Neural Network Verification with Proof Production

Omri Isac[*], Clark Barrett[†], Min Zhang[‡], Guy Katz[*]

[*]The Hebrew University of Jerusalem, [†]Stanford University, [‡]East China Normal University

FMCAD 2022

Agenda

- Introduction
- Neural Network Verification
- Proof Production
- Implementation and Evaluation
- Future Work

Deep Neural Networks (DNNs)

- Accomplish groundbreaking results in many fields, including safety-critical.
- Vulnerable to input perturbations.

Motivation

DNN verifiers were already presented, but they contain errors and numerical stability problems [Jia & Rinard, 2021].

Objective

- Produce proofs of the verifiers' results:
  - Checked using an independent, trusted and simple proof-checker.
  - Witness the correctness of the verifier or reveal errors.

Agenda

- Introduction
- Neural Network Verification
- Proof Production
- Implementation and Evaluation
- Future Work

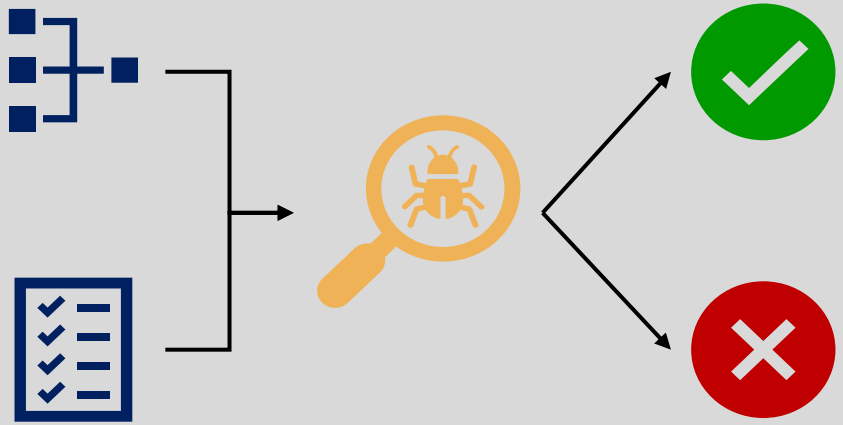
DNN Verification

- Given a DNN $\mathcal{N}$ and a property $P(x, y)$, decide whether there exists an input x such that $P(x, \mathcal{N}(x))$.
  - If exists, the problem is satisfiable (SAT);
  - Otherwise unsatisfiable (UNSAT).

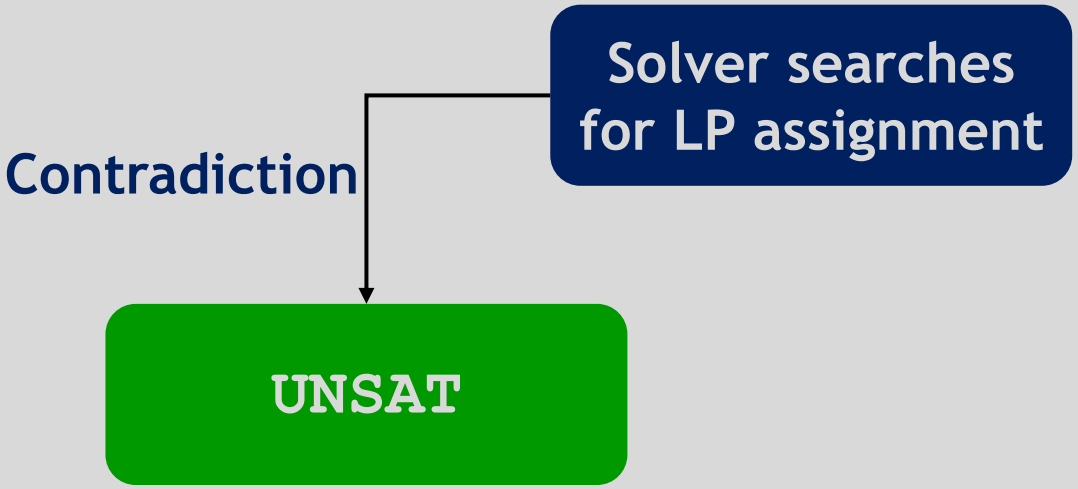
- We consider DNNs with piecewise-linear activations.

