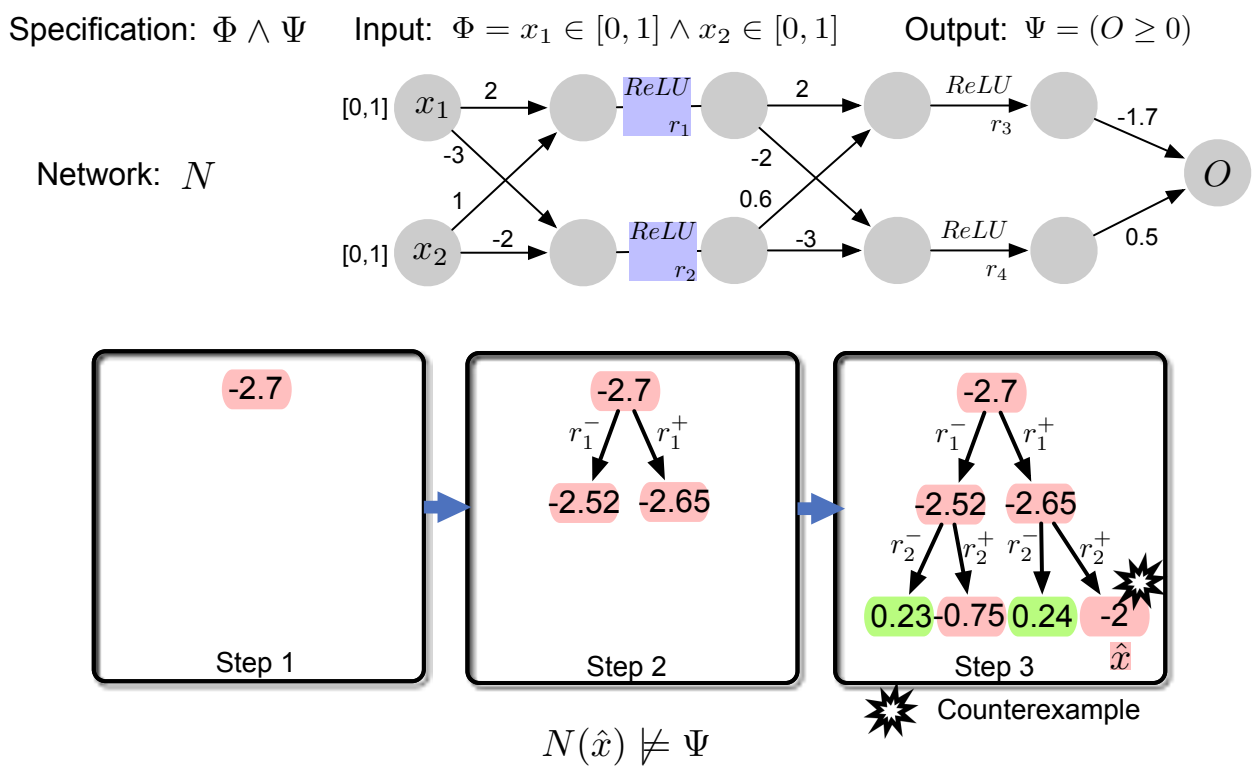
Specification: $\Phi \wedge \Psi$ Input: $\Phi = x_1 \in [0,1] \wedge x_2 \in [0,1]$ Output: $\Psi = (O \geq 0)$

Network: N





A Case Study by BaB-based Sound and Complete DNN Verification



Specification Tree generated after Branch and Bound (BaB), when verifier cannot certify the network.

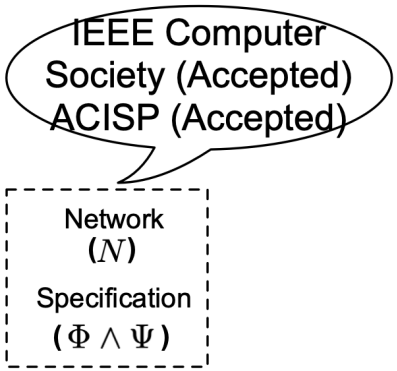
Progress Review Summary

Progress Review - 3: Part II

Candidate: Guanqin Zhang
Main Supervisor: Yulei Sui,
Co-Supervisor: Dilum Bandara, Shiping Chen

University of New South Wales, Sydney, Australia
CSIRO, Data61

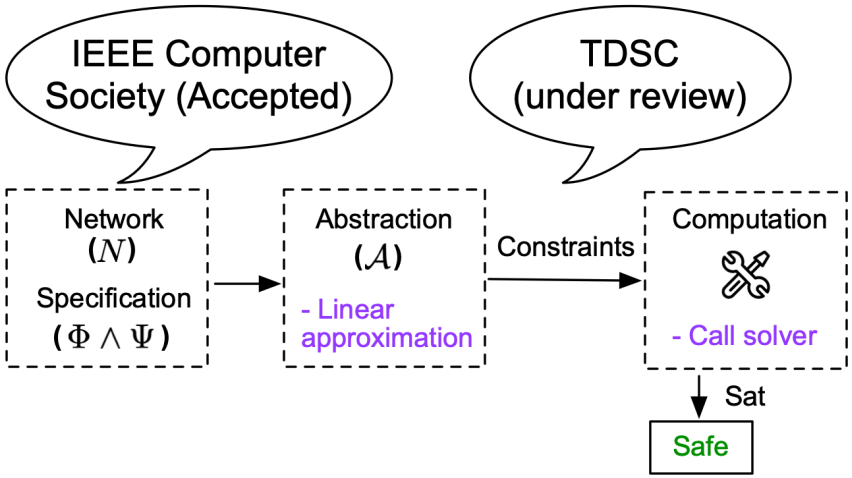
A High-level View of Neural Network Verification Procedures [3] [4]



[3] IEEE Computer: A tale of two cities: data and configuration variances in robust deep learning, **Guanqin Zhang**, J Sun, F Xu, Y Sui, HMND Bandara, S Chen, T Menzies

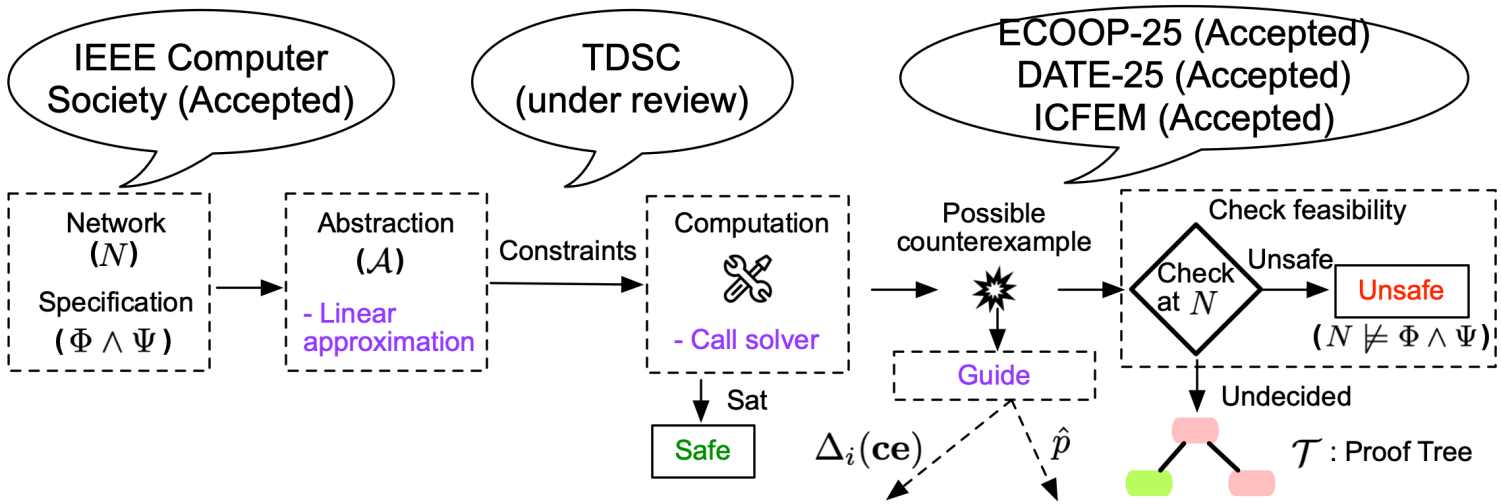
[4] ACISP: Understanding the Robustness of Machine Unlearning Models: **Guanqin Zhang***, F Xu, Dilum Bandara, Shiping Chen and Yulei Sui (conditionally Accept)

A High-level View of Neural Network Verification Procedures [5]



[5] Unveiling the Boundaries of Neural Network Verifiers: A Study on Soundness and Completeness, Feng Xu∗, **Guanqin Zhang**∗, Zhenya Zhang, Xun Li, Ivor W Tsang, Yew-Soon Ong, and Yulei Sui

A High-level View of Neural Network Verification Procedures [6] [7] [8]

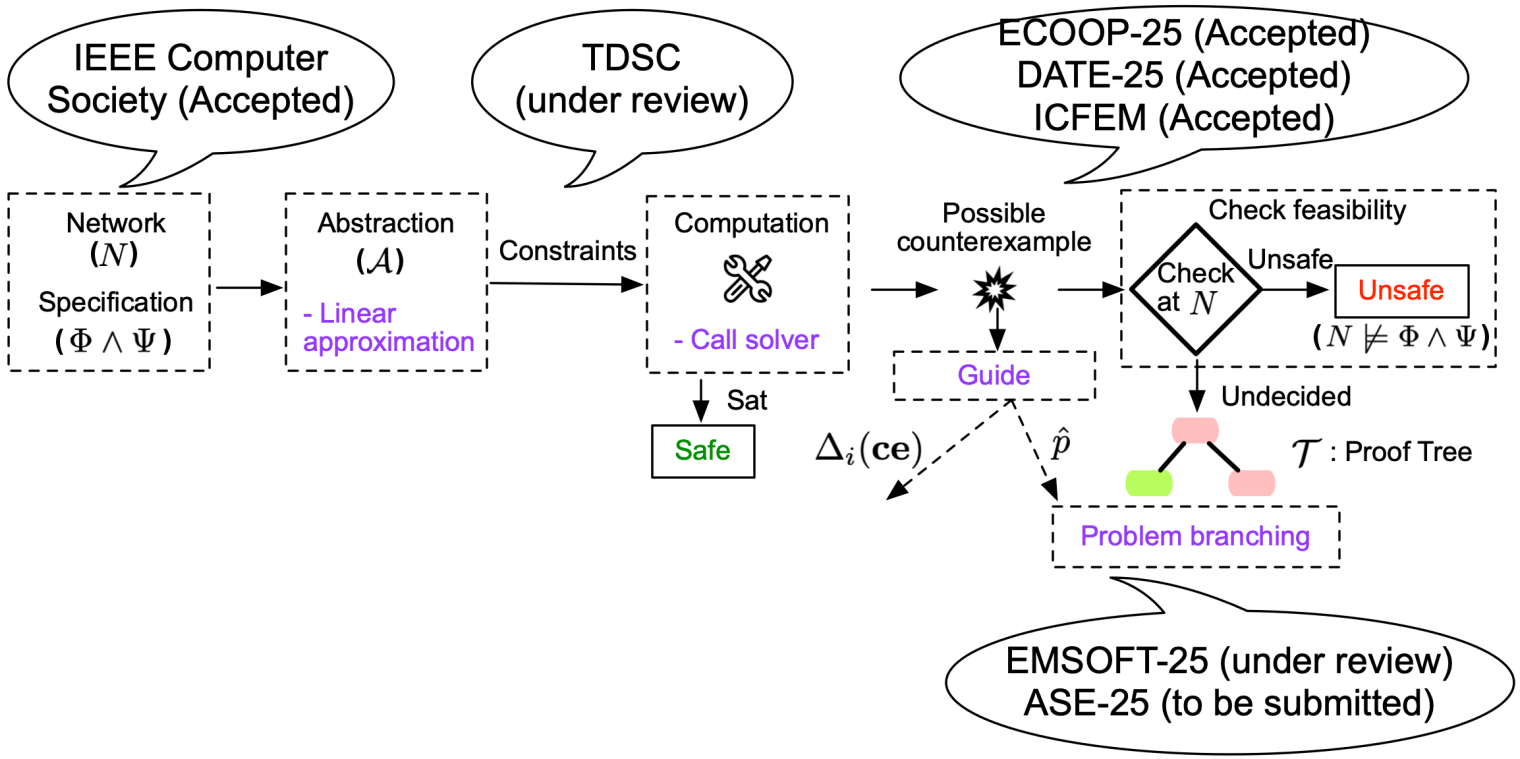


[6] ECOOP25: Efficient Neural Network Verification via Order Leading Exploration of Branch-and-Bound Trees : **Guanqin Zhang**, Kota Fukuda, Zhenya Zhang, Dilum Bandara, Shiping Chen, Jianjun Zhao, Yulei Sui

[7] DATE25: Adaptive Branch-and-Bound Tree Exploration for Neural Network Verification: Kota Fukuda, **Guanqin Zhang**, Zhenya Zhang, Yulei Sui, Jianjun Zhao

[8] ICEFM25: Eager to Stop: Efficient Falsification of Deep Neural Networks: **Guanqin Zhang**

