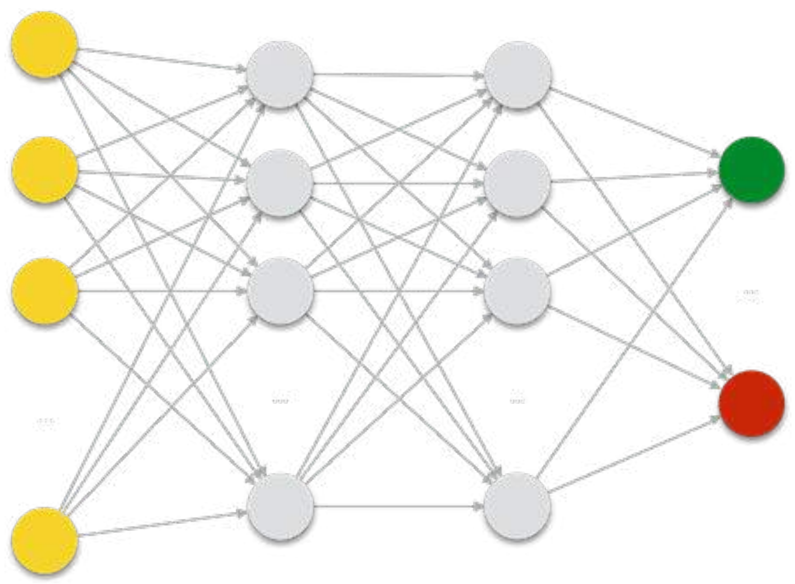
Formal Verification of Machine Learning

MPRI 2-6: Abstract Interpretation, Application to Verification and Static Analysis



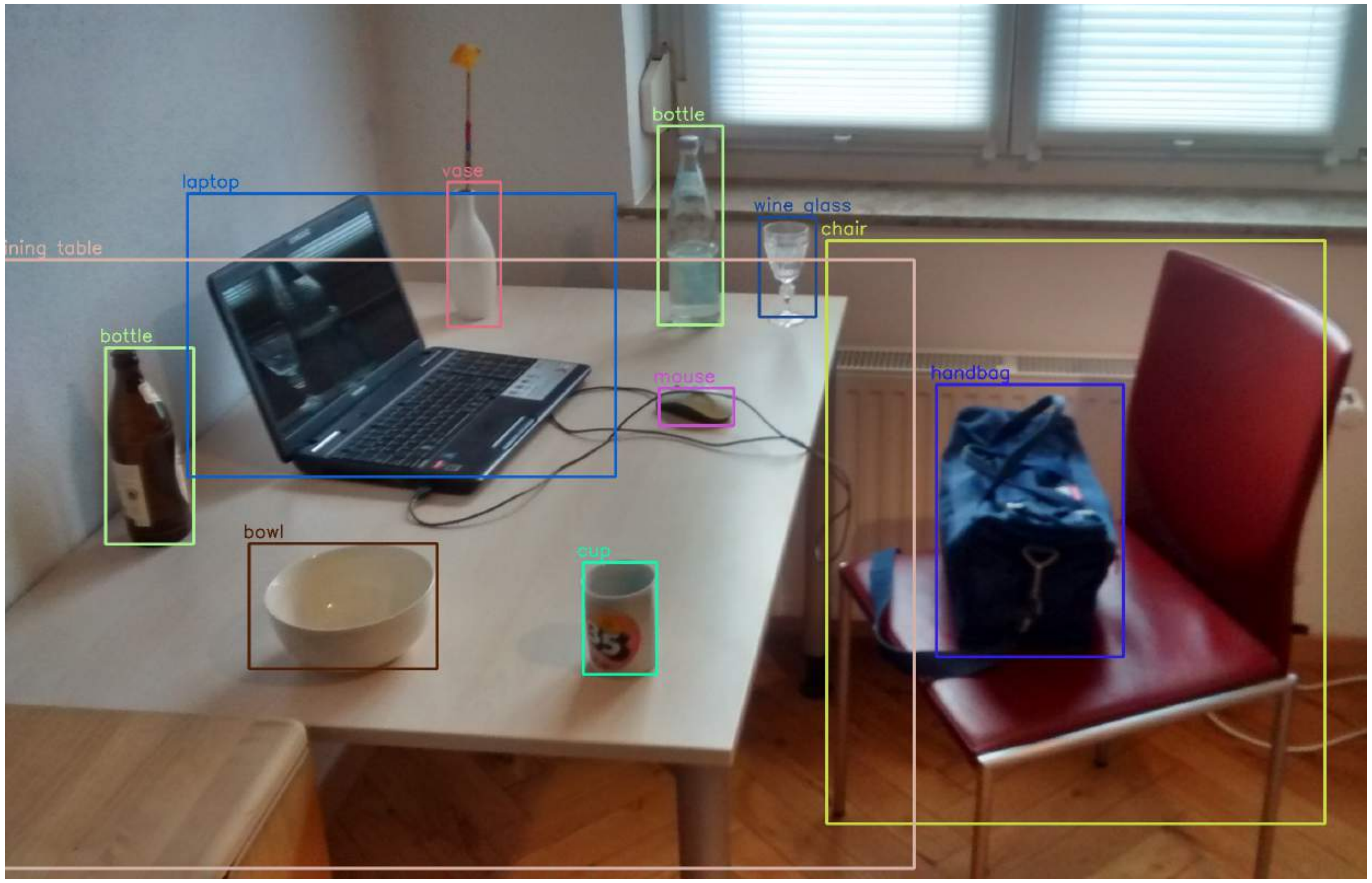
Caterina Urban

February 7th, 2022

Year 2021-2022

Machine Learning Revolution

Computer software able to efficiently and **autonomously perform tasks** that are difficult or even *impossible* to design using explicit programming



Examples: object recognition, image classification, speech recognition, etc.

ML in Safety-Critical Applications

Enables **new functions that could not be envisioned before**

Self-Driving Cars

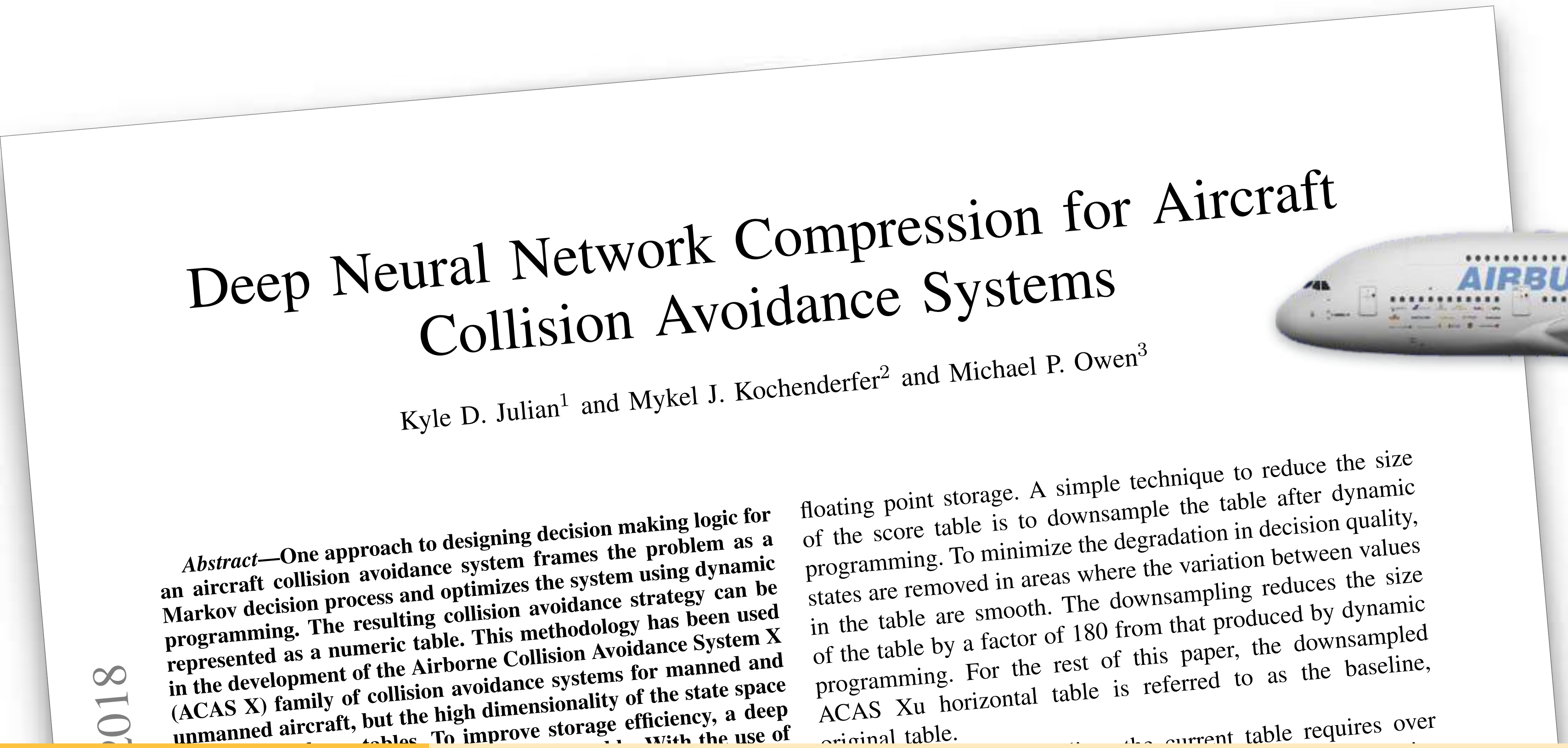
Image-Based Taxiing, Takeoff, Landing

Aircraft Voice Control

ML in Safety-Critical Applications

Approximates complex systems and **automates decision-making**

Diagnosis and Drug Discovery



Deep Neural Network Compression for Aircraft Collision Avoidance Systems

Kyle D. Julian[1] and Mykel J. Kochenderfer[2] and Michael P. Owen[3]

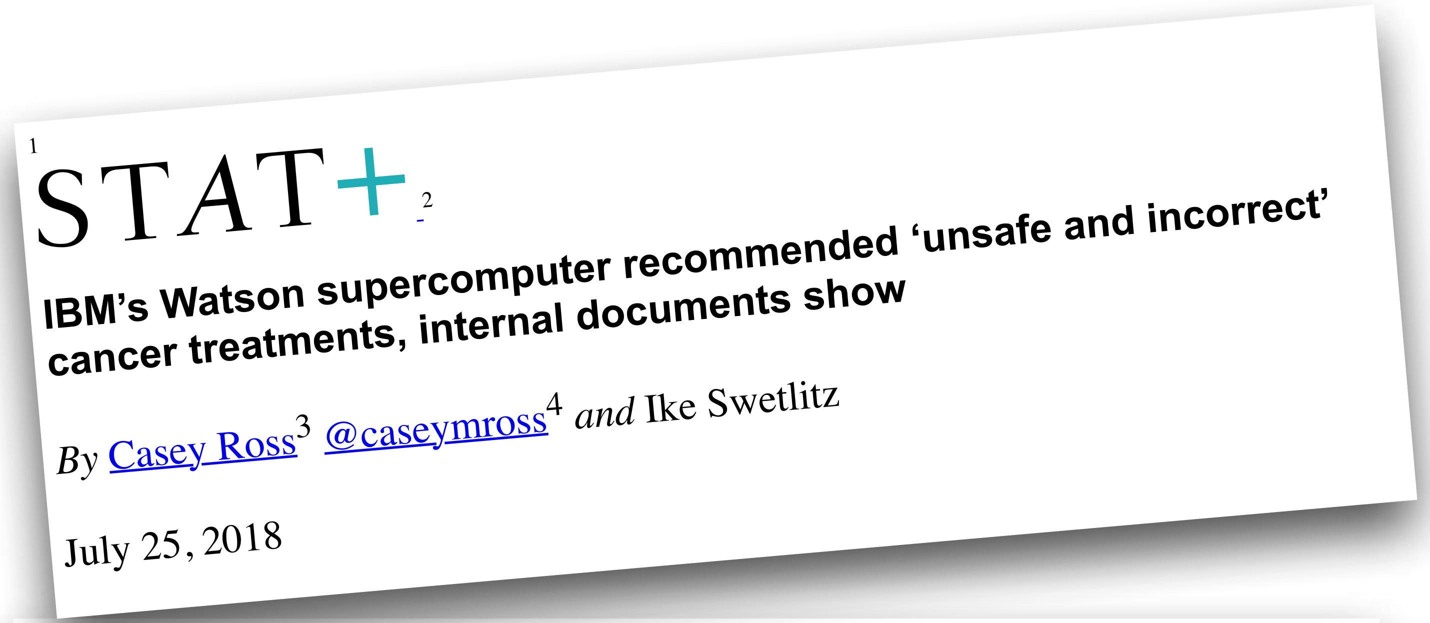
Abstract—One approach to designing decision making logic for an aircraft collision avoidance system frames the problem as a Markov decision process and optimizes the system using dynamic programming. The resulting collision avoidance strategy can be represented as a numeric table. This methodology has been used in the development of the Airborne Collision Avoidance System X (ACAS X) family of collision avoidance systems for manned and unmanned aircraft, but the high dimensionality of the state space

floating point storage. A simple technique to reduce the size of the score table is to downsample the table after dynamic programming. To minimize the degradation in decision quality, states are removed in areas where the variation between values in the table are smooth. The downsampling reduces the size of the table by a factor of 180 from that produced by dynamic programming. For the rest of this paper, the downsampled ACAS Xu horizontal table is referred to as the baseline, original table.

2018

Aircraft Collision Avoidance

ML in Safety-Critical Applications

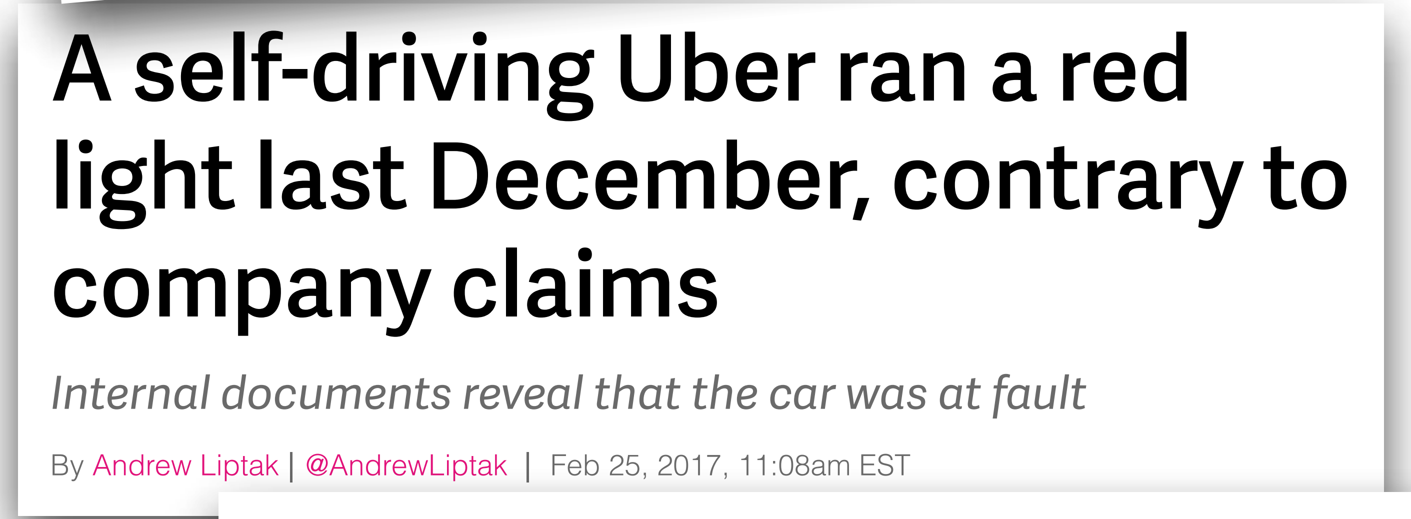


[1] STAT+ [2]

IBM's Watson supercomputer recommended 'unsafe and incorrect' cancer treatments, internal documents show

By Casey Ross[3] @caseymross[4] *and* Ike Swetlitz

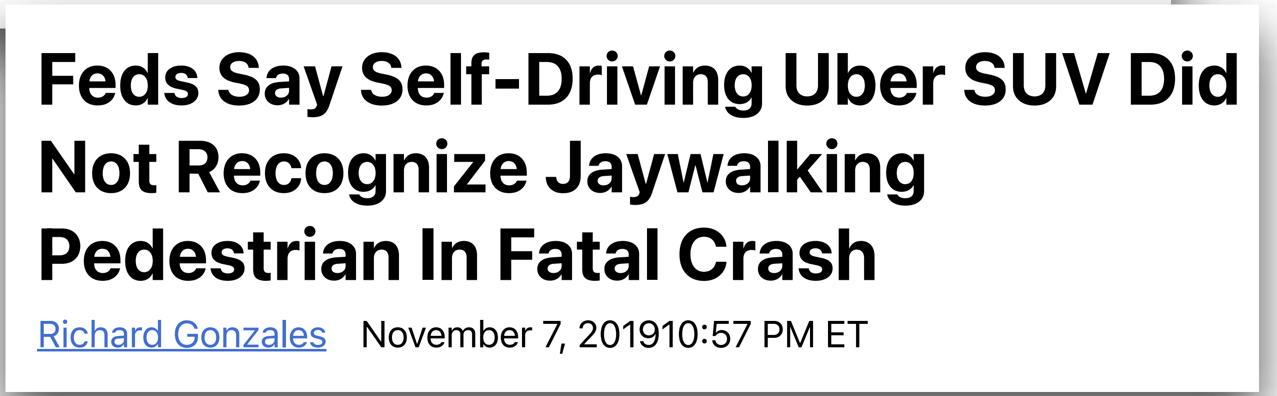
July 25, 2018



A self-driving Uber ran a red light last December, contrary to company claims

Internal documents reveal that the car was at fault

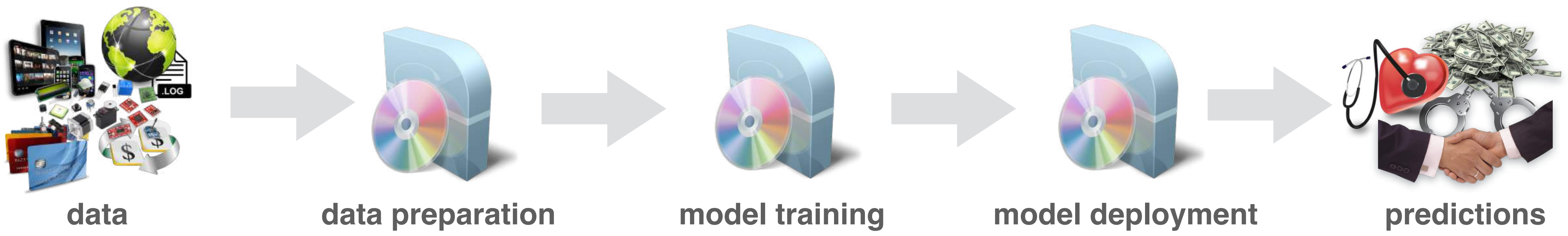
By Andrew Liptak | @AndrewLiptak | Feb 25, 2017, 11:08am EST



Feds Say Self-Driving Uber SUV Did Not Recognize Jaywalking Pedestrian In Fatal Crash

Richard Gonzales November 7, 201910:57 PM ET

Machine Learning Pipeline



Machine Learning Pipeline

Model Training is Highly Non-Deterministic

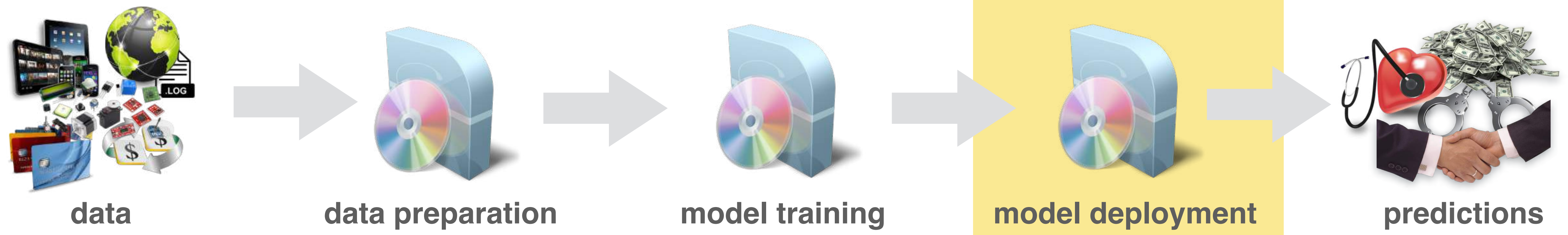
THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

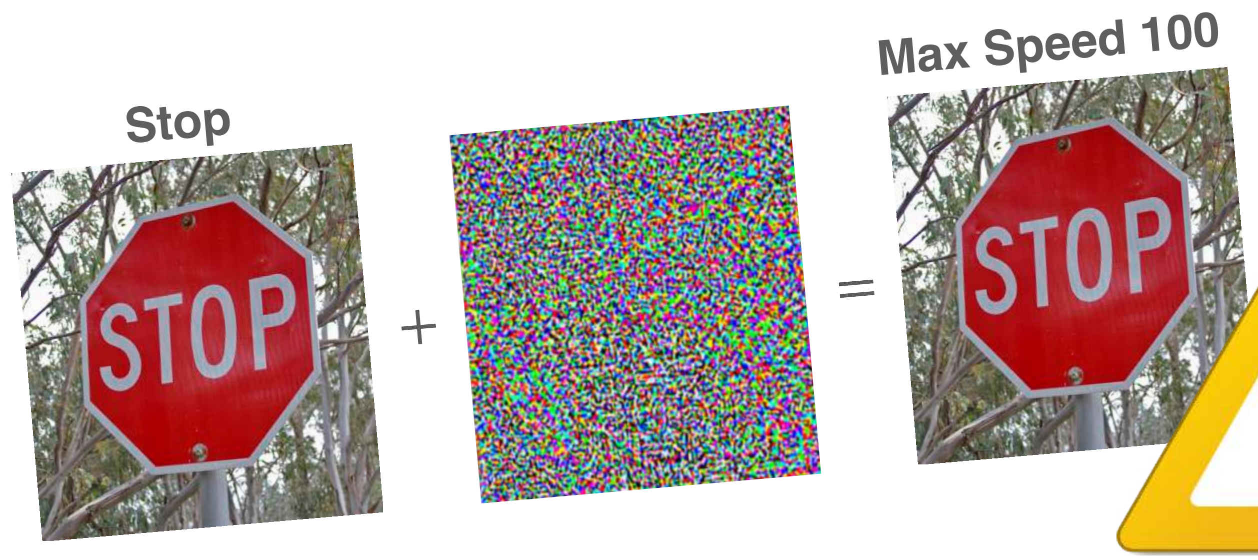
odel training
model deployment
predictions

no predictability and traceability

Machine Learning Pipeline

Models Only Give Probabilistic Guarantees

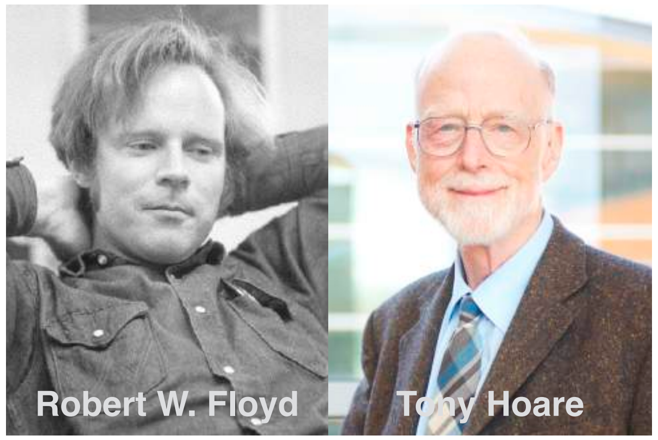




not sufficient for guaranteeing an acceptable failure rate under any circumstance

Formal Methods

Mathematical Guarantees of Safety



Deductive Verification

- extremely **expressive**
- **relies on the user** to guide the proof

Model Checking

- analysis of a **model** of the software
- **sound and complete with respect to the model**

Static Analysis

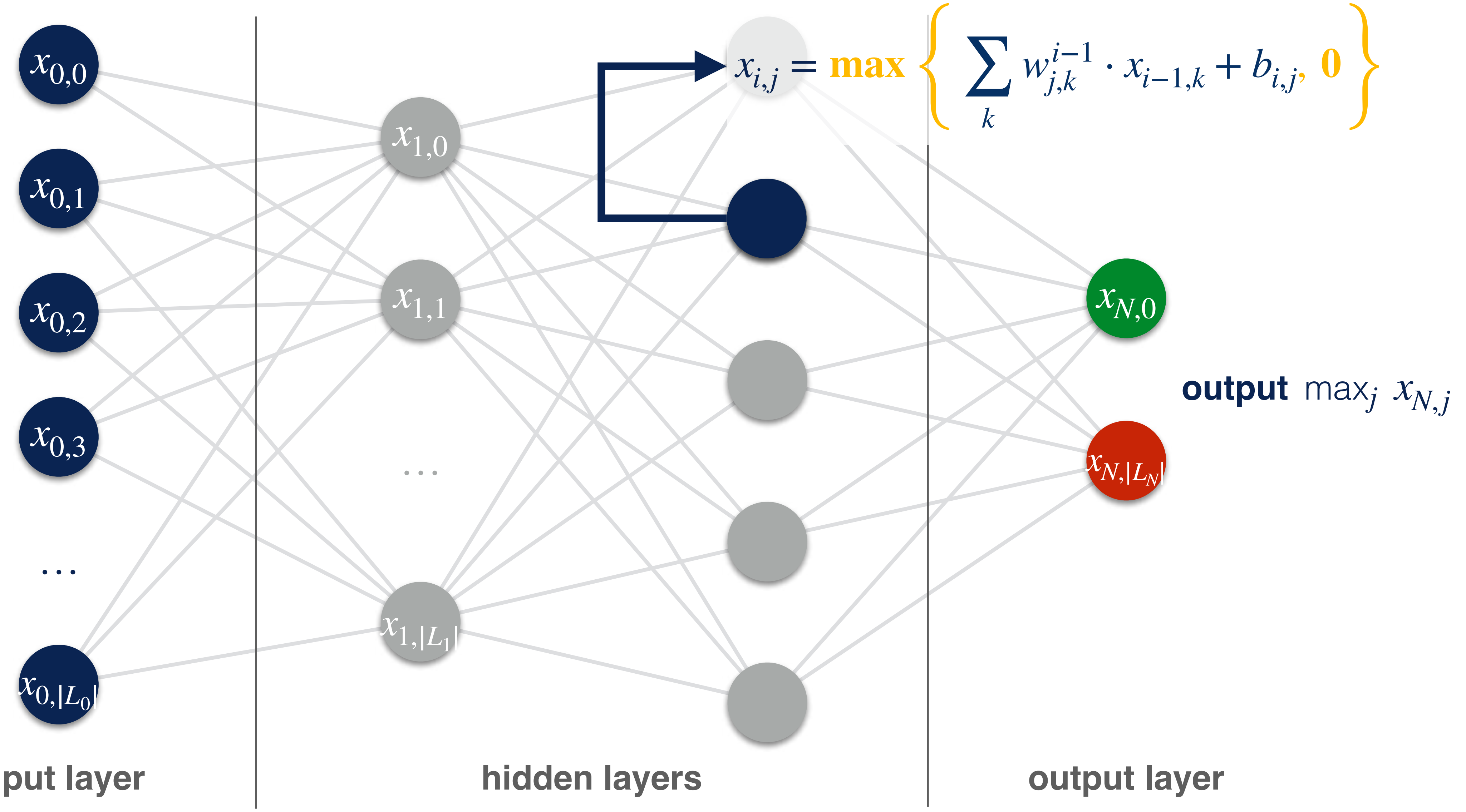
- analysis of the software at some level of **abstraction**
- fully **automatic** and **sound** by construction
- generally **not complete**

Formal Methods for Trained Models

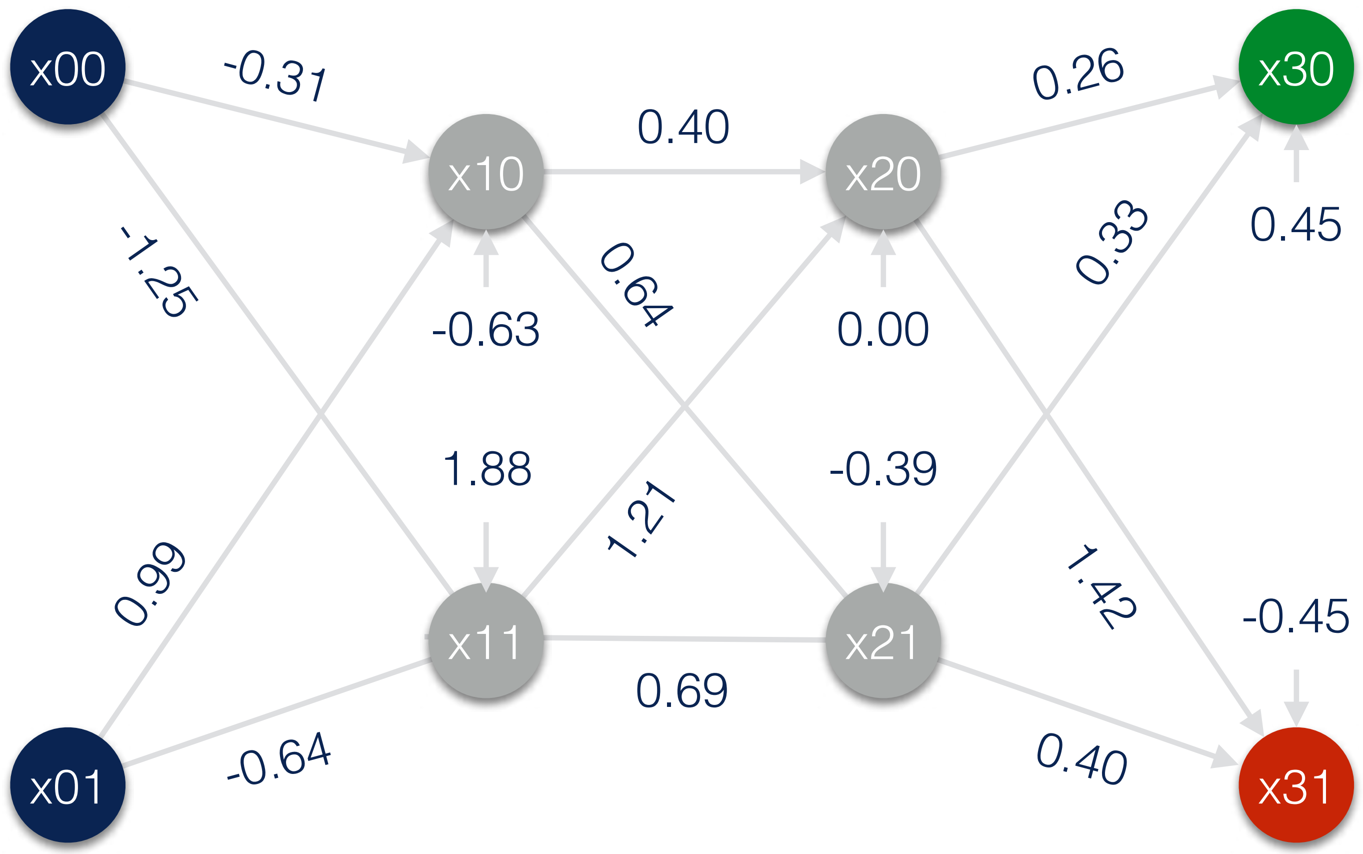
Neural Networks

Neural Networks

Feed-Forward Fully-Connected Neural Networks with ReLU Activation Functions

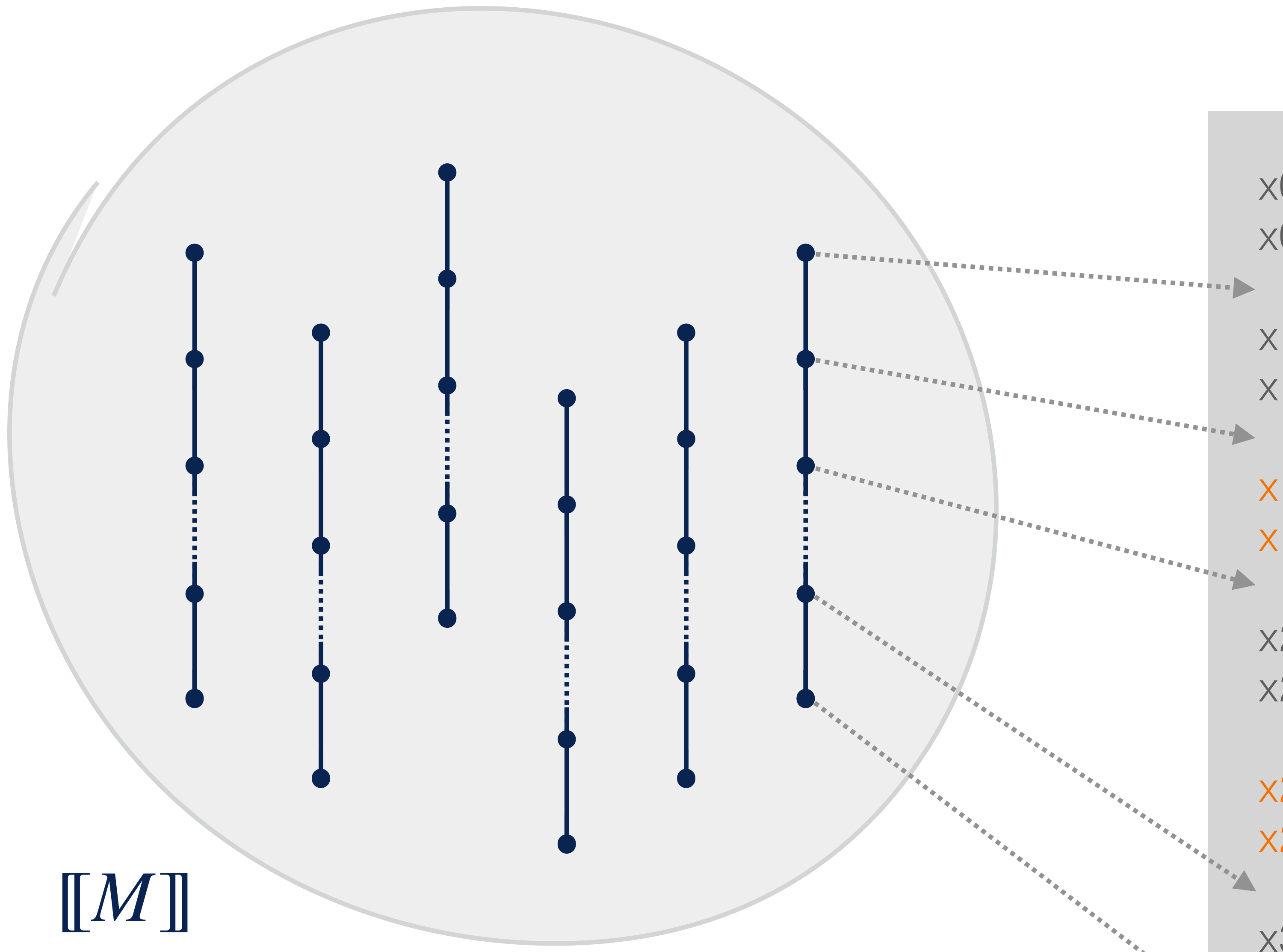


Feed-Forward Fully-Connected ReLU Networks as Programs



```
x00 = input()
x01 = input()

x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88

x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '🟢' if x31 < 30 else '🔴'
```

Maximal Trace Semantics

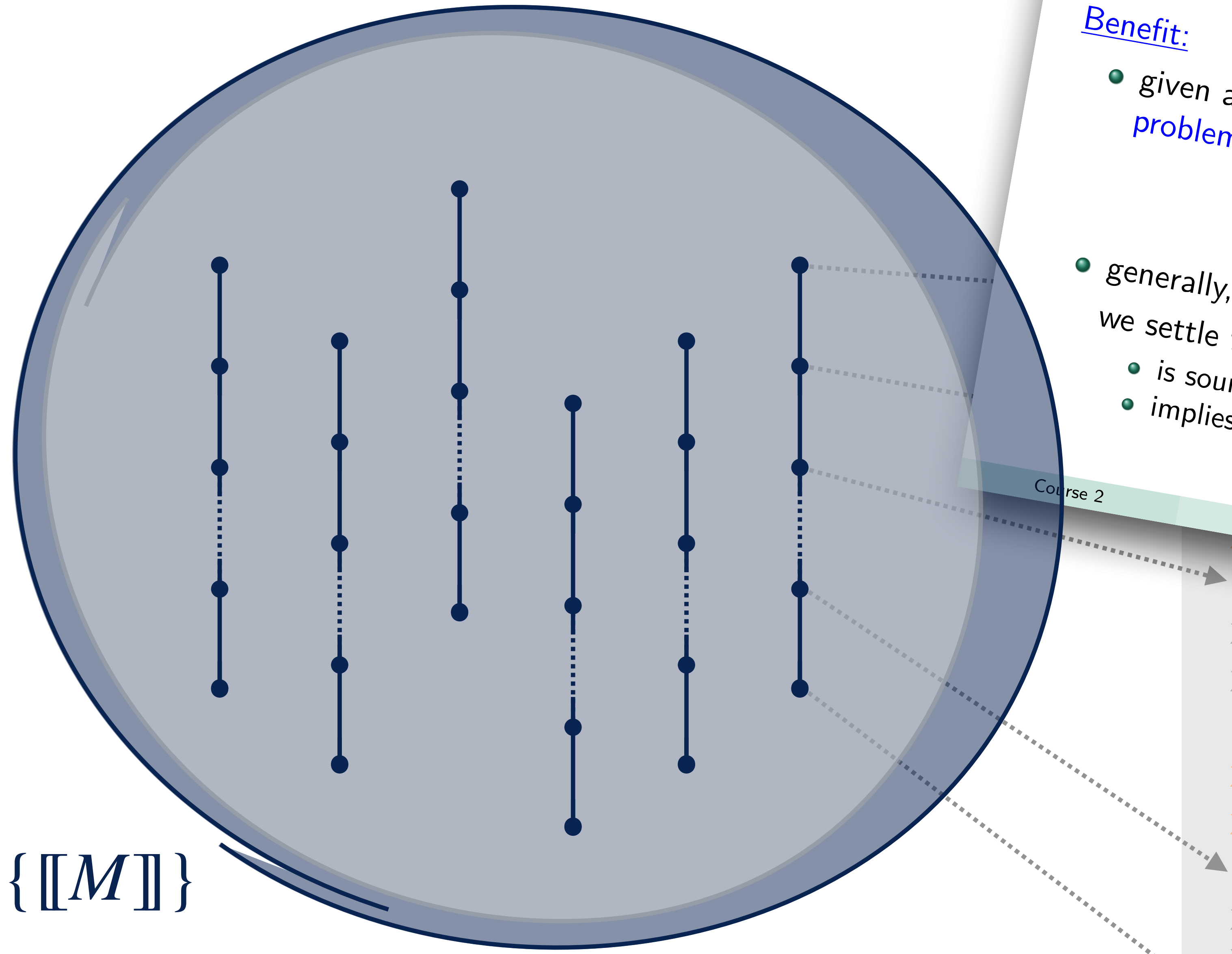


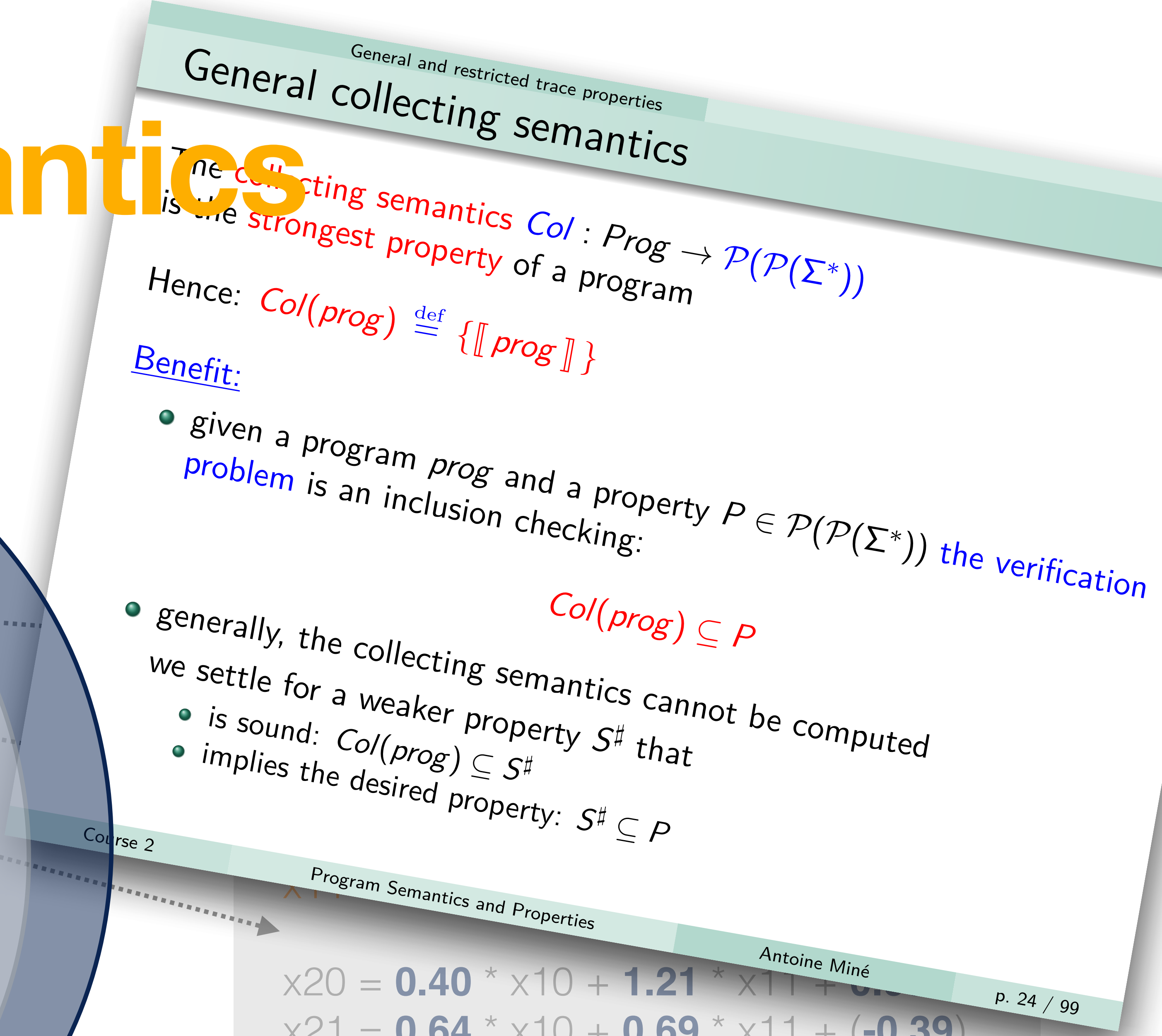
M

```
x00 = input()
x01 = input()

x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88

x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '🟢' if x31 < 30 else '🔴'
```