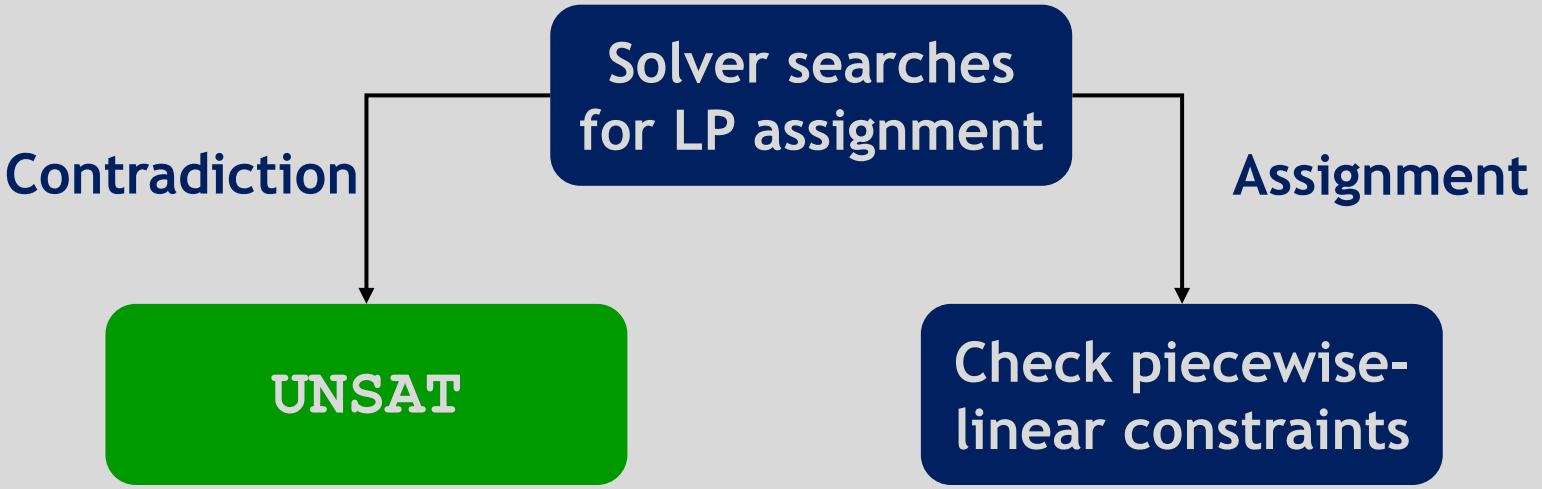
DNN Verification

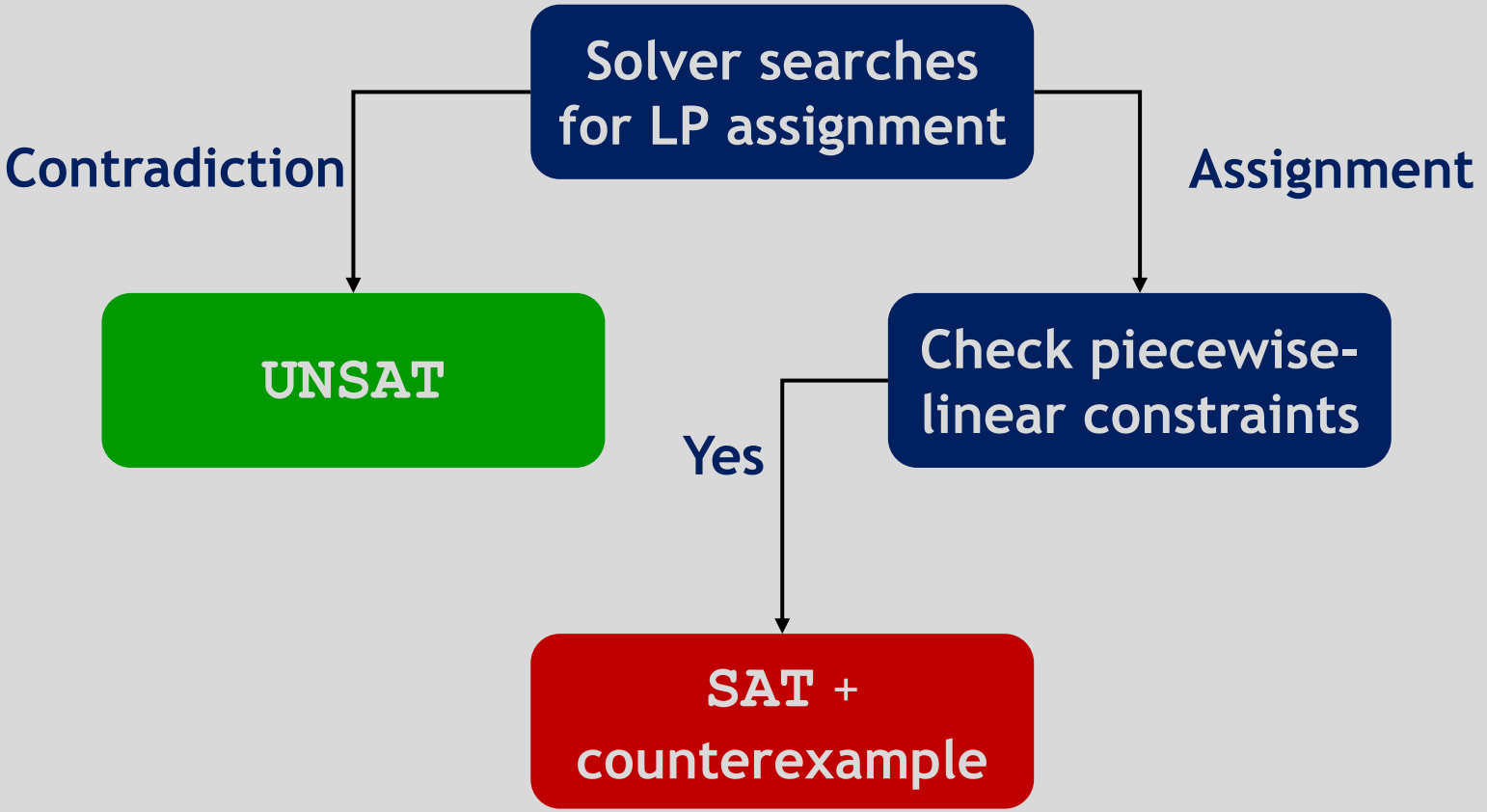
- DNN structure incudes linear + piecewise-linear constraints.
- Many verification algorithms search for a satisfying assignment with LP solvers + splitting approach.