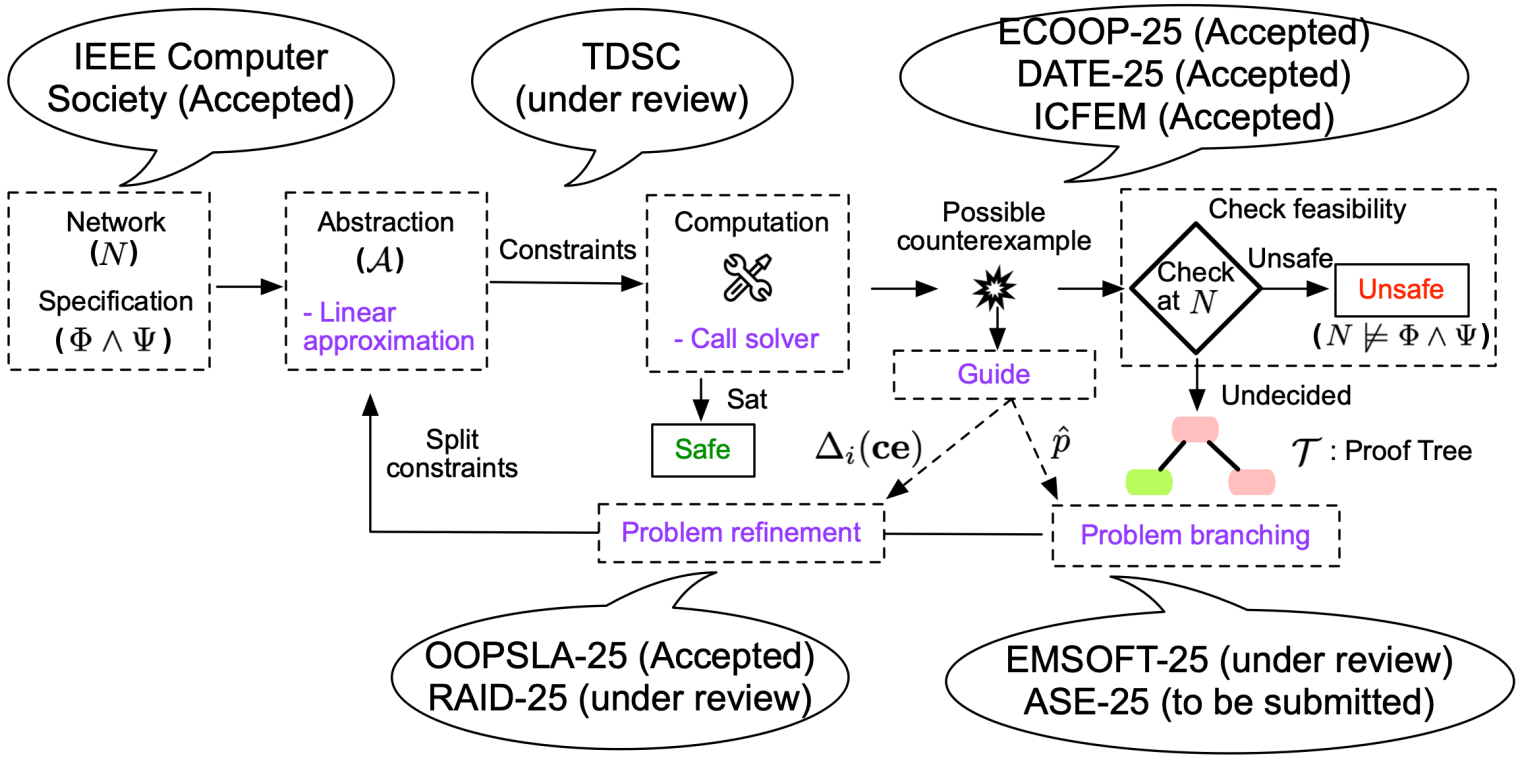
A High-level View of Neural Network Verification Procedures [9] [10]



[9] EMSOFT25: Spurious Counterexample-Guided Branch and Bound Refinement for Neural Network Verification: **Guanqin ZHANG***, Jiawei Ren*, Zhenya Zhang, Yulei Sui

[10] ASE25: Efficient Neural Network Verification by Verdict Boundary Mining: Jiawei Ren*, **Guanqin ZHANG***, Zhenya Zhang, Yulei Sui

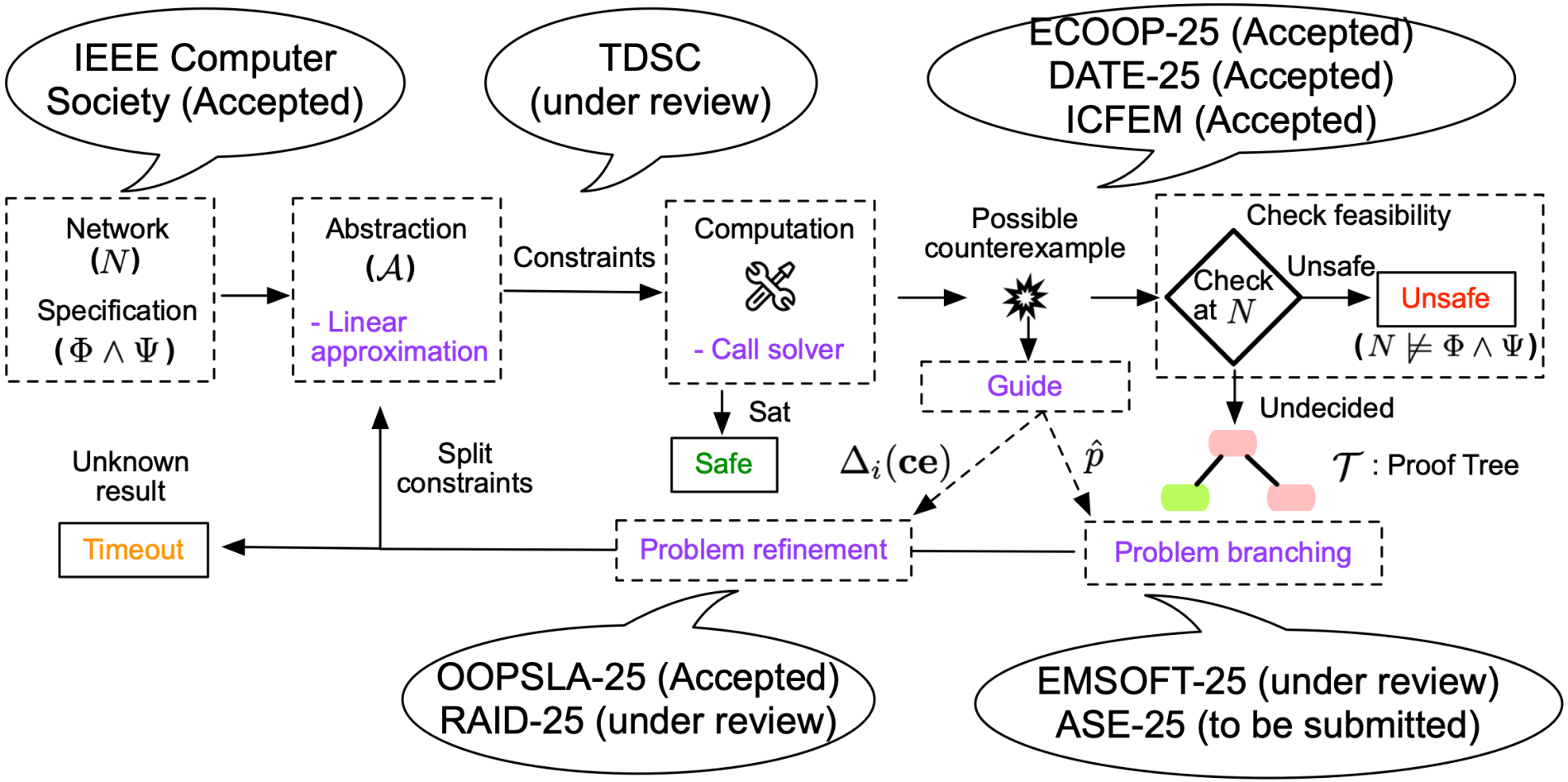
A High-level View of Neural Network Verification Procedures [11] [12]



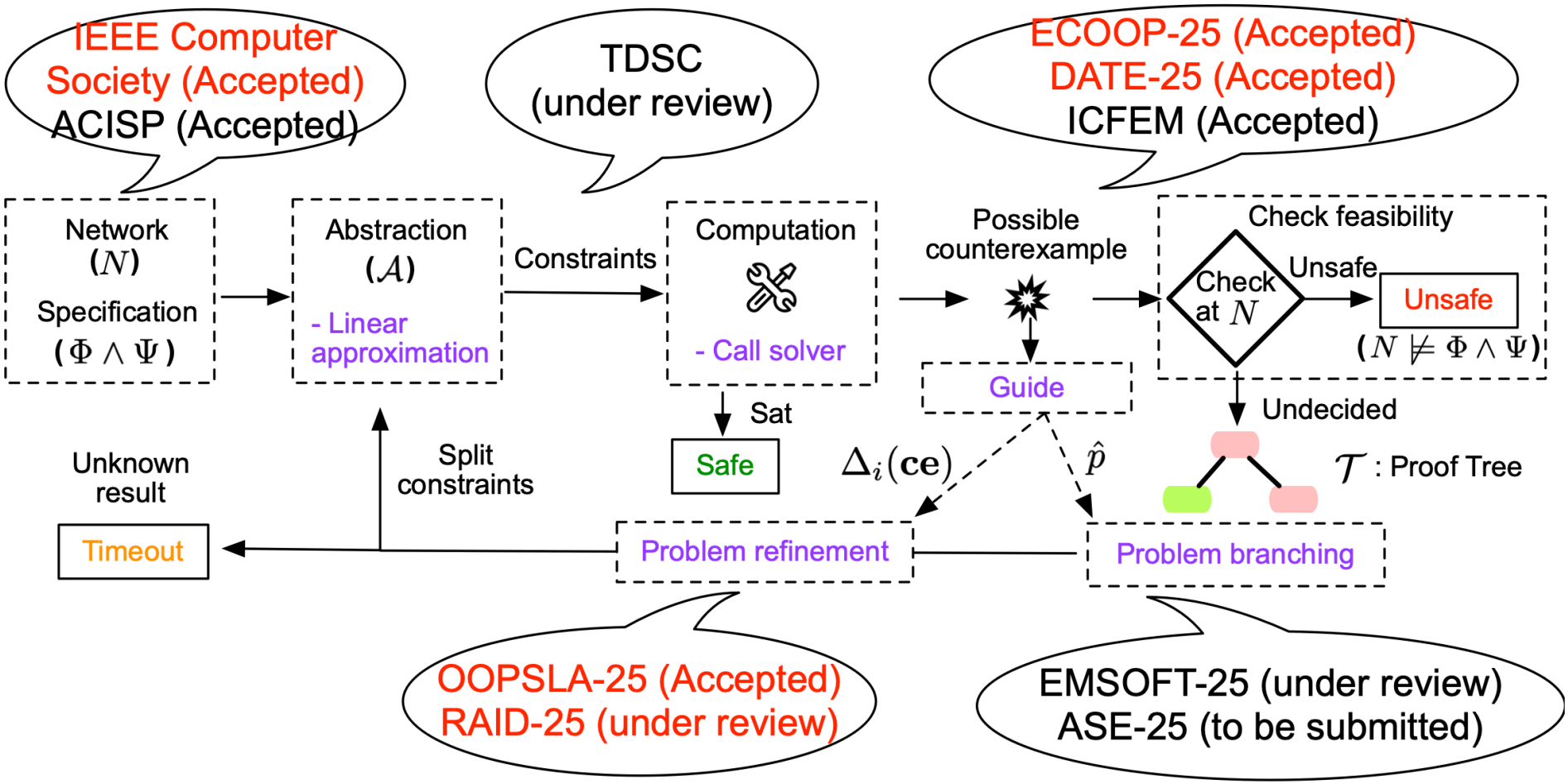
[11] OOSPSLA25: Efficient Incremental Verification of Neural Networks Guided by Counterexample Potentiality: **Guanqin Zhang**, Zhenya Zhang, Dilum Bandara, Shiping Chen, Jianjun Zhao, Yulei Sui

[12] RAID25: Adaptive Branch and Bound Scheduling for Incremental Neural Network

A High-level View of Neural Network Verification Procedures



A High-level View of Neural Network Verification Procedures



Thesis of Neural Network Verification

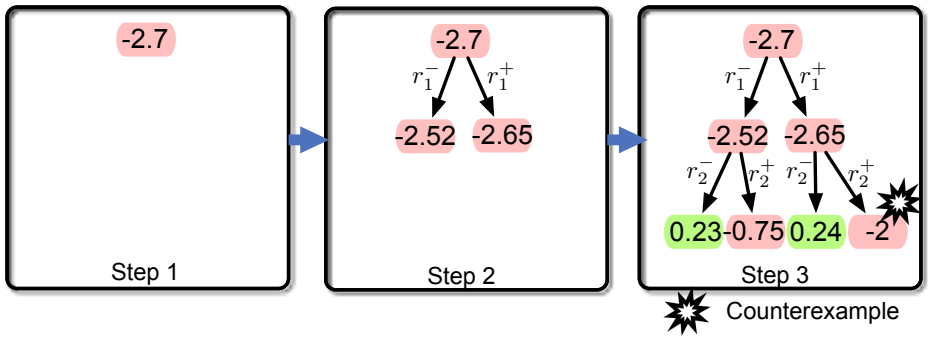
Progress Review - 3: Part III

Candidate: Guanqin Zhang
Main Supervisor: Yulei Sui,
Co-Supervisor: Dilum Bandara, Shiping Chen

University of New South Wales, Sydney, Australia
CSIRO, Data61

Table of Contents

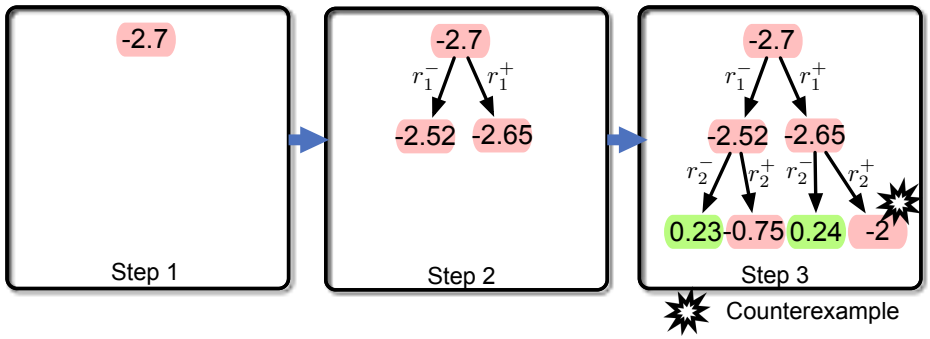
Order Leading Verification Approach (Oliva)[6]



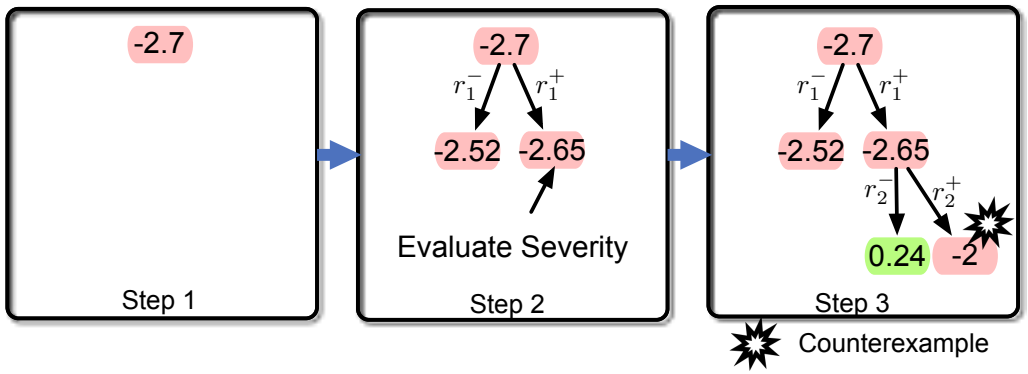
The conventional BaB algorithm manages its subproblems using a **queue** data structure. This approach results in a "first-come, first-served" processing order .