Neural Network Verification

Collecting Semantics





General collecting semantics

The collecting semantics $Col : Prog \to \mathcal{P}(\mathcal{P}(\Sigma^*))$
is the strongest property of a program

Hence: $Col(prog) \overset{\text{def}}{=} \{[\![\, prog \,]\!]\}$

Benefit:

- given a program *prog* and a property $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$ the verification problem is an inclusion checking:

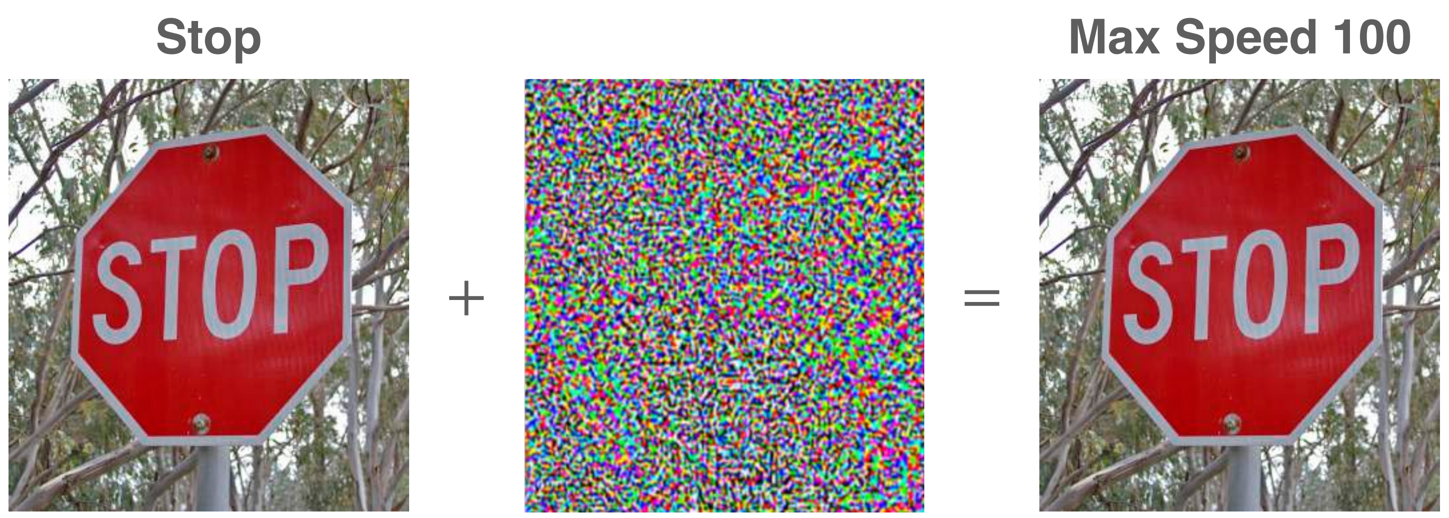
$$Col(prog) \subseteq P$$

- generally, the collecting semantics cannot be computed
  we settle for a weaker property $S^\sharp$ that
  - is sound: $Col(prog) \subseteq S^\sharp$
  - implies the desired property: $S^\sharp \subseteq P$

```
x20 = 0.40 * x10 + 1.21 * x11 + 0.
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '🟢' if x31 < 30 else '🔴'
```

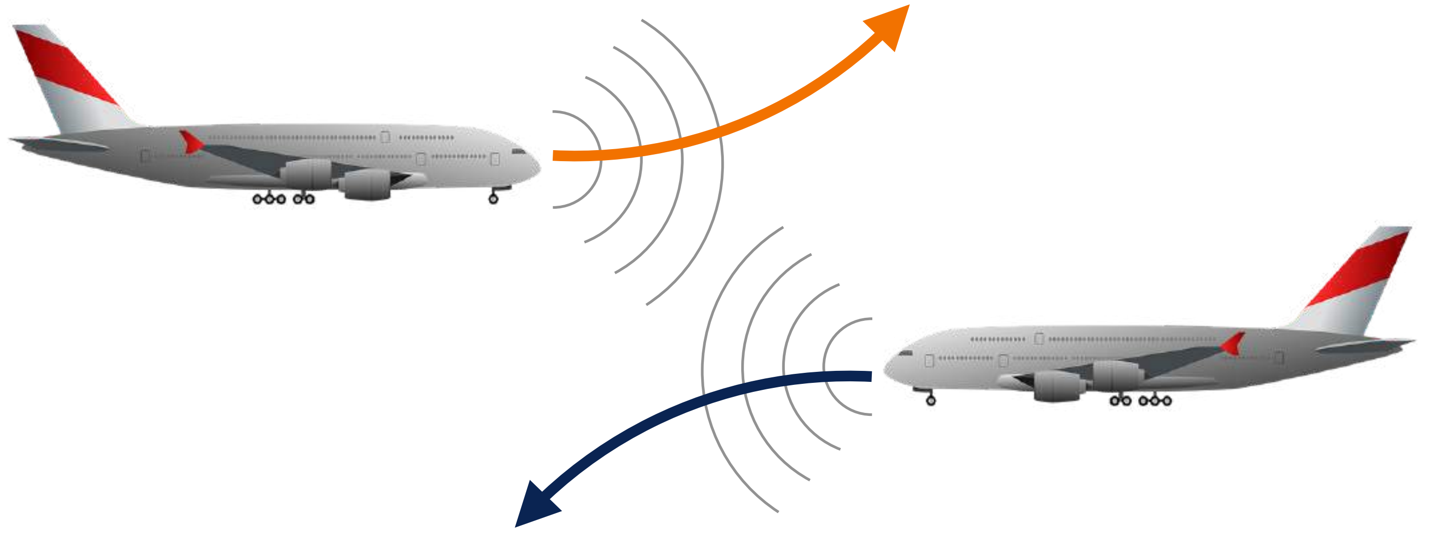
Stability

Goal G3 in [Kurd03]

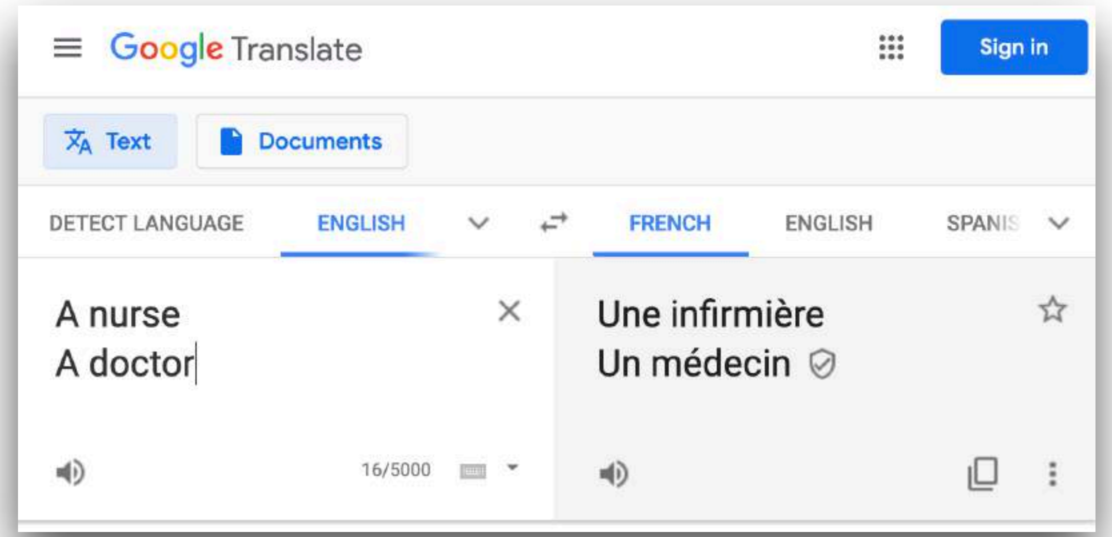


Safety

Goal G4 in [Kurd03]



Fairness

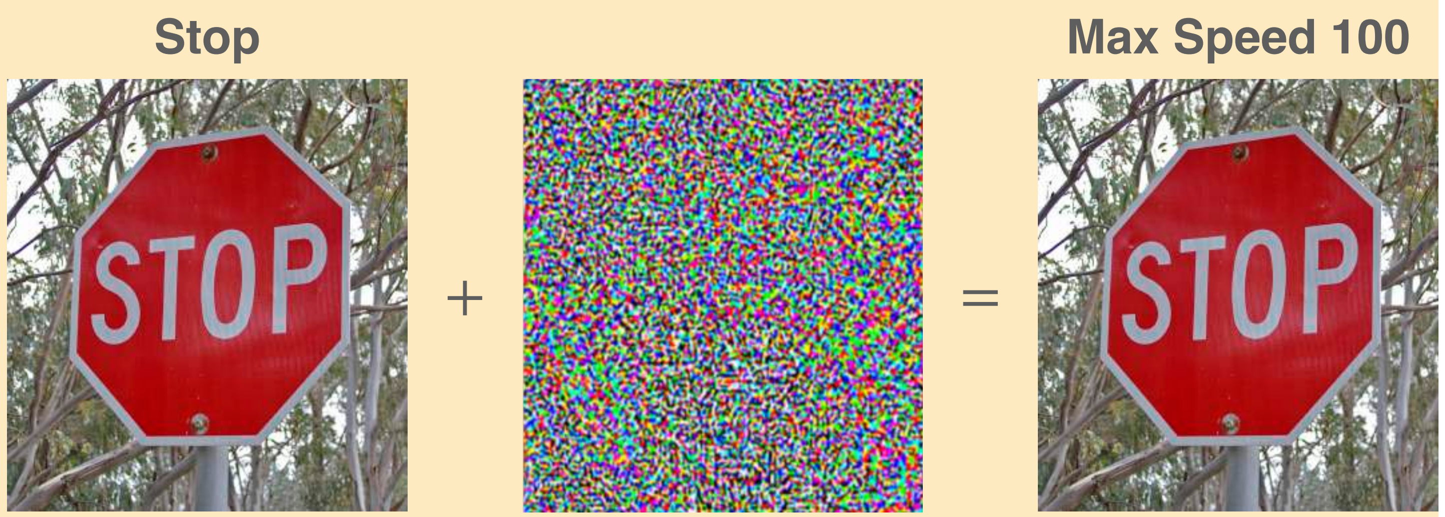


A nurse
A doctor

Une infirmière
Un médecin

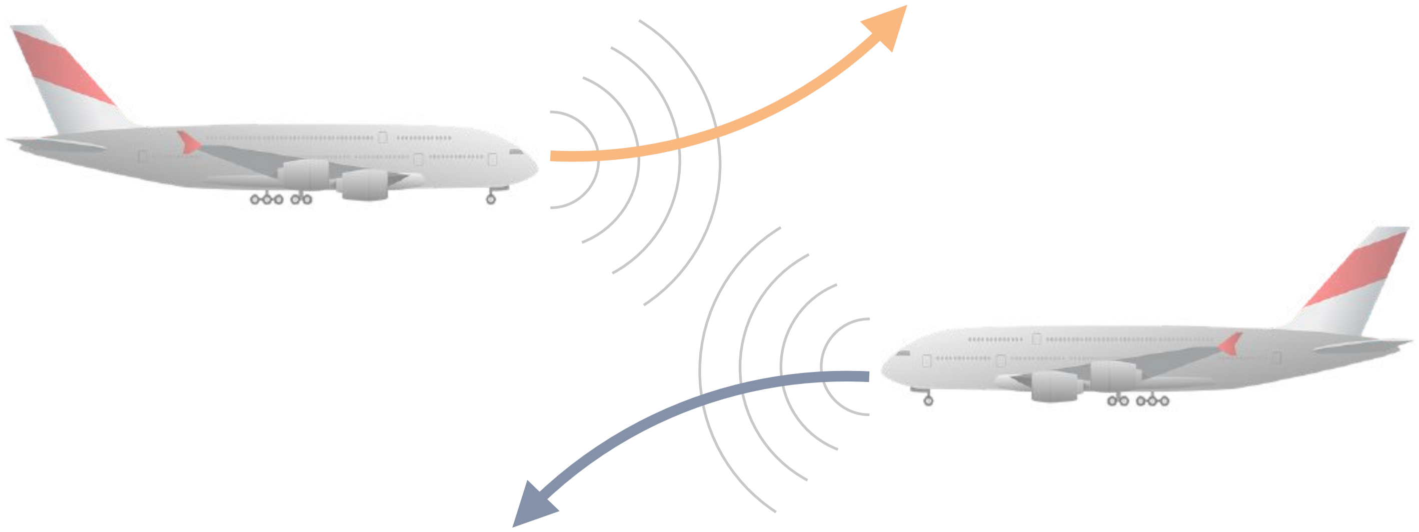
Stability

Goal G3 in [Kurd03]

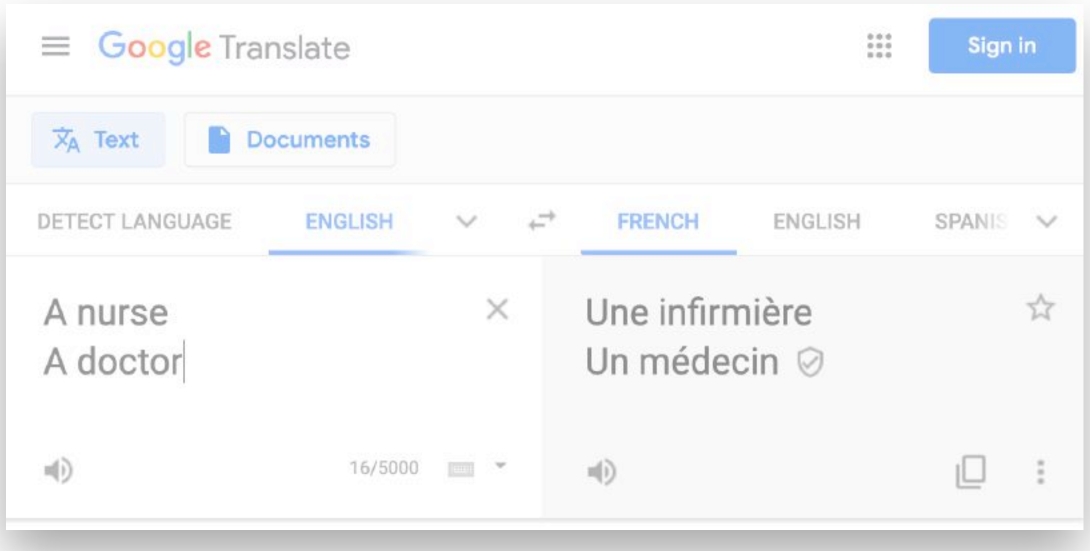


Safety

Goal G4 in [Kurd03]



Fairness

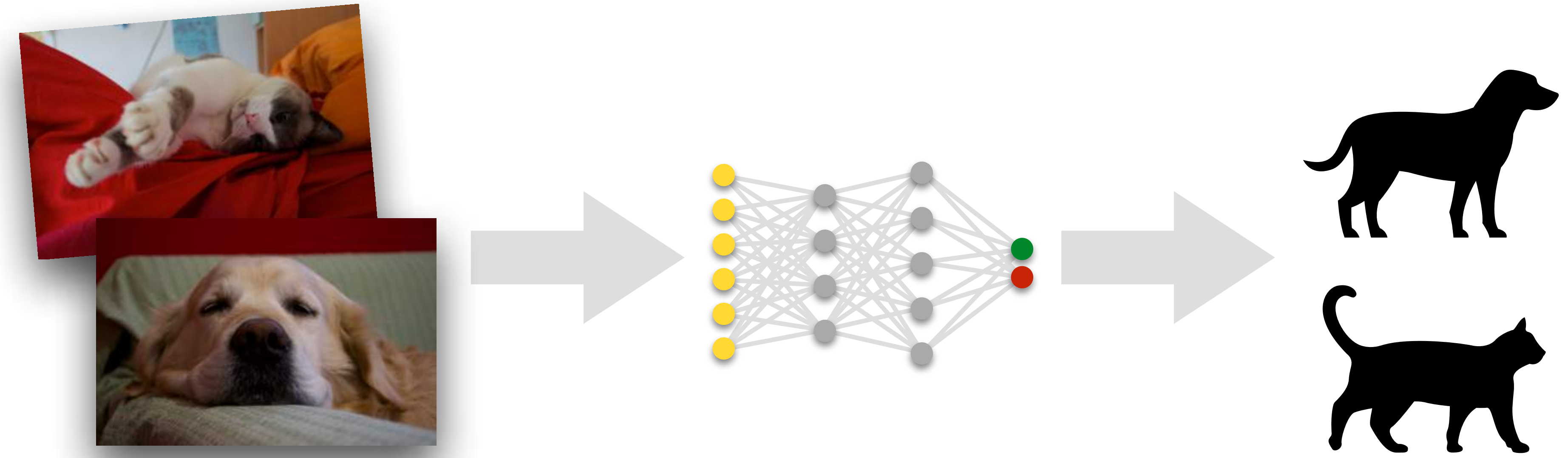


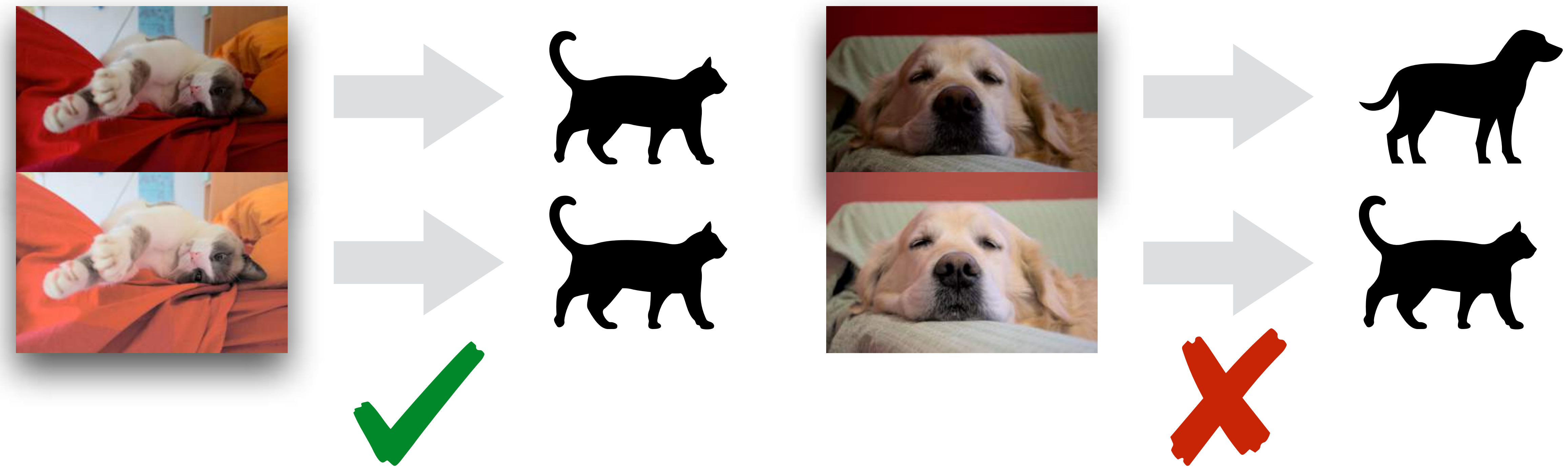
A nurse
A doctor

Une infirmière
Un médecin

Local Stability

The **classification** is **unaffected by small input perturbations**





Local Stability

Distance-Based Perturbations

$$P_{\delta,\epsilon}(\mathbf{x}) \overset{\text{def}}{=} \{\mathbf{x}' \in \mathscr{R}^{|L_0|} \mid \delta(\mathbf{x}, \mathbf{x}') \leq \epsilon\}$$

Example (L_∞ distance): $P_{\infty,\epsilon}(\mathbf{x}) \overset{\text{def}}{=} \{\mathbf{x}' \in \mathscr{R}^{|L_0|} \mid \max_i |\mathbf{x}_i - \mathbf{x}'_i| \leq \epsilon\}$

$$\mathscr{R}_{\mathbf{x}}^{\delta,\epsilon} \overset{\text{def}}{=} \{[\![M]\!] \in \mathscr{P}(\Sigma^*) \mid \text{STABLE}_{\mathbf{x}}^{\delta,\epsilon}([\![M]\!])\}$$

$\mathscr{R}_{\mathbf{x}}^{\delta,\epsilon}$ is the set of all neural networks M (or, rather, their semantics $[\![M]\!]$) that are **stable** in the neighborhood $P_{\delta,\epsilon}(\mathbf{x})$ of a given input $\mathbf{x}$

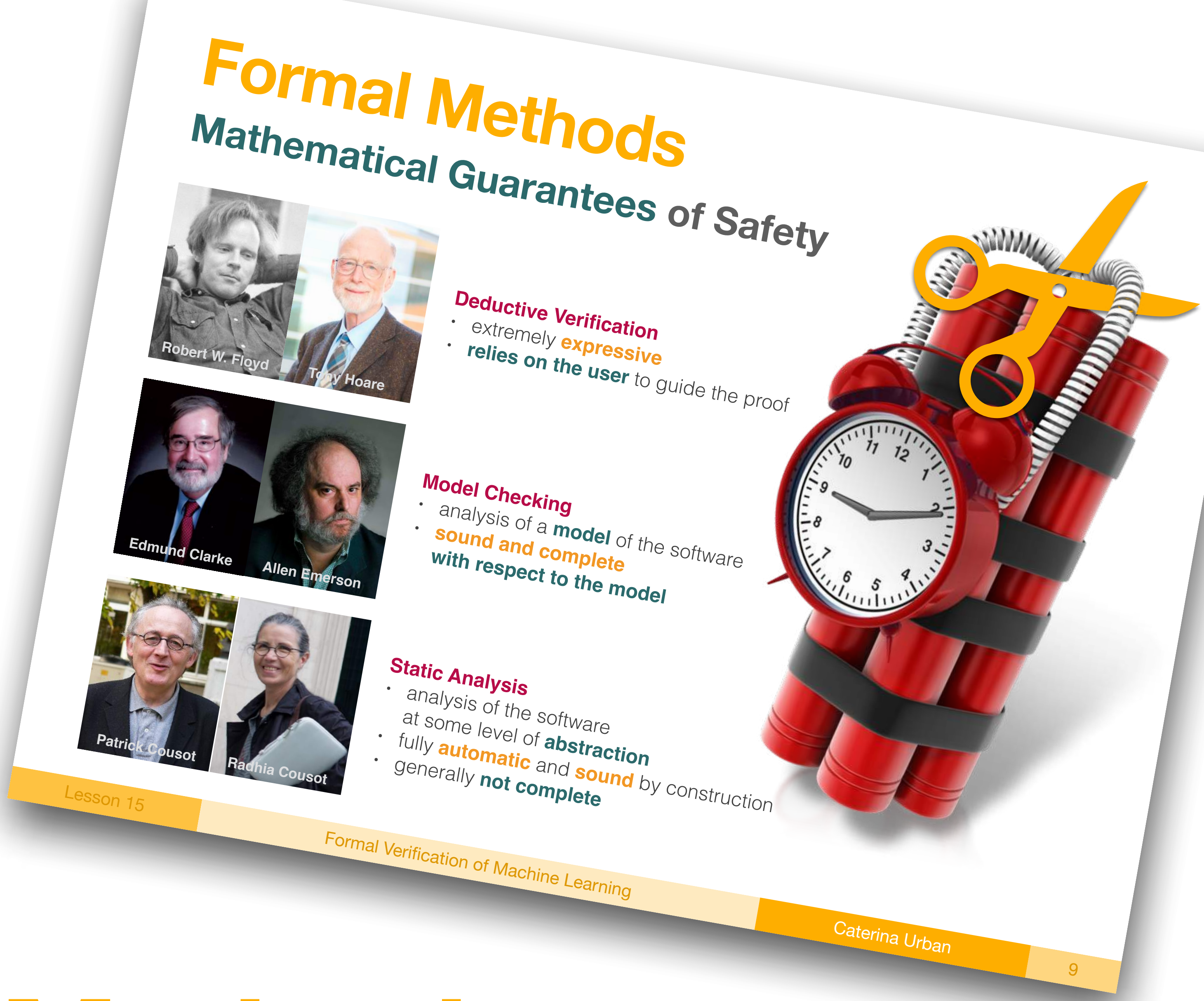
$$\begin{aligned}\text{STABLE}_{\mathbf{x}}^{\delta,\epsilon}([\![M]\!]) \overset{\text{def}}{=} & \; \forall t \in [\![M]\!] : (\exists t' \in [\![M]\!] : \forall 0 \leq i \leq |L_0| : t'_0(x_{0,i}) = \mathbf{x}_i) \\ & \land (\exists \mathbf{x}' \in P_{\delta,\epsilon}(\mathbf{x}) : \forall 0 \leq i \leq |L_0| : t_0(x_{0,i}) = \mathbf{x}'_i) \\ & \Rightarrow \max_j t_\omega(x_{N,j}) = \max_j t'_\omega(x_{N,j})\end{aligned}$$

Theorem

$$M \models \mathscr{R}_{\mathbf{x}}^{\delta,\epsilon} \Leftrightarrow \{[\![M]\!]\} \subseteq \mathscr{R}_{\mathbf{x}}^{\delta,\epsilon}$$

Corollary

$$M \models \mathscr{R}_{\mathbf{x}}^{\delta,\epsilon} \Leftrightarrow [\![M]\!] \subseteq \bigcup \mathscr{R}_{\mathbf{x}}^{\delta,\epsilon}$$



Formal Methods

Mathematical Guarantees of Safety

Deductive Verification
- extremely **expressive**
- **relies on the user** to guide the proof

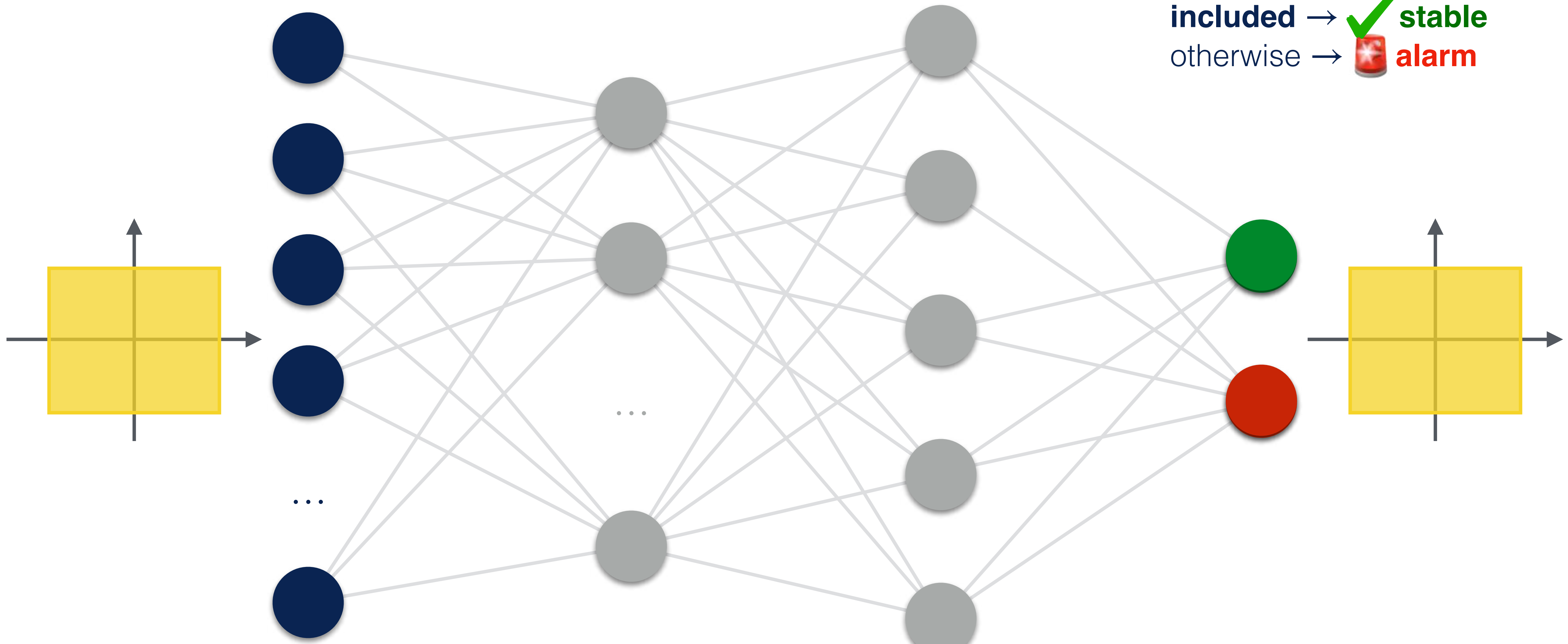
Model Checking
- analysis of a **model** of the software
- **sound and complete with respect to the model**

Static Analysis
- analysis of the software at some level of **abstraction**
- fully **automatic** and **sound** by construction
- generally **not complete**

Lesson 15 | Formal Verification of Machine Learning | Caterina Urban | 9

Static Analysis Methods

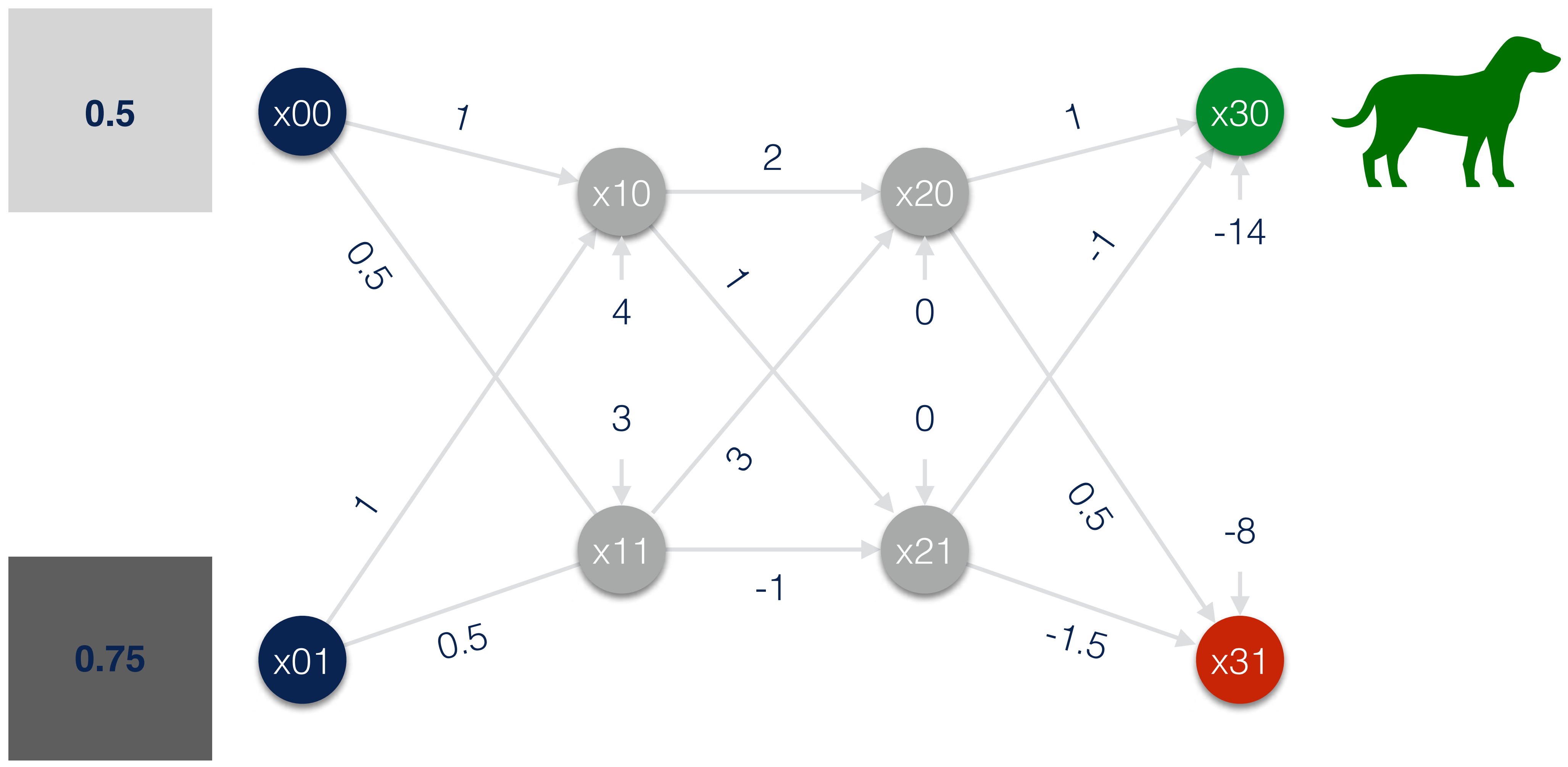
Forward Analysis



② check output for **inclusion** in **expected output**:
included → ✓ **stable**
otherwise → 🚨 **alarm**

① proceed **forwards from an abstraction** of all possible perturbations

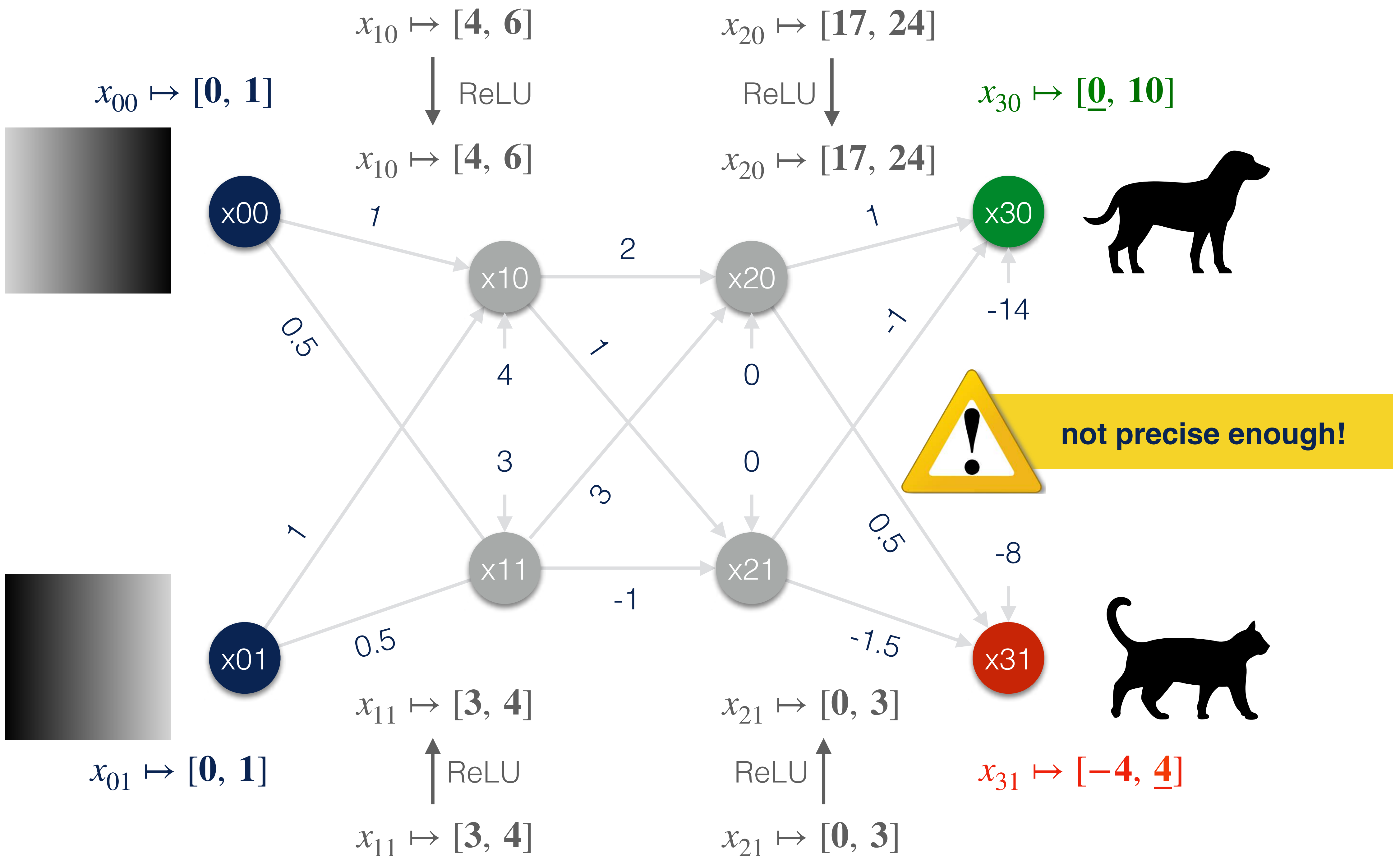
Example



$$P(\langle 0.5, 0.75 \rangle) \overset{\text{def}}{=} \{ \mathbf{x} \in \mathscr{R} \times \mathscr{R} \mid 0 \leq \mathbf{x}_0 \leq 1 \wedge 0 \leq \mathbf{x}_1 \leq 1 \}$$

Interval Domain

$x_{i,j} \mapsto [a, b]$
$a, b \in \mathscr{R}$



Interval Domain

with Symbolic Constant Propagation [Li19]

each neuron as a linear combination of the inputs and the previous ReLUs

$$x_{i,j} \mapsto \begin{cases} \sum_{k=0}^{i-1} \mathbf{c}_k \cdot \mathbf{x}_k + \mathbf{c} & \mathbf{c}_k, \mathbf{c} \in \mathscr{R}^{|\mathbf{X}_k|} \\ [a,\ b] & a, b \in \mathscr{R} \end{cases}$$

$$\begin{array}{l} x_{i-1,0} \mapsto \mathbf{E_{i-1,0}} \\ \dots \\ x_{i-1,j} \mapsto \mathbf{E_{i-1,j}} \\ \dots \end{array} \xrightarrow[x_{i,j} = \sum_k w_{j,k}^{i-1} \cdot x_{i-1,k} + b_{i,j}]{} x_{i,j} \mapsto \sum_k w_{j,k}^{i-1} \cdot \mathbf{E_{i-1,k}} + b_{i,j}$$

$$x_{i,j} \mapsto \begin{cases} \mathbf{E_{i,j}} \\ \mathbf{[a, b]} \end{cases} \xrightarrow{\text{ReLU}} \begin{array}{ll} x_{i,j} \mapsto \begin{cases} \mathbf{E_{i,j}} \\ \mathbf{[a, b]} \end{cases} & 0 \leq a \\ x_{i,j} \mapsto \begin{cases} \mathbf{x_{i,j}} \\ \mathbf{[0, b]} \end{cases} & a < 0 \wedge 0 < b \\ x_{i,j} \mapsto \begin{cases} \mathbf{0} \\ \mathbf{[0, 0]} \end{cases} & b \leq 0 \end{array}$$