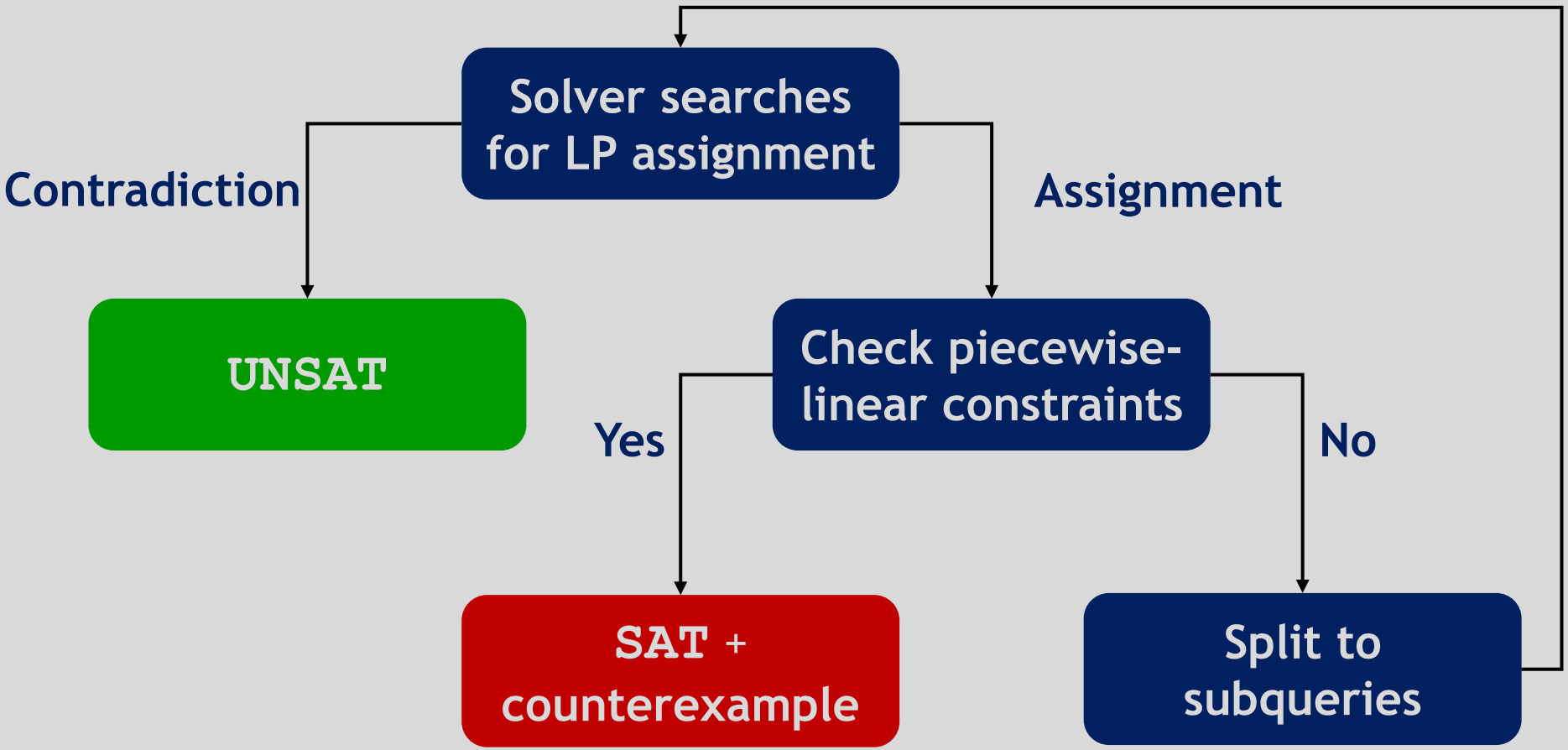


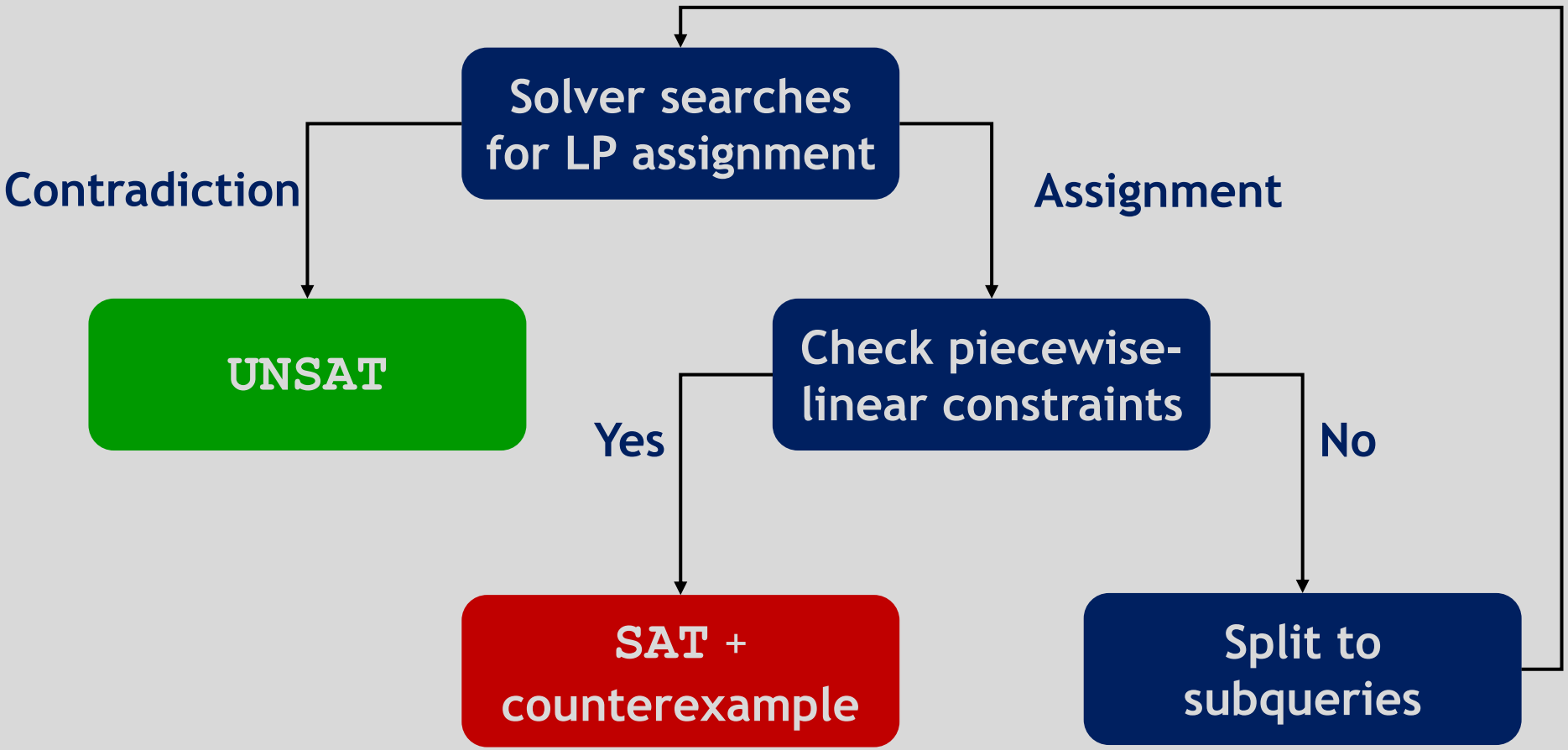
Solver searches for LP assignment


Solver searches
for LP assignment
Contradiction
UNSAT


Solver searches for LP assignment
Contradiction
Assignment
UNSAT
Check piecewise-linear constraints


Solver searches for LP assignment
Contradiction