[6] ECOOP 2025: **Guanqin ZHANG(University of New South Wales)**, Kota Fukuda(Kyushu University), Zhenya Zhang(Kyushu University), Dilum Bandara(Data61 CSIRO), Shiping Chen(Data61 CSIRO), Yulei Sui(University of New South Wales), Jianjun Zhao(Kyushu University)

Order Leading Verification Approach (Oliva)[6]



The conventional BaB algorithm manages its subproblems using a **queue** data structure. This approach results in a "first-come, first-served" processing order .



OLIVA considers the severity of the subproblems, such that speedup the verification process and save **1** times of branching, and **2** times of boounding.

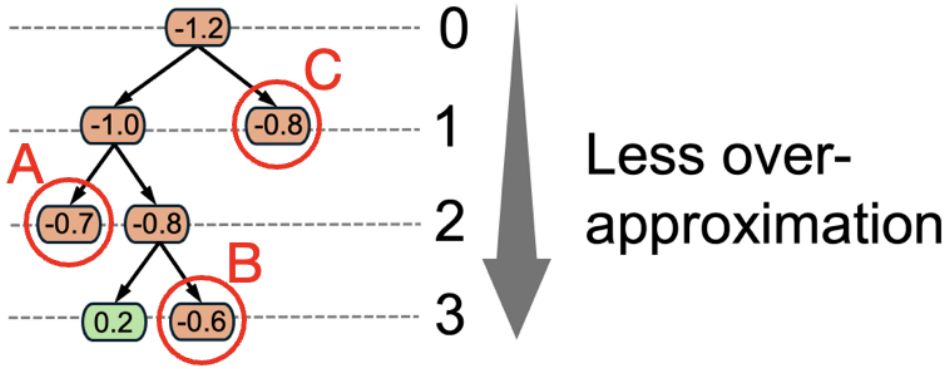
[6] ECOOP 2025: **Guanqin ZHANG(University of New South Wales)**, Kota Fukuda(Kyushu University), Zhenya Zhang(Kyushu University), Dilum Bandara(Data61 CSIRO), Shiping Chen(Data61 CSIRO), Yulei Sui(University of New South Wales), Jianjun Zhao(Kyushu University)

Counterexample Potentiality (CEPO)

- specification distance $\hat{p}$

  ⟹ The nodes with smaller $\hat{p}$ are likely to have high CeP.

- node depth:
  over-approximation level compared to other nodes at different depths.



- Definition of CeP

$$[\![\Gamma]\!] := \begin{cases} -\infty & \text{if } \hat{p} > 0 \\ +\infty & \text{if } \hat{p} < 0 \text{ and } \texttt{valid}(\hat{x}) \\ \lambda \frac{\texttt{depth}(\Gamma)}{K} + (1-\lambda)\frac{\hat{p}}{\hat{p}_{min}} & \text{otherwise} \end{cases}$$

⟹ A: -0.62, B: -0.65, C: -0.64 ⟹ B is selected

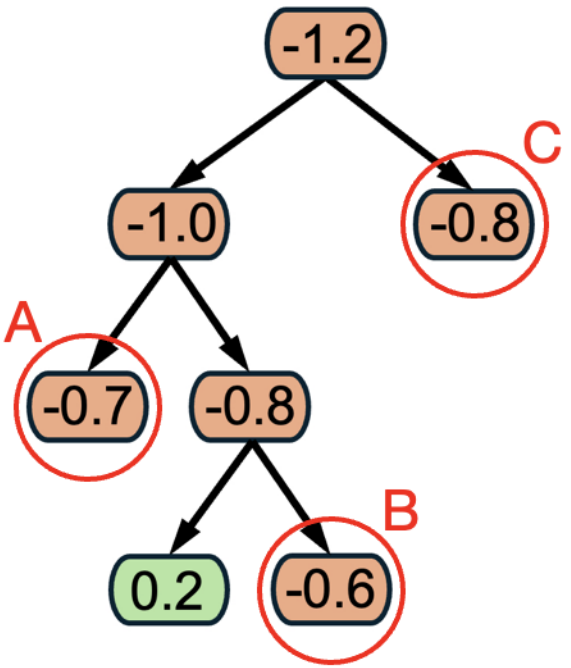
Monte Carlo tree search (MCTS)

- Monte Carlo Tree Search (MCTS) effectively guides node expansion while addressing scenarios where selection based on CeP alone is difficult.

Condition:

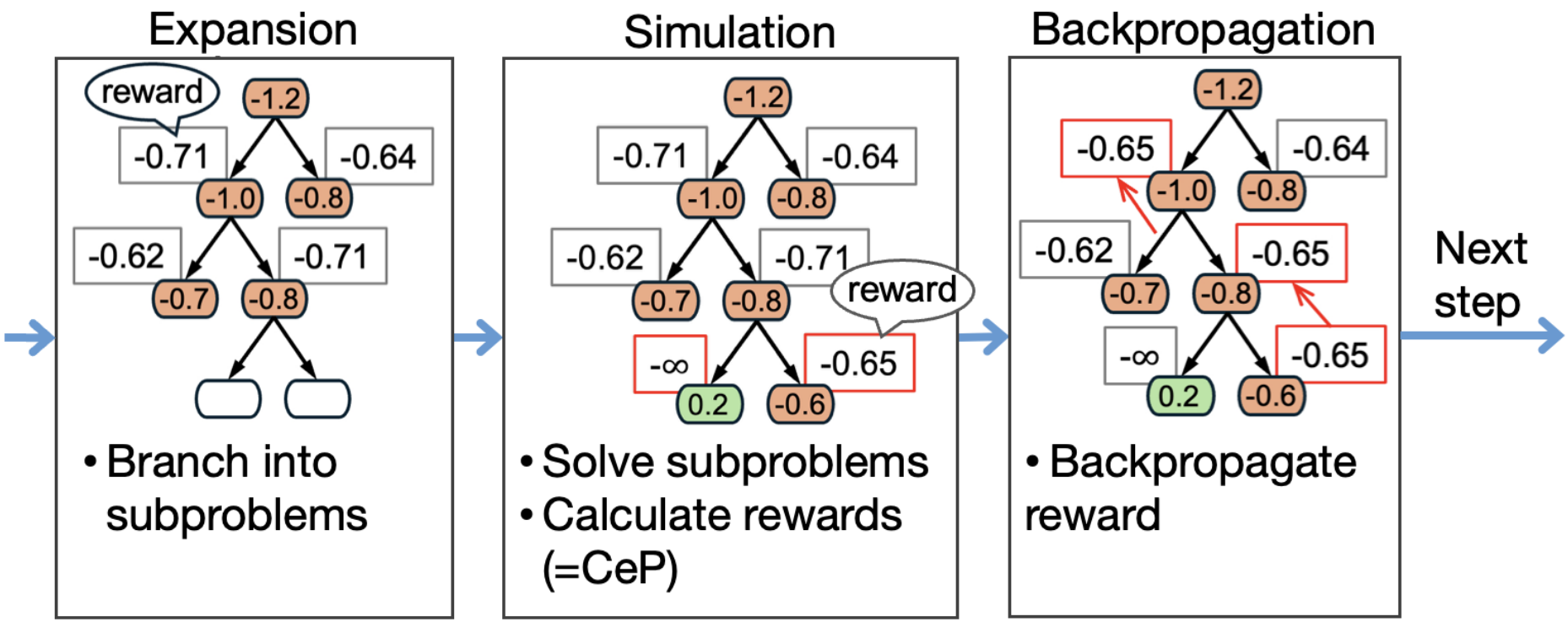
- Node expansion options: A, B, and C have close CeP values.
- The left side: exploited and likely contains counterexamples.
- The right side: unexplored but may also contains counterexamples.

⇒ What is the best choice?

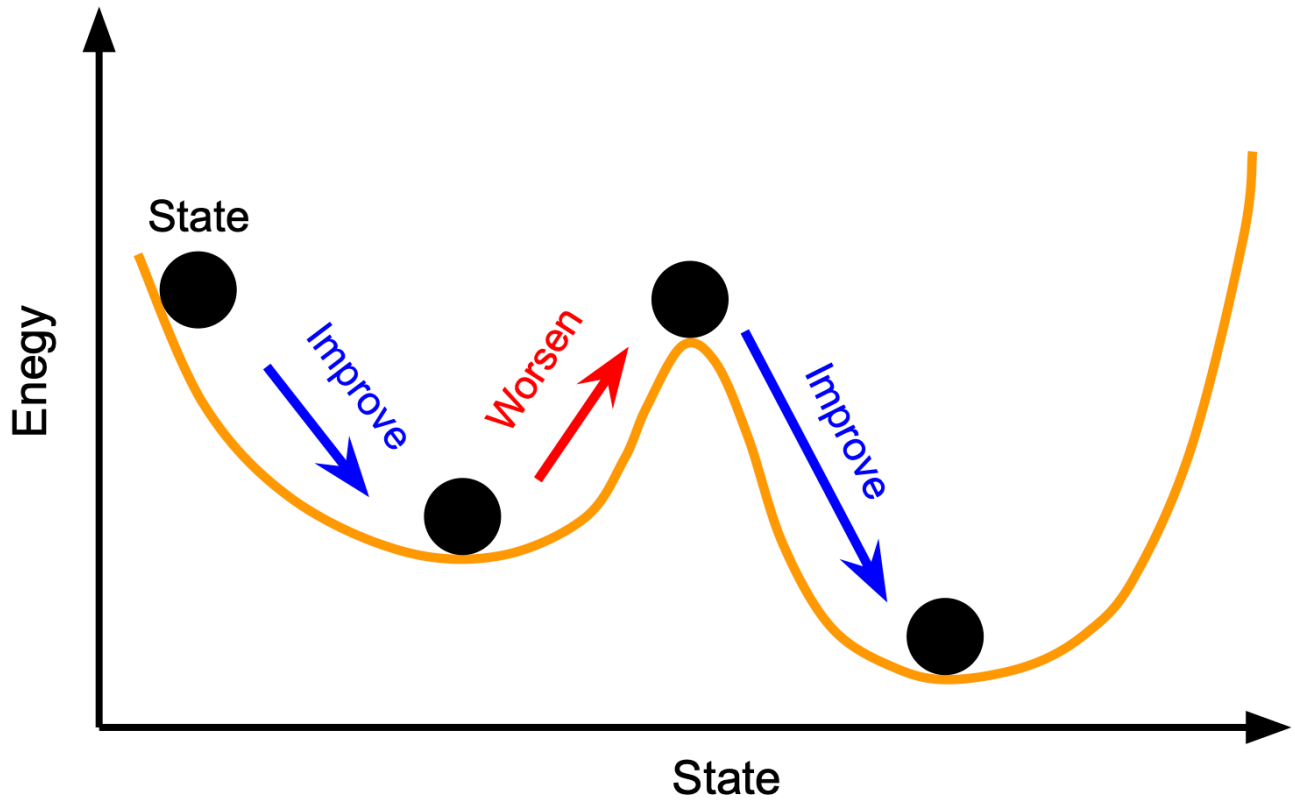


MCTS Steps

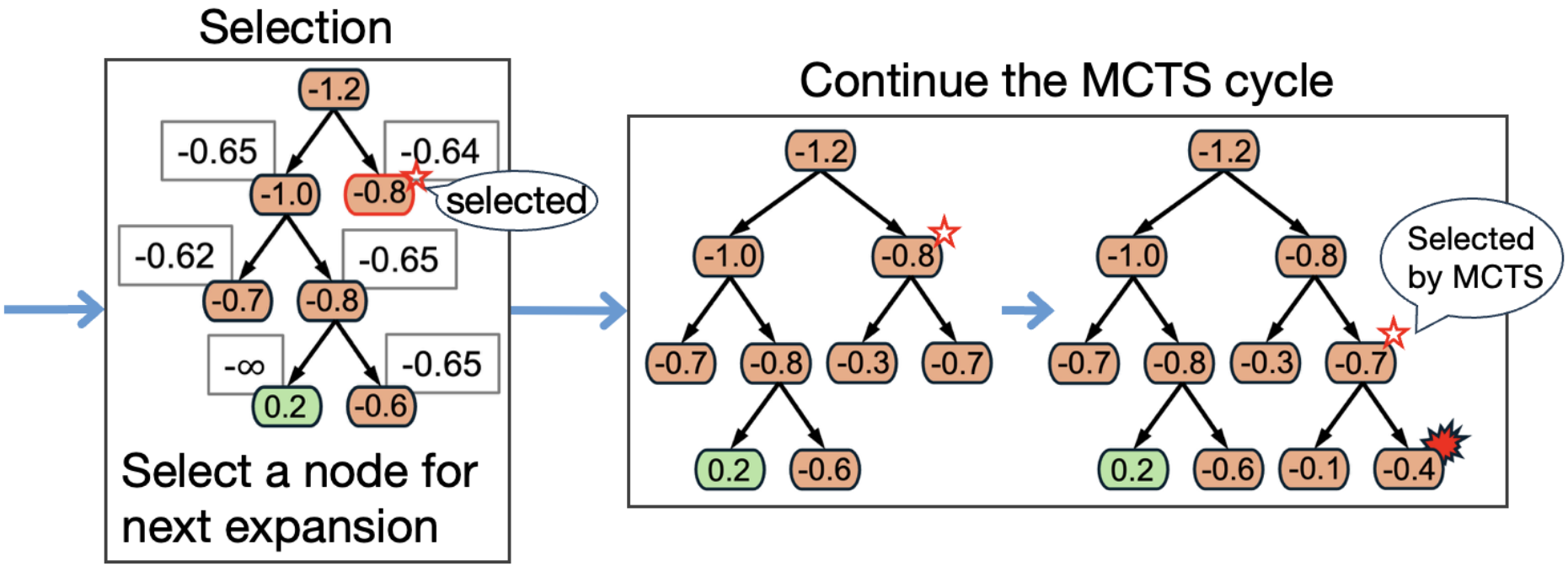
A MCTS cycle on BaB verification consists of 4 steps.