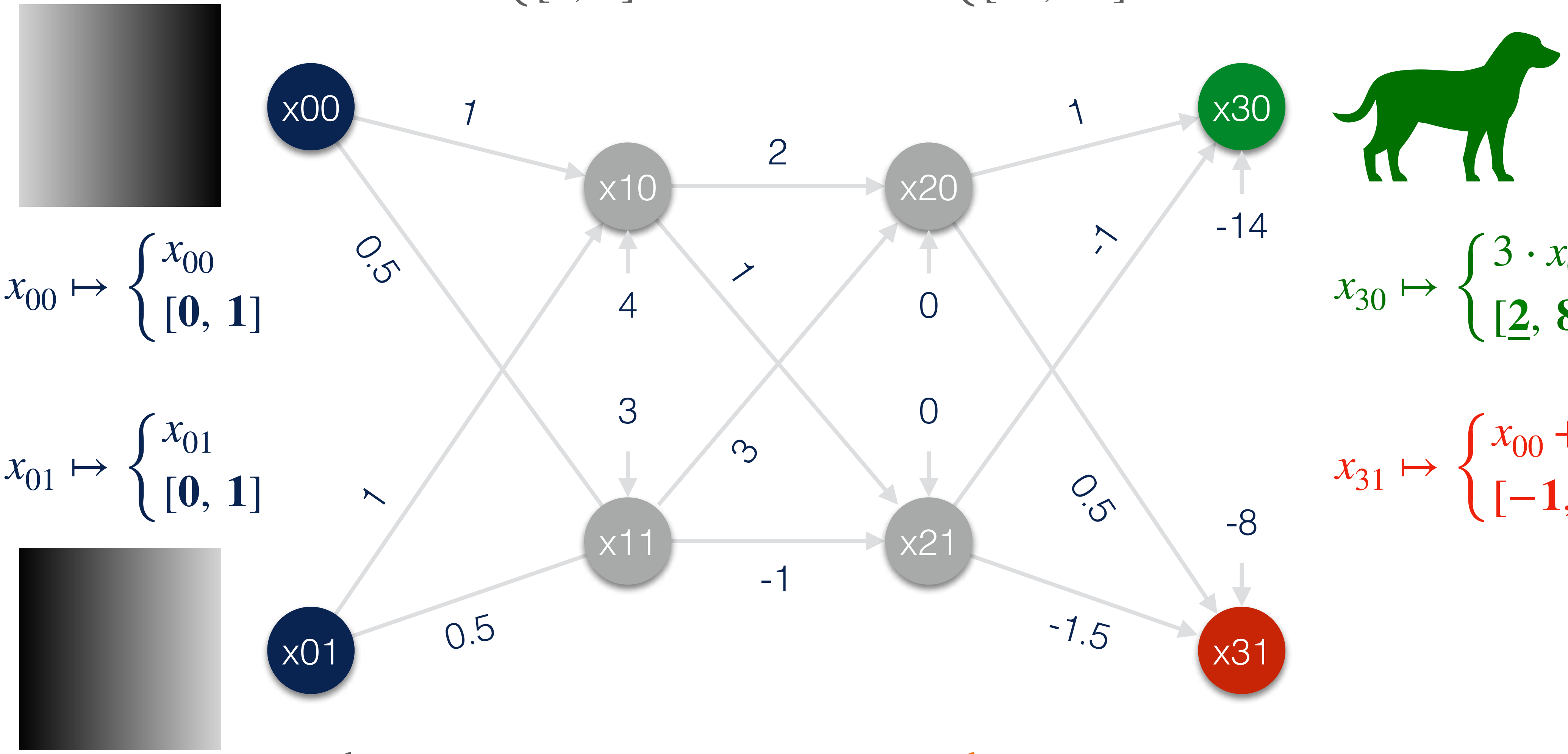
J. Li et al. - Analyzing Deep Neural Networks with Symbolic Propagation (SAS 2019)

Interval Domain

with Symbolic Constant Propagation [Li19]

$x_{10} \mapsto \begin{cases} x_{00} + x_{01} + 4 \\ [\mathbf{4},\ \mathbf{6}] \end{cases}$

$x_{20} \mapsto \begin{cases} 2 \cdot (x_{00} + x_{01} + 4) + 3 \cdot (0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3) \\ [\mathbf{17},\ \mathbf{24}] \end{cases}$



$x_{00} \mapsto \begin{cases} x_{00} \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$

$x_{01} \mapsto \begin{cases} x_{01} \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$

$x_{30} \mapsto \begin{cases} 3 \cdot x_{00} + 3 \cdot x_{01} + 2 \\ [\underline{\mathbf{2}},\ \mathbf{8}] \end{cases}$

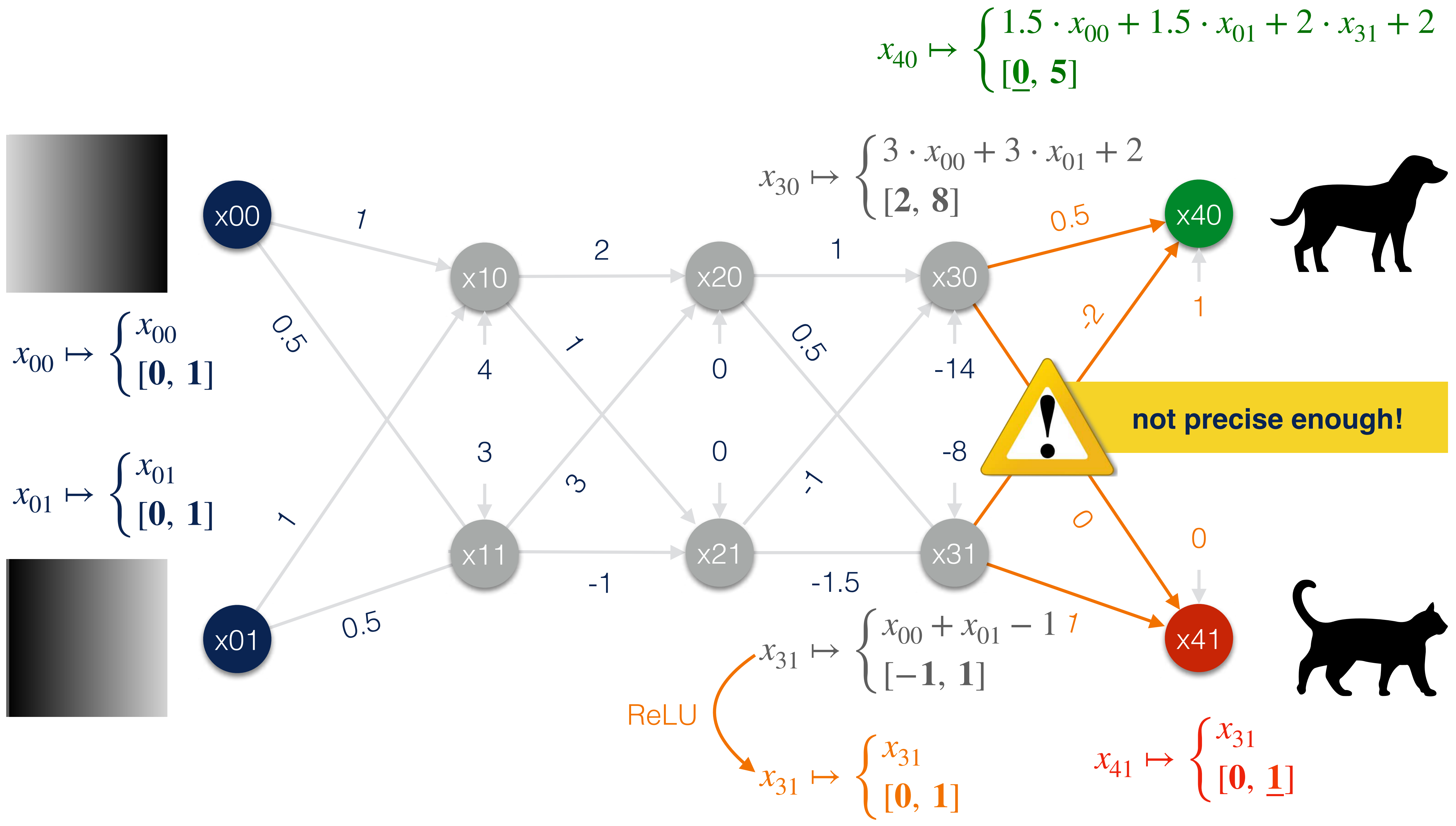
$x_{31} \mapsto \begin{cases} x_{00} + x_{01} - 1 \\ [\mathbf{-1},\ \underline{\mathbf{1}}] \end{cases}$

$x_{11} \mapsto \begin{cases} 0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3 \\ [\mathbf{3},\ \mathbf{4}] \end{cases}$

$x_{21} \mapsto \begin{cases} (x_{00} + x_{01} + 4) - 1 \cdot (0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3) \\ [\mathbf{1},\ \mathbf{2}] \end{cases}$

Interval Domain

with Symbolic Constant Propagation [Li19]



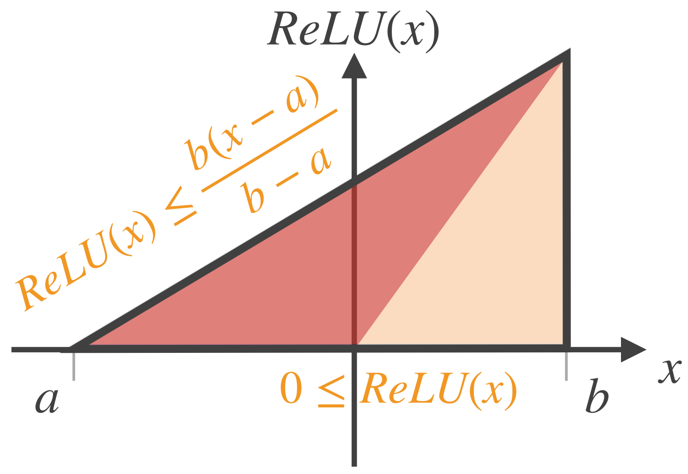
$x_{40} \mapsto \begin{cases} 1.5 \cdot x_{00} + 1.5 \cdot x_{01} + 2 \cdot x_{31} + 2 \\ [\underline{0}, 5] \end{cases}$
$x_{30} \mapsto \begin{cases} 3 \cdot x_{00} + 3 \cdot x_{01} + 2 \\ [2, 8] \end{cases}$
$x_{00} \mapsto \begin{cases} x_{00} \\ [0, 1] \end{cases}$
$x_{01} \mapsto \begin{cases} x_{01} \\ [0, 1] \end{cases}$
x00
x01
x10
x11
x20
x21
x30
x31
x40
x41
1
0.5
1
0.5
2
4
1
3
3
-1
1
0
0.5
0
-1
-1.5
-14
-8
0.5
-2
1
0
0
not precise enough!
$x_{31} \mapsto \begin{cases} x_{00} + x_{01} - 1 \\ [-1, 1] \end{cases}$
ReLU
$x_{31} \mapsto \begin{cases} x_{31} \\ [0, 1] \end{cases}$
$x_{41} \mapsto \begin{cases} x_{31} \\ [0, \underline{1}] \end{cases}$

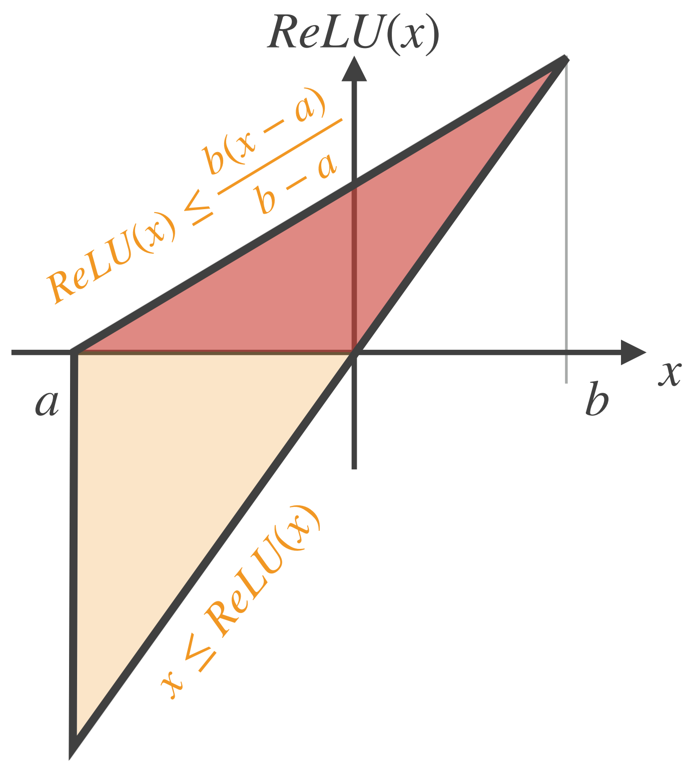
DeepPoly [Singh19]

maintain symbolic lower- and upper-bounds for each neuron + convex ReLU approximations

$$x_{i+1,j} \mapsto \begin{cases} \left[\sum_k c_{i,k}\cdot x_{i,k} + c,\ \sum_k d_{i,k}\cdot x_{i,k} + d\right] & c_{i,k}, c, d_{i,k}, d \in \mathscr{R} \\ [a,\ b] & a, b \in \mathscr{R} \end{cases}$$

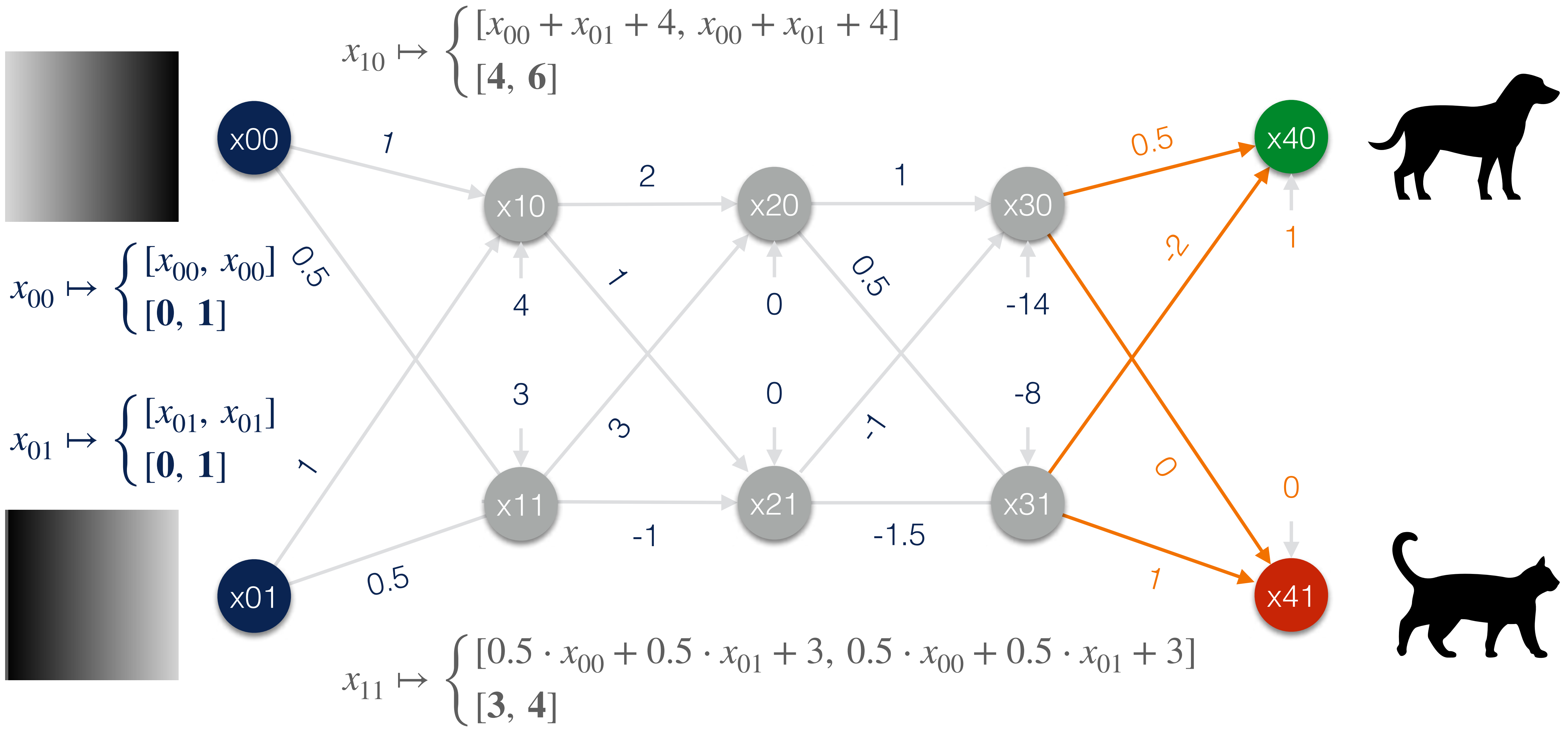
$$\begin{tikzcd}
 & x_{i,j} \mapsto \begin{cases} [\mathbf{L_{i,j}}, \mathbf{U_{i,j}}] \\ [\mathbf{a}, \mathbf{b}] \end{cases} \\
 & x_{i,j} \mapsto \begin{cases} \left[\mathbf{0}, \dfrac{\mathbf{b(x_{i,j}-a)}}{\mathbf{b-a}}\right] \\ [\mathbf{0}, \mathbf{b}] \end{cases} \\
x_{i,j} \mapsto \begin{cases} [\mathbf{L_{i,j}}, \mathbf{U_{i,j}}] \\ [\mathbf{a}, \mathbf{b}] \end{cases} \arrow[ruu, "\text{ReLU}", "0 \le a"'] \arrow[ru, "b \le -a"] \arrow[rd, "-a < b"'] \arrow[rdd, "\text{ReLU}", "b \le 0"'] \arrow[r, phantom, "\text{ReLU}\ a < 0 \wedge 0 < b"] & {} \\
 & x_{i,j} \mapsto \begin{cases} \left[\mathbf{x_{i,j}}, \dfrac{\mathbf{b(x_{i,j}-a)}}{\mathbf{b-a}}\right] \\ [\mathbf{a}, \mathbf{b}] \end{cases} \\
 & x_{i,j} \mapsto \begin{cases} [\mathbf{0}, \mathbf{0}] \\ [\mathbf{0}, \mathbf{0}] \end{cases}
\end{tikzcd}$$





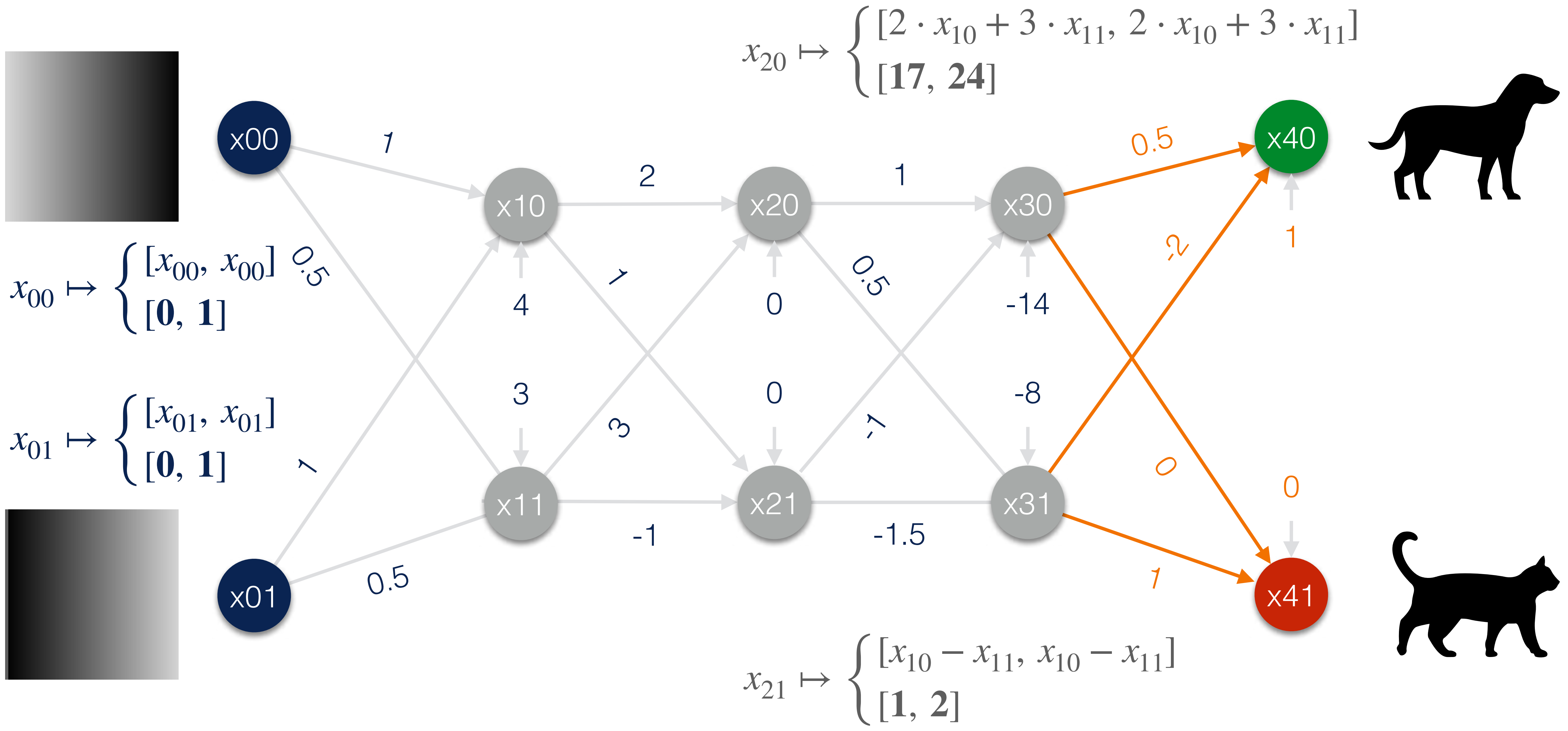
G. Singh, T. Gehr, M. Püschel, and M. Vechev - An Abstract Domain for Certifying Neural Networks (POPL 2019)

DeepPoly [Singh19]

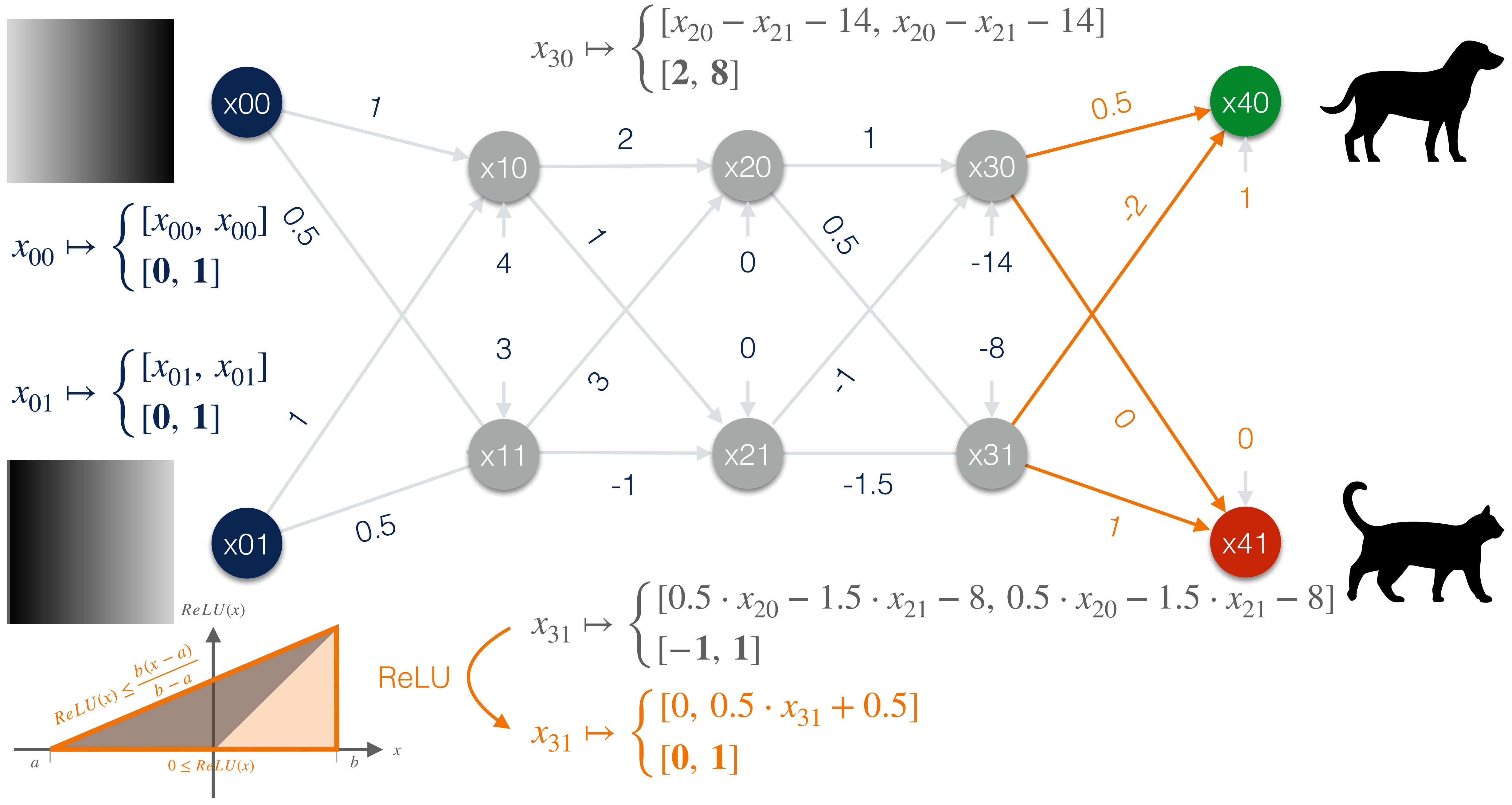


$x_{10} \mapsto \begin{cases}[x_{00} + x_{01} + 4,\ x_{00} + x_{01} + 4] \\ [\mathbf{4},\ \mathbf{6}]\end{cases}$
$x_{00} \mapsto \begin{cases}[x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}]\end{cases}$
$x_{01} \mapsto \begin{cases}[x_{01},\ x_{01}] \\ [\mathbf{0},\ \mathbf{1}]\end{cases}$
$x_{11} \mapsto \begin{cases}[0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3,\ 0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3] \\ [\mathbf{3},\ \mathbf{4}]\end{cases}$
x00
x01
x10
x11
x20
x21
x30
x31
x40
x41
1
0.5
1
0.5
2
1
3
-1
4
3
0
0
1
0.5
-1
-1.5
-14
-8
0.5
2
0
1
1
0

DeepPoly [Singh19]


$x_{20} \mapsto \begin{cases}[2 \cdot x_{10} + 3 \cdot x_{11},\ 2 \cdot x_{10} + 3 \cdot x_{11}] \\ [\mathbf{17},\ \mathbf{24}]\end{cases}$
$x_{00} \mapsto \begin{cases}[x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}]\end{cases}$
$x_{01} \mapsto \begin{cases}[x_{01},\ x_{01}] \\ [\mathbf{0},\ \mathbf{1}]\end{cases}$
$x_{21} \mapsto \begin{cases}[x_{10} - x_{11},\ x_{10} - x_{11}] \\ [\mathbf{1},\ \mathbf{2}]\end{cases}$
x00
x01
x10
x11
x20
x21
x30
x31
x40
x41
1
0.5
1
0.5
4
3
2
1
3
-1
0
0
1
0.5
-1
-1.5
-14
-8
0.5
-2
1
0
1
0

DeepPoly [Singh19]



$$x_{30} \mapsto \begin{cases} [x_{20} - x_{21} - 14,\ x_{20} - x_{21} - 14] \\ [\mathbf{2},\ \mathbf{8}] \end{cases}$$

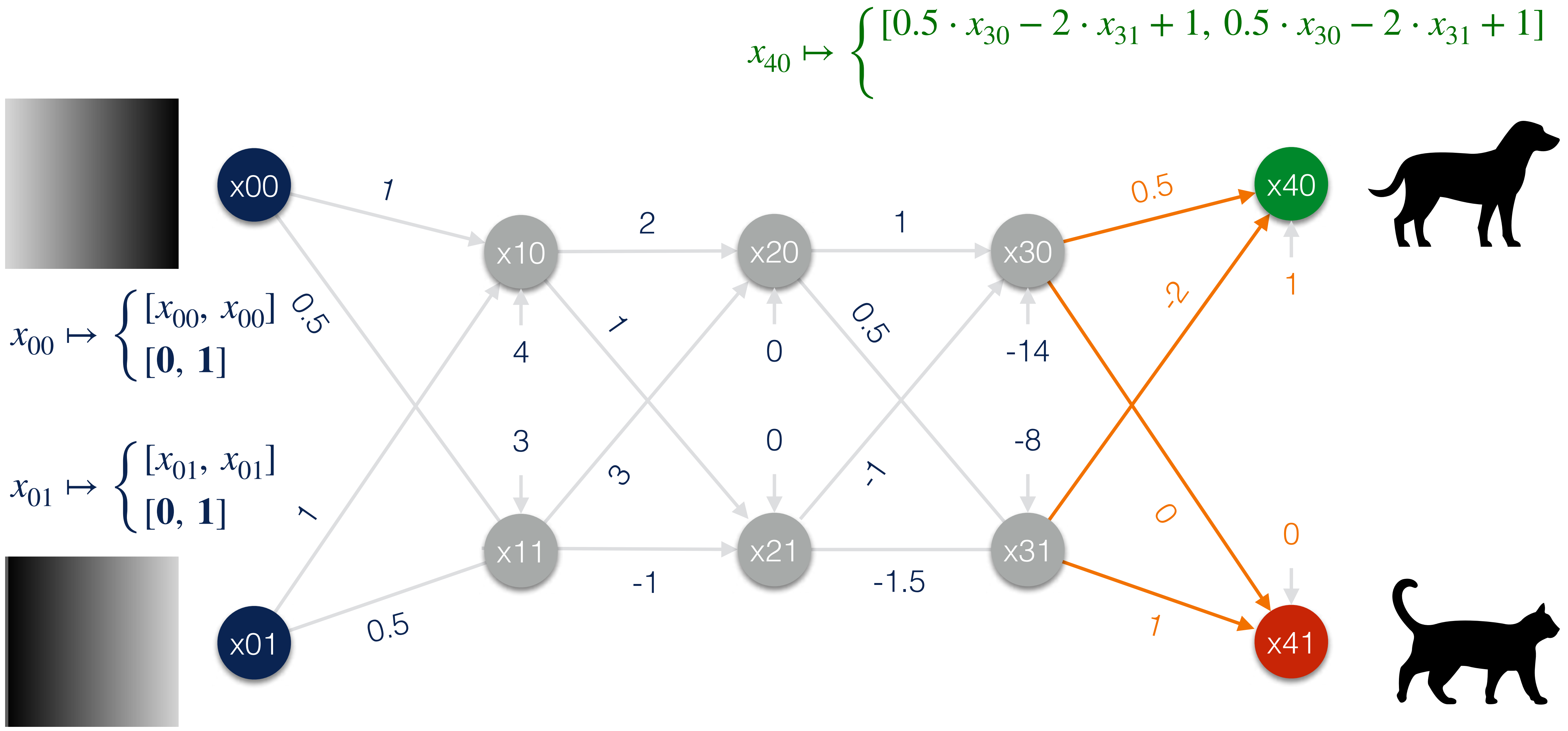
$$x_{00} \mapsto \begin{cases} [x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$$

$$x_{01} \mapsto \begin{cases} [x_{01},\ x_{01}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$$

$$x_{31} \mapsto \begin{cases} [0.5 \cdot x_{20} - 1.5 \cdot x_{21} - 8,\ 0.5 \cdot x_{20} - 1.5 \cdot x_{21} - 8] \\ [\mathbf{-1},\ \mathbf{1}] \end{cases}$$

$$x_{31} \mapsto \begin{cases} [0,\ 0.5 \cdot x_{31} + 0.5] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$$

DeepPoly [Singh19]


$x_{40} \mapsto \begin{cases} [0.5 \cdot x_{30} - 2 \cdot x_{31} + 1,\ 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1] \end{cases}$
$x_{00} \mapsto \begin{cases} [x_{00}, x_{00}] \\ [\mathbf{0}, \mathbf{1}] \end{cases}$
$x_{01} \mapsto \begin{cases} [x_{01}, x_{01}] \\ [\mathbf{0}, \mathbf{1}] \end{cases}$
x00
x01
x10
x11
x20
x21
x30
x31
x40
x41
1
0.5
1
0.5
4
3
2
1
3
-1
0
0
1
0.5
-1
-1.5
-14
-8
0.5
-2
0
1
1
0

DeepPoly [Singh19]

$x_{00} \mapsto \begin{cases} [x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$ $\quad x_{01} \mapsto \begin{cases} [x_{01},\ x_{01}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$

$x_{10} \mapsto \begin{cases} [x_{00} + x_{01} + 4,\ x_{00} + x_{01} + 4] \\ [\mathbf{4},\ \mathbf{6}] \end{cases}$ $\quad x_{11} \mapsto \begin{cases} [0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3,\ 0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3] \\ [\mathbf{3},\ \mathbf{4}] \end{cases}$

$x_{20} \mapsto \begin{cases} [2 \cdot x_{10} + 3 \cdot x_{11},\ 2 \cdot x_{10} + 3 \cdot x_{11}] \\ [\mathbf{17},\ \mathbf{24}] \end{cases}$ $\quad x_{21} \mapsto \begin{cases} [x_{10} - x_{11},\ x_{10} - x_{11}] \\ [\mathbf{1},\ \mathbf{2}] \end{cases}$

$x_{30} \mapsto \begin{cases} [x_{20} - x_{21} - 14,\ x_{20} - x_{21} - 14] \\ [\mathbf{2},\ \mathbf{8}] \end{cases}$ $\quad x_{31} \mapsto \begin{cases} [0,\ 0.5 \cdot (0.5 \cdot x_{20} - 1.5 \cdot x_{21} - 8) + 0.5] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$

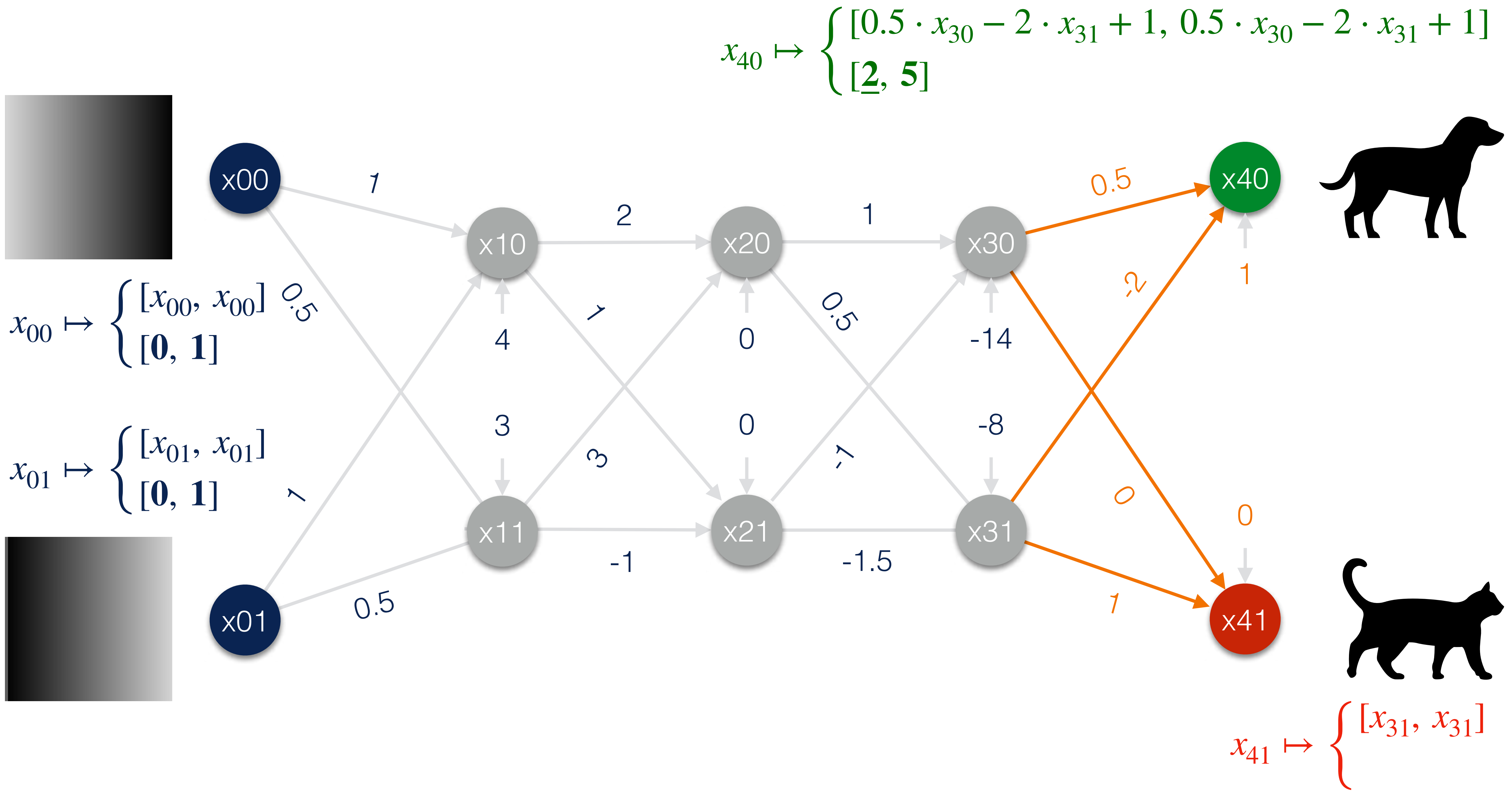
$x_{40} \mapsto \begin{cases} [0.5 \cdot x_{30} - 2 \cdot x_{31} + 1,\ 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1] \\ \end{cases}$

$\mapsto \begin{cases} [x_{21} + 1,\ 0.5 \cdot x_{20} - 0.5 \cdot x_{21} - 6] \\ \end{cases}$

$\mapsto \begin{cases} [x_{10} - x_{11} + 1,\ 0.5 \cdot x_{10} + 2 \cdot x_{11} - 6] \\ \end{cases}$

$\mapsto \begin{cases} [0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 2,\ 1.5 \cdot x_{00} + 1.5 \cdot x_{11} + 2] \\ [\underline{\mathbf{2}},\ \mathbf{5}] \end{cases}$

DeepPoly [Singh19]



$x_{40} \mapsto \begin{cases} [0.5 \cdot x_{30} - 2 \cdot x_{31} + 1,\ 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1] \\ [\underline{\mathbf{2}},\ \mathbf{5}] \end{cases}$
$x_{00} \mapsto \begin{cases} [x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$
$x_{01} \mapsto \begin{cases} [x_{01},\ x_{01}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$
$x_{41} \mapsto \begin{cases} [x_{31},\ x_{31}] \end{cases}$
x00
x01
x10
x11
x20
x21
x30
x31
x40
x41
1
0.5
1
0.5
4
3
2
1
3
-1
0
0
1
0.5
-1
-1.5
-14
-8
0.5
-2
0
1
1
0

DeepPoly [Singh19]

$x_{00} \mapsto \begin{cases} [x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$ $x_{01} \mapsto \begin{cases} [x_{01},\ x_{01}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$

$x_{10} \mapsto \begin{cases} [x_{00} + x_{01} + 4,\ x_{00} + x_{01} + 4] \\ [\mathbf{4},\ \mathbf{6}] \end{cases}$ $x_{11} \mapsto \begin{cases} [0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3,\ 0.5 \cdot x_{00} + 0.5 \cdot x_{01} + 3] \\ [\mathbf{3},\ \mathbf{4}] \end{cases}$

$x_{20} \mapsto \begin{cases} [2 \cdot x_{10} + 3 \cdot x_{11},\ 2 \cdot x_{10} + 3 \cdot x_{11}] \\ [\mathbf{17},\ \mathbf{24}] \end{cases}$ $x_{21} \mapsto \begin{cases} [x_{10} - x_{11},\ x_{10} - x_{11}] \\ [\mathbf{1},\ \mathbf{2}] \end{cases}$

$x_{30} \mapsto \begin{cases} [x_{20} - x_{21} - 14,\ x_{20} - x_{21} - 14] \\ [\mathbf{2},\ \mathbf{8}] \end{cases}$ $x_{31} \mapsto \begin{cases} [0,\ 0.5 \cdot (0.5 \cdot x_{20} - 1.5 \cdot x_{21} - 8) + 0.5] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$