Assignment
UNSAT
Check piecewise-linear constraints
Yes
SAT + counterexample

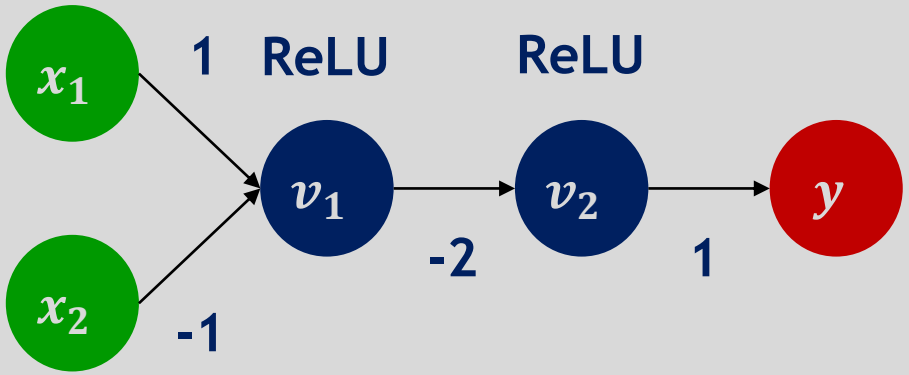

Solver searches for LP assignment
Contradiction
Assignment
UNSAT
Check piecewise-linear constraints
Yes
No
SAT + counterexample
Split to subqueries



Splitting creates a tree structure to the search.