Stochastic: *Oliva*^{SA}



MCTS Steps Cycle [7]



MCTS guides a smarter node expansion order based on CeP while speedup the counterexample finding.

[7] DATE25: Adaptive Branch-and-Bound Tree Exploration for Neural Network Verification: Kota Fukuda, **Guanqin Zhang**, Zhenya Zhang, Yulei Sui, Jianjun Zhao

Evaluation

Table 2 RQ1 – Overall comparison of different verification approaches, in terms of the number of solved problem instances and the time costs (in secs).

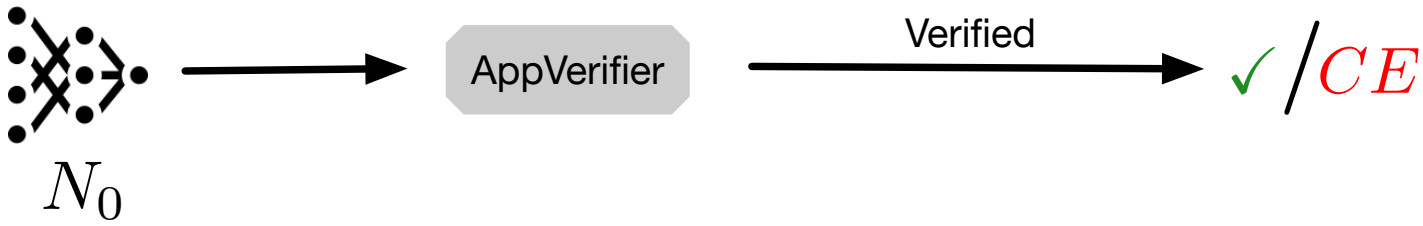
Model	BaB-baseline		$\alpha\beta$-Crown		Neuralsat		Oliva^{GR}		Oliva^{SA}	
	Solved	Time	Solved	Time	Solved	Time	Solved	Time	Solved	Time
$\texttt{MNIST}_{L2}$	96	126.41	87	51.32	99	32.37	95	96.79	99	57.76
$\texttt{MNIST}_{L4}$	65	194.74	44	428.03	54	392.04	67	146.31	53	142.72
$\texttt{OVAL21}_{BASE}$	42	770.7	58	641.7	70	621.21	154	184.96	159	155.29
$\texttt{OVAL21}_{DEEP}$	33	694.74	55	552.24	55	539.59	92	250.72	87	261.68
$\texttt{OVAL21}_{WIDE}$	40	733.73	63	557.51	65	533.01	131	240.16	112	288.0

Table 3 RQ1 – Pairwise comparison on the number of additional solved problem instances from all verification tasks. The number in each cell implies the number of problem instance solved by the approach of the row, but not solved by the approach of the column.

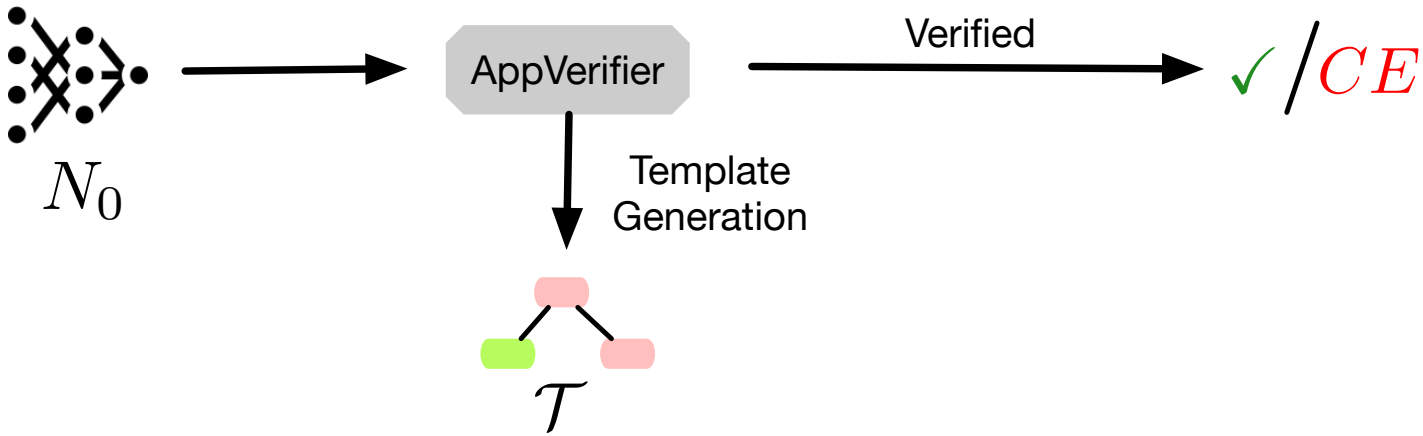
	BaB-Baseline	$\alpha\beta$-Crown	NeuralSAT	Oliva^{GR}	Oliva^{SA}
BaB-Baseline	0	80	59	8	13
$\alpha\beta$-Crown	111	0	11	23	39
NeuralSAT	126	47	0	30	46
Oliva^{GR}	271	255	226	0	40
Oliva^{SA}	247	242	213	11	0

Incremental Neural Network Verification I

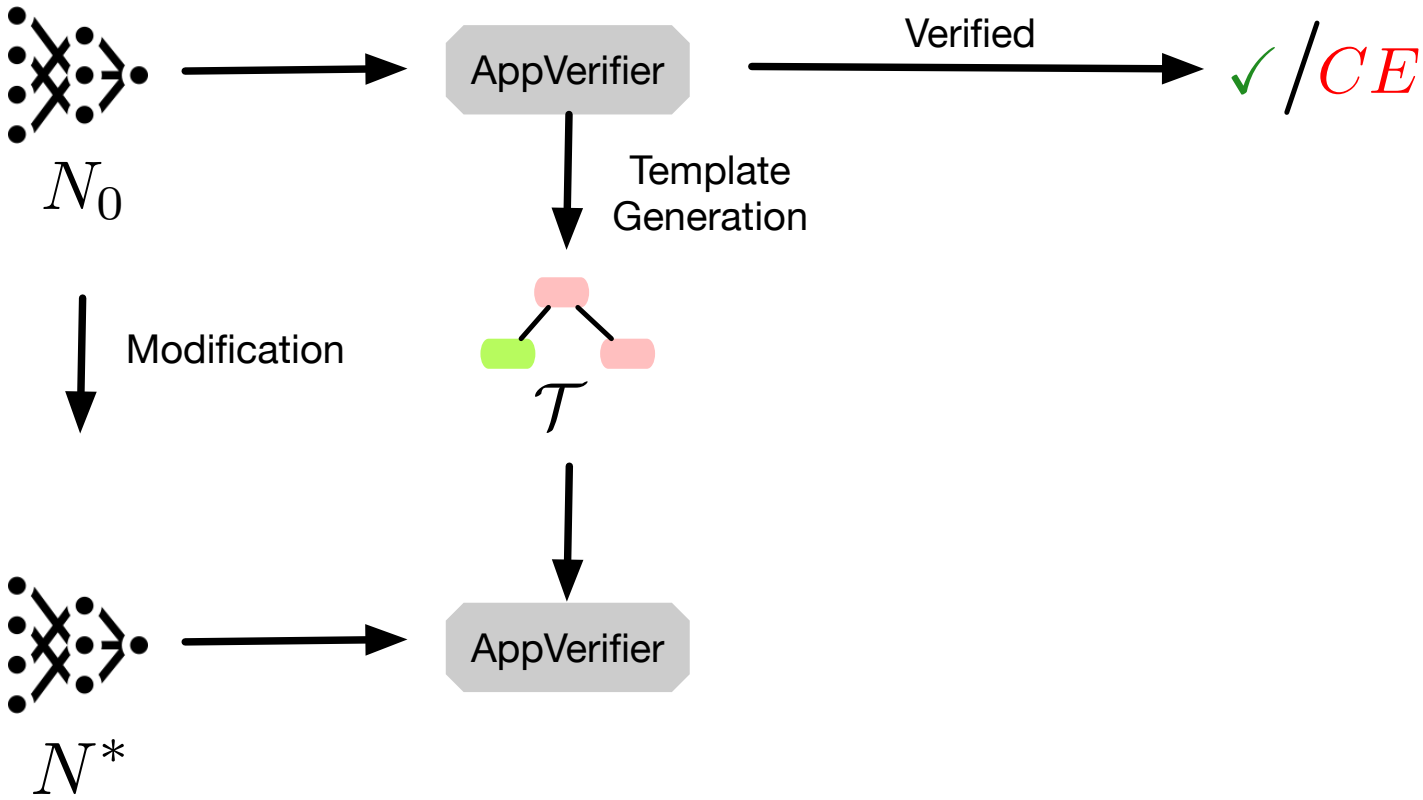
Existing Approach: IVAN



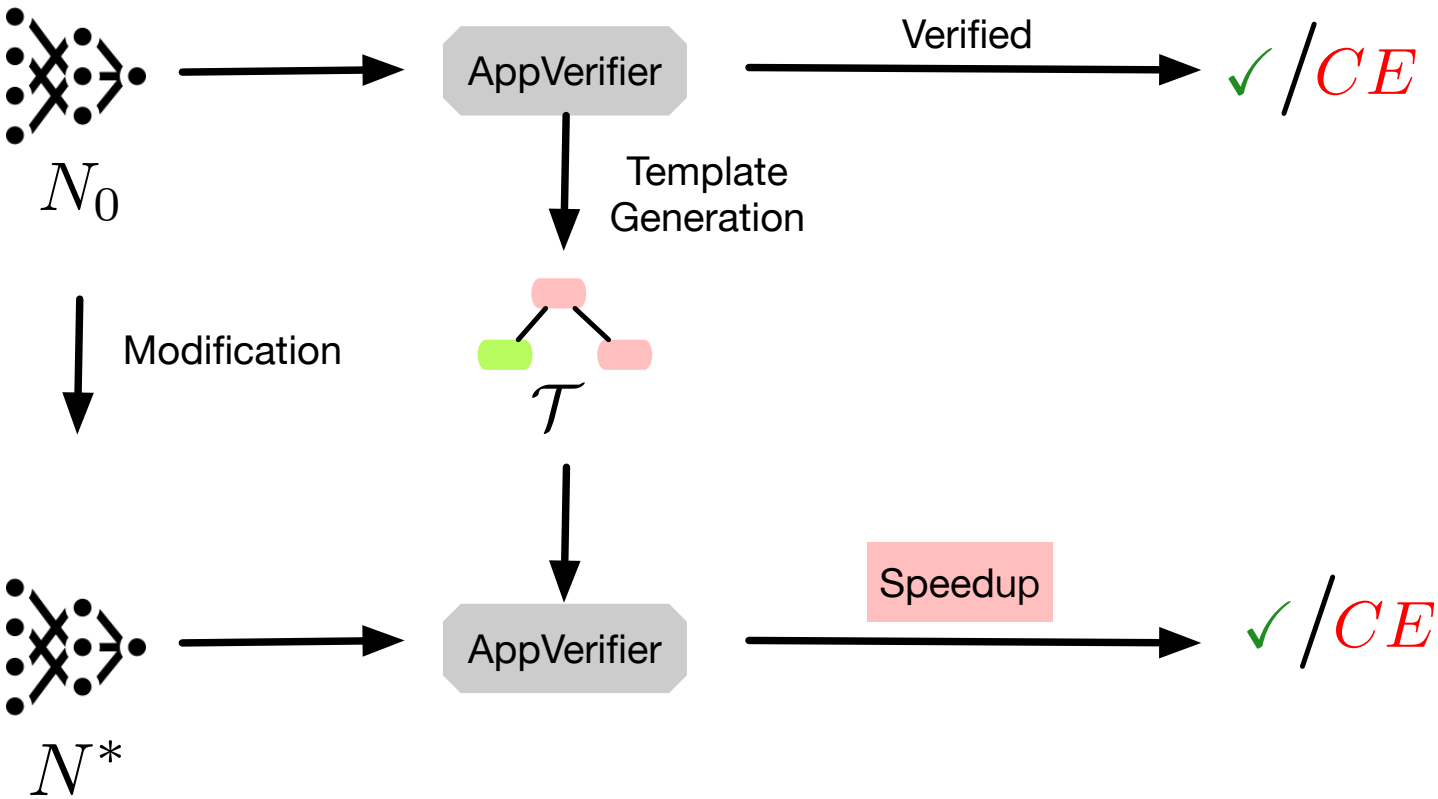
Incremental Neural Network Verification II