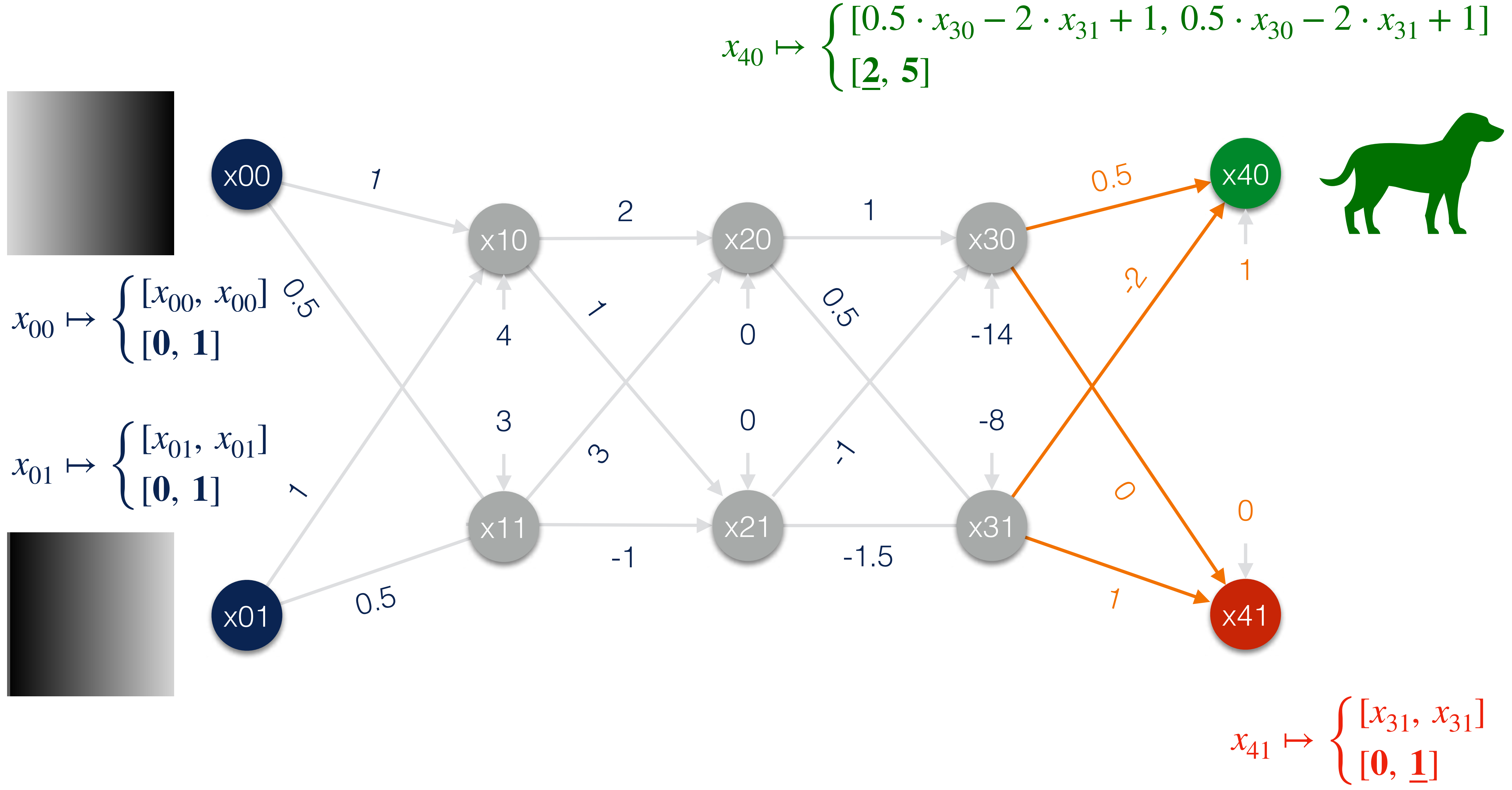
$$x_{41} \mapsto \begin{cases} [x_{31},\ x_{31}] \\ \end{cases}$$

$$\mapsto \begin{cases} [0,\ 0.25 \cdot x_{20} - 0.75 \cdot x_{21} - 3.5] \\ \end{cases}$$

$$\mapsto \begin{cases} [0,\ -0.25 \cdot x_{10} + 1.5 \cdot x_{11} - 3.5] \\ \end{cases}$$

$$\mapsto \begin{cases} [0,\ 0.5 \cdot x_{00} + 0.5 \cdot x_{01}] \\ [\mathbf{0},\ \underline{\mathbf{1}}] \end{cases}$$

DeepPoly [Singh19]



$x_{40} \mapsto \begin{cases} [0.5 \cdot x_{30} - 2 \cdot x_{31} + 1,\ 0.5 \cdot x_{30} - 2 \cdot x_{31} + 1] \\ [\underline{\mathbf{2}},\ \mathbf{5}] \end{cases}$
$x_{00} \mapsto \begin{cases} [x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$
$x_{01} \mapsto \begin{cases} [x_{01},\ x_{01}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$
$x_{41} \mapsto \begin{cases} [x_{31},\ x_{31}] \\ [\mathbf{0},\ \underline{\mathbf{1}}] \end{cases}$
x00
x01
x10
x11
x20
x21
x30
x31
x40
x41
1
0.5
1
0.5
4
3
2
1
3
-1
0
0
1
0.5
-1
-1.5
-14
-8
0.5
-2
0
1
1
0

Other Static Analysis Methods

- **T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev**. *AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation*. In S&P, 2018.
  the first use of abstract interpretation for verifying neural networks

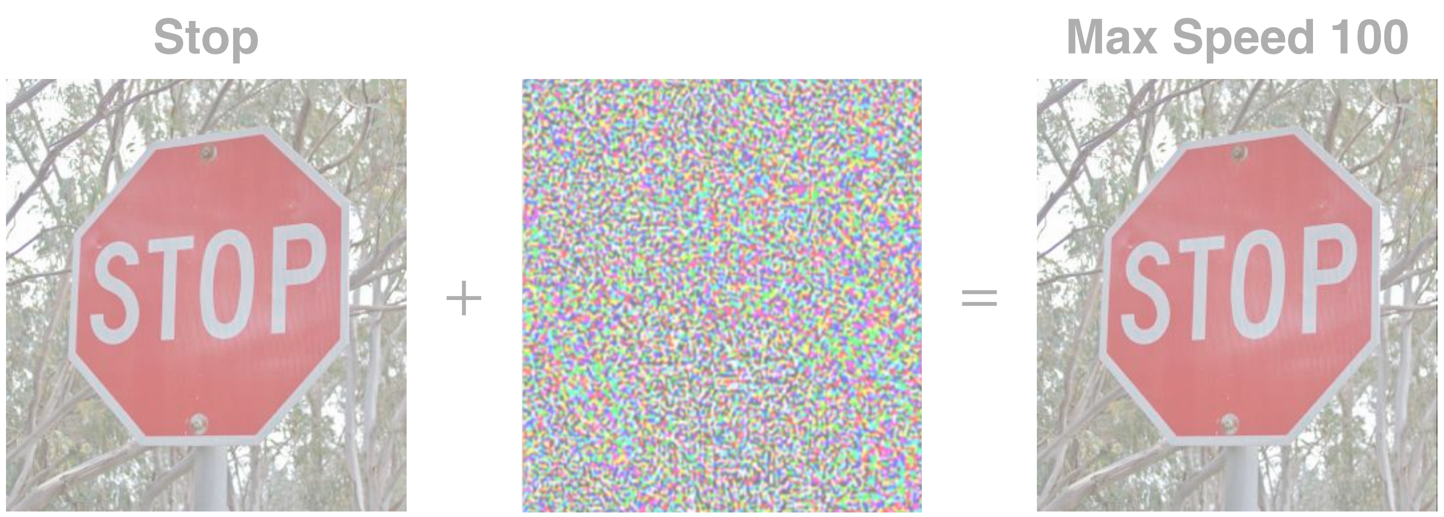
- **G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev**. *Fast and Effective Robustness Certification*. In NeurIPS, 2018.
  a custom zonotope domain for certifying neural networks

- **G. Singh, R. Ganvir, M. Püschel, and M. Vechev**. *Beyond the Single Neuron Convex Barrier for Neural Network Certification*. In NeurIPS, 2019.
  a framework to jointly approximate k ReLU activations

- **M. N. Müller, G. Makarchuk, G. Singh, M. Püschel, and M. Vechev**. *PRIMA: General and Precise Neural Network Certification via Scalable Convex Hull Approximations*. In POPL, 2022.
  a multi-neuron abstraction via a convex-hull approximation algorithm

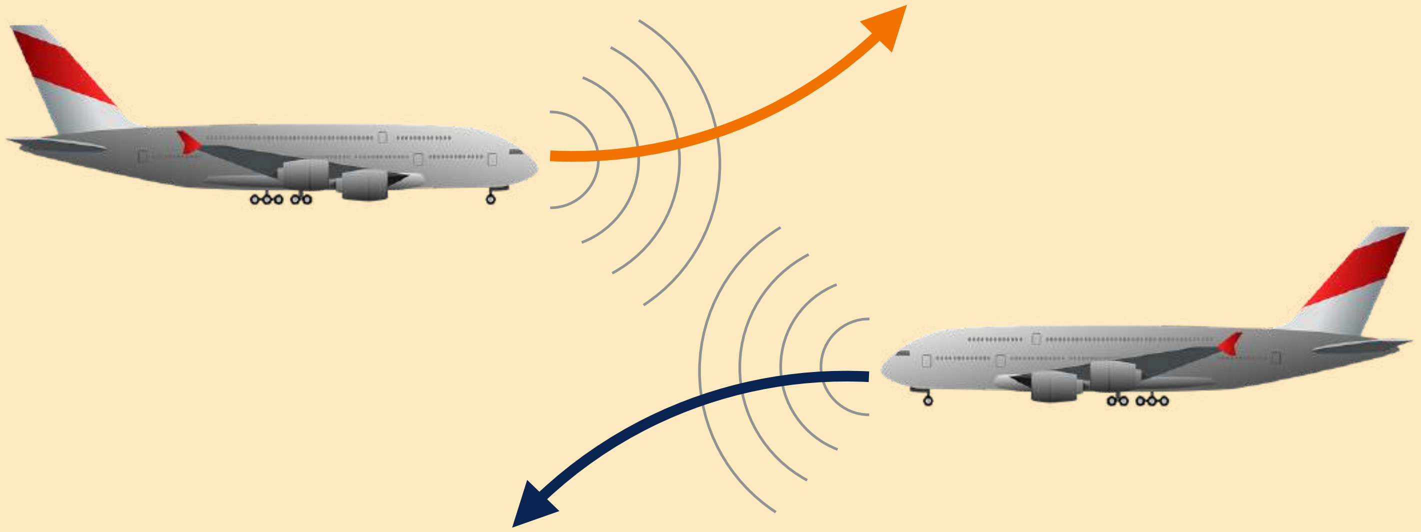
Stability

Goal G3 in [Kurd03]

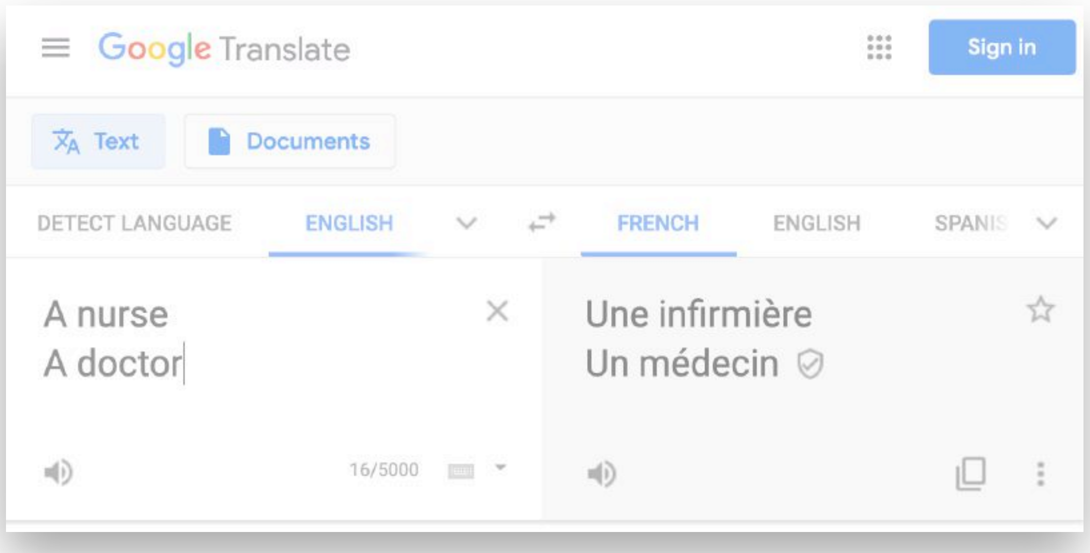


Safety

Goal G4 in [Kurd03]



Fairness



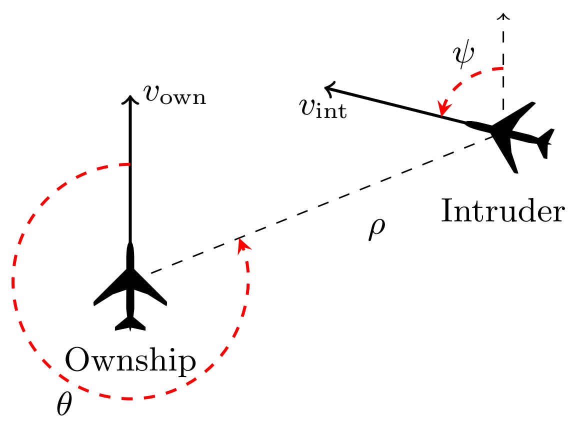
A nurse
A doctor

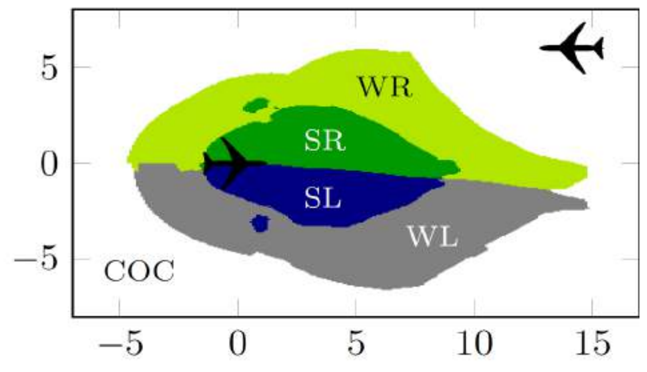
Une infirmière
Un médecin

ACAS Xu [Julian16][Katz17]

Airborne Collision Avoidance System for Unmanned Aircraft

implemented using **45 feed-forward fully-connected ReLU networks**





5 input **sensor measurements**

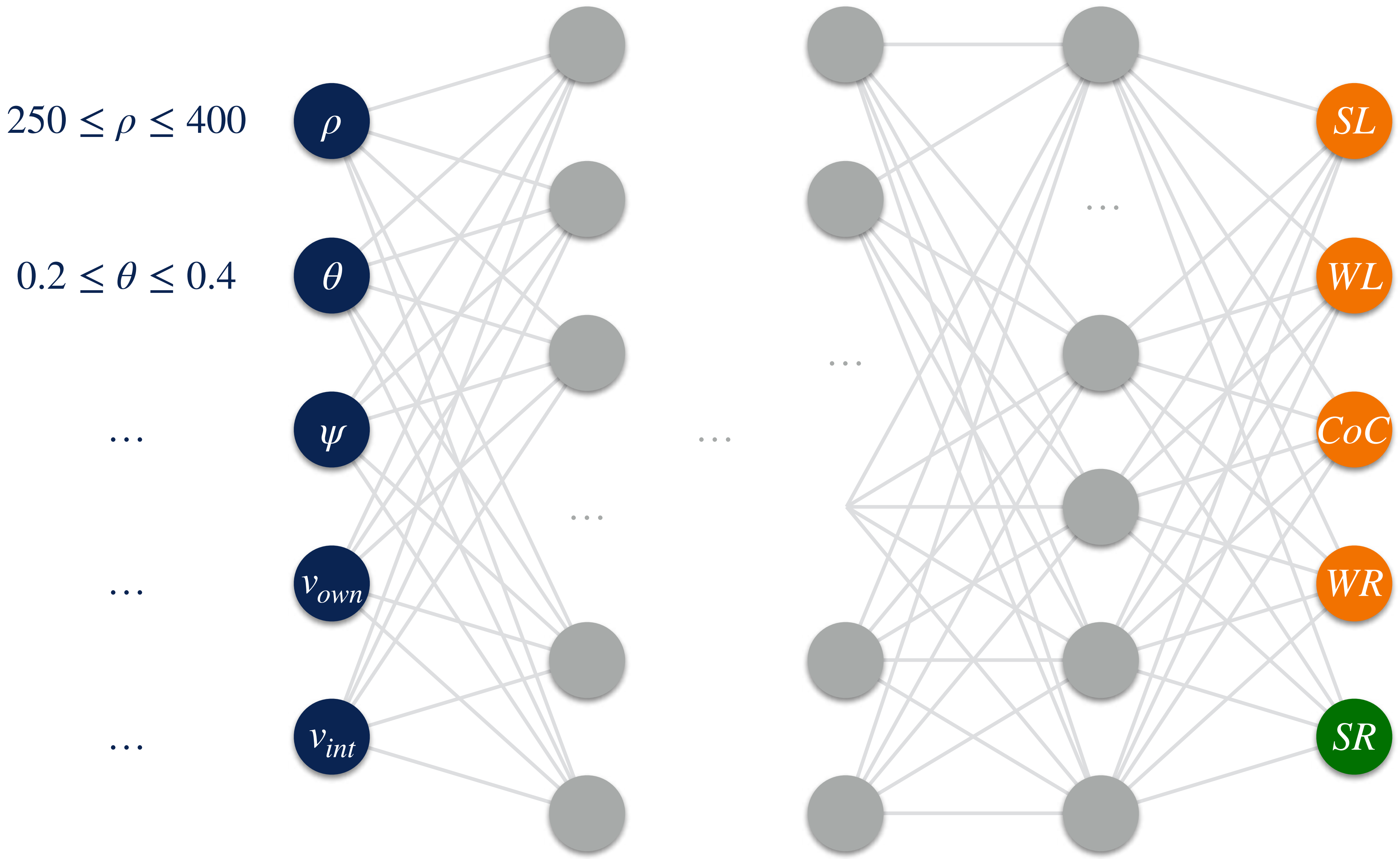
- ρ: distance from ownship to intruder
- θ: angle to intruder relative to ownship heading direction
- ψ: heading angle to intruder relative to ownship heading direction
- v_{own}: speed of ownship
- v_{int}: speed of intruder

5 output **horizontal advisories**

- Strong Left
- Weak Left
- Clear of Conflict
- Weak Right
- Strong Right

ACAS Xu Properties [Katz17]

Example: “if intruder is **near** and approaching **from the left**, go **Strong Right**”



Safety

Input-Output Properties

$\mathbf{I}$: input specification

$\mathbf{O}$: output specification

$$\mathcal{S}^{\mathbf{I}}_{\mathbf{O}} \overset{\text{def}}{=} \{ [\![M]\!] \in \mathcal{P}(\Sigma^*) \mid \text{SAFE}^{\mathbf{I}}_{\mathbf{O}}([\![M]\!]) \}$$

$\mathcal{S}^{\mathbf{I}}_{\mathbf{O}}$ is the set of all neural networks M (or, rather, their semantics $[\![M]\!]$) that **satisfy** the input and output specification $\mathbf{I}$ and $\mathbf{O}$

$$\text{SAFE}^{\mathbf{I}}_{\mathbf{O}}([\![M]\!]) \overset{\text{def}}{=} \forall t \in [\![M]\!] : t_0 \models \mathbf{I} \Rightarrow t_\omega \models \mathbf{O}$$

Theorem

$$M \models \mathcal{S}^{\mathbf{I}}_{\mathbf{O}} \Leftrightarrow \{ [\![M]\!] \} \subseteq \mathcal{S}^{\mathbf{I}}_{\mathbf{O}}$$

Corollary

$$M \models \mathcal{S}^{\mathbf{I}}_{\mathbf{O}} \Leftrightarrow [\![M]\!] \subseteq \bigcup \mathcal{S}^{\mathbf{I}}_{\mathbf{O}}$$

Formal Methods

Mathematical Guarantees of Safety

Deductive Verification
- extremely **expressive**
- **relies on the user** to guide the proof

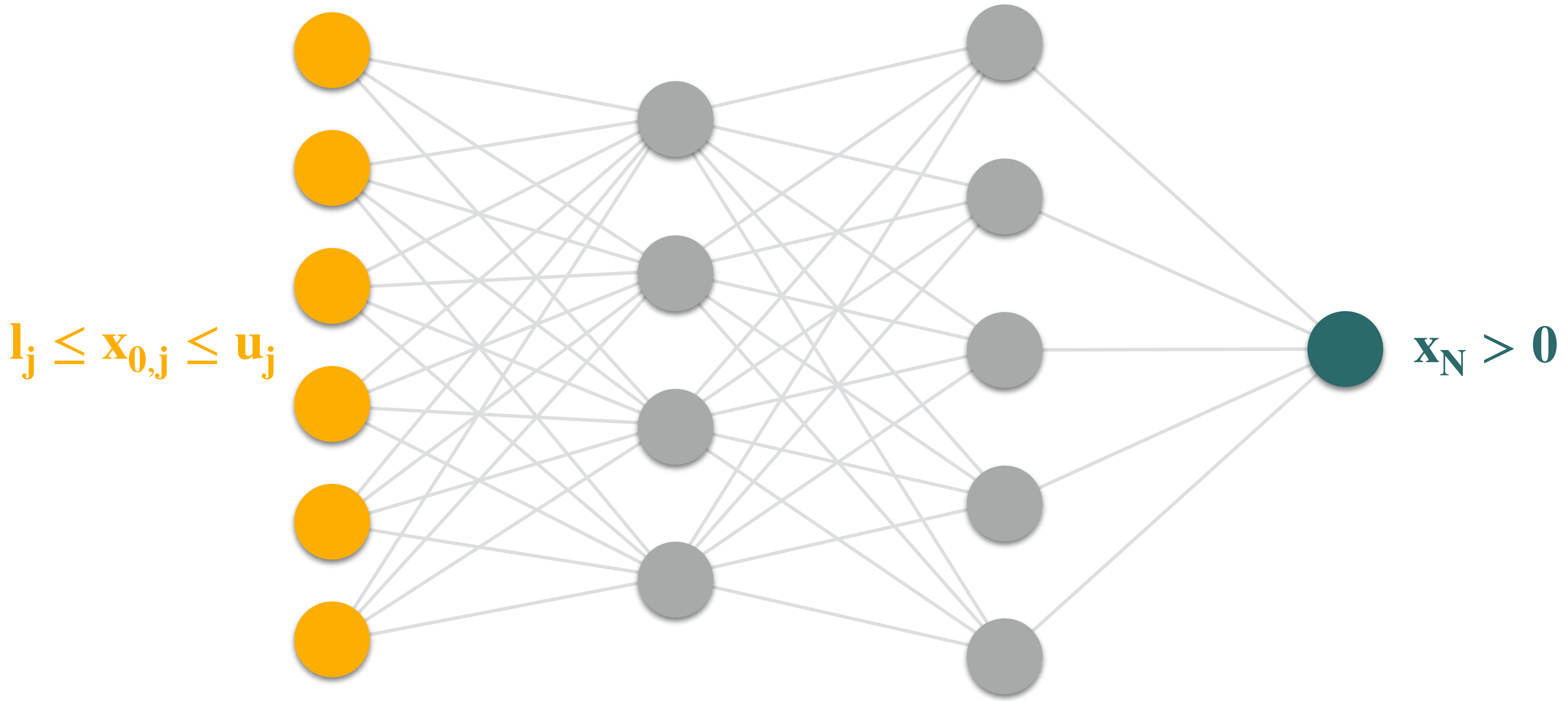
Model Checking
- analysis of a **model** of the software
- **sound and complete with respect to the model**

Static Analysis
- analysis of the software at some level of **abstraction**
- fully **automatic** and **sound** by construction
- generally **not complete**

Model Checking Methods

Safety

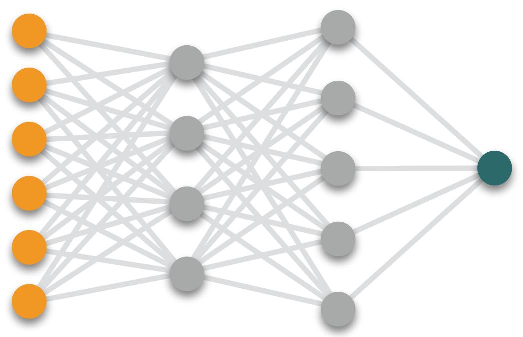
Example



SMT-Based Methods

Verification Reduced to Constraint Satisfiability

$\mathbf{l_j} \leq \mathbf{x_{0,j}} \leq \mathbf{u_j}$ $\quad j \in \{0, \ldots, |\mathbf{X}_0|\}$ **input specification**



$$\hat{x}_{i+1,j} = \sum_{k=0}^{|\mathbf{X}_i|} w^i_{j,k} \cdot x_{i,k} + b_{i,j} \qquad i \in \{0, \ldots, n-1\}$$

$$x_{i,j} = \max\{0, \hat{x}_{i,j}\} \qquad i \in \{1, \ldots, n-1\}, \; j \in \{0, \ldots, |\mathbf{X}_i|\}$$

$\mathbf{x_N} \leq \mathbf{0}$ (negation of) **output specification**

satisfiable → ✗ **counterexample**

otherwise → ✓ **safe**

Planet

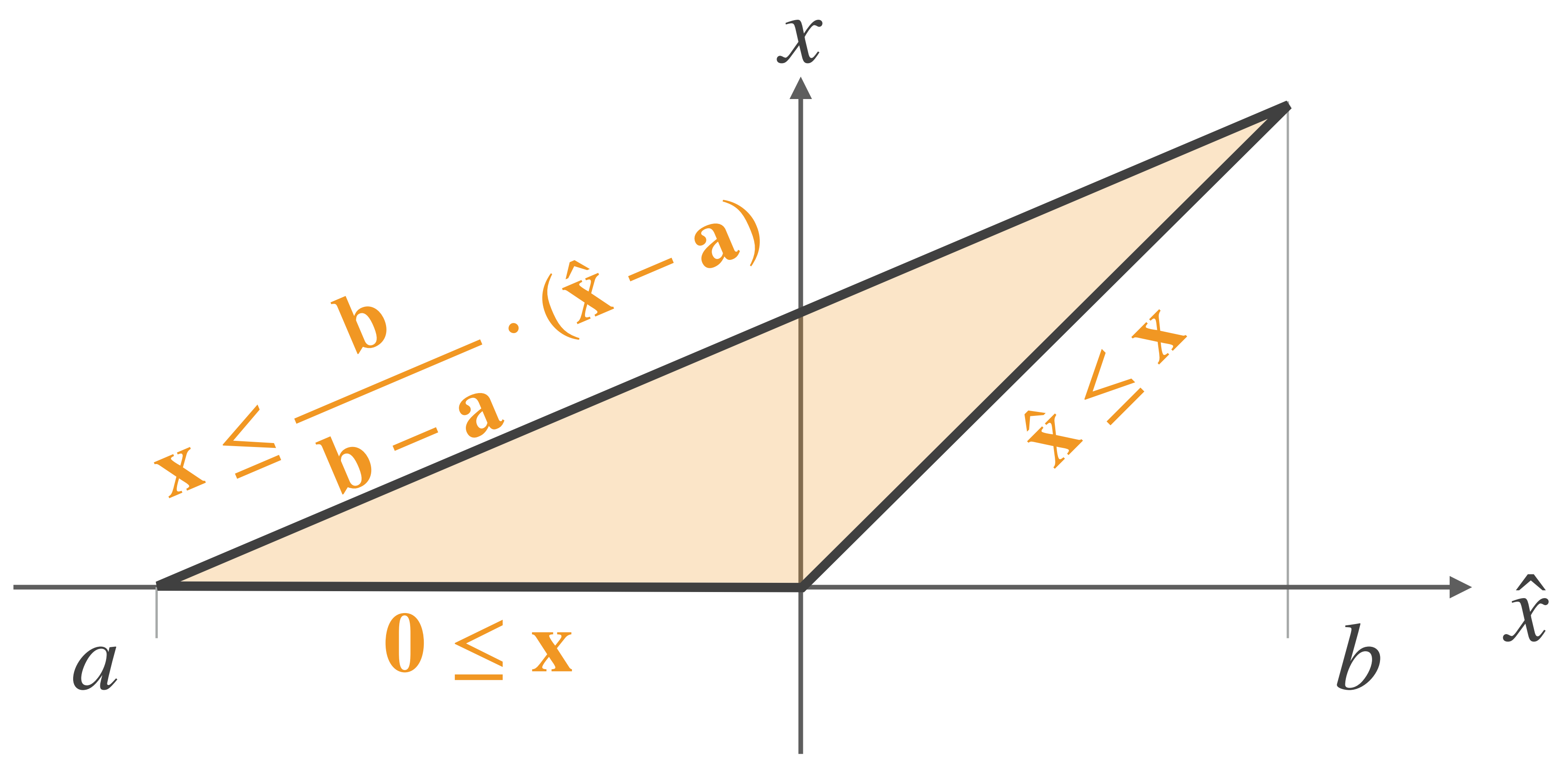
use **approximations** to reduce the solution search space

$$x_{i,j} = \max\{0, \hat{x}_{i,j}\}$$

$$\mathbf{0 \leq x_{i,j}}$$

$$\mathbf{\hat{x}_{i,j} \leq x_{i,j}}$$

$$\mathbf{x_{i,j} \leq \frac{b_{i,j}}{b_{i,j} - a_{i,j}} \cdot (\hat{x}_{i,j} - a_{i,j})}$$



x
$\mathbf{x \leq \frac{b}{b-a} \cdot (\hat{x} - a)}$
$\mathbf{\hat{x} \leq x}$
$\mathbf{0 \leq x}$
a
b
$\hat{x}$

R. Ehlers - Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks (ATVA 2017)

Reluplex

based on the **simplex algorithm** extended to support ReLUs

Variable	Value
$\mathbf{x}_{\mathbf{00}}$	v_{00}
$\dots$	$\dots$
$\hat{\mathbf{x}}_{\mathbf{ij}}$	$\hat{v}_{ij}$
$\mathbf{x}_{\mathbf{ij}}$	v_{ij}
$\dots$	$\dots$
$\mathbf{x}_{\mathbf{N}}$	v_N

Variable	Value
$\mathbf{x}_{\mathbf{00}}$	v_{00}
$\dots$	$\dots$
$\hat{\mathbf{x}}_{\mathbf{ij}}$	$\hat{v}'_{ij}$
$\mathbf{x}_{\mathbf{ij}}$	v_{ij}
$\dots$	$\dots$
$\mathbf{x}_{\mathbf{N}}$	v_N

Variable	Value
$\mathbf{x}_{\mathbf{00}}$	v_{00}
$\dots$	$\dots$
$\hat{\mathbf{x}}_{\mathbf{ij}}$	$\hat{v}'_{ij}$
$\mathbf{x}_{\mathbf{ij}}$	$\hat{v}'_{ij}$
$\dots$	$\dots$
$\mathbf{x}_{\mathbf{N}}$	v_N

Variable	Value
$\mathbf{x}_{\mathbf{00}}$	v_{00}
$\dots$	$\dots$
$\hat{\mathbf{x}}_{\mathbf{ij}}$	$\hat{v}'_{ij}$
$\mathbf{x}_{\mathbf{ij}}$	0
$\dots$	$\dots$
$\mathbf{x}_{\mathbf{N}}$	v_N

G. Katz et al. - Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks (CAV 2017)

Reluplex

based on th
extended

Follow-up Work

G. Katz et al. - The Marabou Framework for Verification and Analysis of Deep Neural Networks (CAV 2019)

Variable	Value
$\mathbf{x}_{00}$	v_{00}
$\dots$	$\dots$
$\hat{\mathbf{x}}_{\mathbf{ij}}$	$\hat{v}_{ij}$
$\mathbf{x}_{\mathbf{ij}}$	v_{ij}
$\dots$	$\dots$
$\mathbf{x}_{\mathbf{N}}$	v_N

Variable	Value
$\mathbf{x}_{00}$	v_{00}
$\dots$	$\dots$
$\hat{\mathbf{x}}_{\mathbf{ij}}$	$\hat{v}'_{ij}$
$\mathbf{x}_{\mathbf{ij}}$	v_{ij}
$\dots$	$\dots$
$\mathbf{x}_{\mathbf{N}}$	v_N

Variable	
$\mathbf{x}_{00}$	v_{00}
$\dots$	$\dots$
$\hat{\mathbf{x}}_{\mathbf{ij}}$	$\hat{v}'_{ij}$
$\mathbf{x}_{\mathbf{ij}}$	$\hat{v}'_{ij}$
$\dots$	$\dots$
$\mathbf{x}_{\mathbf{N}}$	v_N

Variable	Value
$\mathbf{x}_{00}$	v_{00}
$\dots$	$\dots$
$\hat{\mathbf{x}}_{\mathbf{ij}}$	$\hat{v}'_{ij}$
$\mathbf{x}_{\mathbf{ij}}$	0
$\dots$	$\dots$
$\mathbf{x}_{\mathbf{N}}$	v_N

G. Katz et al. - Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks (CAV 2017)

Other SMT-Based Methods

- **L. Pulina and A. Tacchella**. *An Abstraction-Refinement Approach to Verification of Artificial Neural Networks*. In CAV, 2010.
  the first formal verification method for neural networks

- **O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi**. *Measuring Neural Net Robustness with Constraints*. In NeurIPS, 2016.
  an approach for finding the nearest adversarial example according to the L∞ distance

- **X. Huang, M. Kwiatkowska, S. Wang, and M. Wu**. *Safety Verification of Deep Neural Networks*. In CAV, 2017.
  an approach for proving local robustness to adversarial perturbations

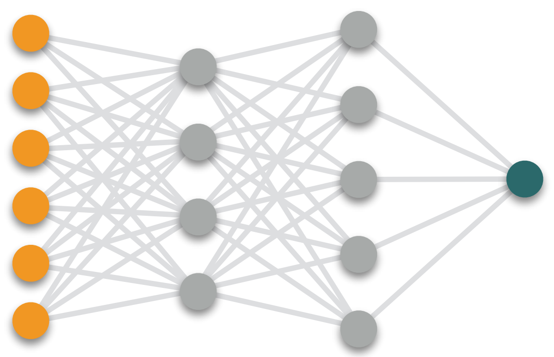
- **N. Narodytska, S. Kasiviswanathan, L. Ryzhyk, M. Sagiv, and T. Walsh**. *Verifying Properties of Binarized Deep Neural Networks*. In AAAI, 2018.
  C. H. Cheng, G. Nührenberg, C. H. Huang, and H. Ruess. *Verification of Binarized Neural Networks via Inter-Neuron Factoring*. In VSTTE, 2018.
  approaches focusing on binarized neural networks

MILP-Based Methods

Verification Reduced to Mixed Integer Linear Program

$\mathbf{l_j} \leq \mathbf{x_{0,j}} \leq \mathbf{u_j}$ $\quad j \in \{0, \ldots, |\mathbf{X}_0|\}$ **input specification**

$$\hat{x}_{i+1,j} = \sum_{k=0}^{|\mathbf{X}_i|} w^i_{j,k} \cdot x_{i,k} + b_{i,j} \qquad i \in \{0, \ldots, n-1\}$$



$$x_{i,j} = \delta_{\mathbf{i,j}} \cdot \hat{x}_{i,j} \qquad \delta_{\mathbf{i,j}} \in \{\mathbf{0}, \mathbf{1}\}$$

$$\delta_{\mathbf{i,j}} = 1 \Rightarrow \hat{x}_{i,j} \geq 0 \qquad i \in \{1, \ldots, n-1\}$$

$$\delta_{\mathbf{i,j}} = 0 \Rightarrow \hat{x}_{i,j} < 0 \qquad j \in \{0, \ldots, |\mathbf{X}_i|\}$$

min $\mathbf{x_N}$ **objective function**

min $\mathbf{x_N} \leq \mathbf{0} \rightarrow$ ✗ **counterexample**

otherwise $\rightarrow$ ✓ **safe**

MILP-Based Methods

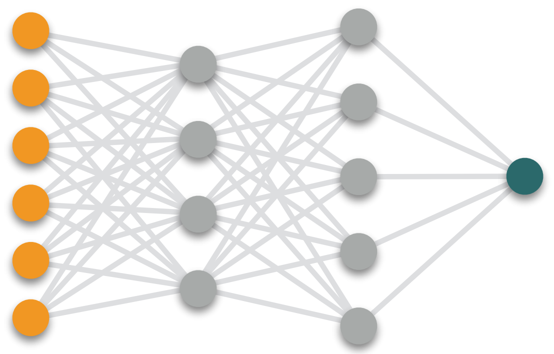
Bounded Encoding with Symmetric Bounds

$$\hat{x}_{i+1,j} = \sum_{k=0}^{|\mathbf{X}_i|} w^i_{j,k} \cdot x_{i,k} + b_{i,j} \qquad i \in \{0, \ldots, n-1\}$$

$$0 \leq x_{i,j} \leq \mathbf{M_{i,j}} \cdot \delta_{i,j} \qquad \delta_{\mathbf{i,j}} \in \{\mathbf{0}, \mathbf{1}\}$$

$$\hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - \mathbf{M_{i,j}} \cdot (1 - \delta_{i,j}) \qquad i \in \{1, \ldots, n-1\}$$

$$\mathbf{M_{i,j}} = \mathbf{max}\{-\mathbf{l_i}, \mathbf{u_i}\} \qquad j \in \{0, \ldots, |\mathbf{X}_i|\}$$



Sherlock

Output Range Analysis

use local search to speed up the MILP solver

$$\mathbf{l_j} \leq \mathbf{x_{0,j}} \leq \mathbf{u_j}$$

$$\hat{x}_{i+1,j} = \sum_{k=0}^{|\mathbf{X}_i|} w^i_{j,k} \cdot x_{i,k} + b_{i,j}$$

$$0 \leq x_{i,j} \leq \mathbf{M_{i,j}} \cdot \delta_{i,j}$$

$$\hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - \mathbf{M_{i,j}} \cdot (1 - \delta_{i,j})$$

$$\mathbf{M_{i,j}} = \max\{-\mathbf{l_i}, \mathbf{u_i}\}$$

$$\min\ \mathbf{x_N}$$

S. Dutta et al. - Output Range Analysis for Deep Feedforward Neural Networks (NFM 2018)

Sherlock

Output Range Analysis

use **local search** to speed up the MILP solver

$$\mathbf{l_j \leq x_{0,j} \leq u_j}$$

$$\hat{x}_{i+1,j} = \sum_{k=0}^{|\mathbf{X}_i|} w^i_{j,k} \cdot x_{i,k} + b_{i,j}$$

$$0 \leq x_{i,j} \leq \mathbf{M_{i,j}} \cdot \delta_{i,j}$$

$$\hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - \mathbf{M_{i,j}} \cdot (1 - \delta_{i,j})$$

$$\mathbf{M_{i,j}} = \mathbf{max\{-l_i, u_i\}}$$

$$\mathbf{x_N < L}$$

sample random input $\mathbf{X}$
and evaluate output $\mathbf{L}$

S. Dutta et al. - Output Range Analysis for Deep Feedforward Neural Networks (NFM 2018)

Sherlock

Output Range Analysis

use **local search** to speed up the MILP solver

$$\mathbf{l_j} \leq \mathbf{x_{0,j}} \leq \mathbf{u_j}$$

$$\hat{x}_{i+1,j} = \sum_{k=0}^{|\mathbf{X}_i|} w^i_{j,k} \cdot x_{i,k} + b_{i,j}$$

$$0 \leq x_{i,j} \leq \mathbf{M_{i,j}} \cdot \delta_{i,j}$$

$$\hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - \mathbf{M_{i,j}} \cdot (1 - \delta_{i,j})$$

$$\mathbf{M_{i,j}} = \mathbf{max}\{-\mathbf{l_i}, \mathbf{u_i}\}$$

$$\mathbf{x_N} < \mathbf{L}$$

find another input $\hat{\mathbf{X}}$ such that $\hat{\mathbf{L}} \leq \mathbf{x_N}$