The query is UNSAT if and only if all leaves are UNSAT.

A Query Example



$$P(x, y) = \{x \in [2, 3] \times [-1, 1] \wedge y \in [0.25, 0.5]\}$$

Query φ

Linear

$$b_1 = x_1 - x_2$$
$$b_2 = -2f_1$$
$$y = f_2$$

$$\begin{bmatrix} 2 \\ -1 \\ 0 \\ 0.25 \\ -0.5 \end{bmatrix} \leq \begin{bmatrix} x_1 \\ x_2 \\ f_1 \\ f_2, y \\ b_1, b_2 \end{bmatrix} \leq \begin{bmatrix} 3 \\ 1 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

Piecewise-Linear

$$f_1 = ReLU(b_1)$$
$$f_2 = ReLU(b_2)$$

Linear Programming (LP)

Optimizing a linear function $c \cdot x$ with linear constraints

$$Ax = 0 \wedge l \leq x \leq u$$

where $x, c \in \mathbb{R}^n$, $l, u \in \mathbb{R}^n \cup \{\pm\infty\}$, $A \in M_{m \times n}(\mathbb{R})$ and x is the variable vector.

When $c \cdot x$ is constant, LP is a satisfiability problem.

The Simplex Algorithm

- Creates a system of equations $x_i = \sum c_j \cdot x_j$ equivalent to $Ax = 0$, called the tableau.
- Initiates $x := 0$; satisfies equations, but not always bounds.
- Shuffles the tableau's equations and updates the assignment.
- Tableau rows are linear combinations of the original rows.