Incremental Neural Network Verification III

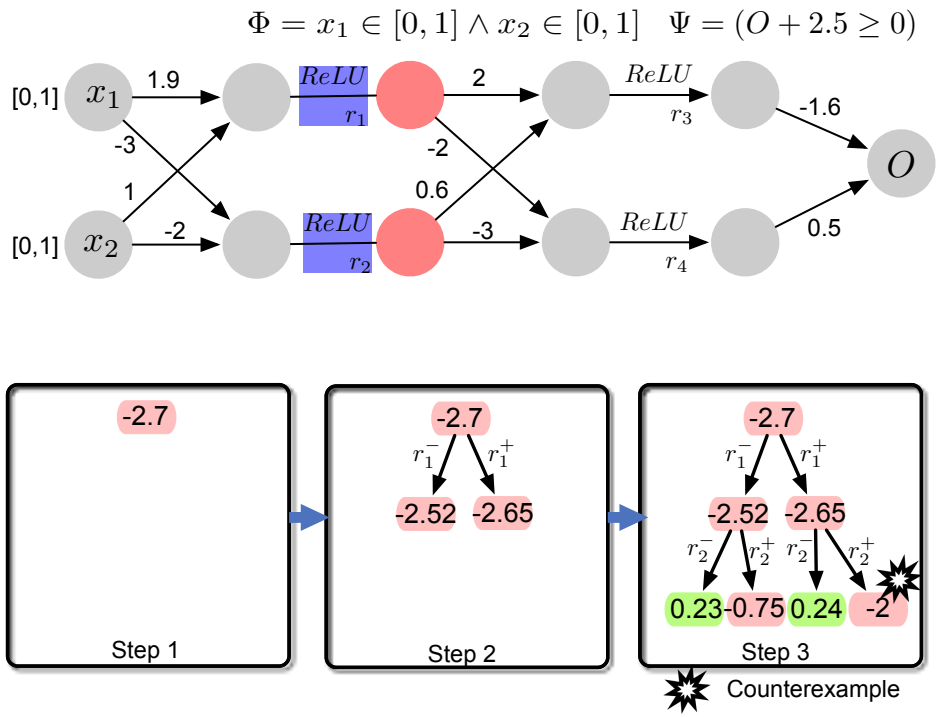


Incremental Neural Network Verification IV



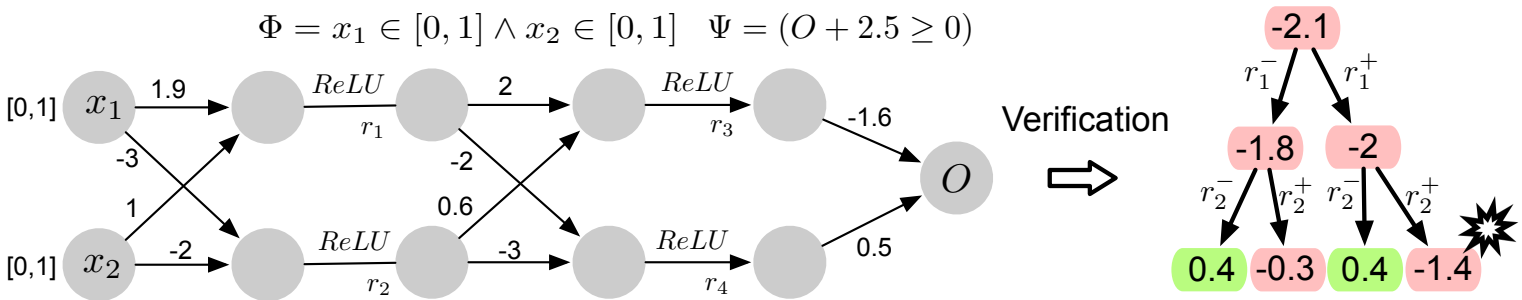
Noted: N^* has the identical structure to N but slightly differs in model parameters.

Revist DNN Verification

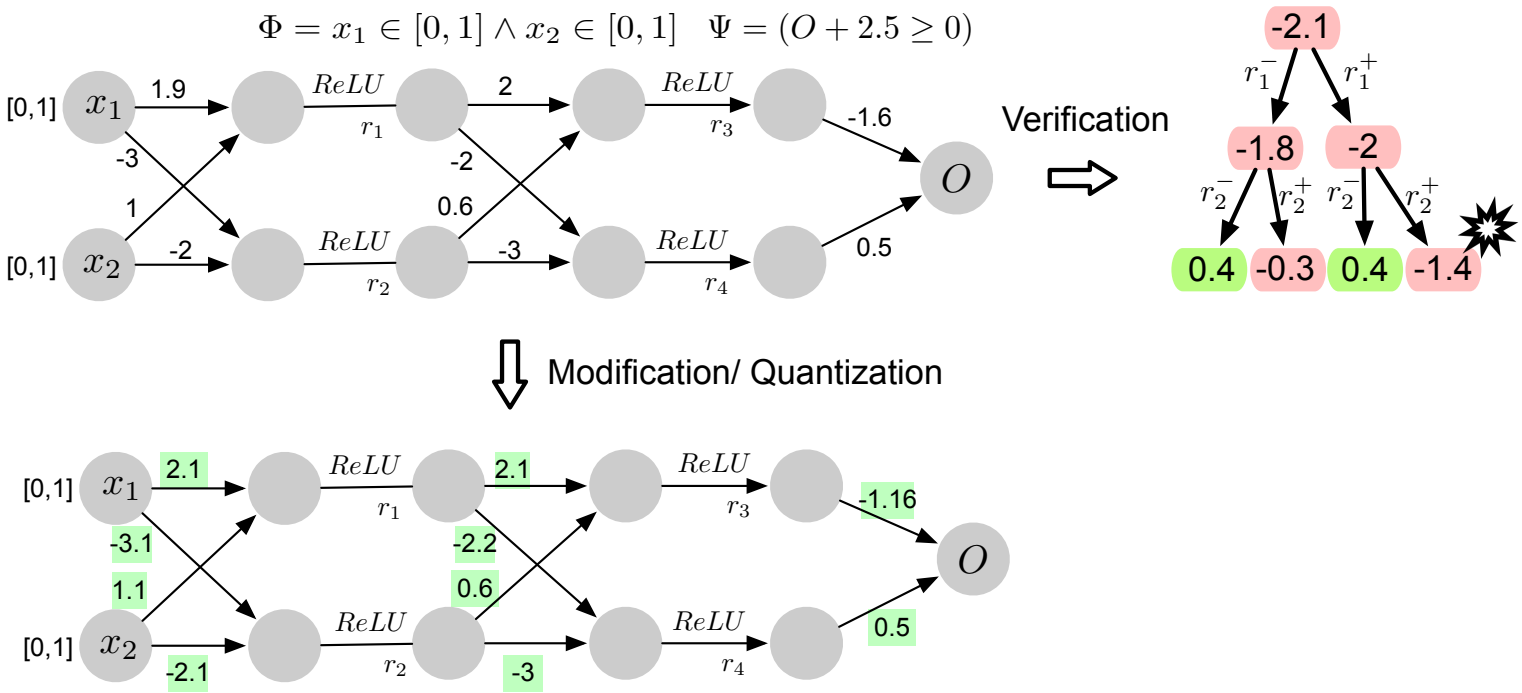


Specification Tree generated after Branch and Bound.