S. Dutta et al. - Output Range Analysis for Deep Feedforward Neural Networks (NFM 2018)

Sherlock

Output Range Analysis

use **local search** to speed up the MILP solver

$$\mathbf{l_j} \leq \mathbf{x_{0,j}} \leq \mathbf{u_j}$$

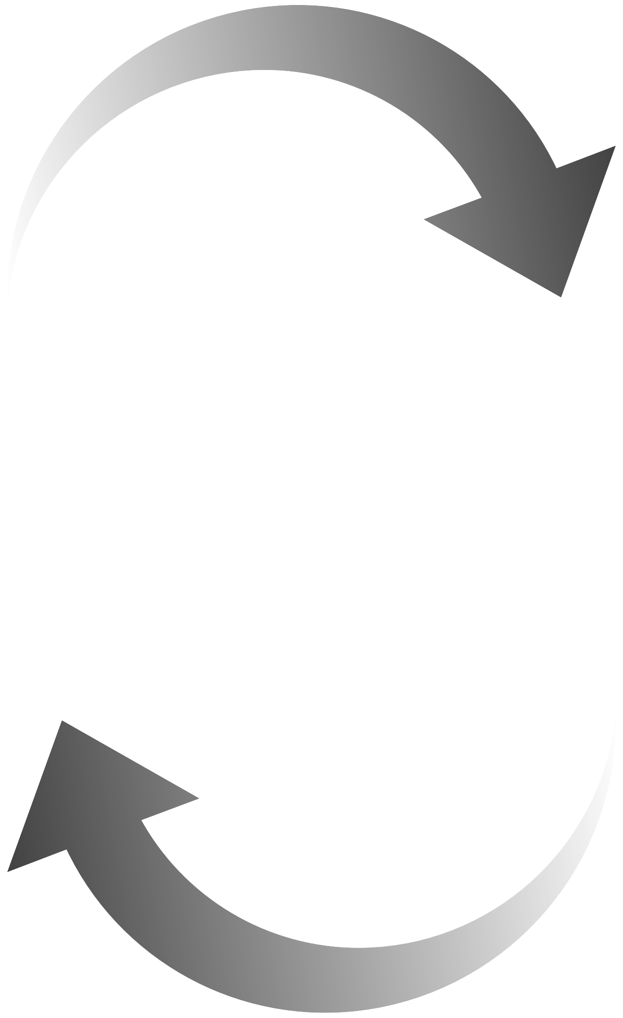
$$\hat{x}_{i+1,j} = \sum_{k=0}^{|\mathbf{X}_i|} w^i_{j,k} \cdot x_{i,k} + b_{i,j}$$

$$0 \leq x_{i,j} \leq \mathbf{M_{i,j}} \cdot \delta_{i,j}$$

$$\hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - \mathbf{M_{i,j}} \cdot (1 - \delta_{i,j})$$

$$\mathbf{M_{i,j}} = \mathbf{max}\{-\mathbf{l_i}, \mathbf{u_i}\}$$

$$\mathbf{x_N} < \hat{\mathbf{L}}$$



find another input $\hat{\mathbf{X}}$ such that $\hat{\mathbf{L}} \leq \mathbf{x_N}$

S. Dutta et al. - Output Range Analysis for Deep Feedforward Neural Networks (NFM 2018)

MILP-Based Methods

Bounded Encoding with Asymmetric Bounds

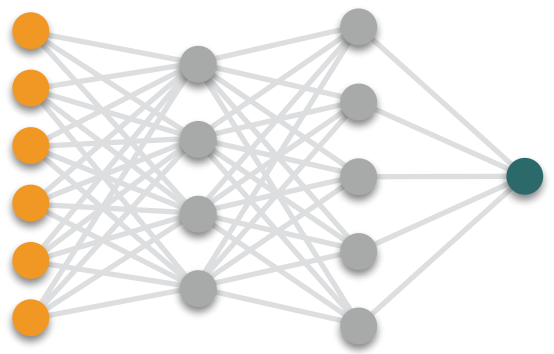
$$\hat{x}_{i+1,j} = \sum_{k=0}^{|\mathbf{X}_i|} w^i_{j,k} \cdot x_{i,k} + b_{i,j} \qquad i \in \{0,\ldots,n-1\}$$

$$\delta_{\mathbf{i,j}} \in \{\mathbf{0},\mathbf{1}\}$$

$$0 \leq x_{i,j} \leq \mathbf{u_{i,j}} \cdot \delta_{i,j}$$

$$\hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - \mathbf{l_{i,j}} \cdot (1 - \delta_{i,j}) \qquad i \in \{1,\ldots,n-1\}$$

$$j \in \{0,\ldots,|\mathbf{X}_i|\}$$



MIPVerify

Finding Nearest Adversarial Example

$\text{min}_{\mathbf{X}'}\ \mathbf{d}(\mathbf{X}, \mathbf{X}')$

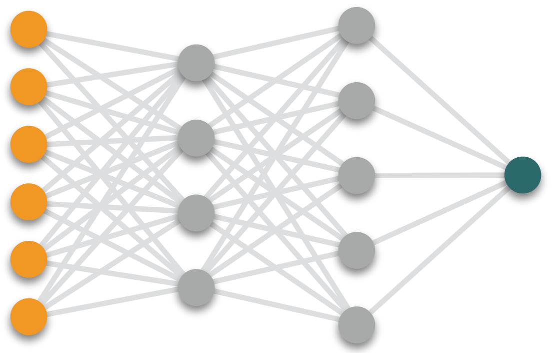
$$\hat{x}_{i+1,j} = \sum_{k=0}^{|\mathbf{X}_i|} w^i_{j,k} \cdot x_{i,k} + b_{i,j} \qquad i \in \{0,\ldots,n-1\}$$

$$\delta_{\mathbf{i,j}} \in \{\mathbf{0}, \mathbf{1}\}$$

$$0 \leq x_{i,j} \leq \mathbf{u_{i,j}} \cdot \delta_{i,j} \qquad i \in \{1,\ldots,n-1\}$$

$$\hat{x}_{i,j} \leq x_{i,j} \leq \hat{x}_{i,j} - \mathbf{l_{i,j}} \cdot (1 - \delta_{i,j}) \qquad j \in \{0,\ldots,|\mathbf{X}_i|\}$$

$$\mathbf{x_N} \neq \mathbf{O}$$



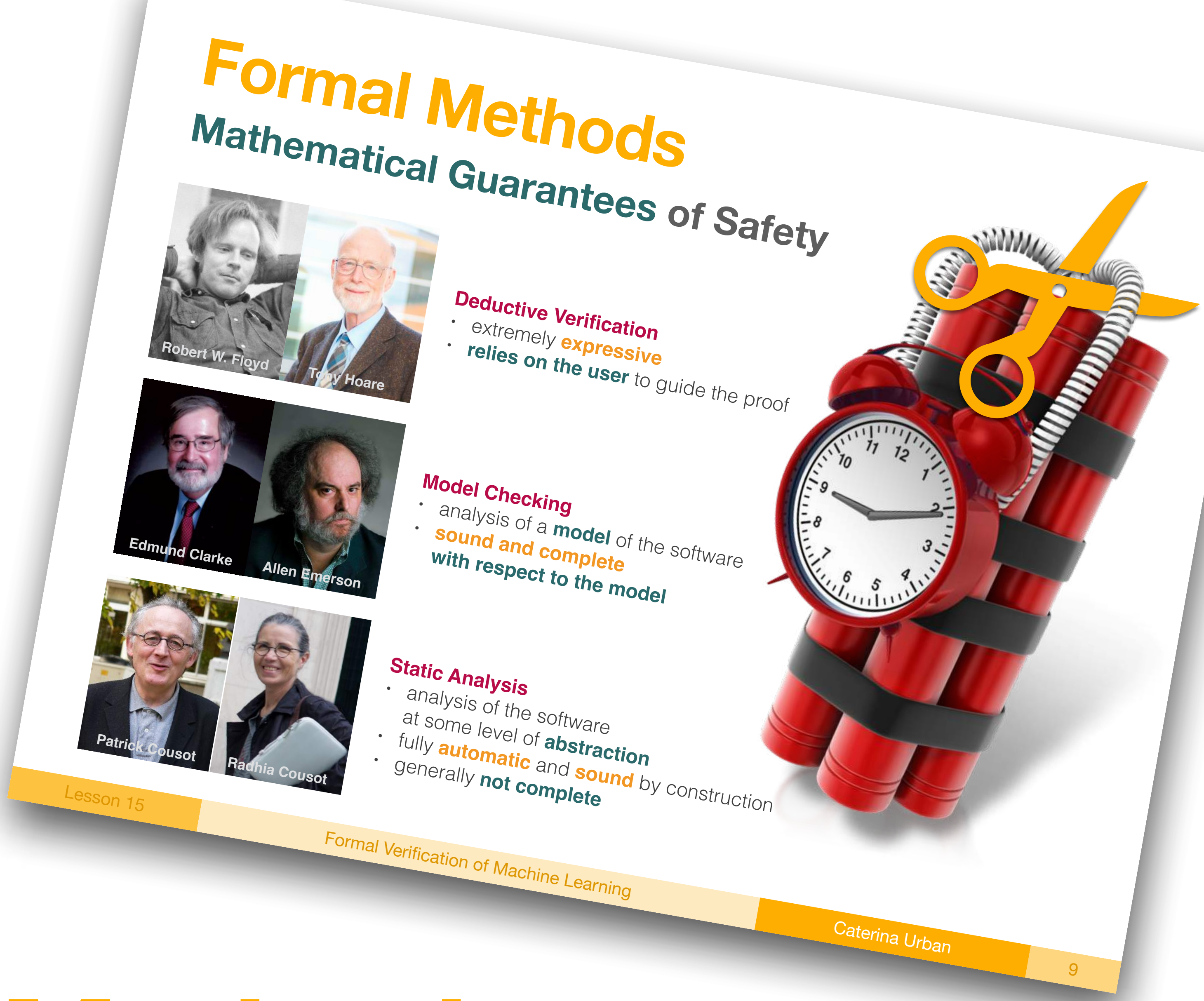
V. Tjeng et al. - Evaluating Robustness of Neural Networks with Mixed Integer Programming (ICLR 2019)

Other MILP-Based Methods

- **R. Bunel, I. Turkaslan, P. H. S. Torr, P. Kohli, and M. P. Kumar**. *A Unified View of Piecewise Linear Neural Network Verification*. In NeurIPS, 2018.
  a unifying verification framework for piecewise-linear ReLU neural networks

- **C.-H. Cheng, G. Nührenberg, and H. Ruess**. *Maximum Resilience of Artificial Neural Networks*. In ATVA, 2017.
  an approach for finding a lower bound on robustness to adversarial perturbations

- **M. Fischetti and J. Jo**. *Deep Neural Networks and Mixed Integer Linear Optimization*. 2018.
  an approach for feature visualization and building adversarial examples



Formal Methods

Mathematical Guarantees of Safety

Deductive Verification
- extremely **expressive**
- **relies on the user** to guide the proof

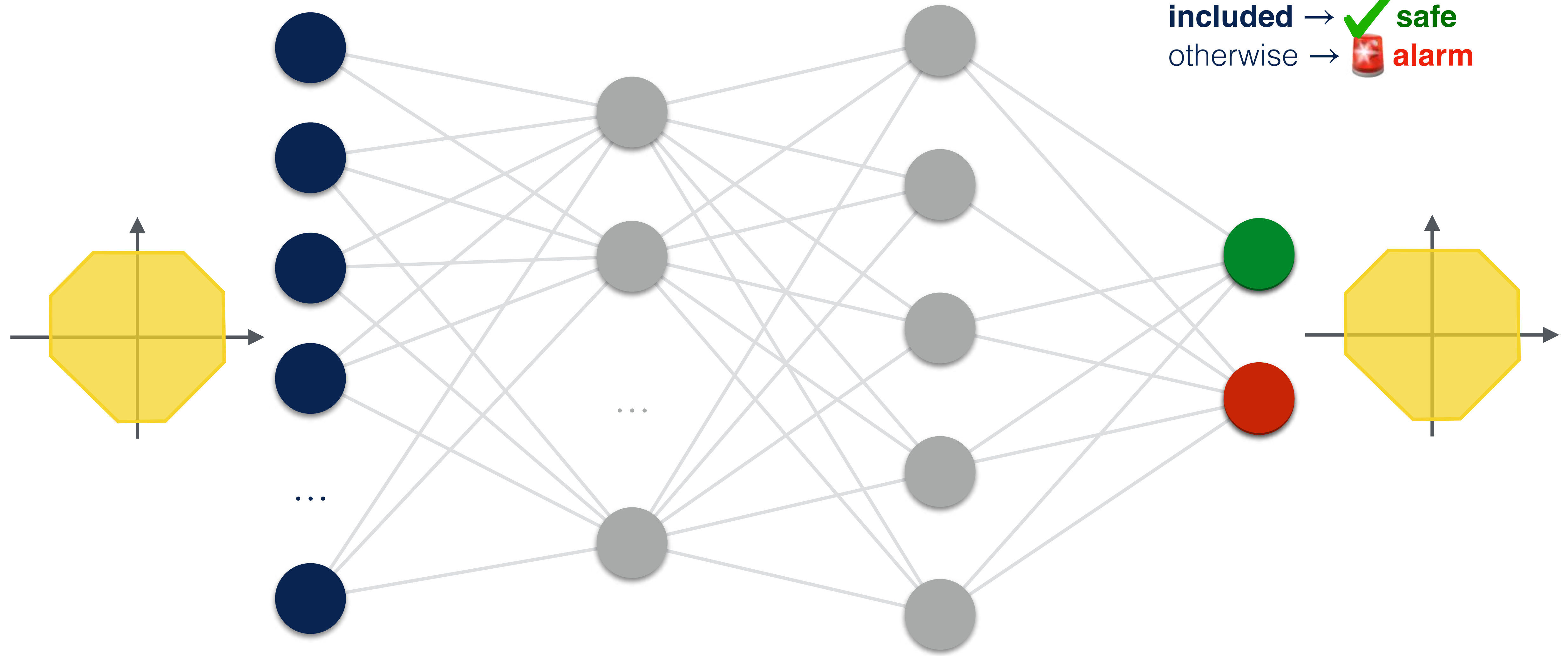
Model Checking
- analysis of a **model** of the software
- **sound and complete** **with respect to the model**

Static Analysis
- analysis of the software at some level of **abstraction**
- fully **automatic** and **sound** by construction
- generally **not complete**

Lesson 15 | Formal Verification of Machine Learning | Caterina Urban | 9

Static Analysis Methods

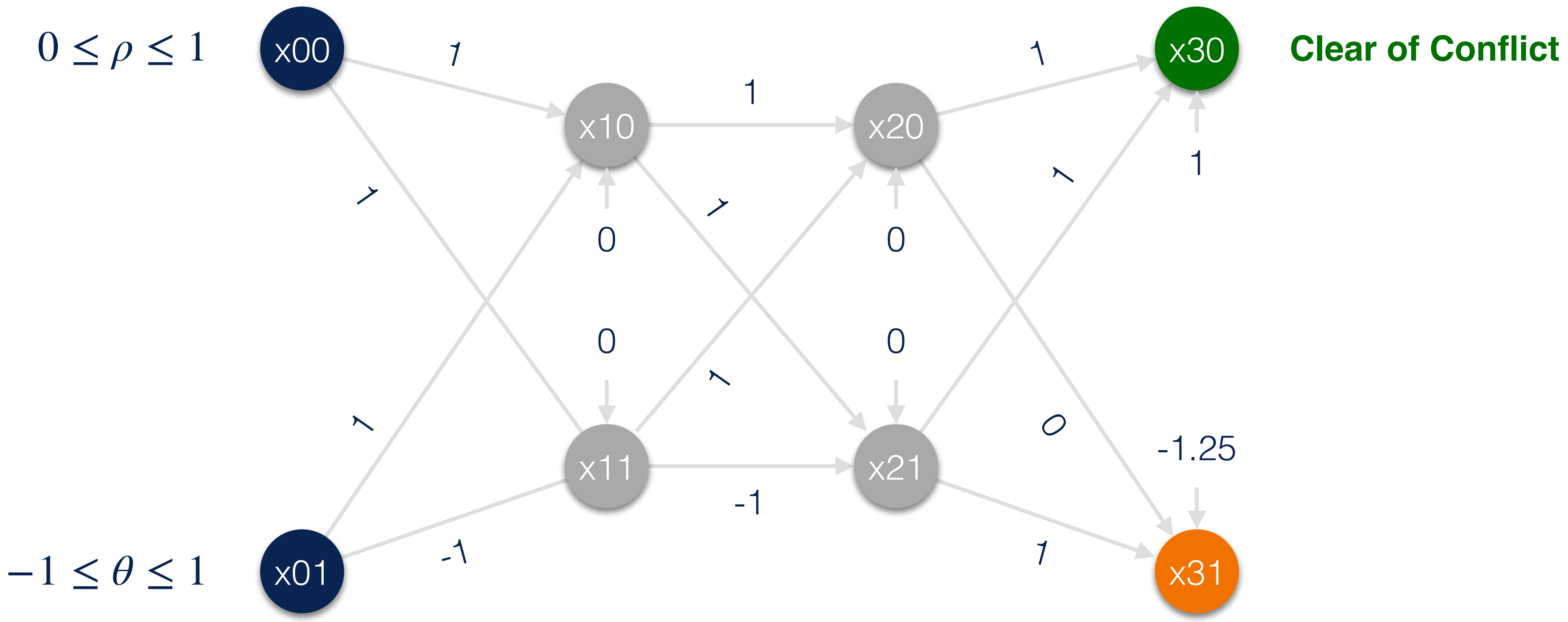
Forward Analysis



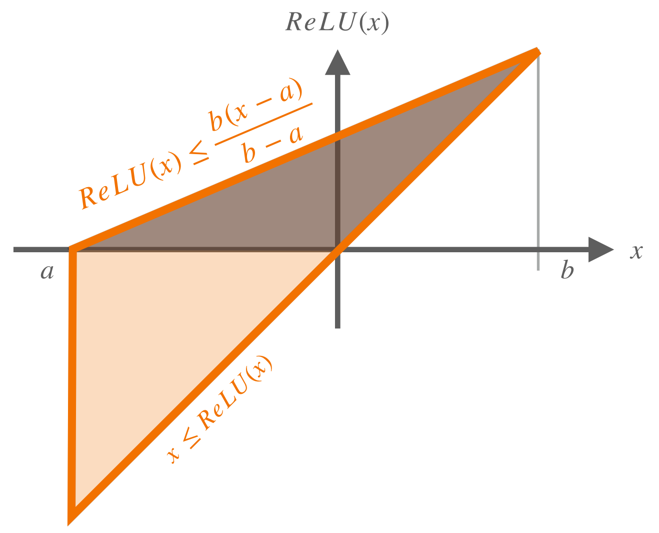
② check output for **inclusion** in **output specification $\mathbf{O}$**:
included → ✓ **safe**
otherwise → 🚨 **alarm**

① proceed **forwards from an abstraction** of the input specification $\mathbf{I}$

Example


$0 \leq \rho \leq 1$
$-1 \leq \theta \leq 1$
x00
x01
x10
x11
x20
x21
x30
x31
Clear of Conflict
1
1
1
-1
0
0
1
1
1
1
-1
0
0
1
1
0
1
1
-1.25

DeepPoly Domain [Singh19]



ReLU

$$x_{10} \mapsto \begin{cases} [x_{10}, \frac{2}{3} \cdot x_{10} + \frac{2}{3}] \\ [\mathbf{-1}, \mathbf{2}] \end{cases}$$

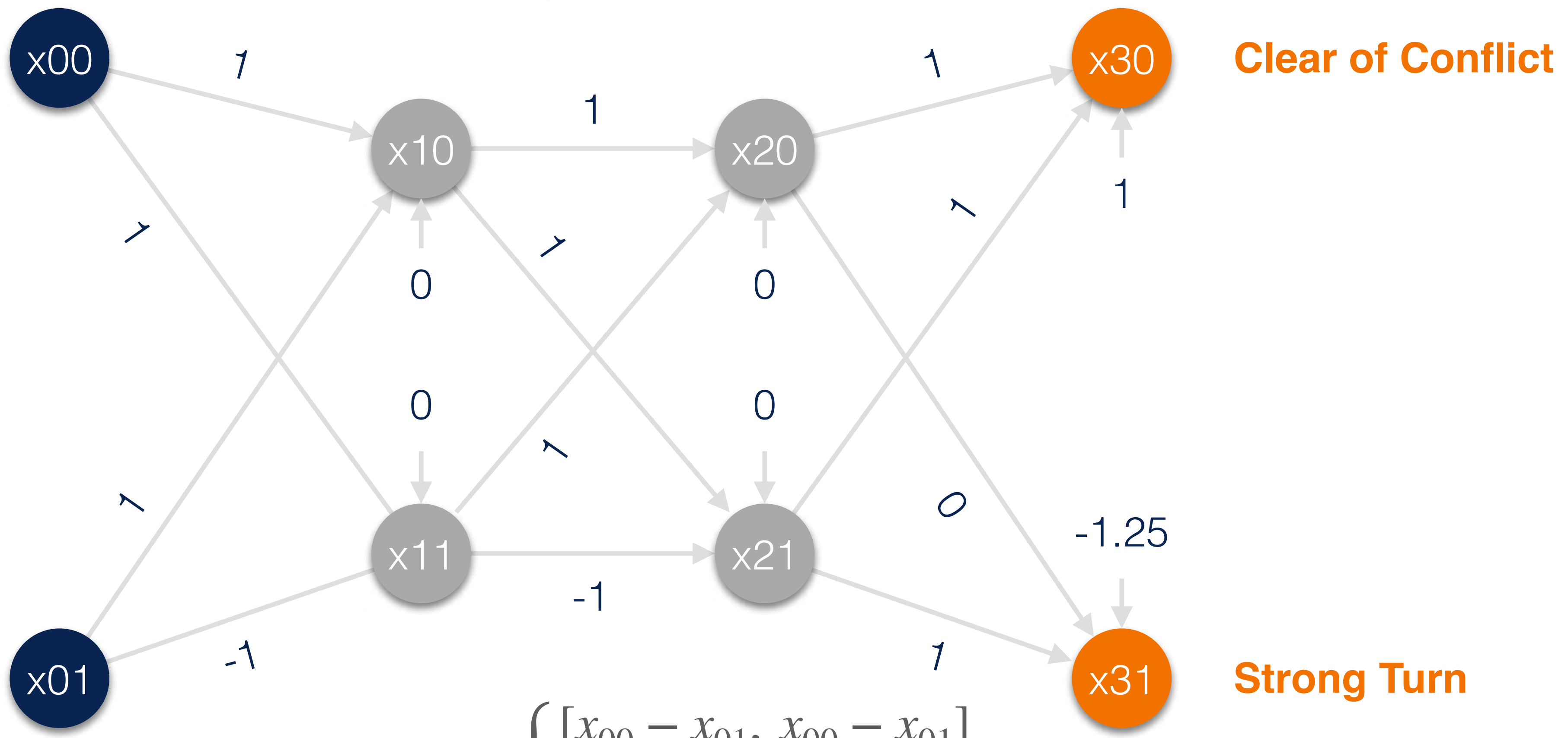
$$x_{10} \mapsto \begin{cases} [x_{00} + x_{01}, x_{00} + x_{01}] \\ [\mathbf{-1}, \mathbf{2}] \end{cases}$$

$0 \leq \rho \leq 1$

$$x_{00} \mapsto \begin{cases} [x_{00}, x_{00}] \\ [\mathbf{0}, \mathbf{1}] \end{cases}$$

$$x_{01} \mapsto \begin{cases} [x_{01}, x_{01}] \\ [\mathbf{-1}, \mathbf{1}] \end{cases}$$

$-1 \leq \theta \leq 1$

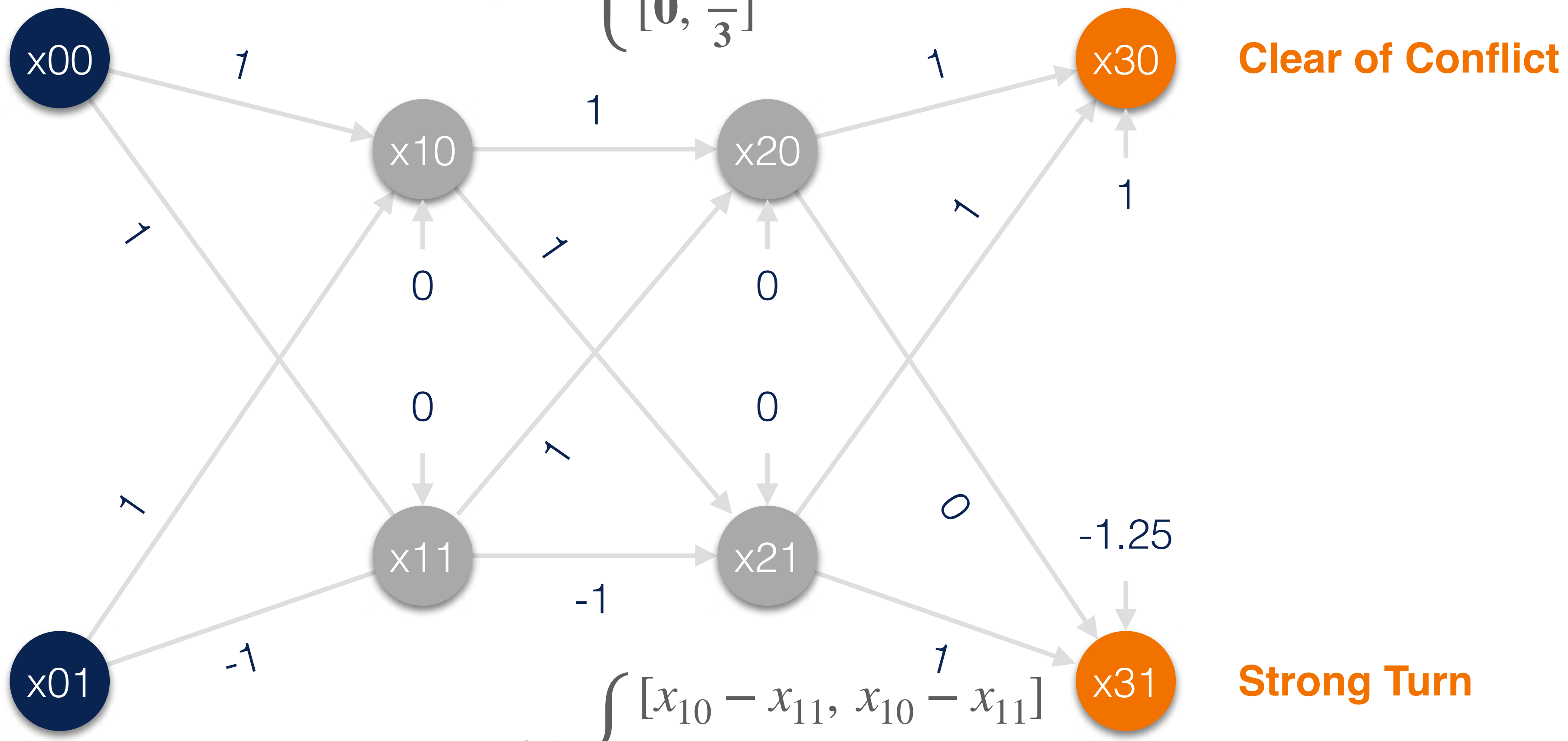


$$x_{11} \mapsto \begin{cases} [x_{00} - x_{01}, x_{00} - x_{01}] \\ [\mathbf{-1}, \mathbf{2}] \end{cases}$$

ReLU

$$x_{11} \mapsto \begin{cases} [x_{11}, \frac{2}{3} \cdot x_{11} + \frac{2}{3}] \\ [\mathbf{-1}, \mathbf{2}] \end{cases}$$

DeepPoly Domain [Singh19]



$$0 \leq \rho \leq 1$$

$$x_{00} \mapsto \begin{cases} [x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$$

$$x_{01} \mapsto \begin{cases} [x_{01},\ x_{01}] \\ [\mathbf{-1},\ \mathbf{1}] \end{cases}$$

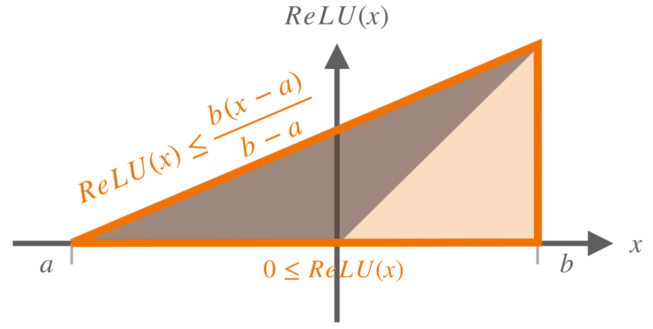
$$-1 \leq \theta \leq 1$$

$$x_{20} \mapsto \begin{cases} [x_{10} + x_{11},\ x_{10} + x_{11}] \\ [\mathbf{0},\ \mathbf{\tfrac{8}{3}}] \end{cases}$$

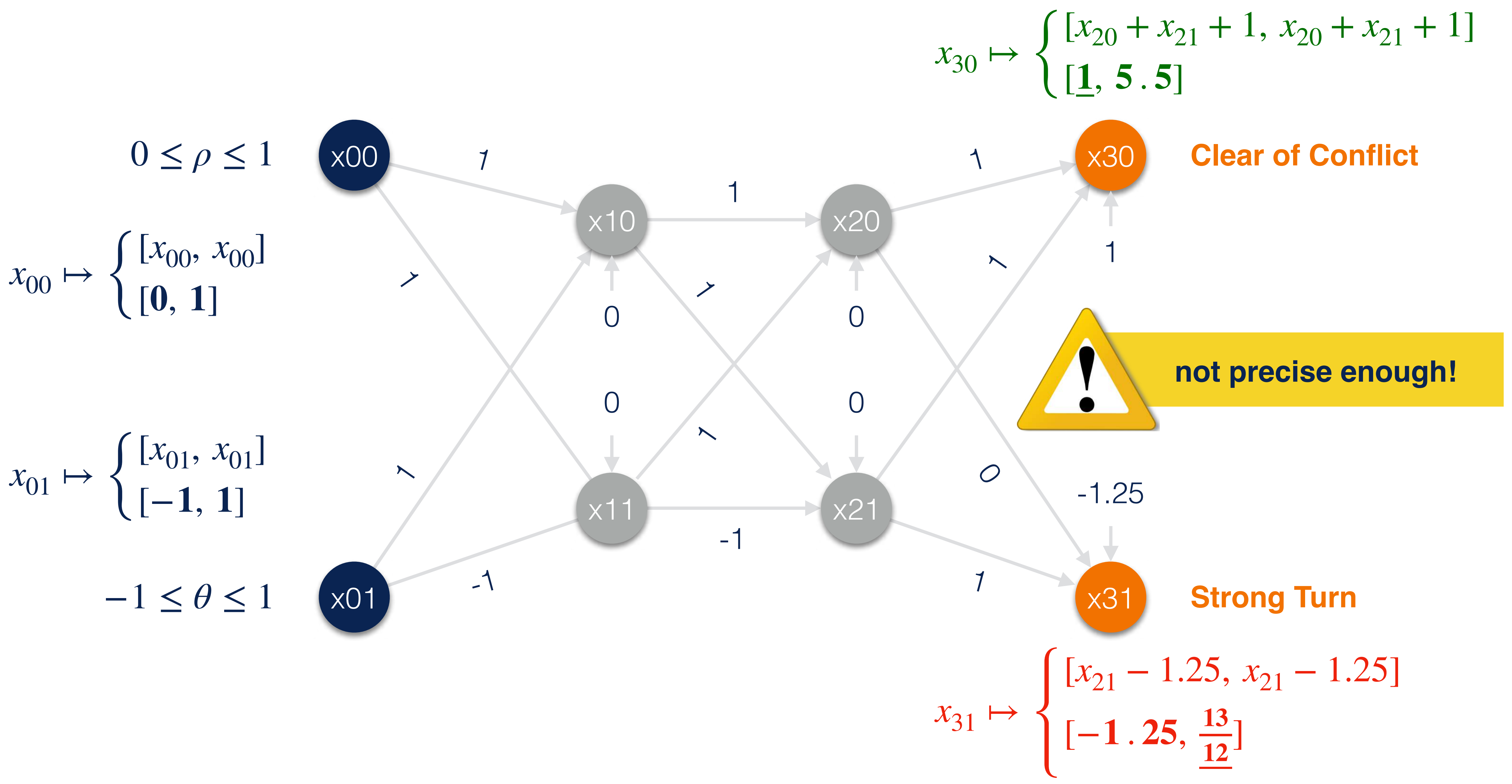
$$x_{21} \mapsto \begin{cases} [x_{10} - x_{11},\ x_{10} - x_{11}] \\ [-\mathbf{\tfrac{7}{3}},\ \mathbf{\tfrac{7}{3}}] \end{cases}$$

ReLU

$$x_{21} \mapsto \begin{cases} [0,\ 0.5 \cdot x_{21} + \tfrac{7}{6}] \\ [\mathbf{0},\ \mathbf{\tfrac{7}{3}}] \end{cases}$$



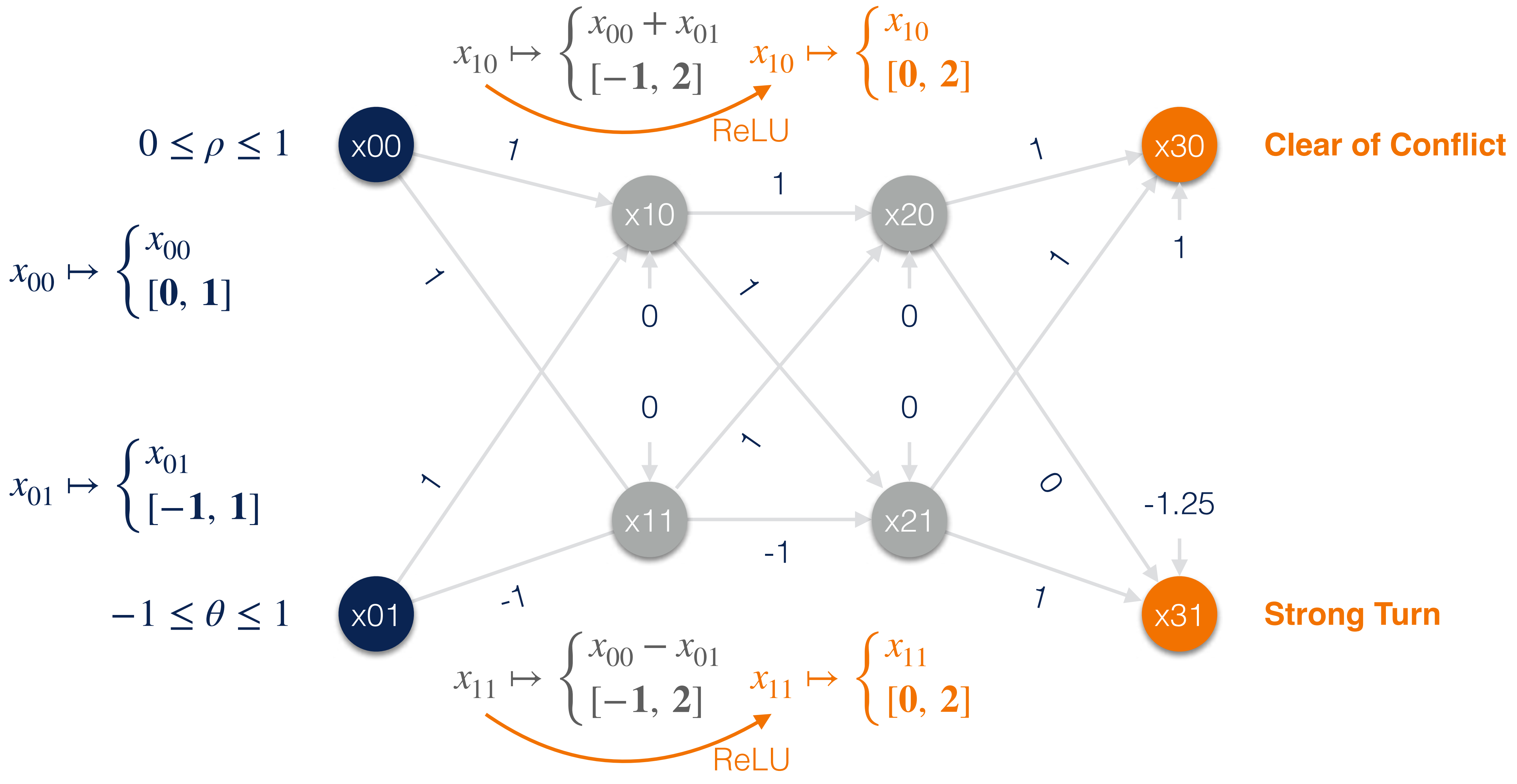
DeepPoly Domain [Singh19]



$x_{30} \mapsto \begin{cases} [x_{20} + x_{21} + 1,\ x_{20} + x_{21} + 1] \\ [\underline{\mathbf{1}},\ \mathbf{5.5}] \end{cases}$
$0 \leq \rho \leq 1$
$x_{00} \mapsto \begin{cases} [x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$
$x_{01} \mapsto \begin{cases} [x_{01},\ x_{01}] \\ [\mathbf{-1},\ \mathbf{1}] \end{cases}$
$-1 \leq \theta \leq 1$
x00
x01
x10
x11
x20
x21
x30
x31
1
1
1
-1
1
0
0
1
-1
1
1
0
0
0
1
1
1
-1.25
Clear of Conflict
not precise enough!
Strong Turn
$x_{31} \mapsto \begin{cases} [x_{21} - 1.25,\ x_{21} - 1.25] \\ [\mathbf{-1.25},\ \underline{\mathbf{\tfrac{13}{12}}}] \end{cases}$

Interval Domain

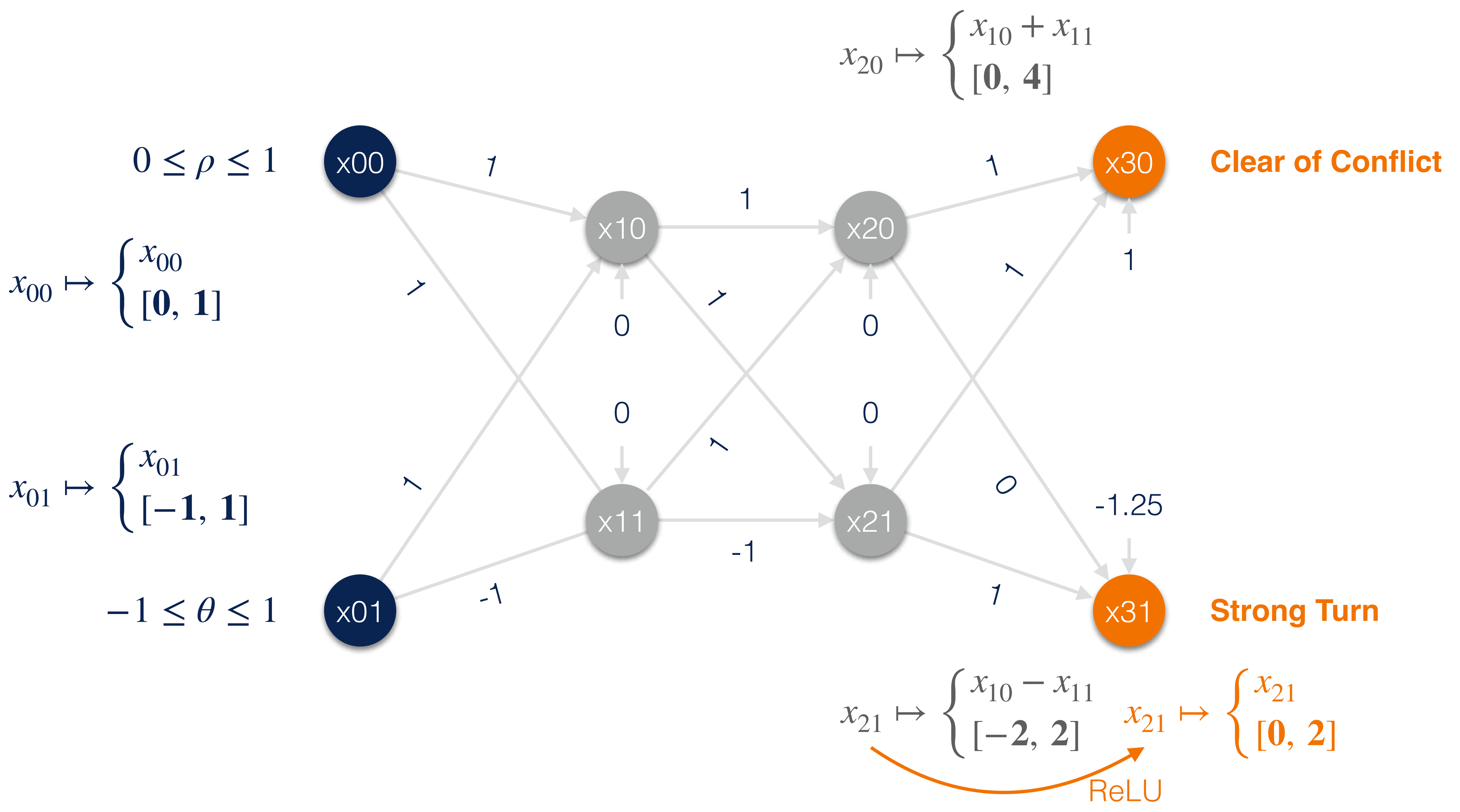
with Symbolic Constant Propagation [Li19]



$x_{10} \mapsto \begin{cases} x_{00} + x_{01} \\ [\mathbf{-1}, \mathbf{2}] \end{cases}$
$x_{10} \mapsto \begin{cases} x_{10} \\ [\mathbf{0}, \mathbf{2}] \end{cases}$
ReLU
$0 \leq \rho \leq 1$
x00
1
x10
1
x20
1
x30
Clear of Conflict
$x_{00} \mapsto \begin{cases} x_{00} \\ [\mathbf{0}, \mathbf{1}] \end{cases}$
1
0
1
0
1
1
0
0
0
1
$x_{01} \mapsto \begin{cases} x_{01} \\ [\mathbf{-1}, \mathbf{1}] \end{cases}$
1
0
-1.25
x11
-1
x21
$-1 \leq \theta \leq 1$
x01
-1
1
x31
Strong Turn
$x_{11} \mapsto \begin{cases} x_{00} - x_{01} \\ [\mathbf{-1}, \mathbf{2}] \end{cases}$
$x_{11} \mapsto \begin{cases} x_{11} \\ [\mathbf{0}, \mathbf{2}] \end{cases}$
ReLU