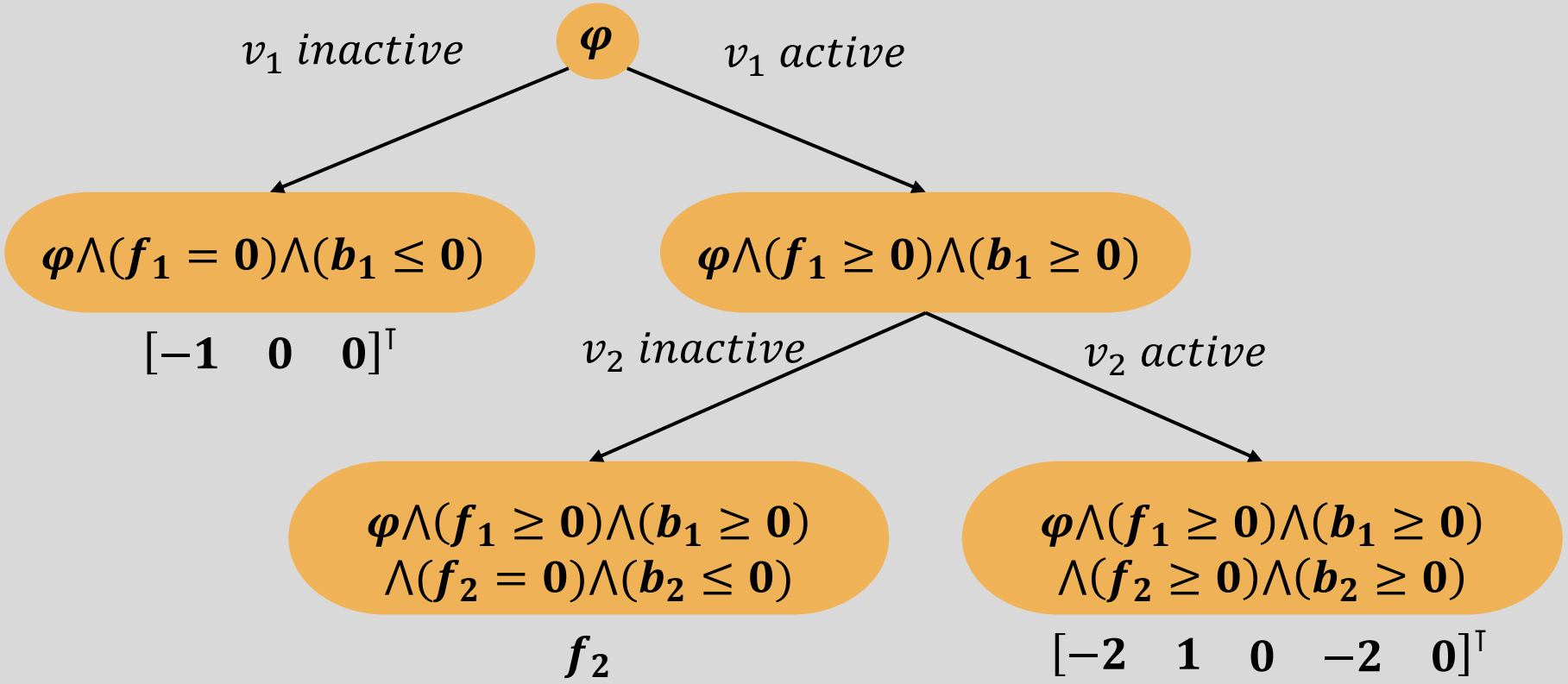
Agenda

- Introduction
- Neural Network Verification
- Proof Production
- Implementation and Evaluation
- Future Work

Proof Production

- Proving `SAT` is straightforward; proving `UNSAT` is more complicated, as DNN verification is NP-Complete for simple cases [Katz et. al, ‘17].
- We suggest a novel proof of `UNSAT`, constructed during execution.

Overview



In the proof-tree, nodes represent case-splits; leaves contain proofs of contradictions, which are represented by a vector.

Producing Proofs

- For Simplex:
  - The simple case
  - Supporting bound tightening
- Extending to DNN verification

The Simple Case

- Simplex reaches `UNSAT` if and only if a variable has inconsistent bounds; either from input or deduced by some tableau row discovered by the algorithm.
- Goal: construct a corresponding vector.
- Solution:
  - If input bounds are inconsistent - trivial.
  - If deduced by some tableau row - use its coefficients vector.

Example

Assume both ReLU constraints in φ are active:

Tableau (simplified):

$$b_1 = x_1 - x_2$$
$$b_2 = -2f_1$$
$$y = f_2$$
$$f_1 = b_1$$
$$f_2 = b_2$$

Bounds:

$$2 \leq x_1 \leq 3$$
$$-1 \leq x_2 \leq 1$$
$$0 \leq b_1, b_2, f_1 \leq 0.5$$
$$0.25 \leq f_2, y \leq 0.5$$

Example

$$-2(b_1 = x_1 - x_2) +$$
$$1(b_2 = -2f_1) +$$
$$0(y = f_2) +$$
$$-2(f_1 = b_1) +$$
$$0(f_2 = b_2)$$

$$b_2 = -2x_1 + 2x_2$$

Example

- Given:

$$b_2 = -2x_1 + 2x_2$$

- We have that $b_2 \leq -2l(x_1) + 2u(x_2) = -2 \cdot 2 + 2 \cdot 1 = -2$, but $b_2 \geq 0$ (v_2 active).
- The coefficients vector $[-2, 1, 0, -2, 0]^T$ is a proof of contradiction.

Bound Tightening

- Verifiers dynamically tighten bounds to improve performance.
- In one approach, given a row $x_i = \sum_j c_j \cdot x_j$, a new bound can be tightened by:

$$u'(x_i) = \sum_{j:c_j>0} [c_j \cdot u'(x_j)] + \sum_{j:c_j<0} [c_j \cdot l'(x_j)]$$

And similarly for $l'(x_j)$.

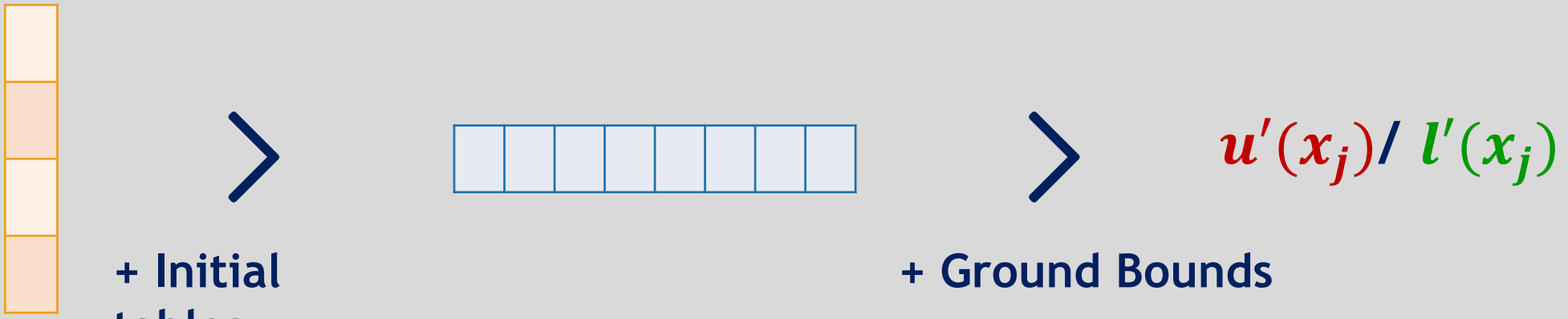
- We call the input bounds, ground bounds u, l, as opposed to dynamic bounds u', l' which are discovered on the fly.