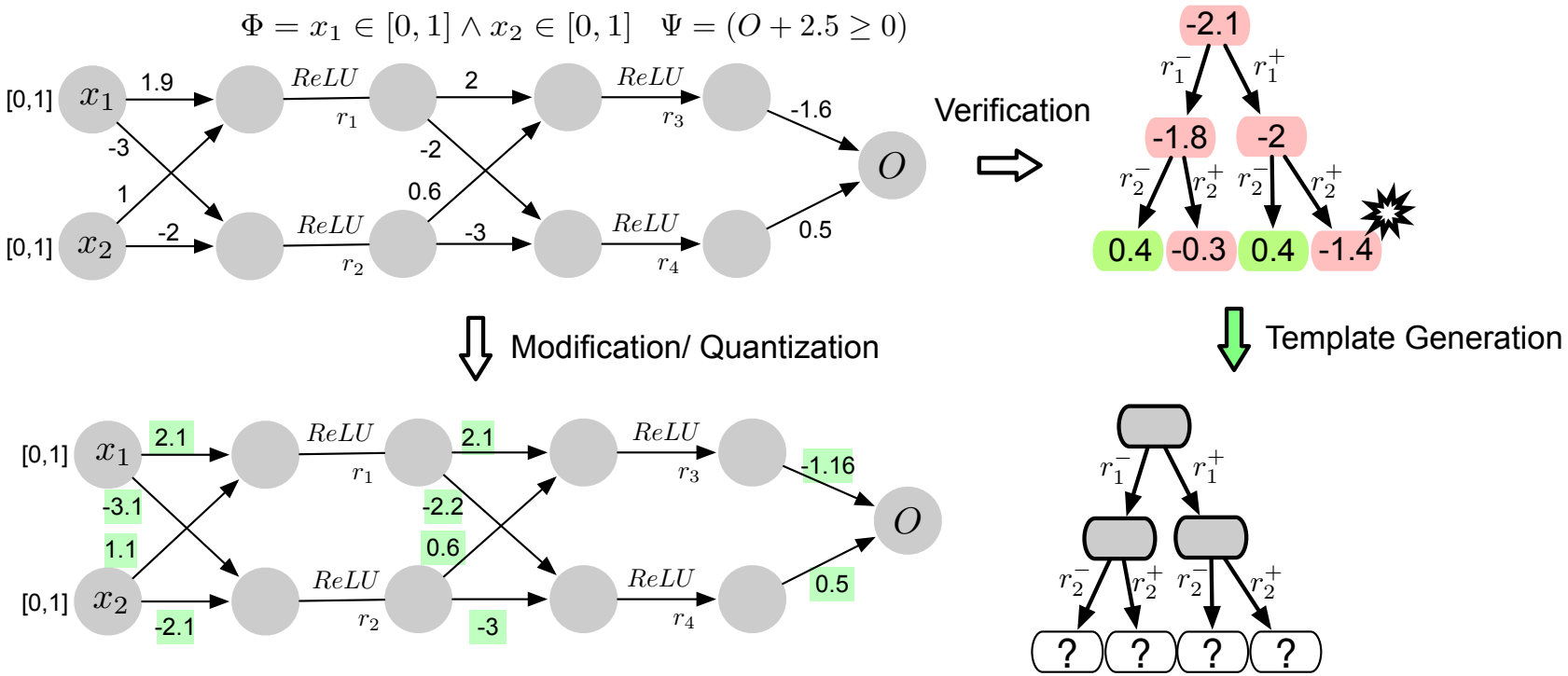
Incremental Neural Network Verification



Incremental Neural Network Verification

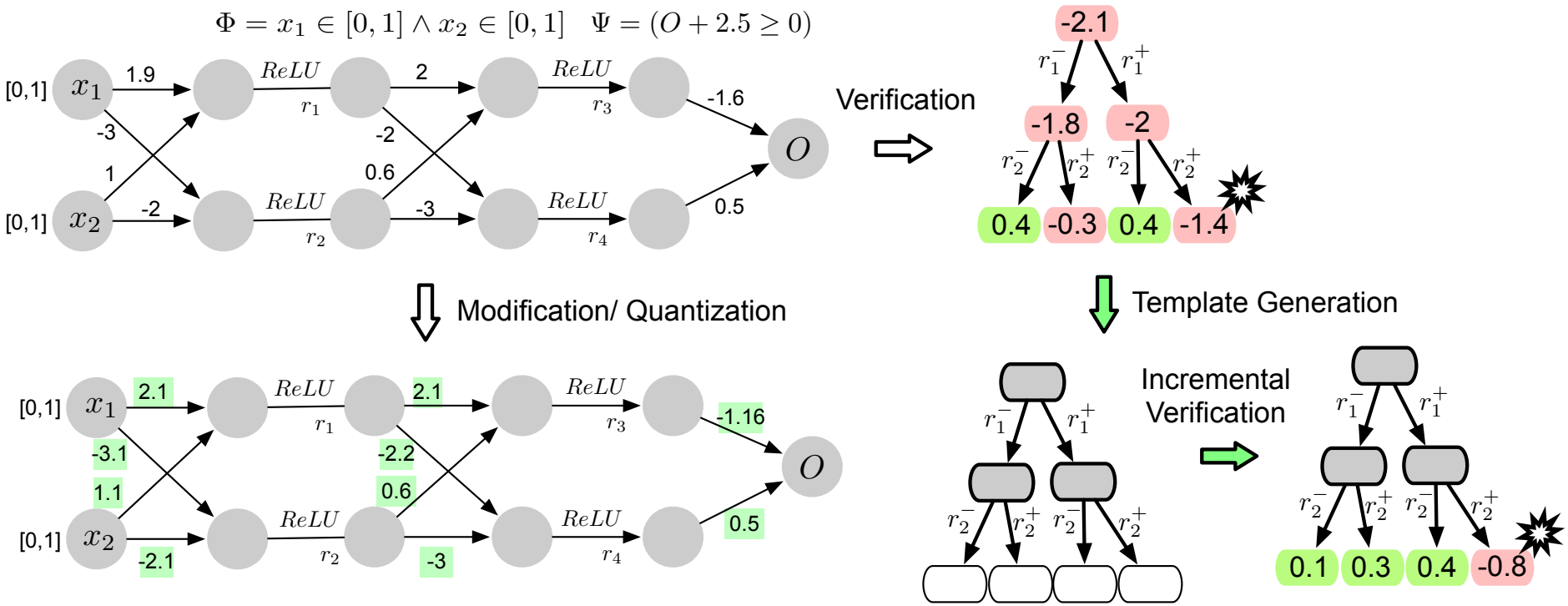


Incremental Neural Network Verification

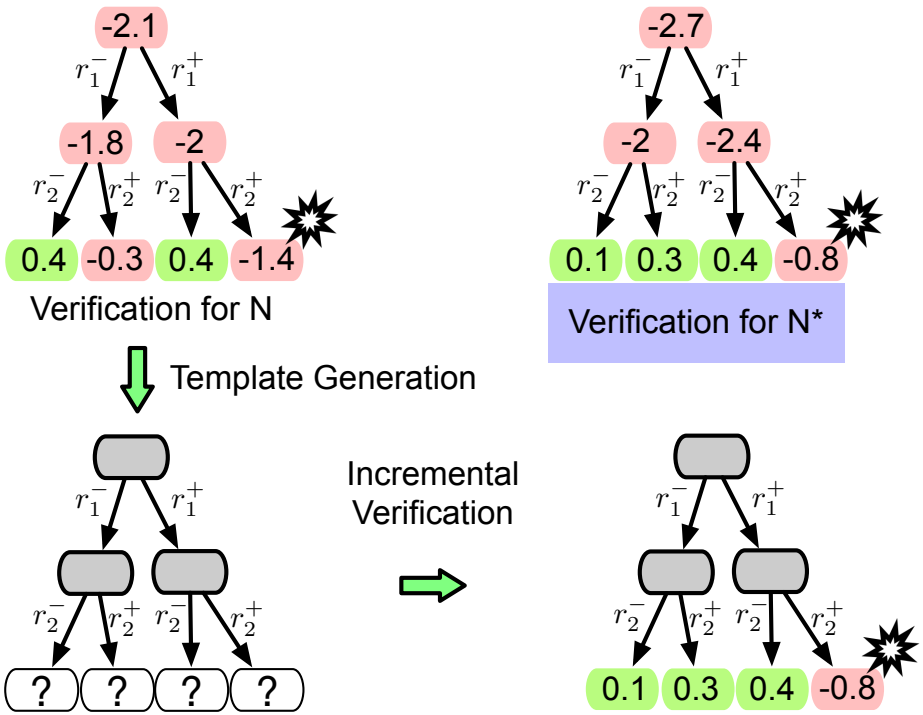


We only recompute the bounds in the **leaf nodes.**

Incremental Neural Network Verification



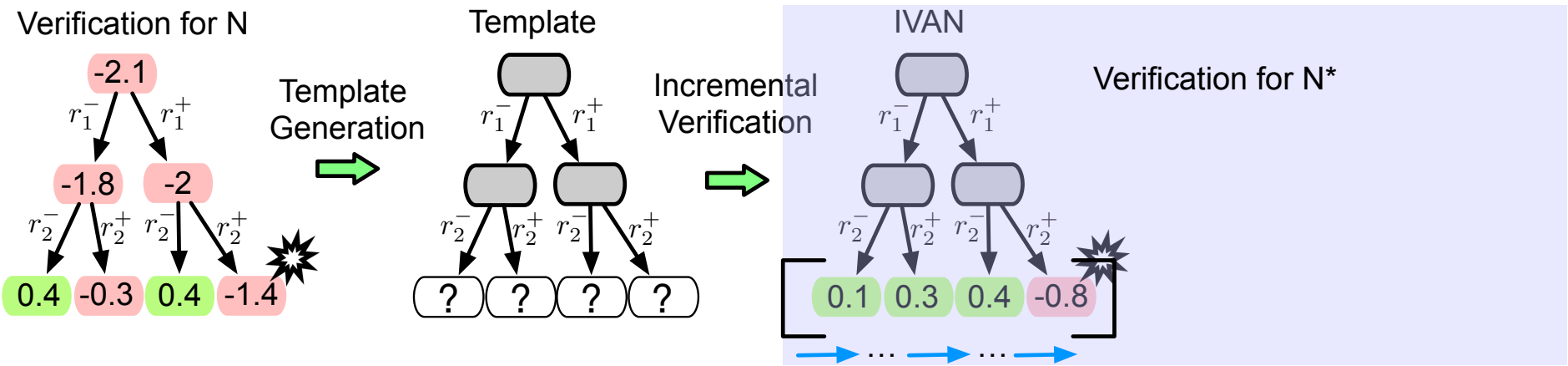
Incremental Nerual Network Verification (IVAN [13].)



- IVAN build on top of branch and bound based method, and compared with non-incremental verification, gaining the more scalabilites.
- IVAN saves extra **2** branching selection, **3** bounding computation.
- Assuming the non-incremental successfully verified $\mathbf{2^{n+1} - 1}$ nodes, the incremental verification only executes $\mathbf{2^n}$ bounding.

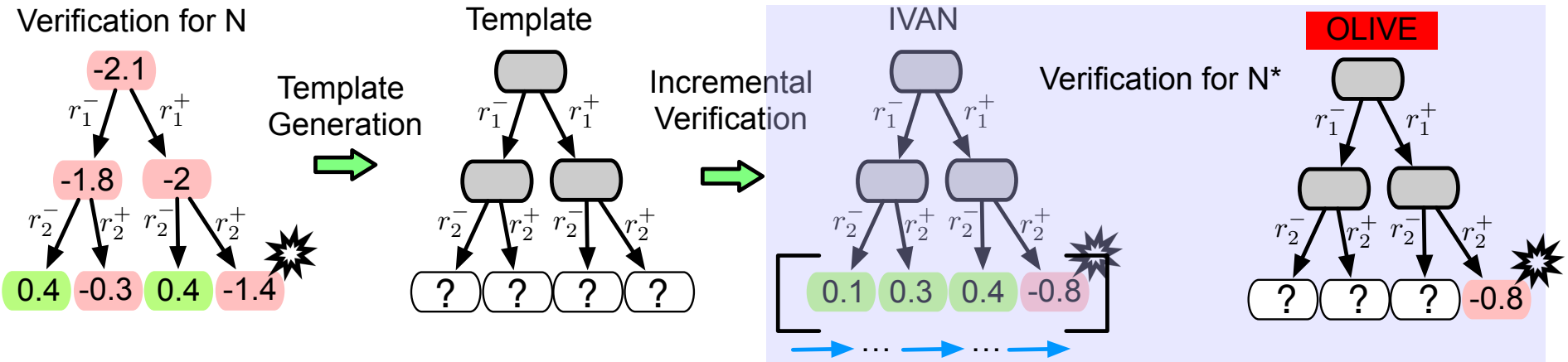
[13] Ugare, Shubham, et al. "Incremental verification of neural networks." Proceedings of the ACM on Programming Languages 7.PLDI (2023): 1920-1945.

Aim of This Work: Order-leading Incremental Verification



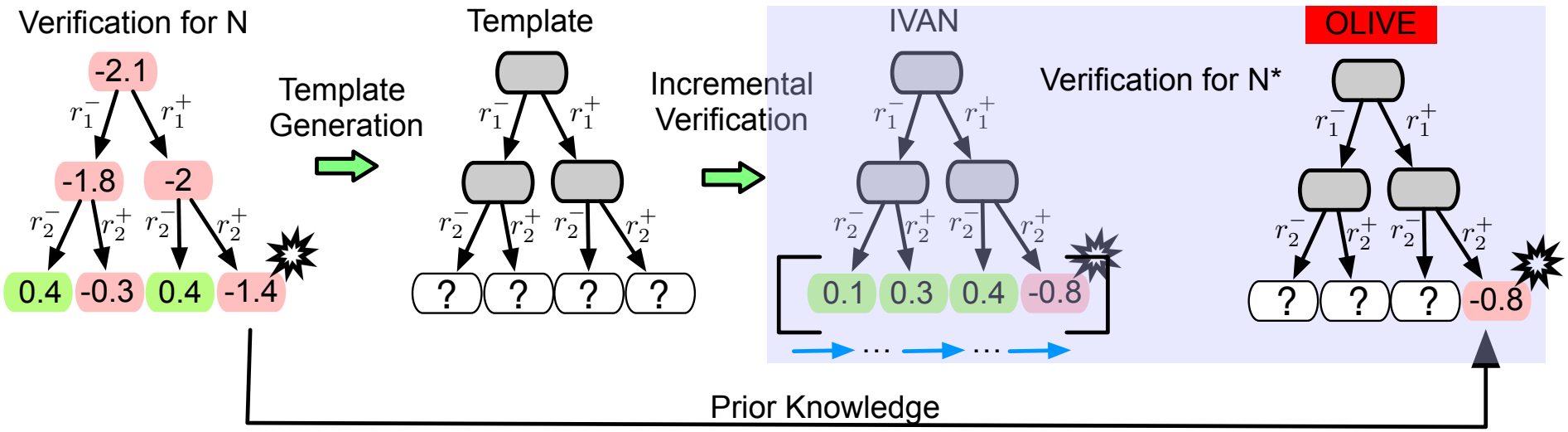
Existing work, such as IVAN, does not consider the **order** of the subproblem issues. It serves the template subproblem as in a **queue**, assuming as a “first come first serve”.

Order-leading Incremental Verification (1): *Olive*^{g}



We shortcuts the queue with less bounding times, in this case, we **speedup** 4×.

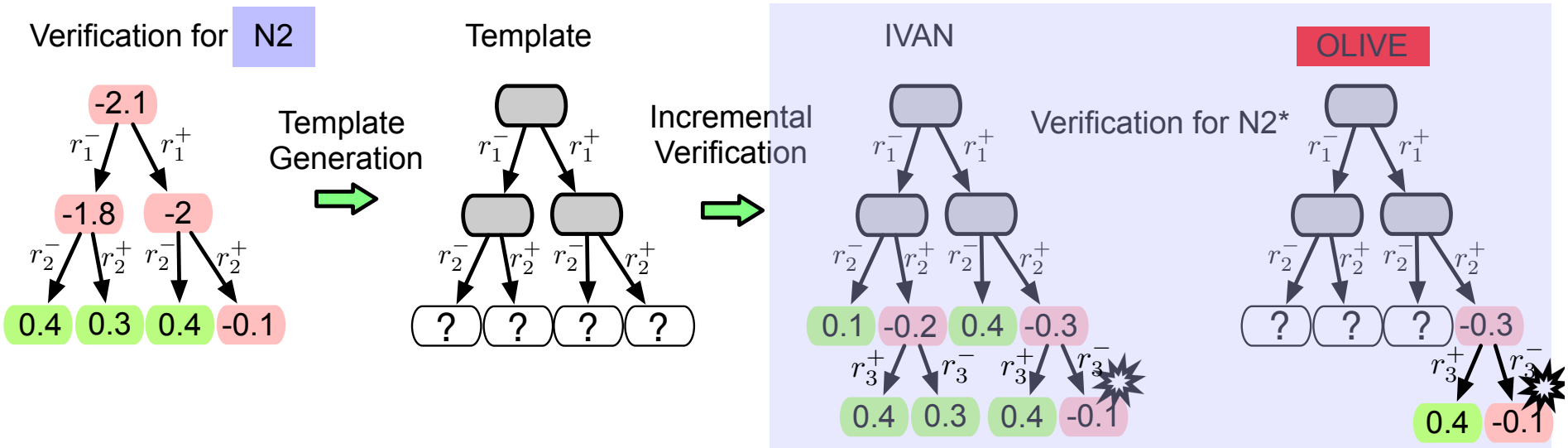
Order-leading Incremental Verification (1): *Olive*^{g}



We consider the **counterexample potentiality** [14] of each node in the specification tree.

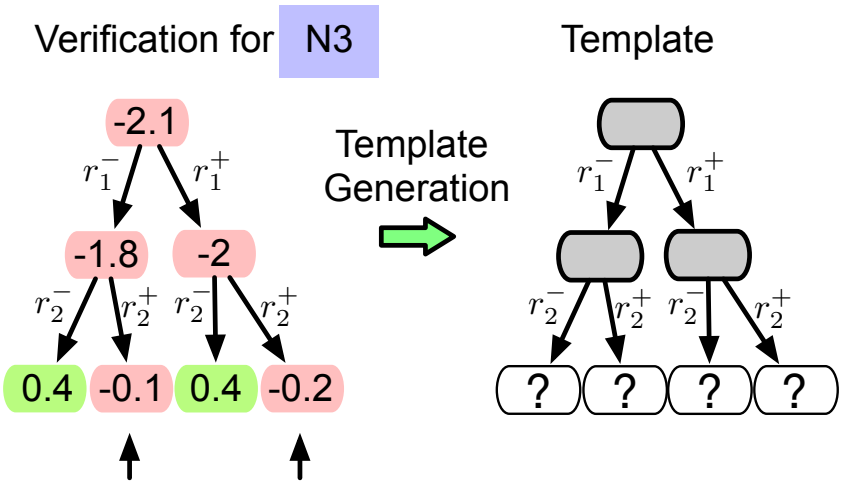
[14] if a node $\langle \Gamma_1, \hat{p}_1 \rangle$ is superior than another node $\langle \Gamma_2, \hat{p}_2 \rangle$ in only one of the attributions (e.g., $|\Gamma_1| > |\Gamma_2|$ but $\hat{p}_1 > \hat{p}_2$).

Order-leading Incremental Verification (2): *Olive*^{g}

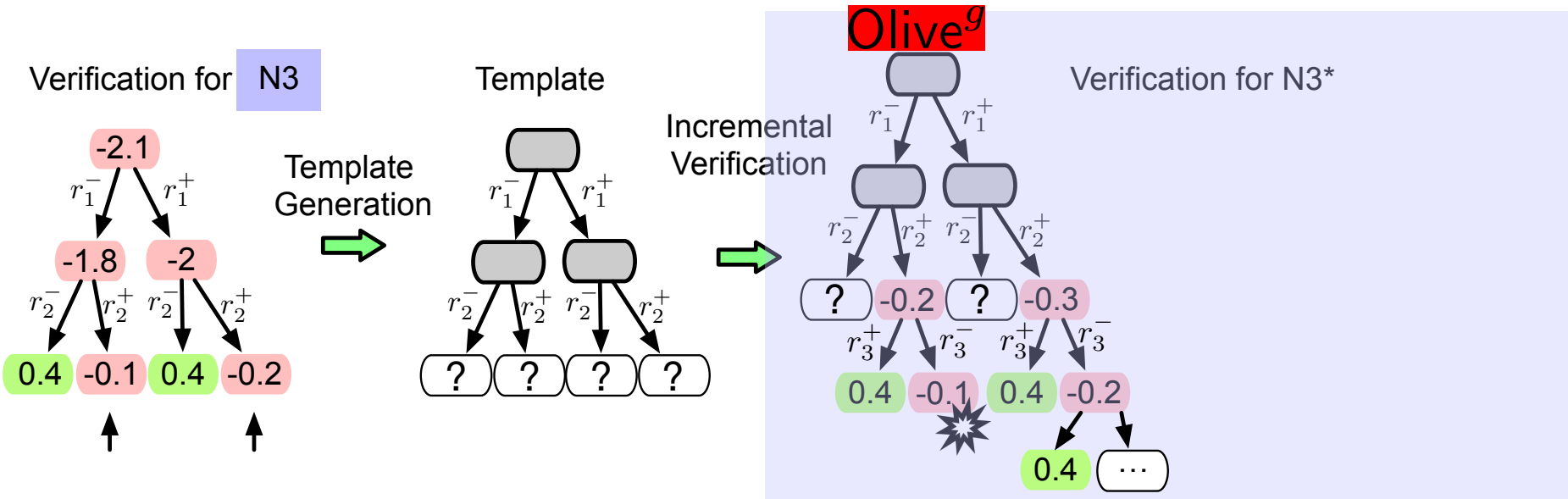


This order is essential, because we exploit the most promising counterexample potential order for speedup. In this case, IVAN require **2** branching, and **8** bounding. *Olive*^{g} only need **1** branching, and **3** bounding.