Interval Domain

with Symbolic Constant Propagation [Li19]



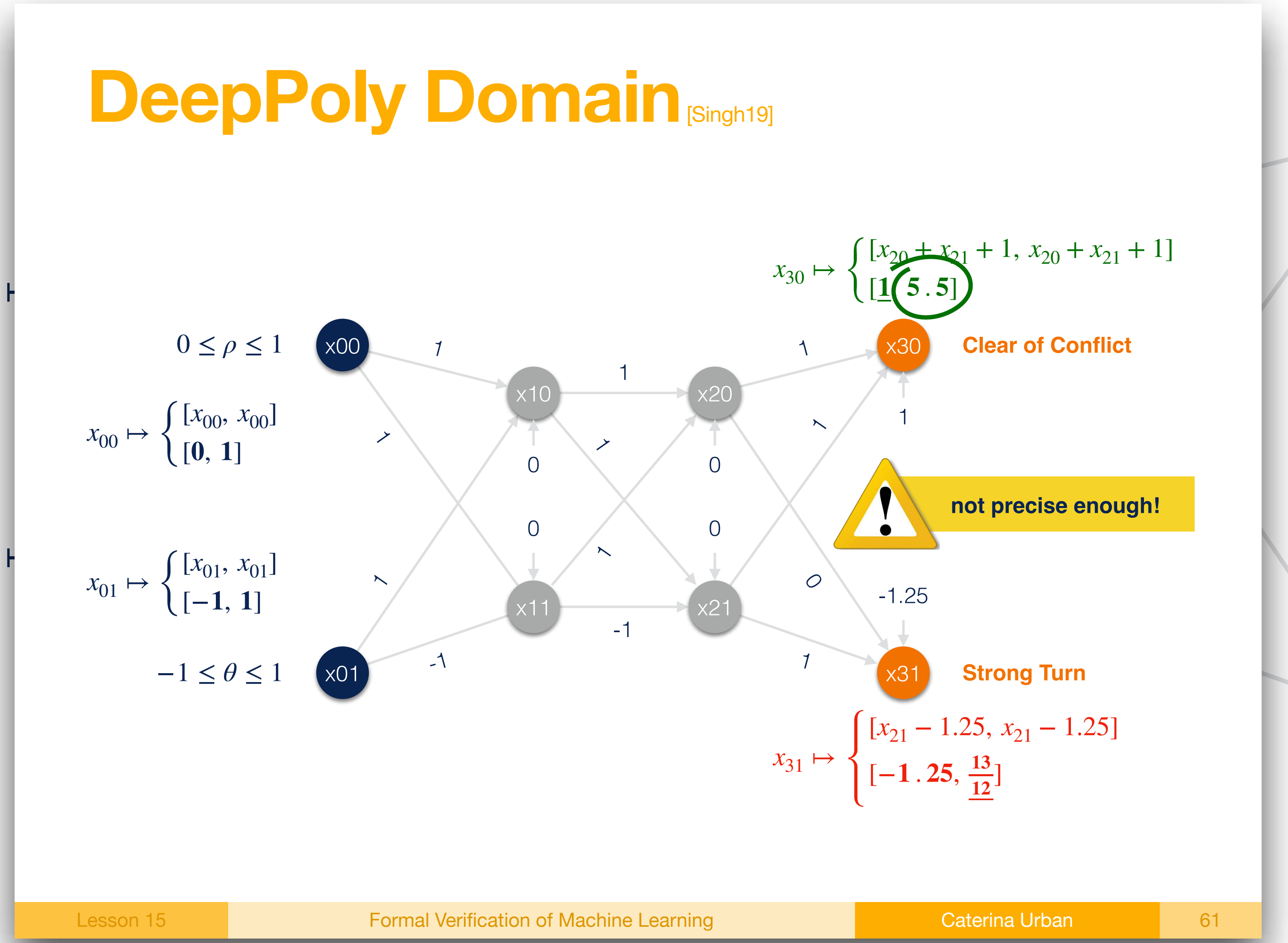
$0 \leq \rho \leq 1$
$-1 \leq \theta \leq 1$
$x_{00} \mapsto \begin{cases} x_{00} \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$
$x_{01} \mapsto \begin{cases} x_{01} \\ [\mathbf{-1},\ \mathbf{1}] \end{cases}$
$x_{20} \mapsto \begin{cases} x_{10} + x_{11} \\ [\mathbf{0},\ \mathbf{4}] \end{cases}$
$x_{21} \mapsto \begin{cases} x_{10} - x_{11} \\ [\mathbf{-2},\ \mathbf{2}] \end{cases}$
$x_{21} \mapsto \begin{cases} x_{21} \\ [\mathbf{0},\ \mathbf{2}] \end{cases}$
ReLU
x00
x01
x10
x11
x20
x21
x30
x31
1
1
1
-1
0
0
1
1
1
-1
0
0
1
1
0
1
1
-1.25
Clear of Conflict
Strong Turn

Interval Domain

with Symbolic Constant Propagation [Li19]

$x_{00} \mapsto$

$x_{01} \mapsto$



DeepPoly Domain [Singh19]

$0 \leq \rho \leq 1$

$x_{00} \mapsto \begin{cases} [x_{00}, x_{00}] \\ [\mathbf{0}, \mathbf{1}] \end{cases}$

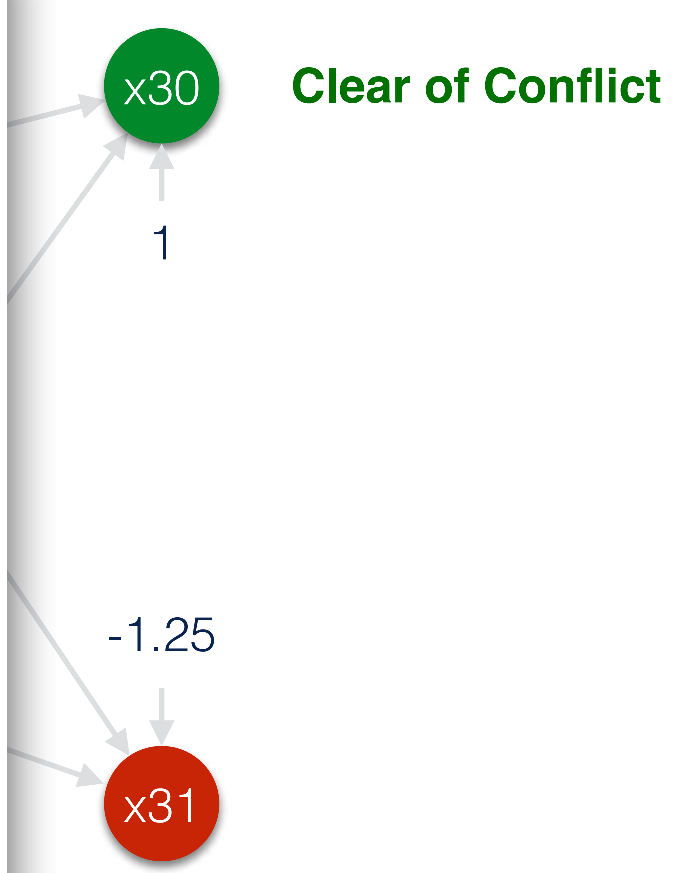
$x_{01} \mapsto \begin{cases} [x_{01}, x_{01}] \\ [\mathbf{-1}, \mathbf{1}] \end{cases}$

$-1 \leq \theta \leq 1$

$x_{30} \mapsto \begin{cases} [x_{20} + x_{21} + 1, x_{20} + x_{21} + 1] \\ [\mathbf{1}, \mathbf{5.5}] \end{cases}$

$x_{31} \mapsto \begin{cases} [x_{21} - 1.25, x_{21} - 1.25] \\ [\mathbf{-1.25}, \mathbf{\frac{13}{12}}] \end{cases}$

Lesson 15 | Formal Verification of Machine Learning | Caterina Urban | 61



$x_{30} \mapsto \begin{cases} x_{10} + x_{11} + x_{21} + 1 \\ [\underline{\mathbf{1}}, \mathbf{7}] \end{cases}$

Clear of Conflict

$x_{31} \mapsto \begin{cases} x_{21} - 1.25 \\ [\mathbf{-1.25}, \underline{\mathbf{0.75}}] \end{cases}$

Product Domain [Mazzucato21]

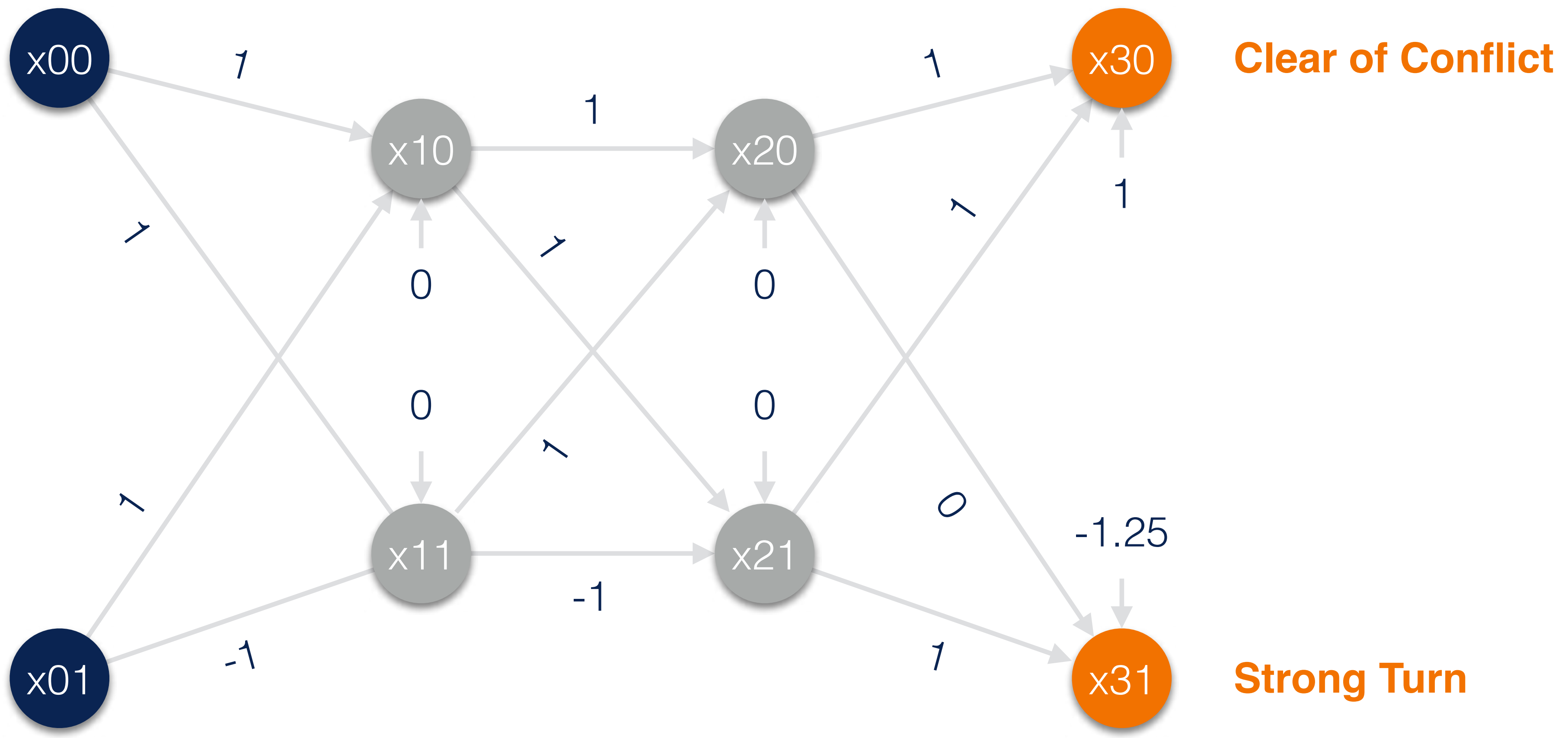
DeepPoly with Symbolic Constant Propagation

$0 \leq \rho \leq 1$

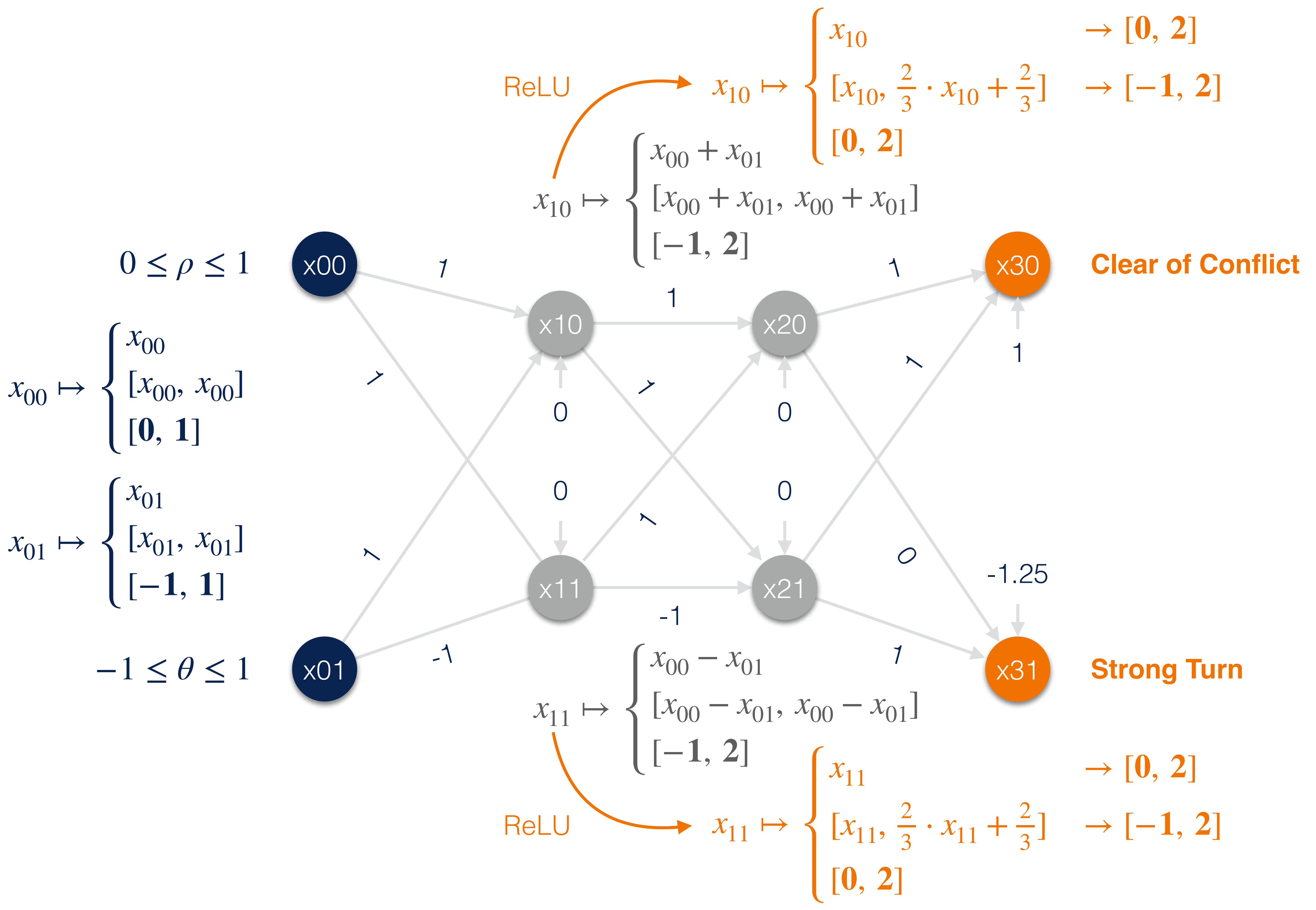
$$x_{00} \mapsto \begin{cases} x_{00} \\ [x_{00}, x_{00}] \\ [\mathbf{0}, \mathbf{1}] \end{cases}$$

$$x_{01} \mapsto \begin{cases} x_{01} \\ [x_{01}, x_{01}] \\ [\mathbf{-1}, \mathbf{1}] \end{cases}$$

$-1 \leq \theta \leq 1$

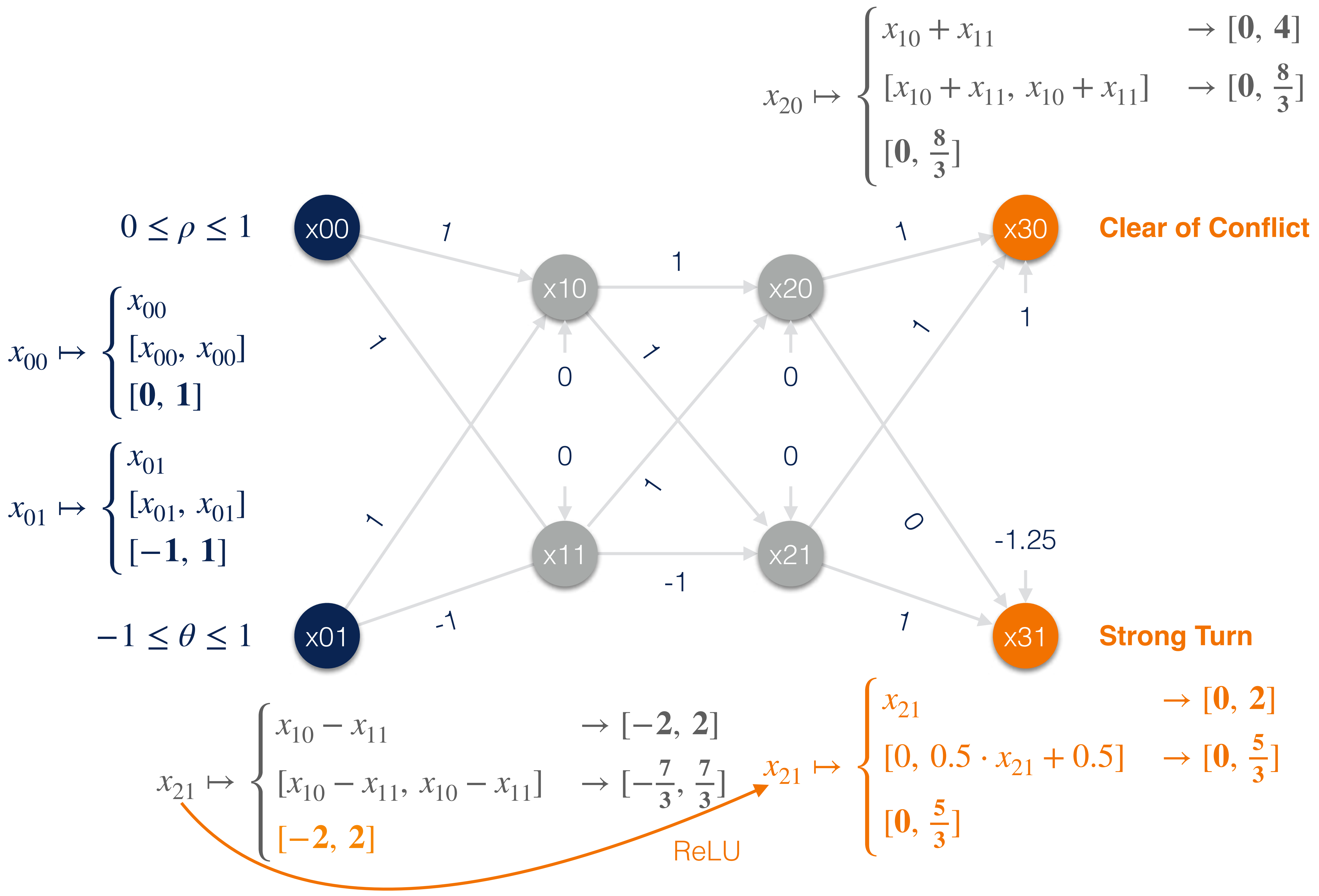


Product Domain [Mazzucato21]



ReLU
$x_{10} \mapsto \begin{cases} x_{10} & \rightarrow \mathbf{[0,\ 2]} \\ [x_{10},\ \frac{2}{3} \cdot x_{10} + \frac{2}{3}] & \rightarrow \mathbf{[-1,\ 2]} \\ \mathbf{[0,\ 2]} \end{cases}$
$x_{10} \mapsto \begin{cases} x_{00} + x_{01} \\ [x_{00} + x_{01},\ x_{00} + x_{01}] \\ \mathbf{[-1,\ 2]} \end{cases}$
$0 \leq \rho \leq 1$
$x_{00} \mapsto \begin{cases} x_{00} \\ [x_{00},\ x_{00}] \\ \mathbf{[0,\ 1]} \end{cases}$
$x_{01} \mapsto \begin{cases} x_{01} \\ [x_{01},\ x_{01}] \\ \mathbf{[-1,\ 1]} \end{cases}$
$-1 \leq \theta \leq 1$
x00
x01
x10
x11
x20
x21
x30
x31
1
1
1
1
-1
1
0
0
1
1
-1
0
0
1
1
1
0
1
-1.25
Clear of Conflict
Strong Turn
$x_{11} \mapsto \begin{cases} x_{00} - x_{01} \\ [x_{00} - x_{01},\ x_{00} - x_{01}] \\ \mathbf{[-1,\ 2]} \end{cases}$
ReLU
$x_{11} \mapsto \begin{cases} x_{11} & \rightarrow \mathbf{[0,\ 2]} \\ [x_{11},\ \frac{2}{3} \cdot x_{11} + \frac{2}{3}] & \rightarrow \mathbf{[-1,\ 2]} \\ \mathbf{[0,\ 2]} \end{cases}$

Product Domain [Mazzucato21]

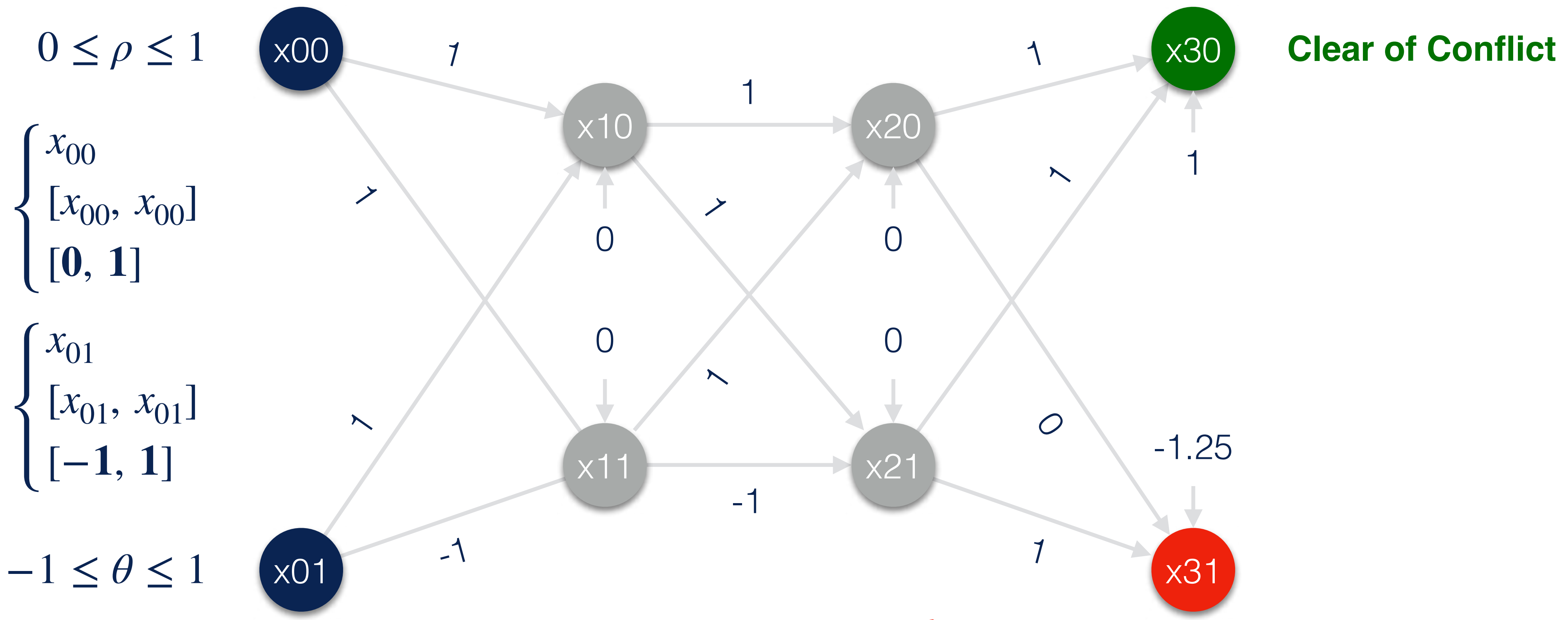


$$x_{20} \mapsto \begin{cases} x_{10} + x_{11} & \to [\mathbf{0},\ \mathbf{4}] \\ [x_{10} + x_{11},\ x_{10} + x_{11}] & \to [\mathbf{0},\ \frac{\mathbf{8}}{\mathbf{3}}] \\ [\mathbf{0},\ \frac{\mathbf{8}}{\mathbf{3}}] \end{cases}$$
$0 \leq \rho \leq 1$
$-1 \leq \theta \leq 1$
$$x_{00} \mapsto \begin{cases} x_{00} \\ [x_{00},\ x_{00}] \\ [\mathbf{0},\ \mathbf{1}] \end{cases}$$
$$x_{01} \mapsto \begin{cases} x_{01} \\ [x_{01},\ x_{01}] \\ [-\mathbf{1},\ \mathbf{1}] \end{cases}$$
x00
x01
x10
x11
x20
x21
x30
x31
Clear of Conflict
Strong Turn
1
1
1
1
1
1
1
1
1
1
1
-1
-1
0
0
0
0
0
-1.25
$$x_{21} \mapsto \begin{cases} x_{10} - x_{11} & \to [-\mathbf{2},\ \mathbf{2}] \\ [x_{10} - x_{11},\ x_{10} - x_{11}] & \to [-\frac{\mathbf{7}}{\mathbf{3}},\ \frac{\mathbf{7}}{\mathbf{3}}] \\ [-\mathbf{2},\ \mathbf{2}] \end{cases}$$
ReLU
$$x_{21} \mapsto \begin{cases} x_{21} & \to [\mathbf{0},\ \mathbf{2}] \\ [0,\ 0.5 \cdot x_{21} + 0.5] & \to [\mathbf{0},\ \frac{\mathbf{5}}{\mathbf{3}}] \\ [\mathbf{0},\ \frac{\mathbf{5}}{\mathbf{3}}] \end{cases}$$

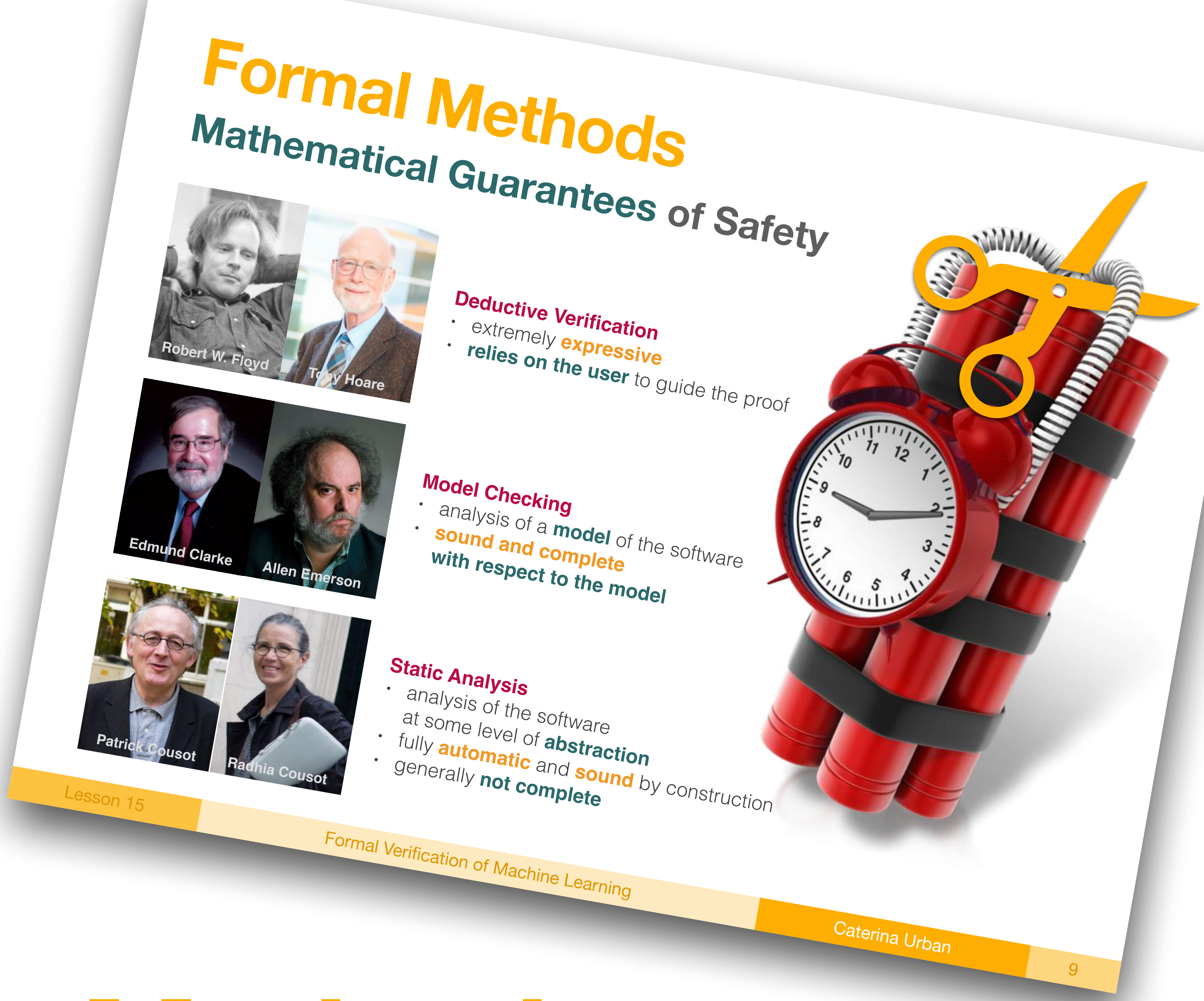
Product Domain [Mazzucato21]

$$x_{30} \mapsto \begin{cases} x_{10} + x_{11} + x_{21} + 1 & \rightarrow [\mathbf{1}, \frac{\mathbf{20}}{\mathbf{3}}] \\ [x_{20} + x_{21} + 1,\ x_{20} + x_{21} + 1] & \rightarrow [\mathbf{1}, \mathbf{4.5}] \\ [\underline{\mathbf{1}}, \mathbf{4.5}] \end{cases}$$

$$x_{00} \mapsto \begin{cases} x_{00} \\ [x_{00}, x_{00}] \\ [\mathbf{0}, \mathbf{1}] \end{cases} \qquad x_{01} \mapsto \begin{cases} x_{01} \\ [x_{01}, x_{01}] \\ [\mathbf{-1}, \mathbf{1}] \end{cases}$$



$$x_{31} \mapsto \begin{cases} x_{21} - 1.25 & \rightarrow [\mathbf{-1.25}, \frac{\mathbf{5}}{\mathbf{12}}] \\ [x_{21} - 1.25,\ x_{21} - 1.25] & \rightarrow [\mathbf{-1.25}, \frac{\mathbf{5}}{\mathbf{12}}] \\ [\mathbf{-1.25}, \underline{\frac{\mathbf{5}}{\mathbf{12}}}] \end{cases}$$



Formal Methods

Mathematical Guarantees of Safety

Robert W. Floyd, Tony Hoare

Deductive Verification
- extremely **expressive**
- **relies on the user** to guide the proof

Edmund Clarke, Allen Emerson

Model Checking
- analysis of a **model** of the software
- **sound and complete with respect to the model**

Patrick Cousot, Radhia Cousot

Static Analysis
- analysis of the software at some level of **abstraction**
- fully **automatic** and **sound** by construction
- generally **not complete**

Lesson 15 | Formal Verification of Machine Learning | Caterina Urban | 9

Other Complete Methods

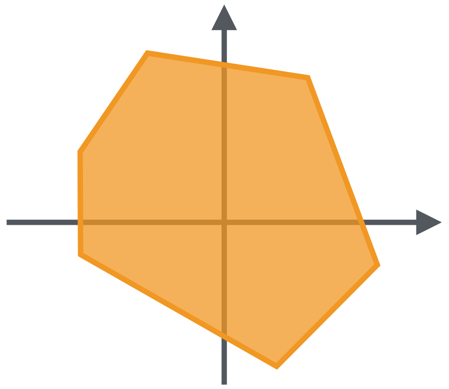
Star Sets

Exact Static Analysis Method

use union of efficient representations of bounded convex polyhedra

$$\Theta \overset{\text{def}}{=} \langle c, V, P \rangle$$

$c \in \mathcal{R}^n$: center
$V = \{v_1, \ldots, v_m\}$: basis vectors in $\mathcal{R}^n$
$P : \mathcal{R}^m \to \{ \perp, \top \}$: predicate



$$[\![\Theta]\!] = \{x \mid x = c + \sum_{i=1}^{m} \alpha_i v_i \text{ such that } P(\alpha_1, \ldots, \alpha_m) = \top \}$$

- fast and cheap **affine mapping operations** → neural network layers
- inexpensive **intersections with half-spaces** → ReLU activations

H.-D. Tran et al. - Star-Based Reachability Analysis of Deep Neural Networks (FM 2018)

Star Sets

Exact Static Analysis Method

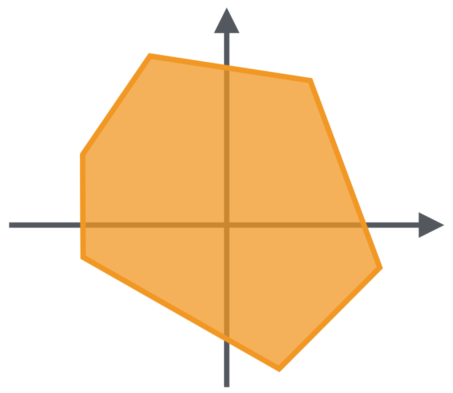
us
efficient
of bounded

Follow-up Work

H.-D. Tran et al. - Verification of Deep Convolutional Neural Networks Using ImageStars (CAV 2020)

$$\Theta \overset{\text{def}}{=} \langle c, V, P \rangle$$

$c \in \mathcal{R}^n$: center
$V = \{v_1, \ldots, v_m\}$: basis vectors in $\mathcal{R}^n$
$P: \mathcal{R}^m \to \{ \perp, \top \}$: predicate



$$[\![\Theta]\!] = \{ x \mid x = c + \sum_{i=1}^{m} \alpha_i v_i \text{ such that } P(\alpha_1, \ldots, \alpha_m) = \top \}$$

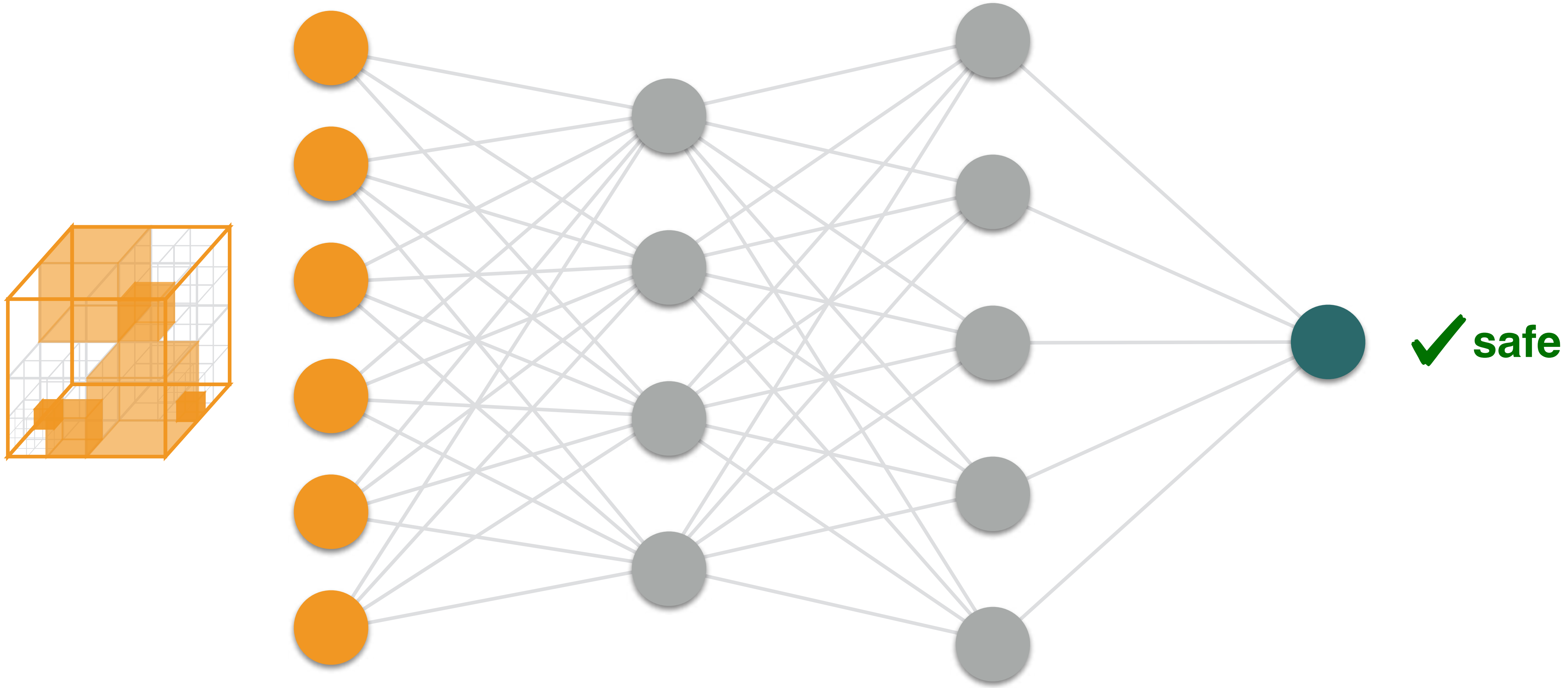
- fast and cheap **affine mapping operations** → neural network layers
- inexpensive **intersections with half-spaces** → ReLU activations

H.-D. Tran et al. - Star-Based Reachability Analysis of Deep Neural Networks (FM 2018)

ReluVal

Asymptotically Complete Method

use symbolic propagation
+ iterative input refinement



S. Wang et al. - Formal Security Analysis of Neural Networks Using Symbolic Intervals (USENIX Security 2018)

Neurify

Asymptotically Complete Method

use symbolic propagation +
convex ReLU approximation +
iterative input/ReLU refinement

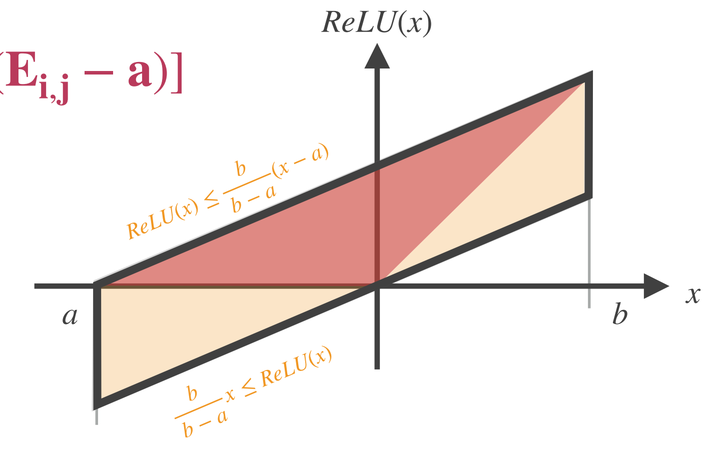
$$x_{i,j} \mapsto \begin{cases} [\sum_k c_{0,k} \cdot x_{0,k} + c,\ \sum_k d_{0,k} \cdot x_{0,k} + d] & c_{0,k}, c, d_{0,k}, d \in \mathcal{R} \\ [a,\ b] & a, b \in \mathcal{R} \end{cases}$$

$$\begin{tikzcd} & x_{i,j} \mapsto \begin{cases} [\mathbf{E_{i,j}}, \mathbf{E_{i,j}}] \\ [\mathbf{a}, \mathbf{b}] \end{cases} \\ x_{i,j} \mapsto \begin{cases} [\mathbf{E_{i,j}}, \mathbf{E_{i,j}}] \\ [\mathbf{a}, \mathbf{b}] \end{cases} \arrow[ur, "\text{ReLU}"] \arrow[r, "\text{ReLU}"] \arrow[dr, "\text{ReLU}"] & x_{i,j} \mapsto \begin{cases} [\frac{\mathbf{b}}{\mathbf{b}-\mathbf{a}}\mathbf{E_{i,j}}, \frac{\mathbf{b}}{\mathbf{b}-\mathbf{a}}(\mathbf{E_{i,j}} - \mathbf{a})] \\ [\mathbf{a}, \mathbf{b}] \end{cases} \\ & x_{i,j} \mapsto \begin{cases} [\mathbf{0}, \mathbf{0}] \\ [\mathbf{0}, \mathbf{0}] \end{cases} \end{tikzcd}$$