Bound Tightening

- The inconsistent bounds of a variable can be deduced using many previous tightenings.
- Naively proving each is possible, but wasteful.
- We prove the tightest bound using the ground data only.

UNSAT Proof for Simplex

- Dynamic bounds are explained with column vectors $f_u(x_j), f_l(x_j)$ that represent a linear combination of initial tableau rows.



- The vectors are updated when a bound is tightened and enable deducing dynamic bounds using ground data only.

Explanation Update Rule

- Initiate as zero vectors.
- Given $x_i = \sum_j c_j \cdot x_j$ and $f_u(x_j), f_l(x_j)$, whenever we update:

$$u'(x_i) = \sum_{j:c_j>0} [c_j \cdot u'(x_j)] + \sum_{j:c_j<0} [c_j \cdot l'(x_j)]$$

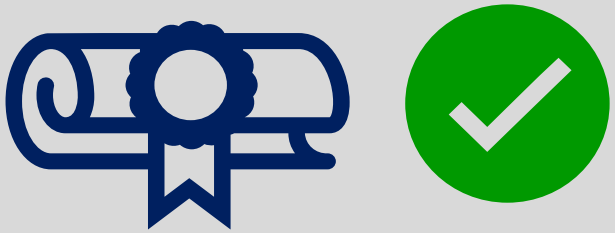
Also update:

$$f_u(x_i) = \sum_{j:c_j>0} [c_j \cdot f_u(x_j)] + \sum_{j:c_j<0} [c_j \cdot f_l(x_j)] + coef(Row)$$

And similarly for $f_l(x_j)$.

Proving Contradiction

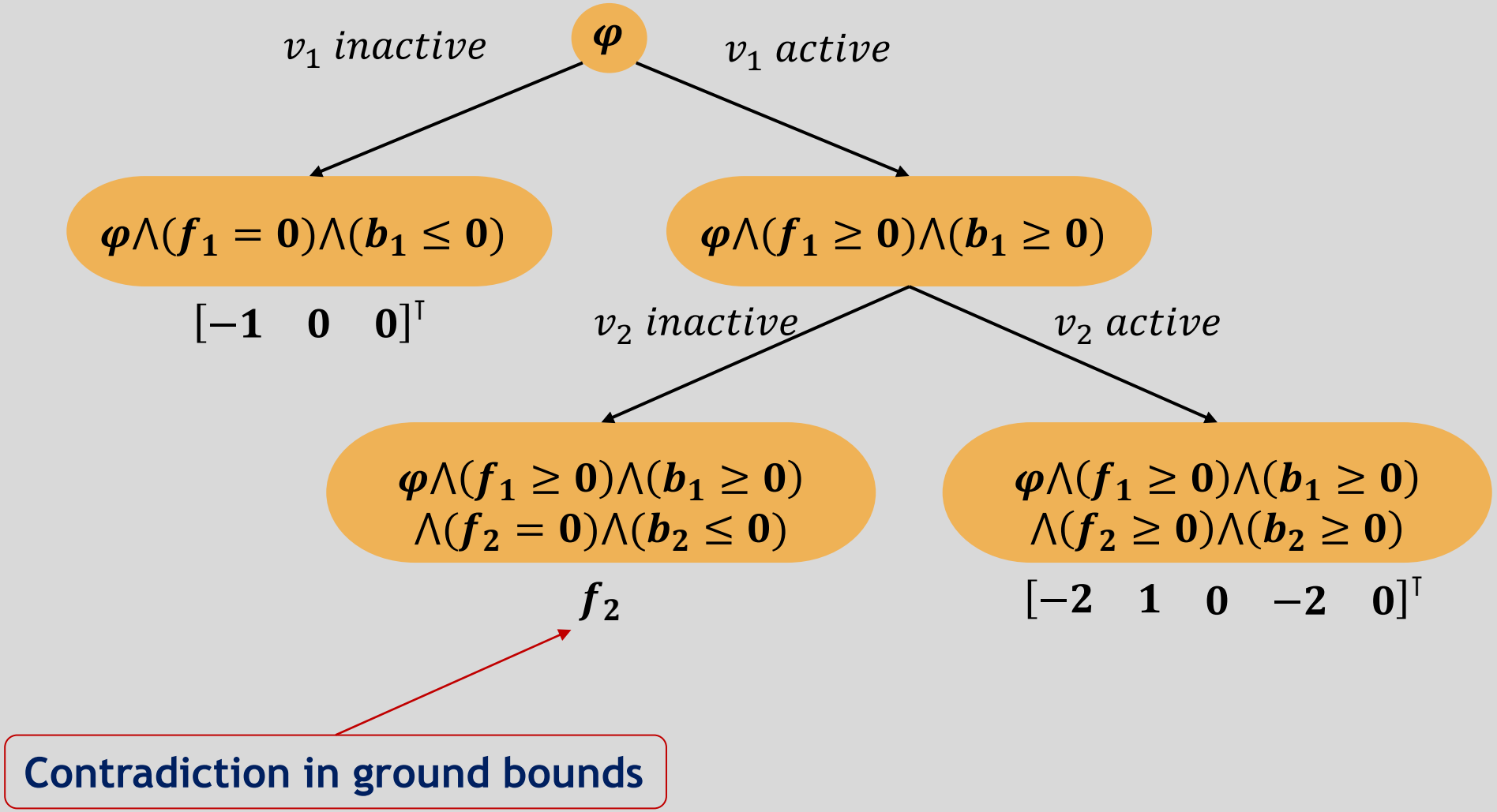
- When `UNSAT`, a variable has inconsistent bounds $u'(x_j) < l'(x_j)$, explained with $f_u(x_j), f_l(x_j)$.
- The vector $w = f_u(x_j) - f_l(x_j)$ witnesses contradiction.
- Essentially, a constructive proof for the Farkas' Lemma, for LP of the form $Ax = 0 \wedge l \leq x \leq u$.