Olive^g Insights

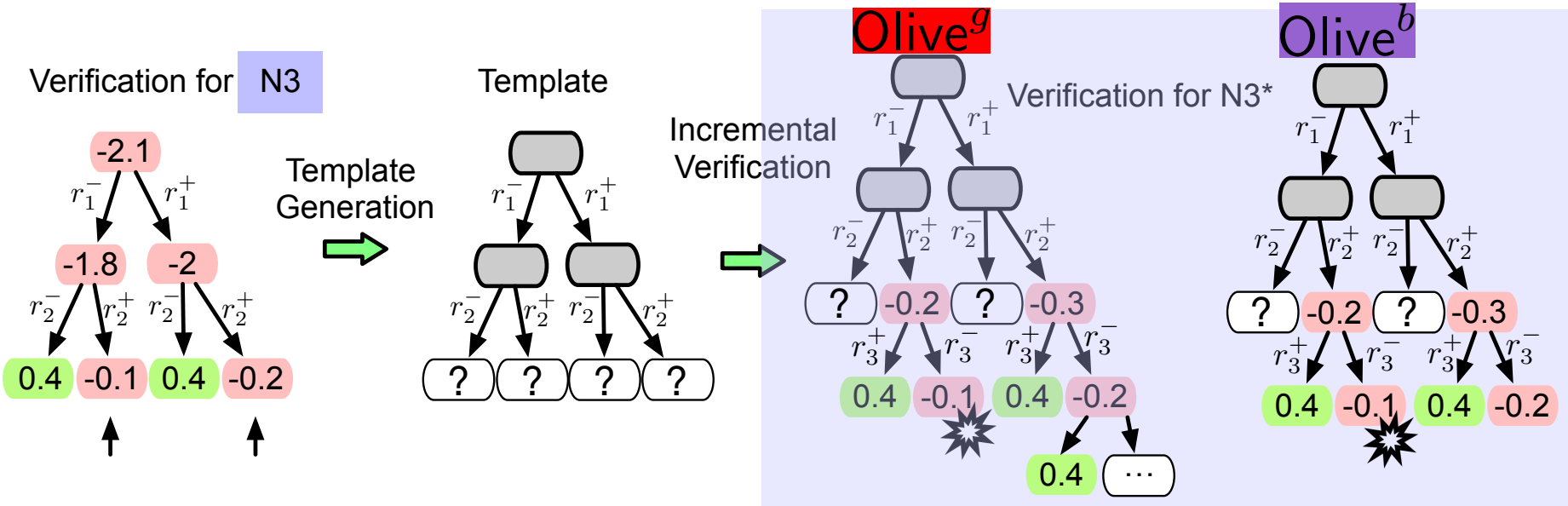


Olive^g Insights



While the greedy strategy *Olive*^{g} favors exploitation of the suspicious sub-problem suggested by the specification tree $\mathcal{T}_N$ of N, it can **fail** when the suggestion given by $\mathcal{T}_N$ is not precise for N^*.

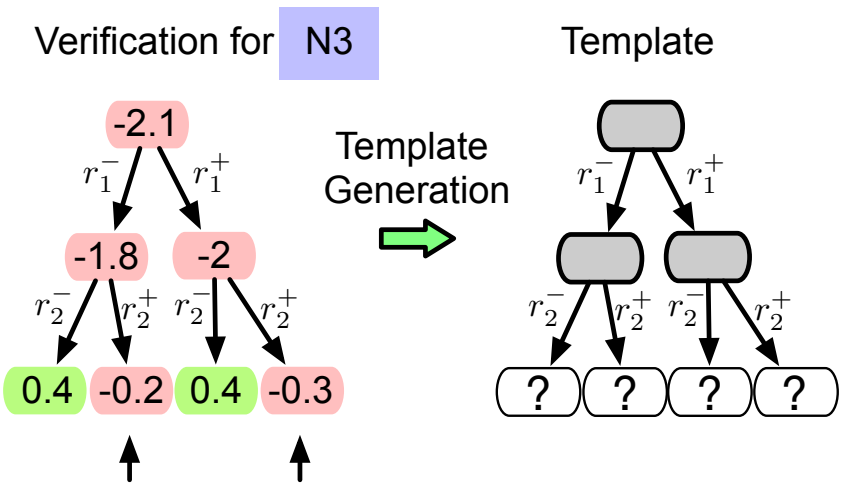
Order-leading Incremental Verification (2): *Olive*^{b}

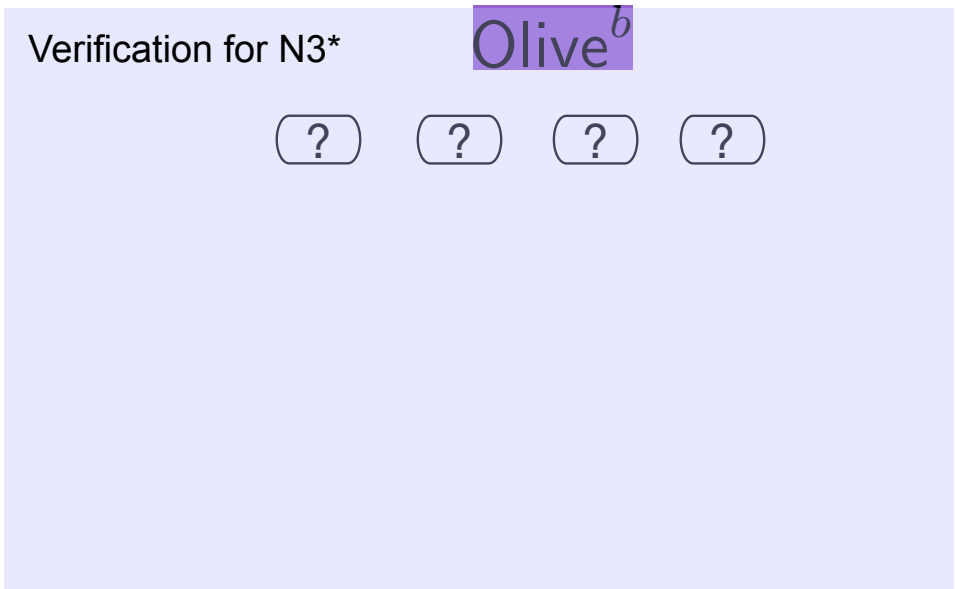


Namely, there could exist a different branch in *N*3* other than the one being exploited, in which counterexamples are easier to be found. This raises the classic problem of *"exploration and exploitation" trade-off* in search-based techniques. In light of this, we propose a balanced strategy that models the incremental verification problem as a *multi-armed bandit (MAB)* [15].) problem.

[15] Slivkins, Aleksandrs. "Introduction to multi-armed bandits." Foundations and Trends in Machine Learning 12.1-2 (2019): 1-286.

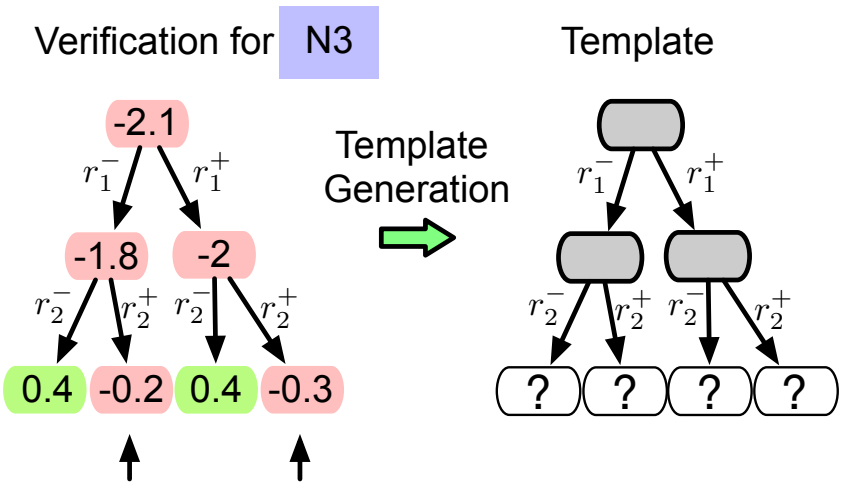
Olive^b with MAB

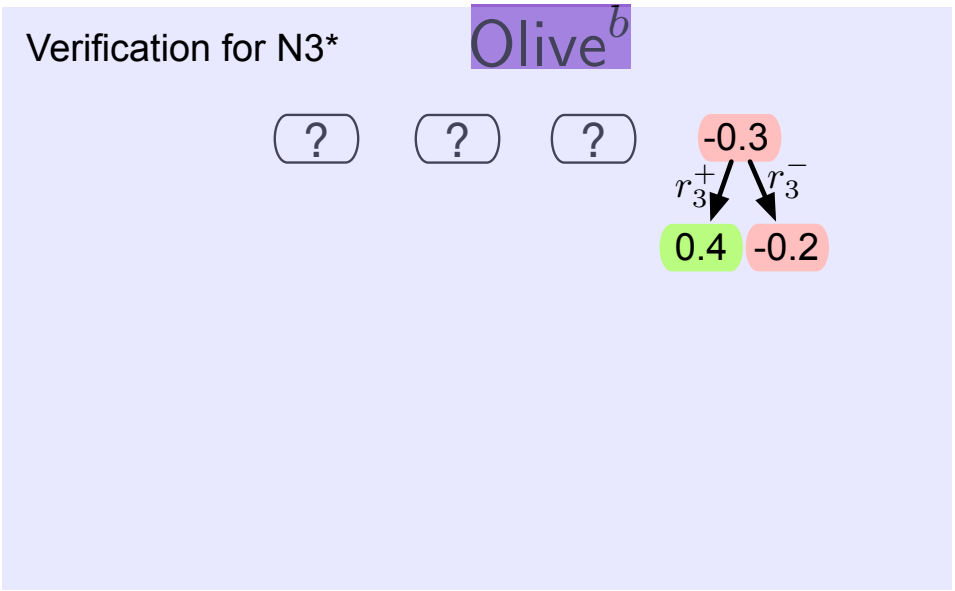




In *Olive*^{b}, we consider the classic problem of ***"exploration and exploitation"*** *trade-off* in search-based techniques.

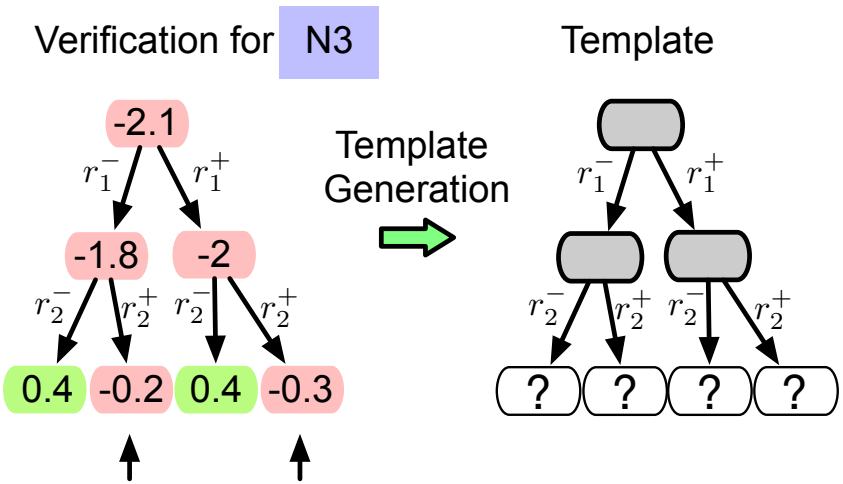
Olive^b with MAB - Compute Reward

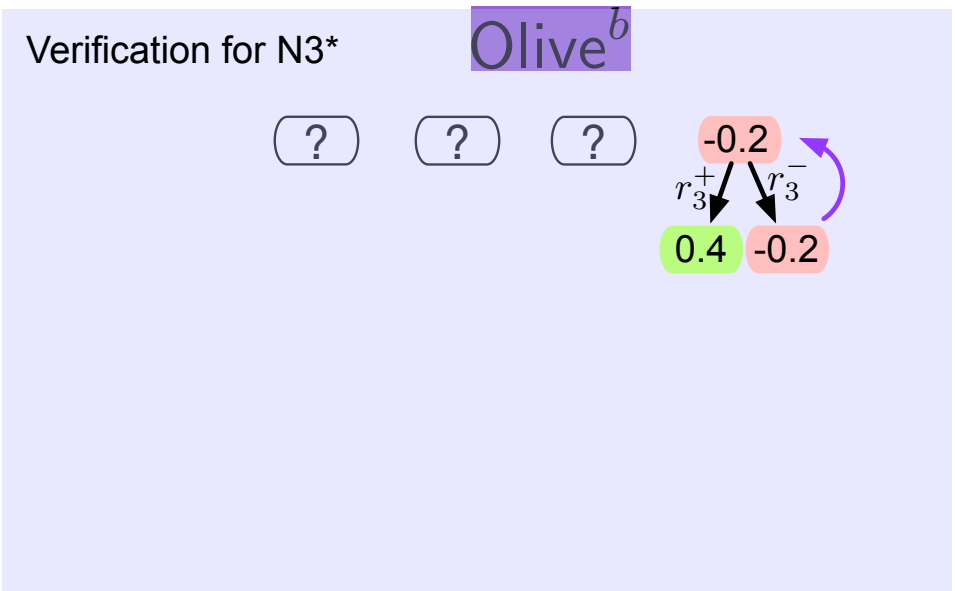




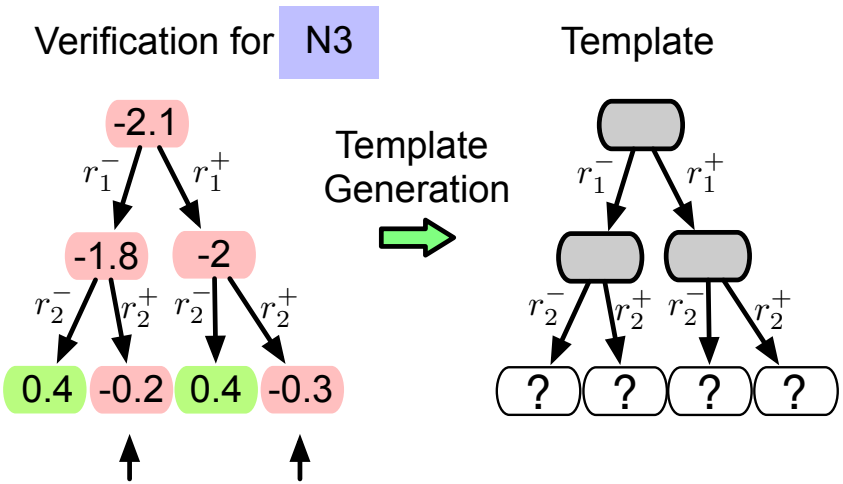
$$R(|\Gamma|, \hat{p}) := \begin{cases} +\infty & \text{if } \hat{p} < 0 \text{ and } \mathit{Valid}(\hat{x}, N^*, \Psi) \\ -\infty & \text{if } \hat{p} > 0 \\ \sigma \frac{|\Gamma|}{K} + (1 - \sigma) \frac{\hat{p}}{\hat{p}_{Min}} & \text{otherwise} \end{cases}$$