$0 \leq a$

$a < 0 \wedge 0 < b$

$b \leq 0$


$ReLU(x)$
$ReLU(x) \leq \frac{b}{b-a}(x-a)$
a
b
x
$\frac{b}{b-a}x \leq ReLU(x)$

S. Wang et al. - Formal Security Analysis of Neural Networks Using Symbolic Intervals (USENIX Security 2018)

Further Complete Methods

- **W. Ruan, X. Huang, and M. Kwiatkowska**. *Reachability Analysis of Deep Neural Networks with Provable Guarantees*. In IJCAI, 2018.
  a global optimization-based approach for verifying Lipschitz continuous neural networks

- **G. Singh, T. Gehr, M. Püschel, and M. Vechev**. *Boosting Robustness Certification of Neural Networks*. In ICLR, 2019.
  an approach combining abstract interpretation and (mixed integer) linear programming

Formal Methods

Mathematical Guarantees of Safety

Deductive Verification
- extremely **expressive**
- **relies on the user** to guide the proof

Model Checking
- analysis of a **model** of the software
- **sound and complete with respect to the model**

Static Analysis
- analysis of the software at some level of **abstraction**
- fully **automatic** and **sound** by construction
- generally **not complete**

Lesson 15 | Formal Verification of Machine Learning | Caterina Urban | 9

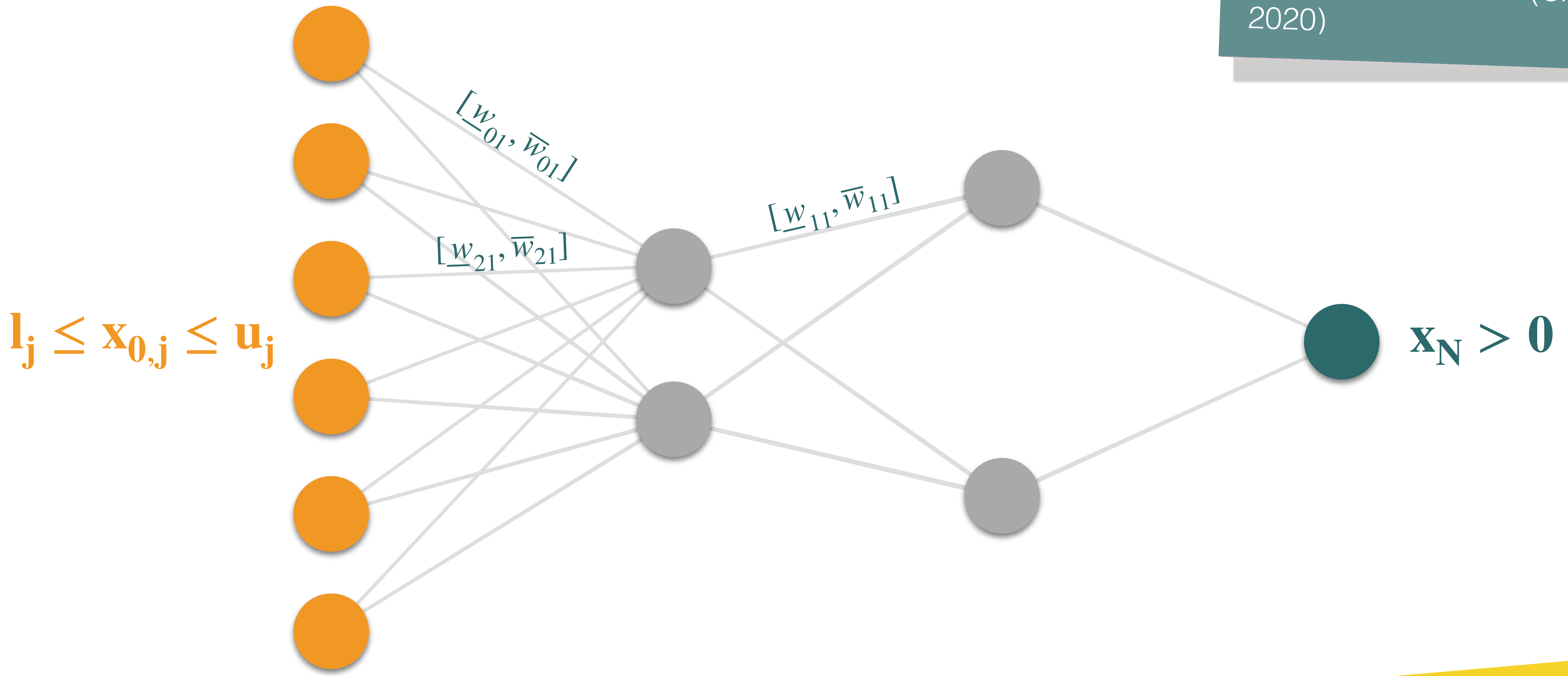
Other Incomplete Methods

Interval Neural Networks

Abstraction-Based Method

Related Work

Y. Y. Elboher et al. - An Abstraction-Based Framework for Neural Network Verification (CAV 2020)



$l_j \leq x_{0,j} \leq u_j$
$[\underline{w}_{01}, \overline{w}_{01}]$
$[\underline{w}_{21}, \overline{w}_{21}]$
$[\underline{w}_{11}, \overline{w}_{11}]$
$x_N > 0$

merge neurons layer-wise based on partitioning strategy + replace weights with intervals

P. Prabhakar and Z. R. Afza - Abstraction based Output Range Analysis for Neural Networks (NeurIPS 2019)

Further Incomplete Methods

- **W. Xiang, H.-D. Tran, and T. T. Johnson**. *Output Reachable Set Estimation and Verification for Multi-Layer Neural Networks*. 2018.
  an approach combining simulation and linear programming

- **K. Dvijotham, R. Stanforth, S. Gowal, T. Mann, and P. Kohli**. *A Dual Approach to Scalable Verification of Deep Networks*. In UAI, 2018.
  an approach based on duality for verifying neural networks

Further Incomplete Methods

- **E. Wong and Z. Kolter**. *Provable Defenses Against Adversarial Examples via the Convex Outer Adversarial Polytope*. In ICML, 2018.
  A. Raghunathan, J. Steinhardt, and P. Liang. *Certified Defenses against Adversarial Examples*. In ICML, 2018.
  T.-W. Weng, H. Zhang, H. Chen, Z. Song, C.-J. Hsieh, L. Daniel, D. Boning, and I. Dhillon. *Towards Fast Computation of Certified Robustness for ReLU Networks*. In ICML, 2018.
  H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel. Efficient *Neural Network Robustness Certification with General Activation Functions*. In NeurIPS, 2018.
  approaches for finding a lower bound on robustness to adversarial perturbations

Further Incomplete Methods

- **A. Boopathy, T.-W. Weng, P.-Y. Chen, S. Liu, and L. Daniel**. *CNN-Cert: An Efficient Framework for Certifying Robustness of Convolutional Neural Networks*. In AAAI, 2019.
  approach focusing on convolutional neural networks

- **C.-Y. Ko, Z. Lyu, T.-W. Weng, L. Daniel, N. Wong, and D. Lin**. *POPQORN: Quantifying Robustness of Recurrent Neural Networks*. In ICML, 2019.
  H. Zhang, M. Shinn, A. Gupta, A. Gurfinkel, N. Le, and N. Narodytska. *Verification of Recurrent Neural Networks for Cognitive Tasks via Reachability Analysis*. In ECAI, 2020.
  approaches focusing on recurrent neural networks

- **D. Gopinath, H. Converse, C. S. Pasareanu, and A. Taly**. *Property Inference for Deep Neural Networks*. In ASE, 2019.
  an approach for inferring safety properties of neural networks

Complete Methods

Advantages

sound and complete

Disadvantages

soundness not typically guaranteed with respect to floating-point arithmetic

do not scale to large models

often limited to certain model architectures

Incomplete Methods

Disadvantages

suffer from false positives

Advantages

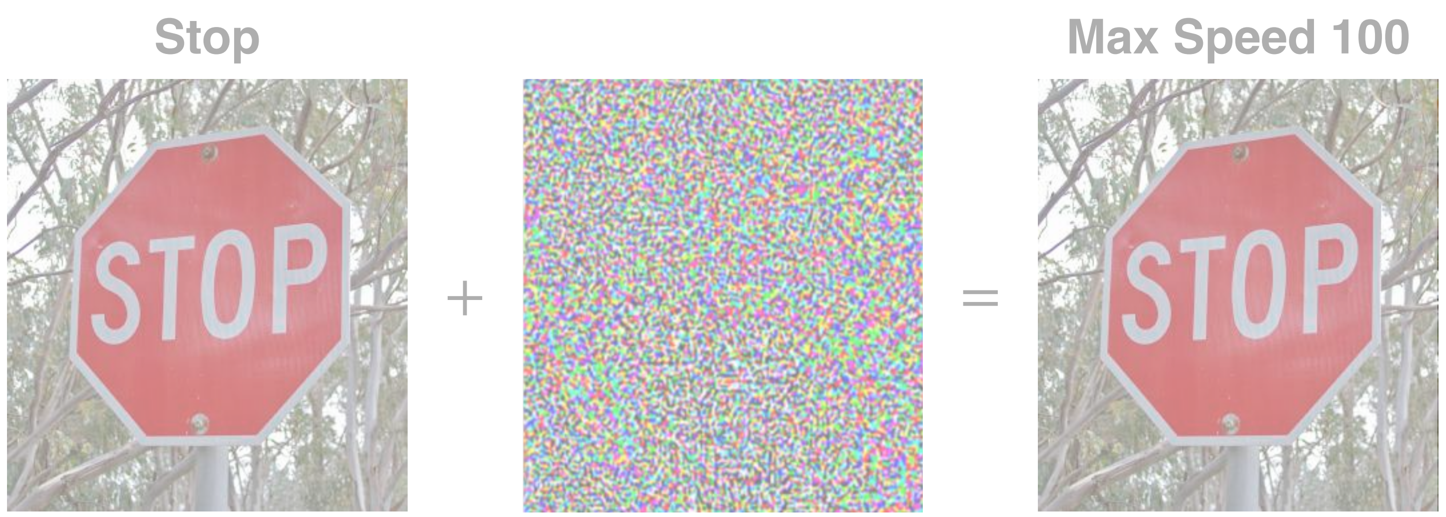
able to scale to large models

sound often also with respect to floating-point arithmetic

less limited to certain model architectures

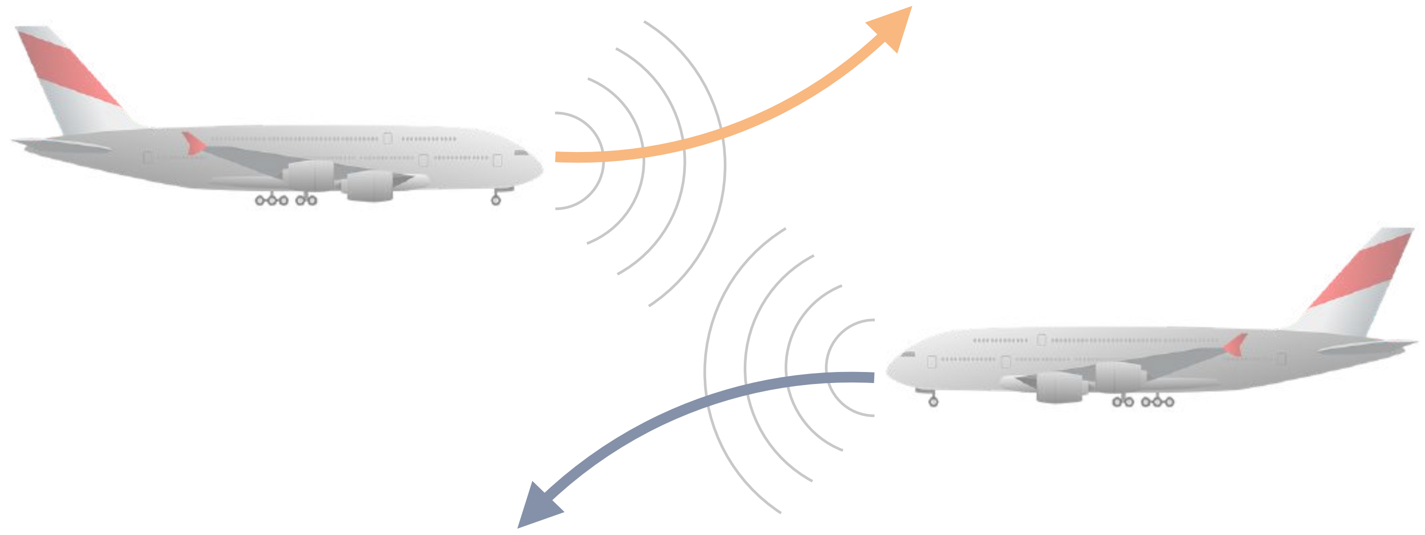
Stability

Goal G3 in [Kurd03]

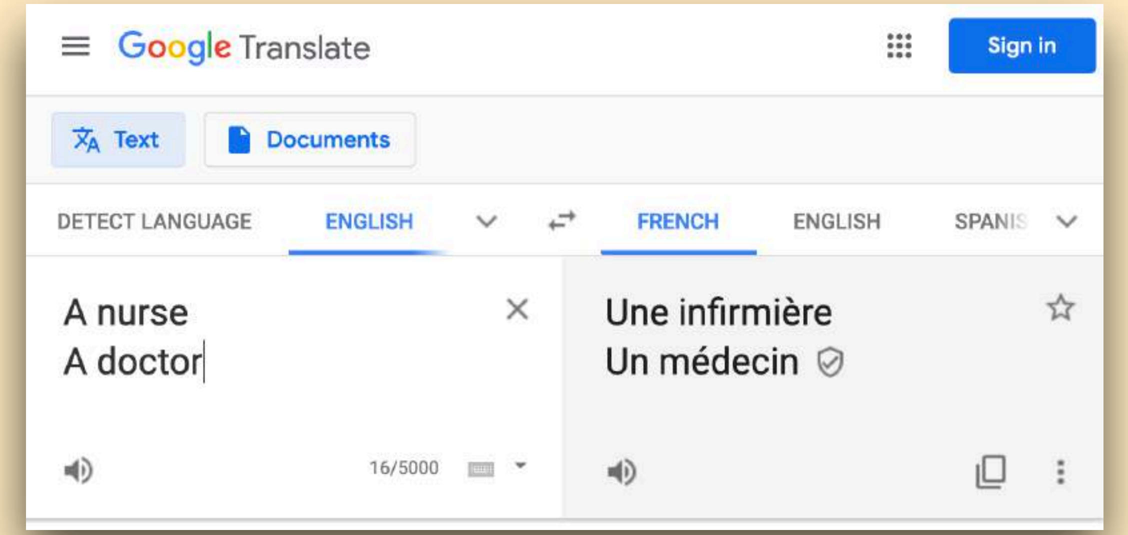


Safety

Goal G4 in [Kurd03]



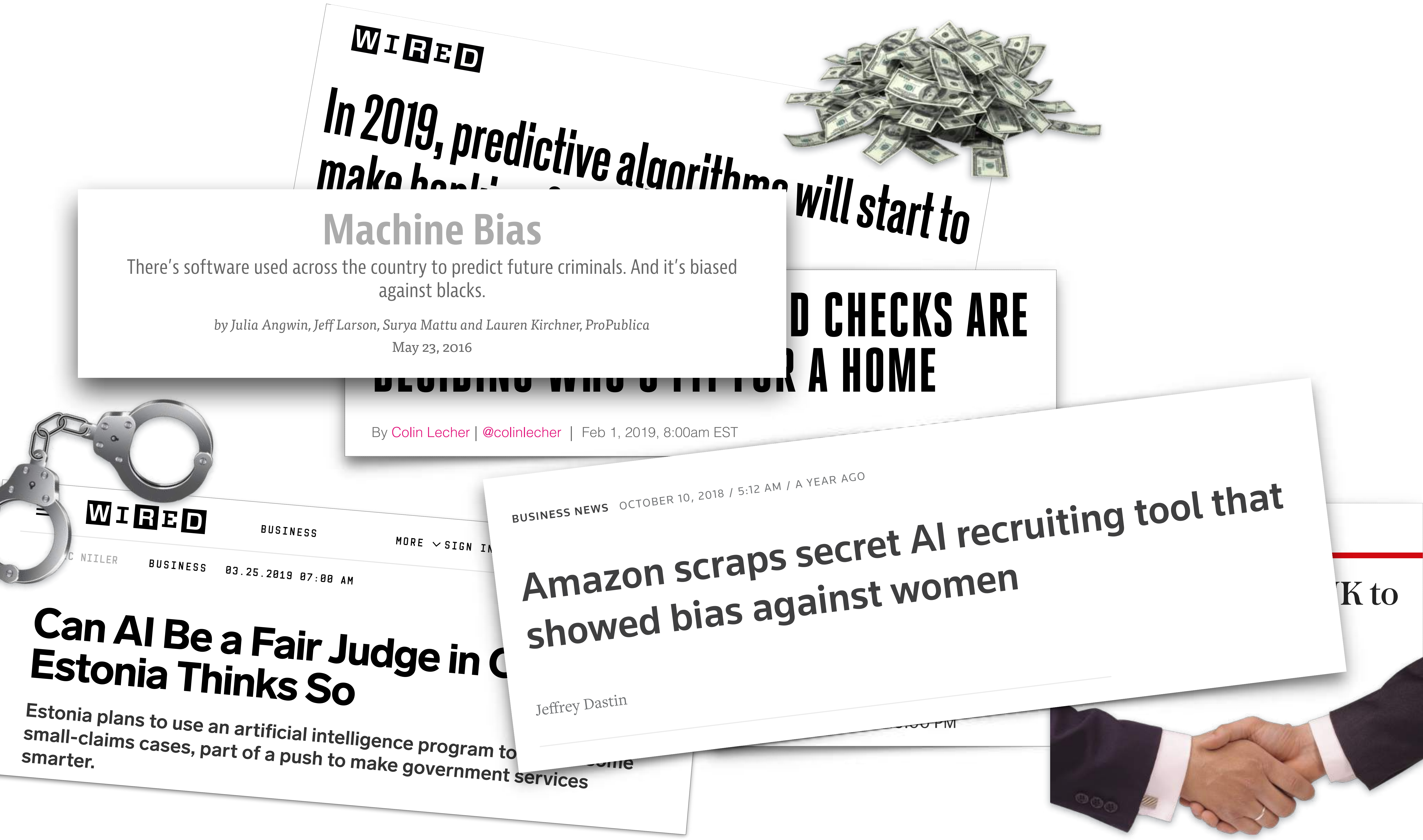
Fairness



A nurse
A doctor

Une infirmière
Un médecin

ML Impacts Our Society



WIRED

In 2019, predictive algorithms will start to make bankin

Machine Bias

There's software used across the country to predict future criminals. And it's biased against blacks.

by Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner, ProPublica

May 23, 2016

D CHECKS ARE

R A HOME

By Colin Lecher | @colinlecher | Feb 1, 2019, 8:00am EST

WIRED BUSINESS MORE SIGN IN

C NIILER BUSINESS 03.25.2019 07:00 AM

Can AI Be a Fair Judge in C

Estonia Thinks So

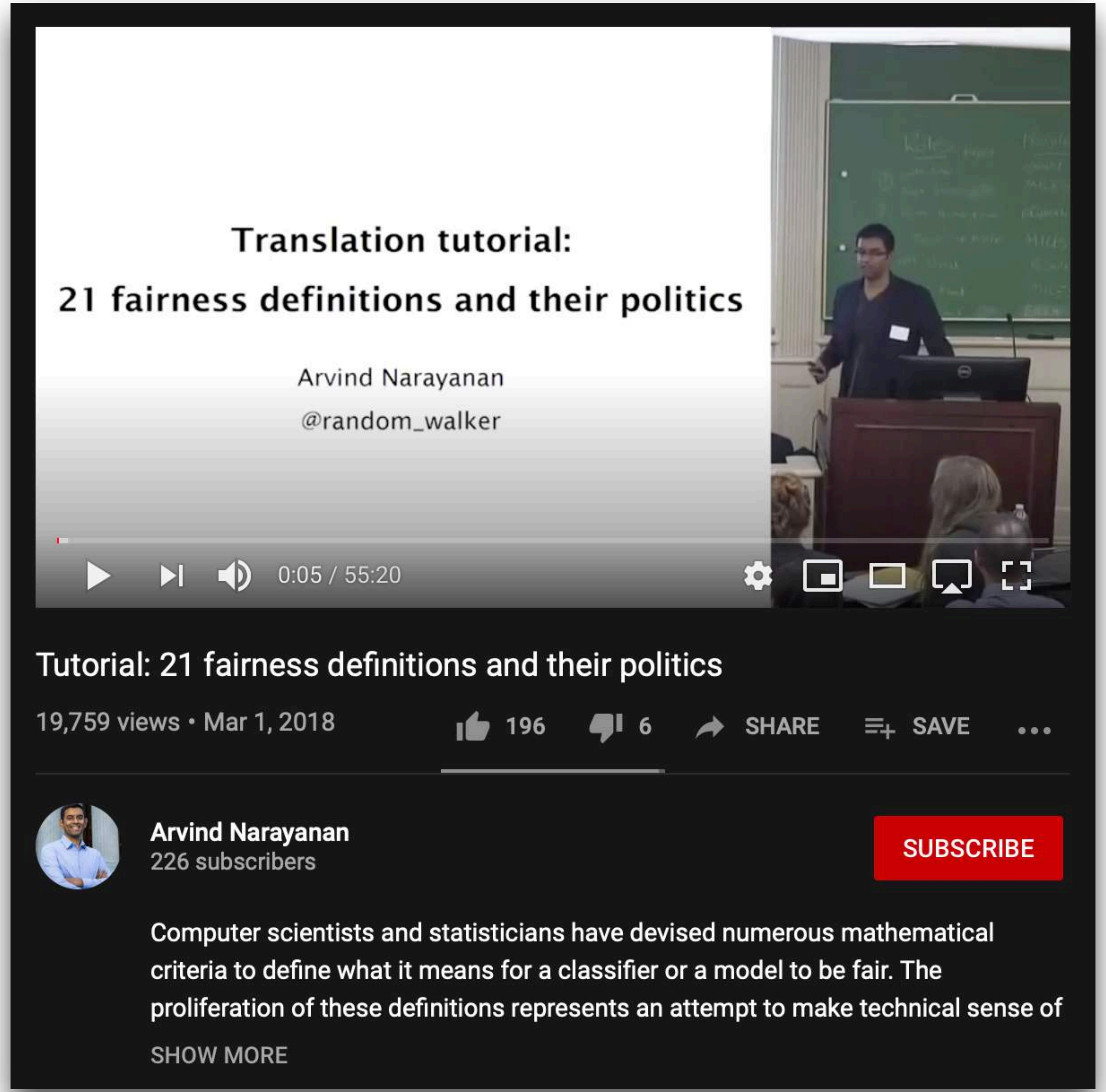
Estonia plans to use an artificial intelligence program to small-claims cases, part of a push to make government services smarter.

BUSINESS NEWS OCTOBER 10, 2018 / 5:12 AM / A YEAR AGO

Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin

K to



Translation tutorial:

21 fairness definitions and their politics

Arvind Narayanan

@random_walker

Tutorial: 21 fairness definitions and their politics

19,759 views • Mar 1, 2018

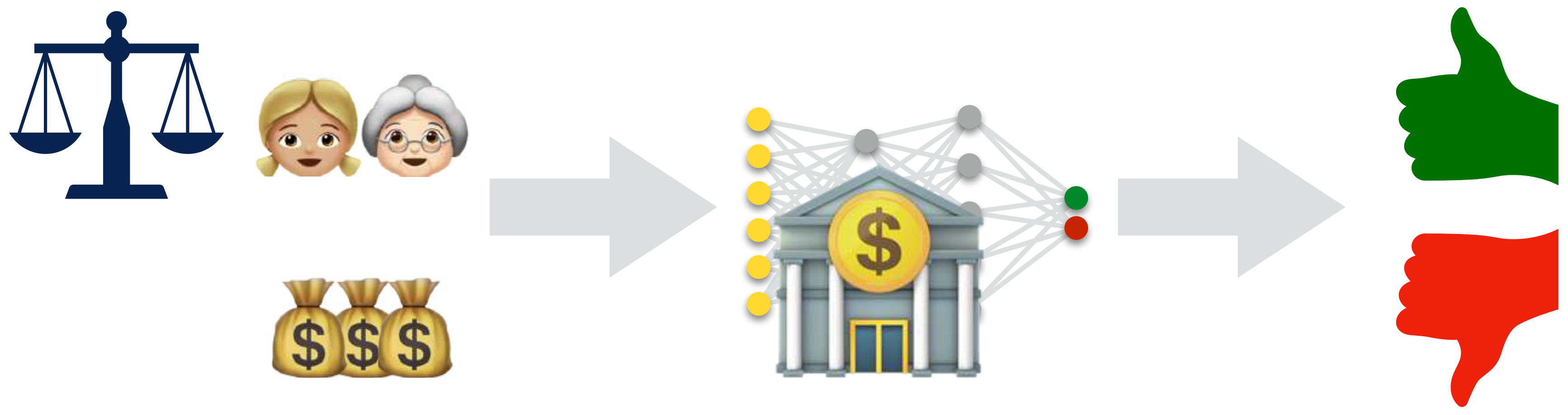
Arvind Narayanan

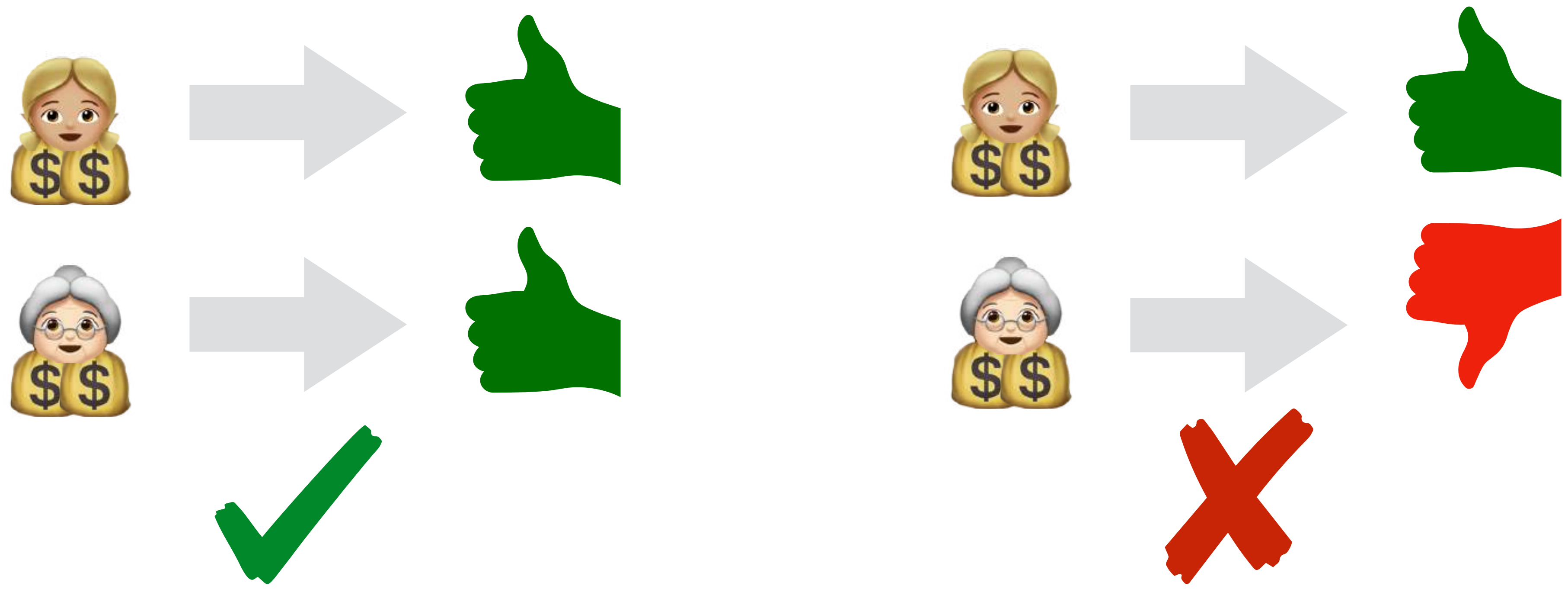
226 subscribers

Computer scientists and statisticians have devised numerous mathematical criteria to define what it means for a classifier or a model to be fair. The proliferation of these definitions represents an attempt to make technical sense of

Dependency Fairness [Galhotra17]

The **classification** is **independent of** the values of the **sensitive inputs**





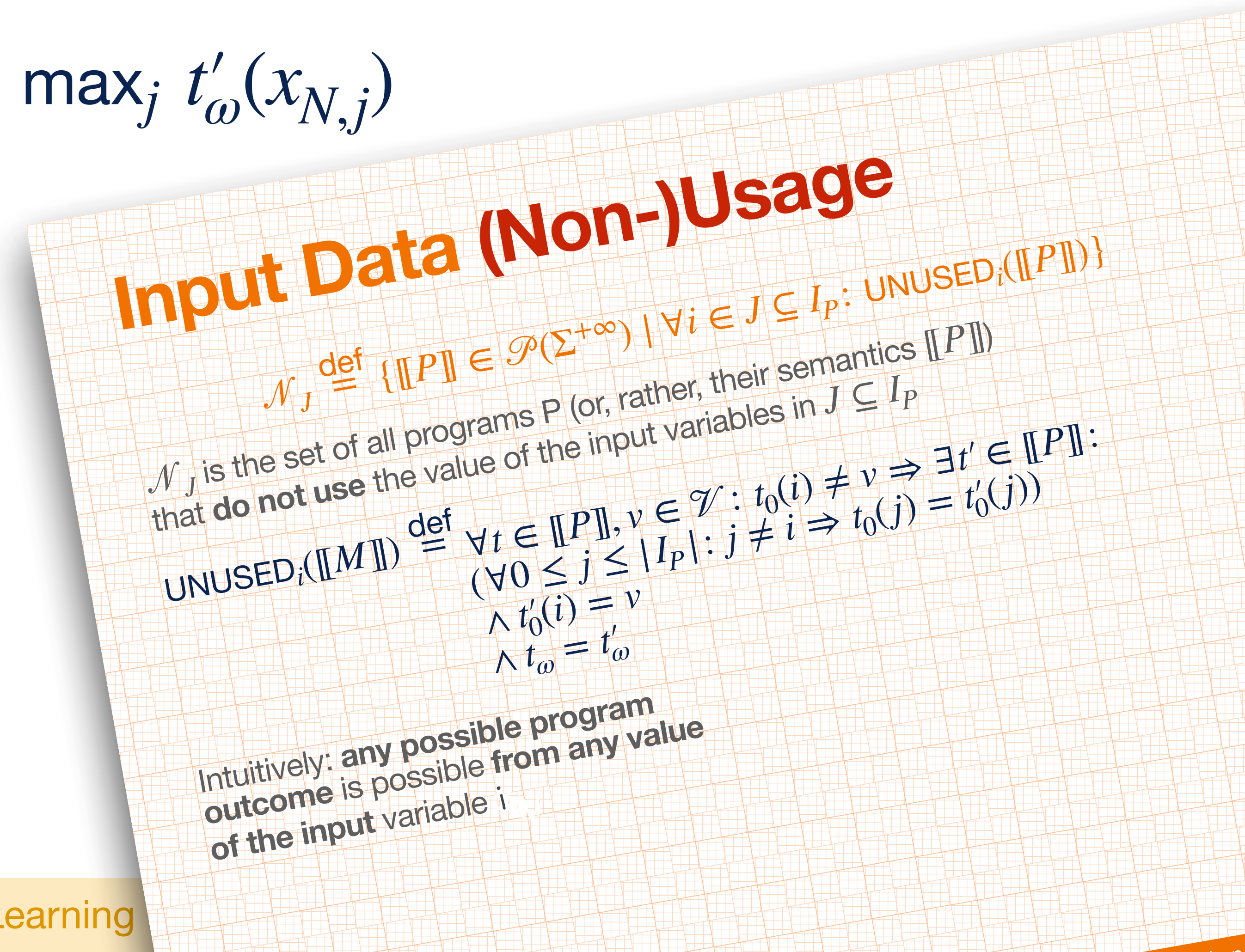
Dependency Fairness

$$\mathcal{F}_i \overset{\text{def}}{=} \{[\![M]\!] \in \mathcal{P}(\Sigma^*) \mid \text{UNUSED}_i([\![M]\!])\}$$

$\mathcal{F}_i$ is the set of all neural networks M (or, rather, their semantics $[\![M]\!]$) that **do not use** the value of the sensitive input node $x_{0,i}$ for classification

$$\begin{aligned}\text{UNUSED}_i([\![M]\!]) \overset{\text{def}}{=} & \forall t \in [\![M]\!], v \in \mathcal{R} \colon t_0(x_{0,i}) \neq v \Rightarrow \exists t' \in [\![M]\!] \colon \\ & (\forall 0 \leq j \leq |L_0| \colon j \neq i \Rightarrow t_0(x_{0,j}) = t'_0(x_{0,j})) \\ & \wedge t'_0(x_{0,i}) = v \\ & \wedge \max_j t_\omega(x_{N,j}) = \max_j t'_\omega(x_{N,j})\end{aligned}$$

Intuitively: **any possible classification outcome** is possible **from any value of the sensitive input** node $x_{0,i}$



Input Data (Non-)Usage

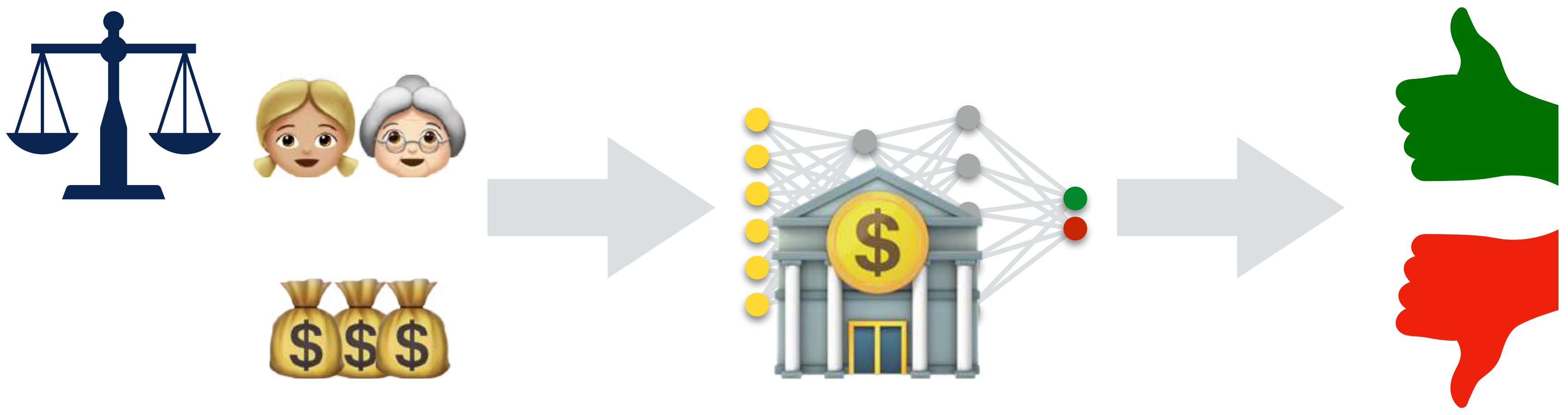
$$\mathcal{N}_J \overset{\text{def}}{=} \{[\![P]\!] \in \mathcal{P}(\Sigma^{+\infty}) \mid \forall i \in J \subseteq I_P \colon \text{UNUSED}_i([\![P]\!])\}$$

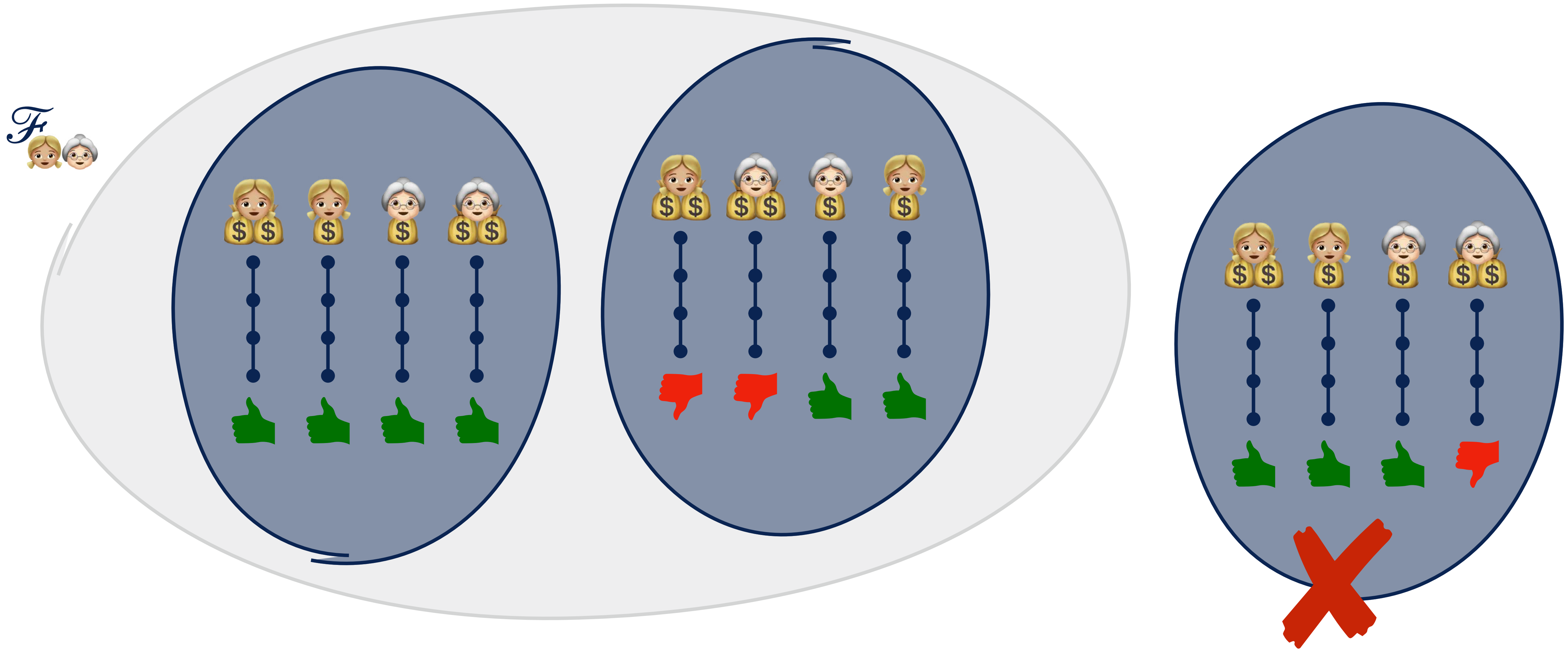
$\mathcal{N}_J$ is the set of all programs P (or, rather, their semantics $[\![P]\!]$) that **do not use** the value of the input variables in $J \subseteq I_P$

$$\begin{aligned}\text{UNUSED}_i([\![M]\!]) \overset{\text{def}}{=} & \forall t \in [\![P]\!], v \in \mathcal{V} \colon t_0(i) \neq v \Rightarrow \exists t' \in [\![P]\!] \colon \\ & (\forall 0 \leq j \leq |I_P| \colon j \neq i \Rightarrow t_0(j) = t'_0(j)) \\ & \wedge t'_0(i) = v \\ & \wedge t_\omega = t'_\omega\end{aligned}$$

Intuitively: **any possible program outcome** is possible **from any value of the input** variable i

Dependency Fairness





Dependency Fairness

$$\mathcal{F}_i \overset{\text{def}}{=} \{[\![M]\!] \in \mathcal{P}(\Sigma^*) \mid \text{UNUSED}_i([\![M]\!])\}$$

$\mathcal{F}_i$ is the set of all neural networks M (or, rather, their semantics $[\![M]\!]$) that **do not use** the value of the sensitive input node $x_{0,i}$ for classification

$$\begin{aligned}\text{UNUSED}_i([\![M]\!]) \overset{\text{def}}{=}\ & \forall t \in [\![M]\!], v \in \mathcal{R} : t_0(x_{0,i}) \neq v \Rightarrow \exists t' \in [\![M]\!] : \\ & (\forall 0 \leq j \leq |L_0| : j \neq i \Rightarrow t_0(x_{0,j}) = t'_0(x_{0,j})) \\ & \wedge t'_0(x_{0,i}) = v \\ & \wedge \max_j t_\omega(x_{N,j}) = \max_j t'_\omega(x_{N,j})\end{aligned}$$

Intuitively: **any possible classification outcome** is possible **from any value of the sensitive input** node $x_{0,i}$