Extending Proofs to DNN Verification

- On every split in the search tree, split the proof tree.
- Split bounds are set as the new ground bounds, their vectors are reset to zero, and all other vectors relate to them.

Example - Proof Checking



Agenda

- Introduction
- Neural Network Verification
- Proof Production
- Implementation and Evaluation
- Future Work

Implementation

- Enhanced the Marabou DNN verifier with a proof-production mechanism and a proof-checker.
- 180 ACAS-Xu queries - 113 returned `UNSAT` by Marabou.

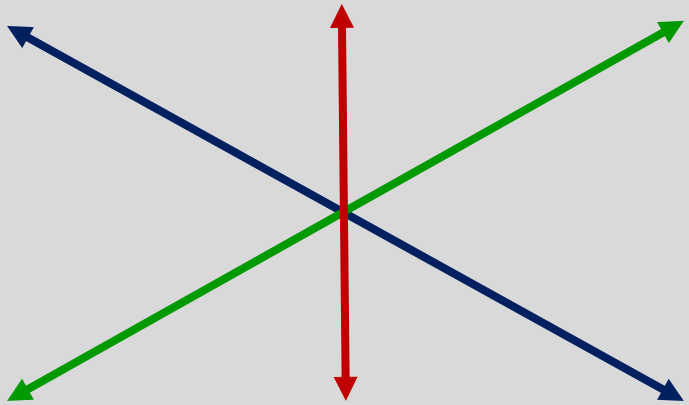
Evaluation

- In 1.46M leaves, 99.99% included a correct proof. All other were found `UNSAT` by a trusted (slower) SMT solver.
- All benchmark answers by Marabou were correct, even when proofs were not.

Agenda

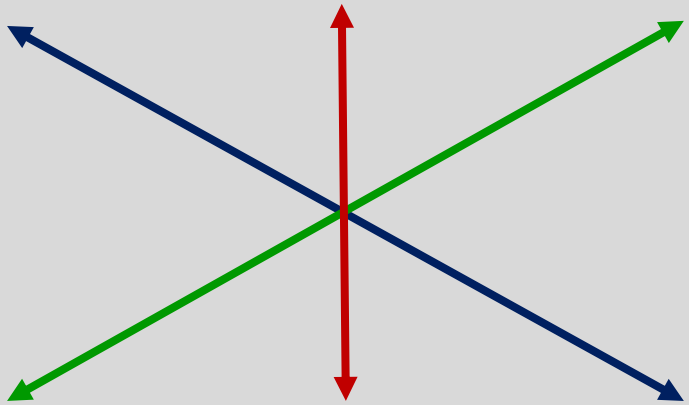
- Introduction
- Neural Network Verification
- Proof Production
- Implementation and Evaluation
- Future Work

Future Work

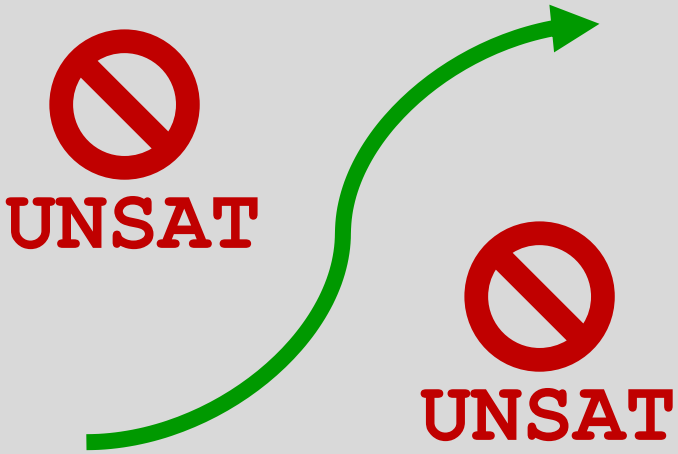


Support additional
procedures and topologies

Future Work



Support additional procedures and topologies



Use proofs for CDCL

Q&A