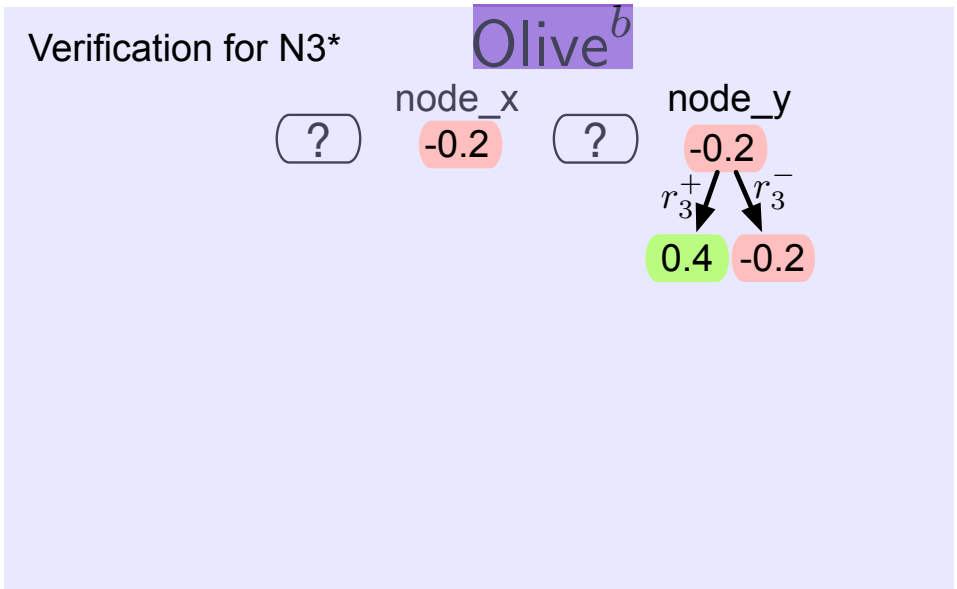
Olive^b with MAB - Back Propagation





Olive^b with MAB - Arms Expansion

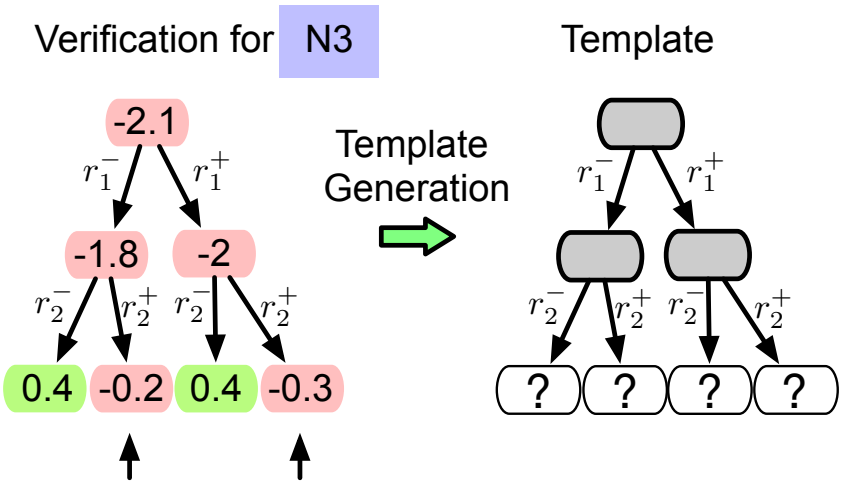


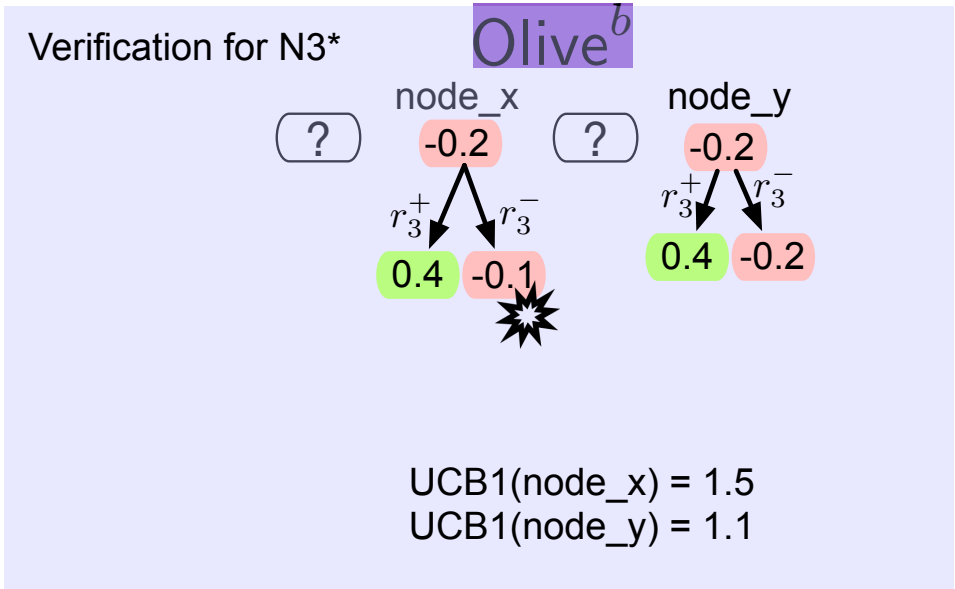


Conditional Expansion : *if* $|W| \leq C \cdot (\sum_{\Gamma \in W} |T(\Gamma)|)^{\alpha}$ and $|W| \neq |\mathcal{L}|$ [16]

[16] Here, $W = 4$

Olive^b with MAB - UCB1





Select **next**: $\Gamma^* \leftarrow argmax_{\Gamma \in w} \left(R(\Gamma) + c\sqrt{\frac{2 \ln \left(\sum_{\Gamma \in w} |T(\Gamma)| \right)}{|T(\Gamma)|}} \right)$

Evaluation

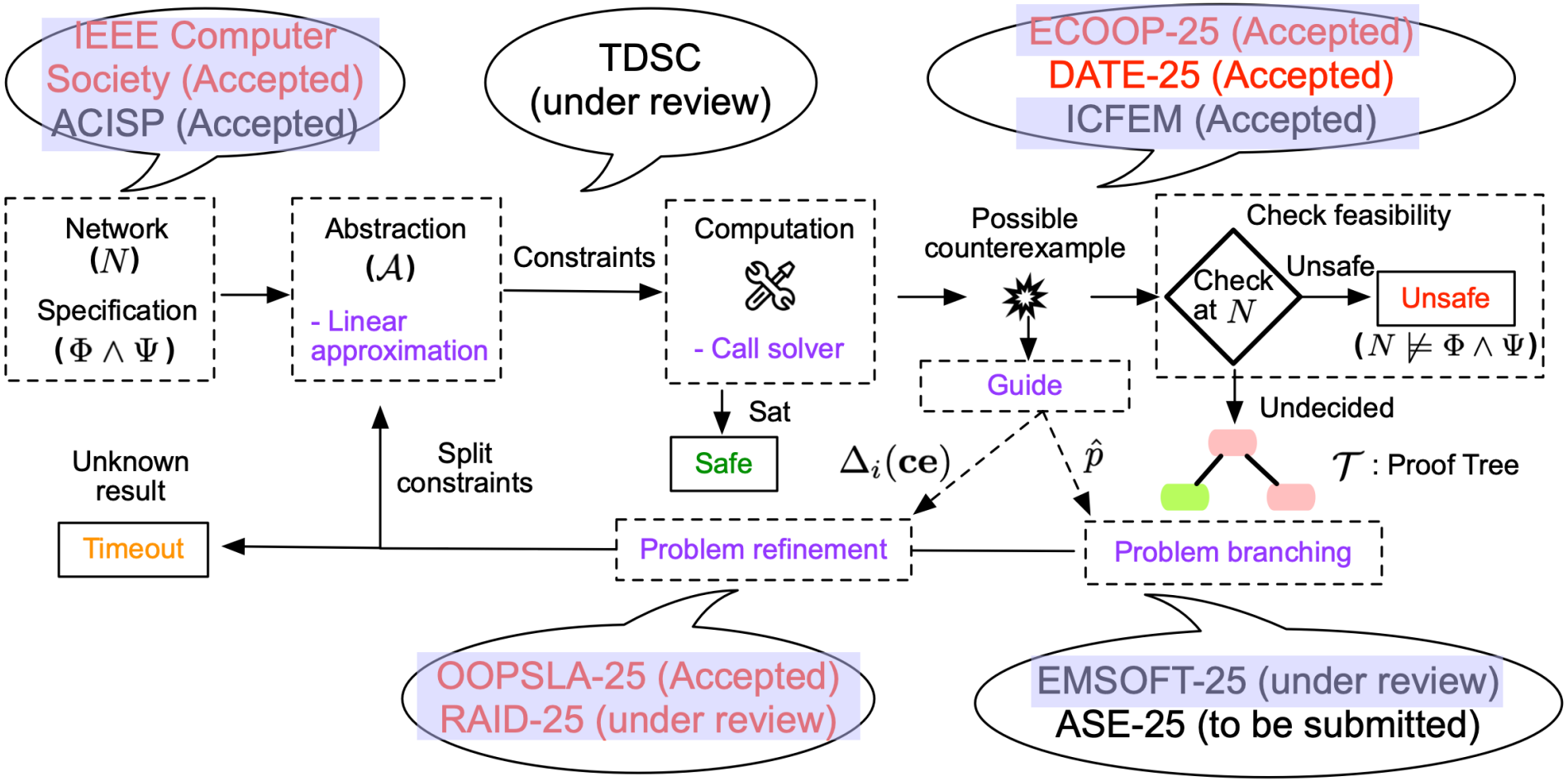
Model	Alteration	$\alpha\beta$-Crown		Marabou		Ivan		Olive^{g}		Olive^{b}	
		Speedup	+Solved	Speedup	+Solved	Speedup	+Solved	Speedup	+Solved	Speedup	+Solved
MNIST_{L2}	Quant(Int8)	1.08×	0	0.93×	0	3.59×	8	3.28×	5	2.51×	3
MNIST_{L2}	Prune(7%)	1.05×	0	1.02×	0	1.60×	13	1.75×	10	1.73×	11
MNIST_{L2}	Prune(12%)	2.28×	0	1.22×	0	2.31×	0	34.89×	1	35.98×	4
MNIST_{L4}	Quant(Int8)	2.18×	0	1.28×	0	1.78×	4	2.15×	5	2.14×	6
MNIST_{L4}	Prune(7%)	1.01×	0	0.99×	0	1.59×	4	2.16×	2	2.18×	3
MNIST_{L4}	Prune(12%)	1.06×	0	1.1×	0	2.92×	2	14.74×	3	14.99×	23
$\text{OVAL21}_{\text{BASE}}$	Prune(0.1%)	1.01×	0	0.93×	0	1.74×	16	2.56×	11	2.58×	7
$\text{OVAL21}_{\text{BASE}}$	Prune(1%)	1.04×	0	0.99×	0	1.79×	17	2.37×	12	2.33×	7
$\text{OVAL21}_{\text{BASE}}$	Prune(3%)	1.00×	0	0.96×	0	1.77×	14	2.31×	10	2.29×	6
$\text{OVAL21}_{\text{DEEP}}$	Prune(0.1%)	1.02×	0	1.03×	0	1.91×	10	3.19×	8	3.21×	5
$\text{OVAL21}_{\text{DEEP}}$	Prune(1%)	1.56×	3	1.14×	0	1.92×	7	3.14×	5	3.18×	3
$\text{OVAL21}_{\text{DEEP}}$	Prune(3%)	1.03×	0	1.01×	0	1.88×	4	2.79×	2	2.78×	2
$\text{OVAL21}_{\text{WIDE}}$	Prune(0.1%)	1.0×	0	0.88×	0	1.98×	1	5.82×	1	5.98×	1
$\text{OVAL21}_{\text{WIDE}}$	Prune(1%)	0.98×	0	0.83×	0	1.74×	1	7.51×	1	7.34×	2
$\text{OVAL21}_{\text{WIDE}}$	Prune(3%)	1.02×	0	0.93×	0	1.88×	4	2.79×	2	2.78×	2

The performance comparison between three incremental verification approaches and the BaB baseline approach, in terms of the average speedup w.r.t. BaB over the solved verification problems and the number of the additional problems that are not solved by BaB but solved by other approaches.

MileStone Summary

MileStones	Outcome
Complete major revision of the paper submitted to TOSEM	Resubmitted to TDSC
Complete major revision of the paper submitted to OOPSLA25	ACHIEVED
Complete current work (OLIVA) and submit it to CAV 2025	ACHIEVED in ECOOP
Participate in collaborators' work and submit it to VNN-COMP 2025	As planned and undergo
Draft thesis	As planned and undergo
Other Contributed works	Detailed in next slide

Contributed works (All in One)



Thesis Structure

1. Literature Review and Outline:
   - Duration: March 2025 (1 month)
   - Focus: Comprehensive literature review and creating a detailed outline for the thesis
2. Chapter 1 Refinement and Revision:
   - Duration: April 2025 - May 2025 (2 months)
   - IEEE work as in an introduction, refining and revising
3. Chapter 2 Major Work - 1:
   - Duration: April 2025 - May 2025 (2 months)
   - ECOOP, DATE
4. Chapter 3 Major Work - 2:
   - Duration: April 2025 - May 2025 (2 months)
   - OOPSLA
5. Chapter 4 Refinement and Revision:
   - Duration: June 2025 (1 month)
   - Focus: Refining and revising the all chapters of thesis
6. Full Thesis Initial Revision:
   - Duration: June 2025 (1 month)
   - Focus: First comprehensive revision of the entire thesis

Questions?