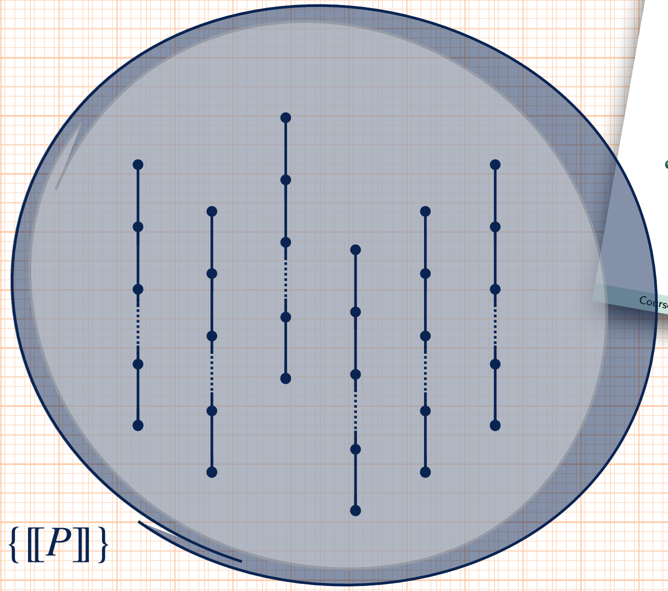
Theorem

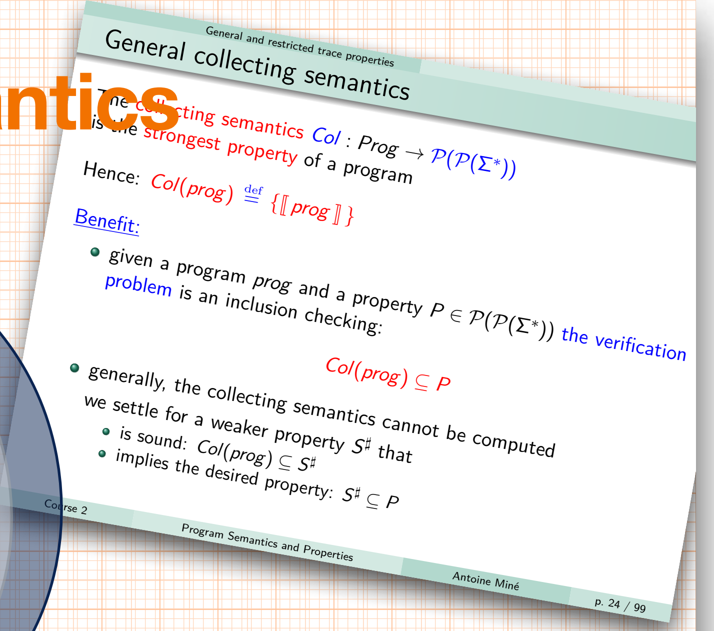
$$M \models \mathcal{F}_i \Leftrightarrow \{[\![M]\!]\} \subseteq \mathcal{F}_i$$

Hierarchy of Semantics

$$\begin{tikzcd}
\text{parallel semantics} & \{[M]\}^{\parallel}_{\rightsquigarrow} & & \\
& & \llbracket M \rrbracket_{\rightsquigarrow} \arrow[ul, "\alpha_{\parallel}"'] & \text{dependency semantics} \\
& \{[M]\}^{\parallel}_{\bullet} \arrow[uu, "\alpha_{\rightsquigarrow}"] & & \\
& & \llbracket M \rrbracket_{\bullet} \arrow[ul, "\alpha_{\parallel}"'] \arrow[uu, "\alpha_{\rightsquigarrow}"] & \text{outcome semantics} \\
& \{[M]\}^{\parallel} \arrow[uu, "\alpha_{\bullet}"] & & \\
& & \{\llbracket M \rrbracket\} \arrow[ul, "\alpha_{\parallel}"'] \arrow[uu, "\alpha_{\bullet}"] & \text{collecting semantics}
\end{tikzcd}$$

Collecting Semantics





General collecting semantics

The collecting semantics $Col : Prog \to \mathcal{P}(\mathcal{P}(\Sigma^*))$ is the strongest property of a program

Hence: $Col(prog) \stackrel{\text{def}}{=} \{[\![\, prog \,]\!]\}$

Benefit:

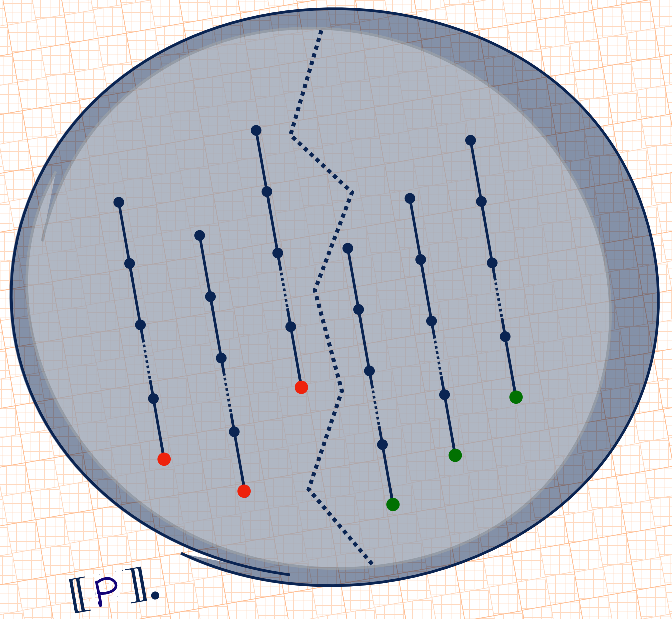
- given a program *prog* and a property $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$ the verification problem is an inclusion checking:

$$Col(prog) \subseteq P$$

- generally, the collecting semantics cannot be computed
  we settle for a weaker property $S^\sharp$ that
  - is sound: $Col(prog) \subseteq S^\sharp$
  - implies the desired property: $S^\sharp \subseteq P$

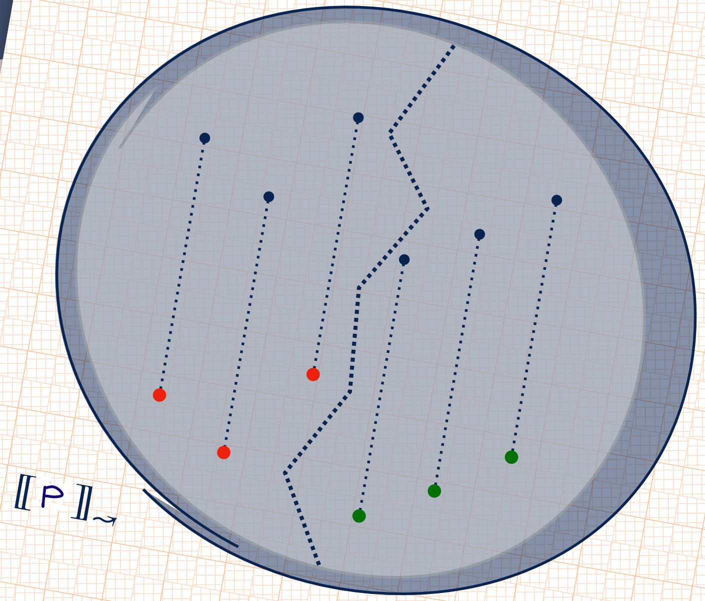
Outcome Semantics

partitioning a set of traces that satisfies input data (non-)usage **with respect to the program outcome** yields sets of traces that also satisfy input data (non-)usage

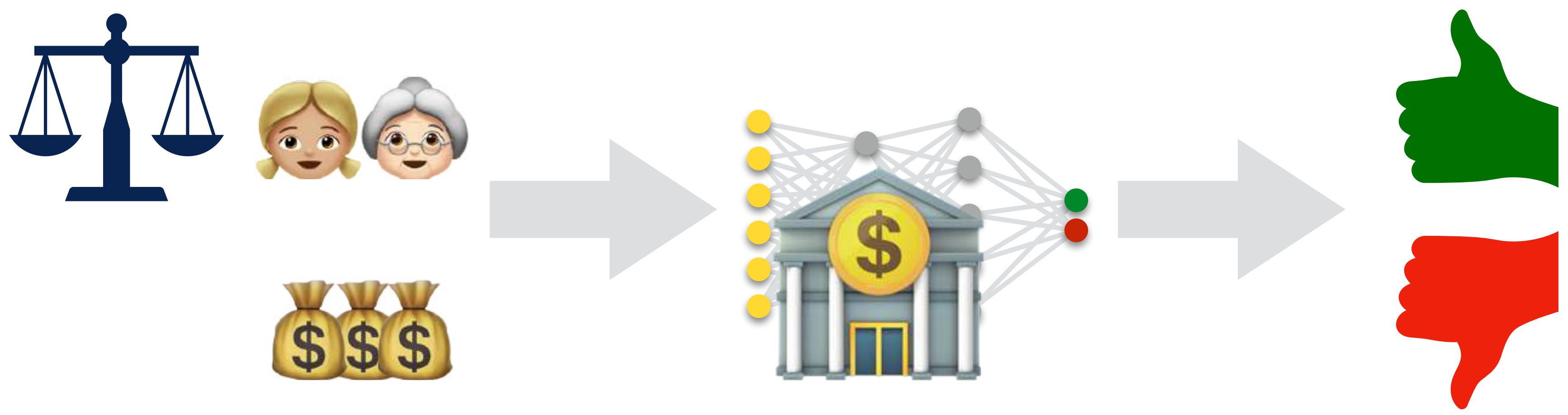


Dependency Semantics

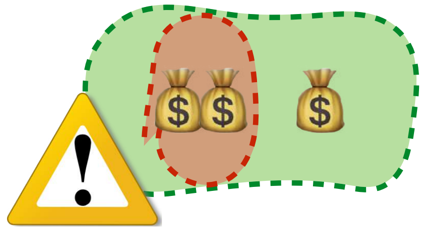
to reason about input data (non-)usage **we do not need to consider all intermediate computations** between the initial and final states of a trace (if any)

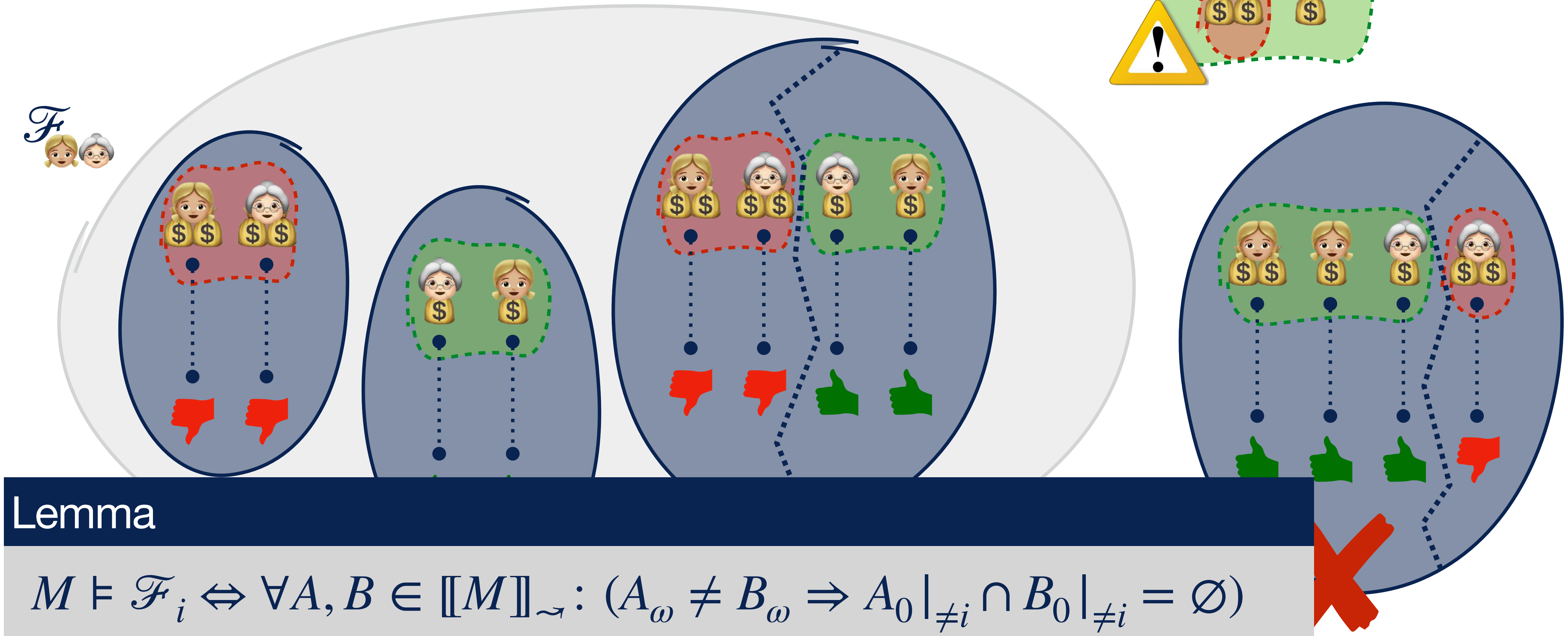


Dependency Semantics



💡 **partitioning with respect to the outcome** classification **induces a partition of the** space of **values** of the input nodes ***used* for classification**





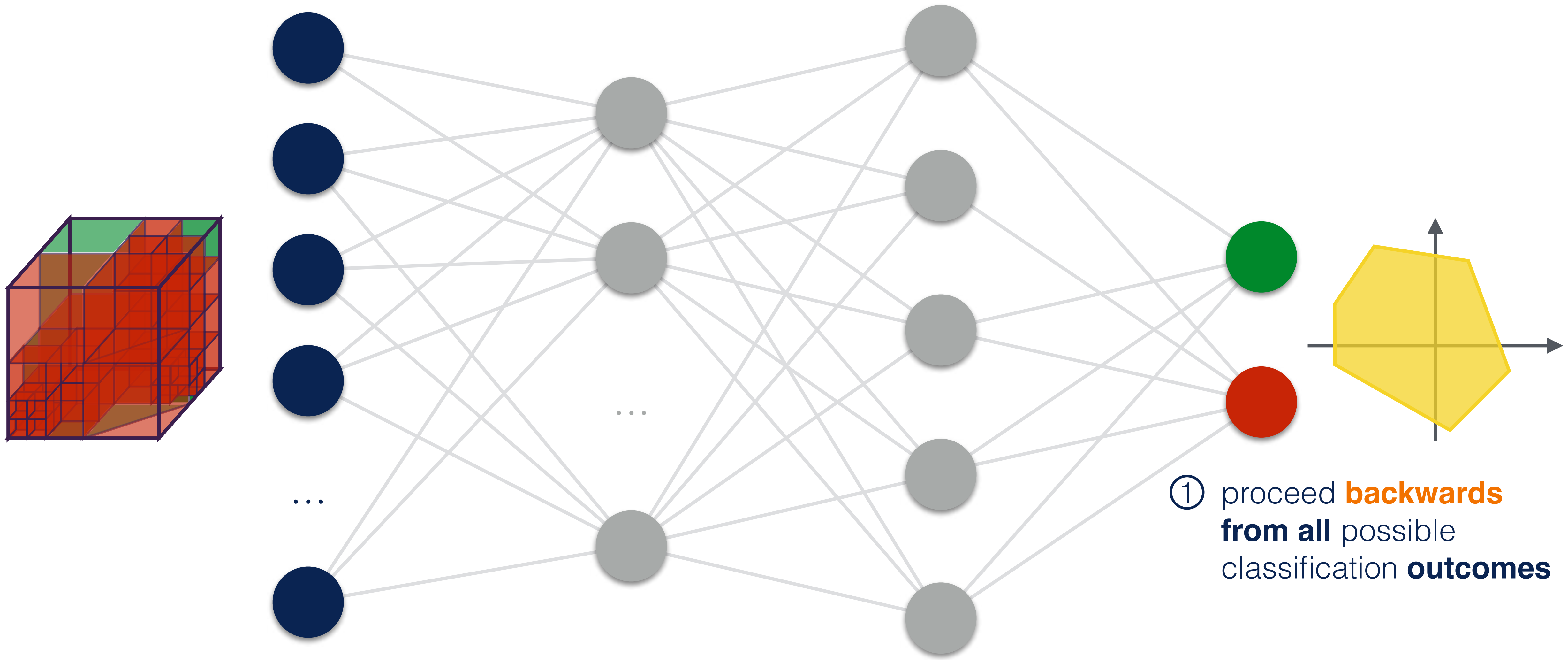
Lemma

$$M \models \mathscr{F}_i \Leftrightarrow \forall A, B \in [\![M]\!]_{\rightsquigarrow} : (A_\omega \neq B_\omega \Rightarrow A_0|_{\neq i} \cap B_0|_{\neq i} = \emptyset)$$

Naïve Abstraction

Naïve Backward Analysis

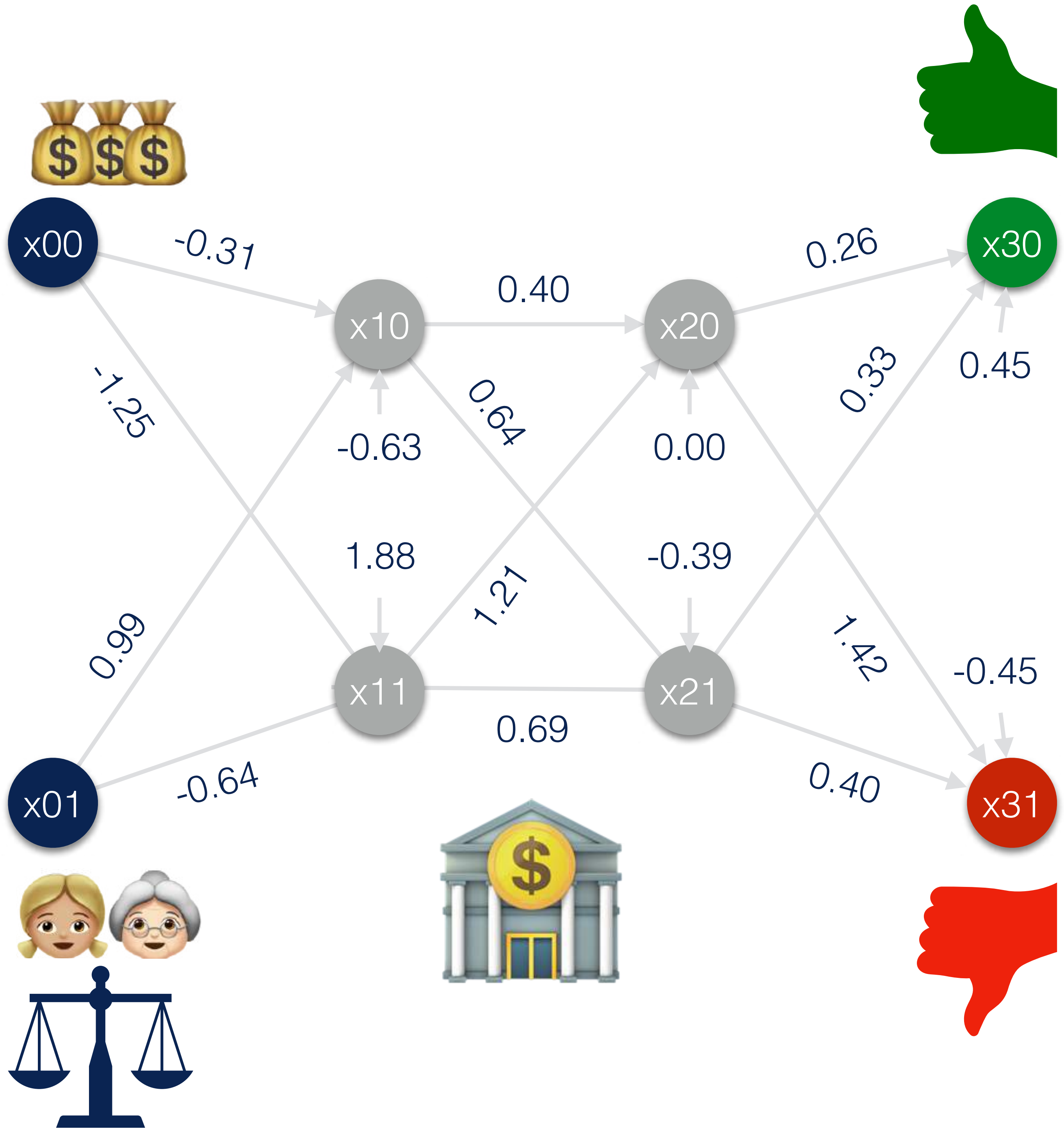
② **forget** the values of the **sensitive input** nodes



① proceed **backwards** **from all** possible classification **outcomes**

③ check for **intersection**:
empty → ✓ **fair**
otherwise → 🚨 **alarm**

Naïve Backward Analysis



```
x00 = input()
x01 = input()
      …                …
x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88
      …                …
x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21
1.16 * x20 + 0.07 * x21 ≤ 0.90     1.16 * x20 + 0.07 * x21 ≥ 0.90
x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)
x30 ≥ x31     x31 ≥ x30
return '👍' if x31 < 30 else '👎'
```

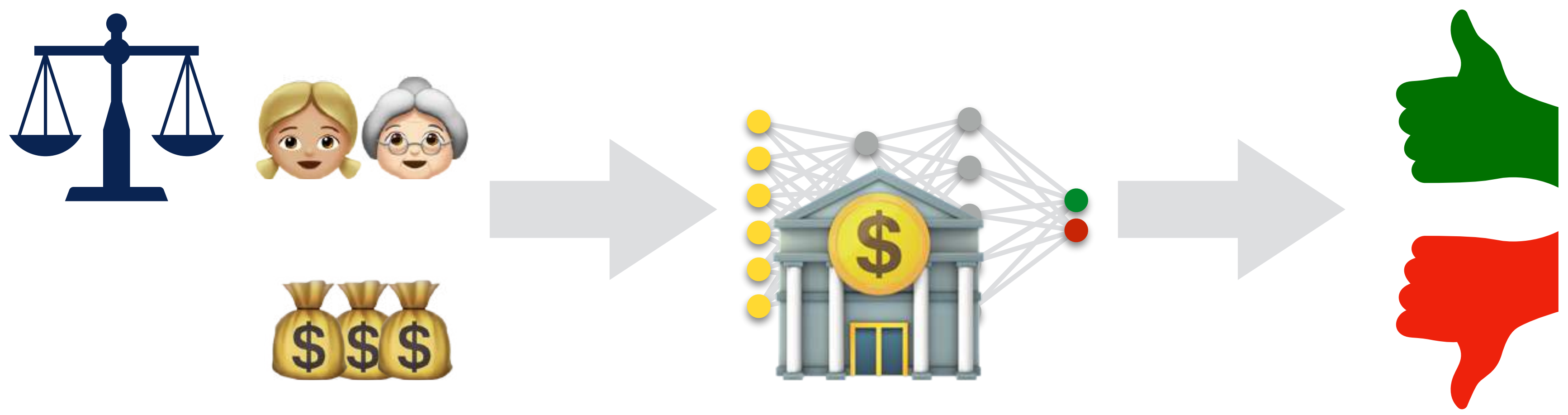
too many disjunctions!

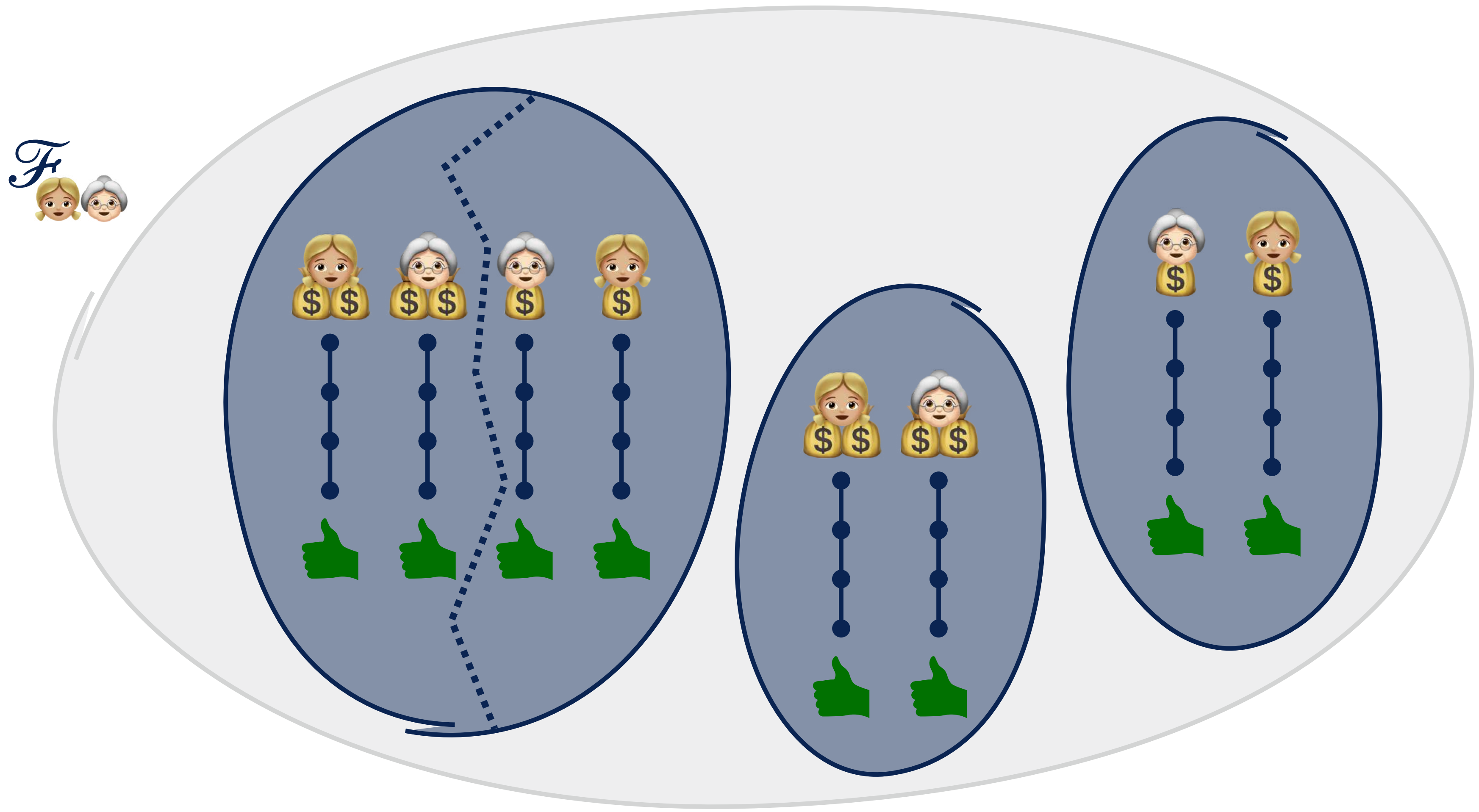
Back to the Semantics…

Hierarchy of Semantics

$$\begin{tikzcd}
\text{parallel semantics} & \{[M]\}^{\parallel}_{\rightsquigarrow} & & \\
& & [\![M]\!]_{\rightsquigarrow} \arrow[ul, "\alpha_{\parallel}"'] & \text{dependency semantics} \\
& \{[M]\}^{\parallel}_{\bullet} \arrow[uu, "\alpha_{\rightsquigarrow}"] & & \\
& & [\![M]\!]_{\bullet} \arrow[uu, "\alpha_{\rightsquigarrow}"] \arrow[ul, "\alpha_{\parallel}"'] & \text{outcome semantics} \\
& \{[M]\}^{\parallel} \arrow[uu, "\alpha_{\bullet}"] & & \\
& & \{[\![M]\!]\} \arrow[uu, "\alpha_{\bullet}"] \arrow[ul, "\alpha_{\parallel}"'] & \text{collecting semantics}
\end{tikzcd}$$

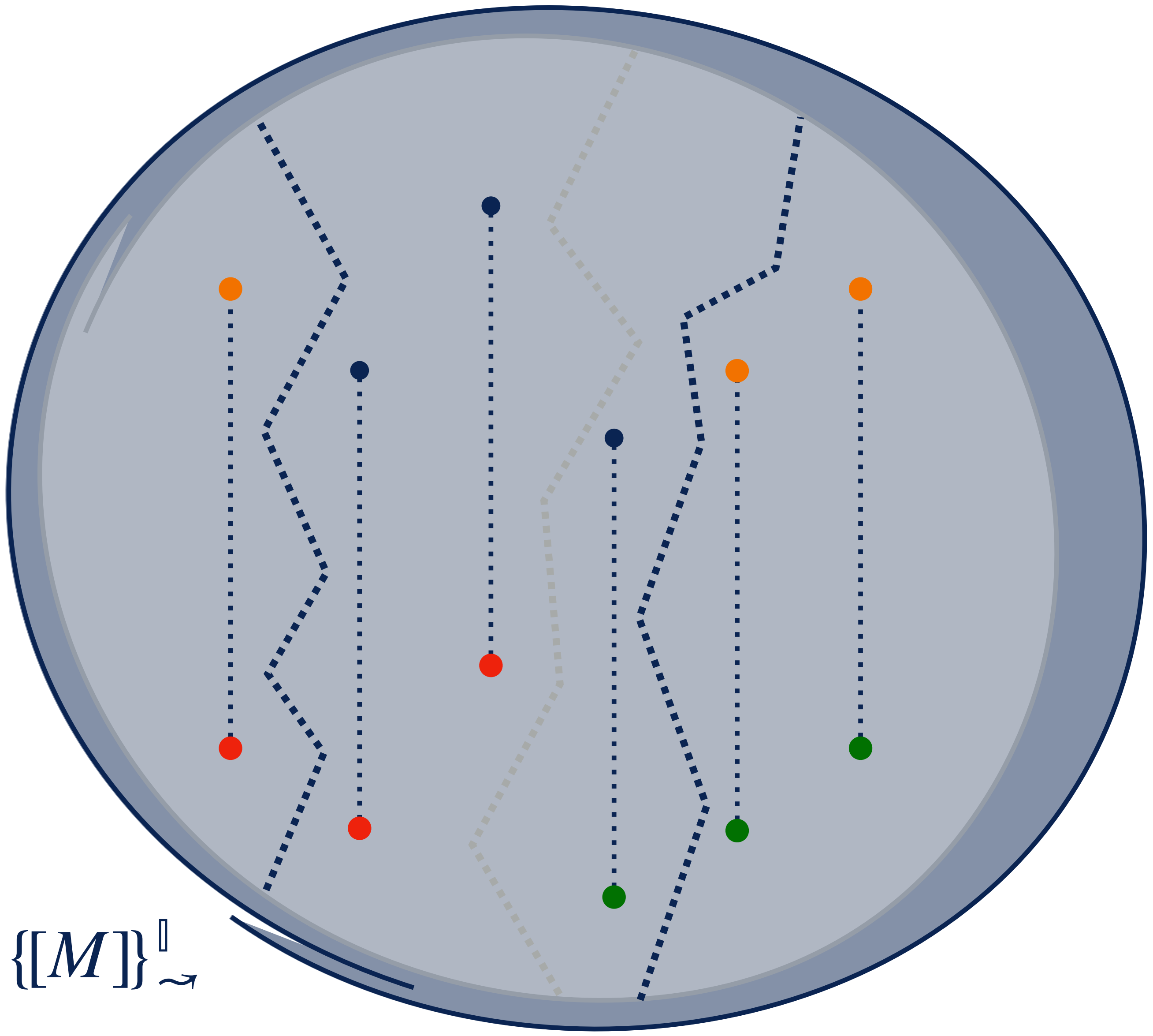
Parallel Semantics





💡 **partitioning** a set of traces that satisfies dependency fairness **with respect to the non-sensitive inputs** yields sets of traces that also satisfy dependency fairness

Parallel Semantics



💡 **partitioning** a set of traces that satisfies dependency fairness **with respect to the non-sensitive inputs** yields sets of traces that also satisfy dependency fairness

Parallel Semantics

$$\begin{tikzcd}
\langle \mathscr{P}(\mathscr{P}(\Sigma \times \Sigma)), \subseteq_{\rightsquigarrow} \rangle \arrow[r, bend right, "\alpha_{\parallel}"'] & \langle \mathscr{P}(\mathscr{P}(\Sigma \times \Sigma)), \subseteq_{\parallel} \rangle \arrow[l, bend right, "\gamma_{\parallel}"']
\end{tikzcd}$$

$$\alpha_{\parallel}(S) \overset{\text{def}}{=} \{ \{ \langle t_0, t_\omega \rangle \in R \mid t_0 \in I \} \mid R \in S \wedge I \in \mathbb{I} \}$$ **parallel abstraction**

$$\{[M]\}^{\parallel}_{\rightsquigarrow} \overset{\text{def}}{=} \alpha_{\parallel}([\![M]\!]_{\rightsquigarrow})$$
$$= \{ \{ \langle t_0, t_\omega \rangle \in \Sigma \times \Sigma \mid t \in [\![M]\!] \wedge t_0 \in I \wedge t_\omega \in O \} \mid I \in \mathbb{I} \wedge O \in \mathbb{O} \}$$

Theorem

$$M \models \mathcal{F}_i \Leftrightarrow \gamma_{\rightsquigarrow}(\{[M]\}^{\parallel}_{\rightsquigarrow}) \subseteq \mathcal{F}_i$$

Lemma

$$M \models \mathcal{F}_i \Leftrightarrow \forall I \in \mathbb{I} \colon \forall A, B \in \{[M]\}^{\parallel}_{\rightsquigarrow} \colon (A^I_\omega \neq B^I_\omega \Rightarrow A^I_0|_{\neq i} \cap B^I_0|_{\neq i} = \emptyset)$$

Better Abstraction

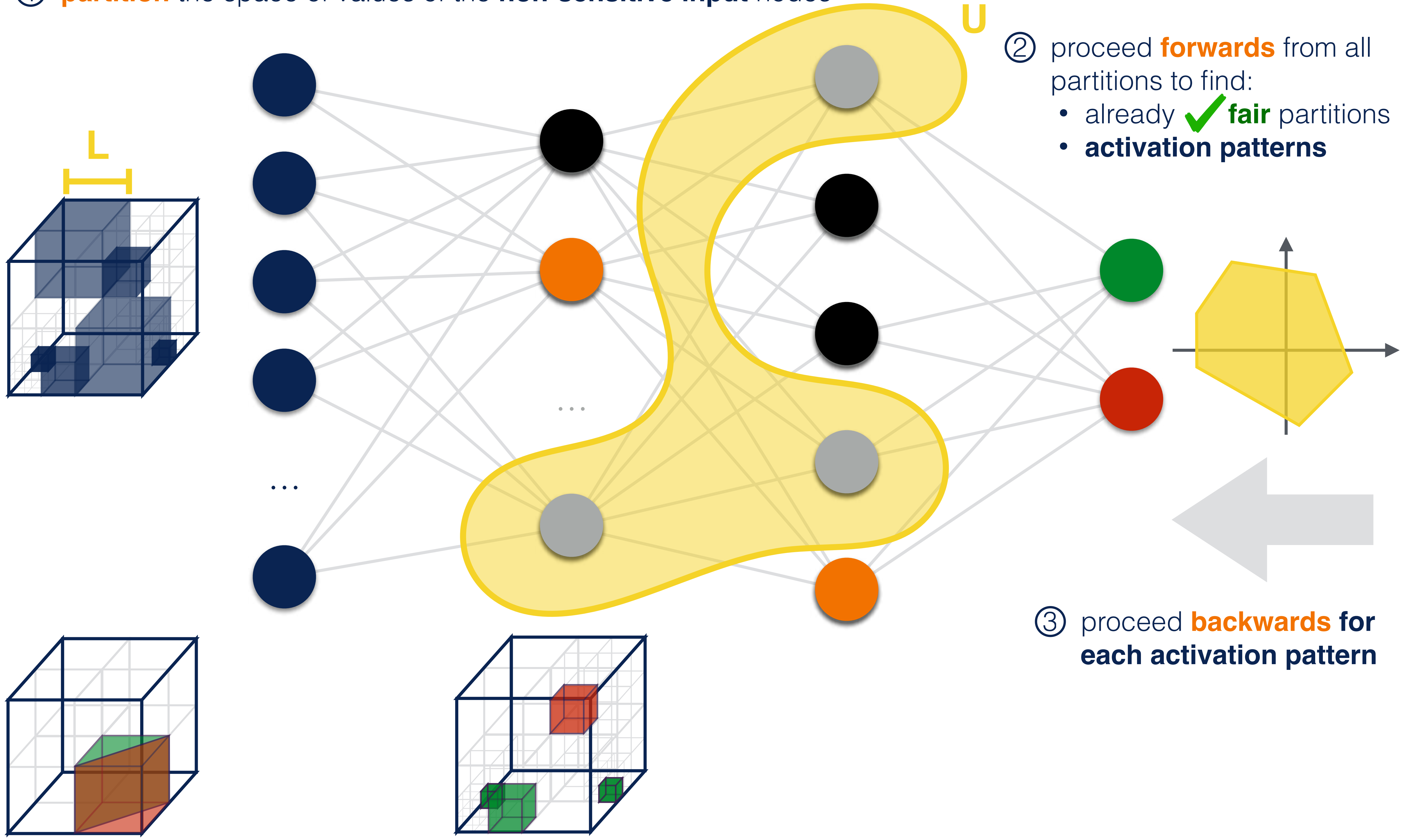
Forward and Backward Analysis

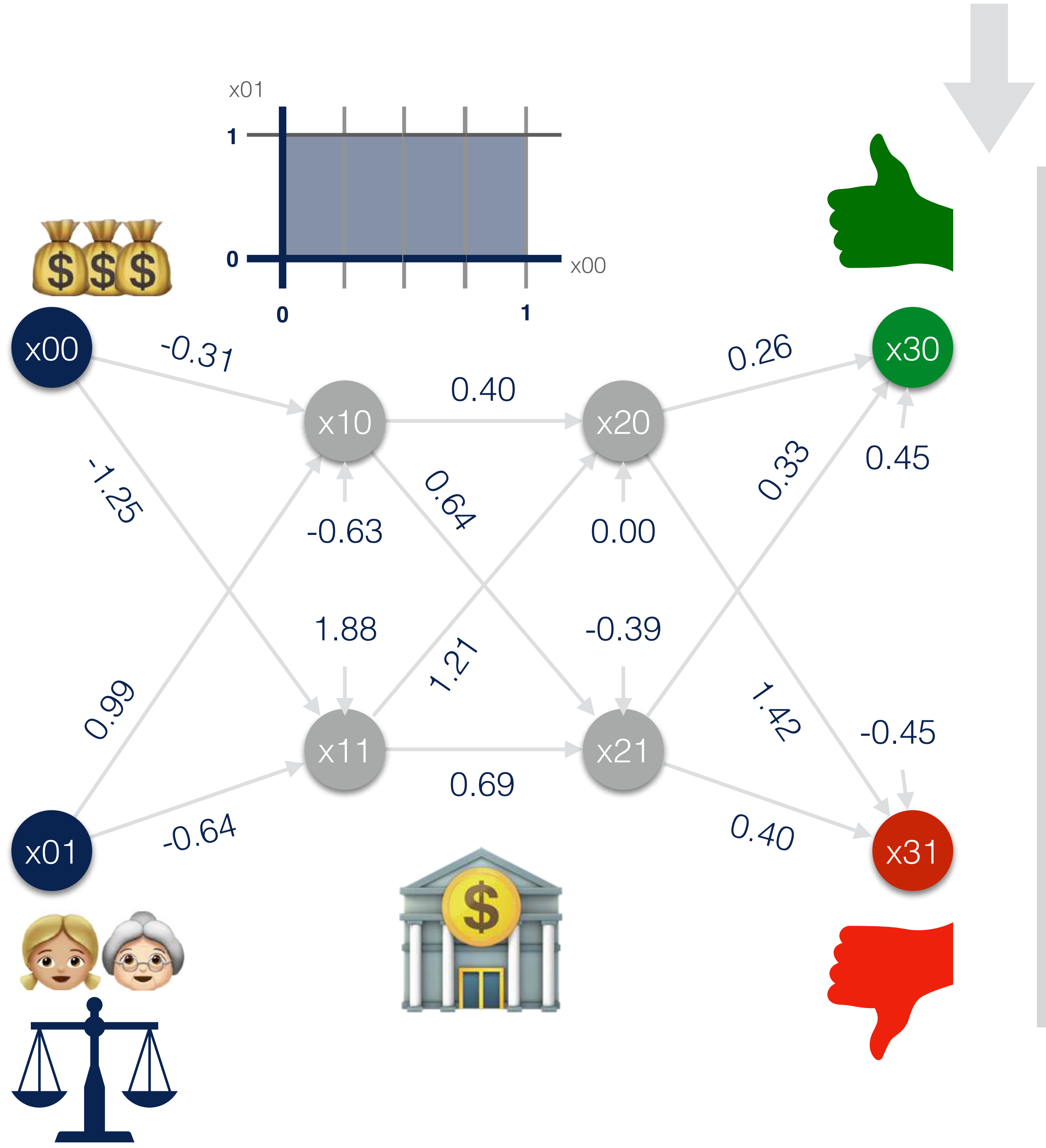
① **partition** the space of values of the **non-sensitive input** nodes

② proceed **forwards** from all partitions to find:

- already ✓ **fair** partitions
- **activation patterns**

③ proceed **backwards for each activation pattern**

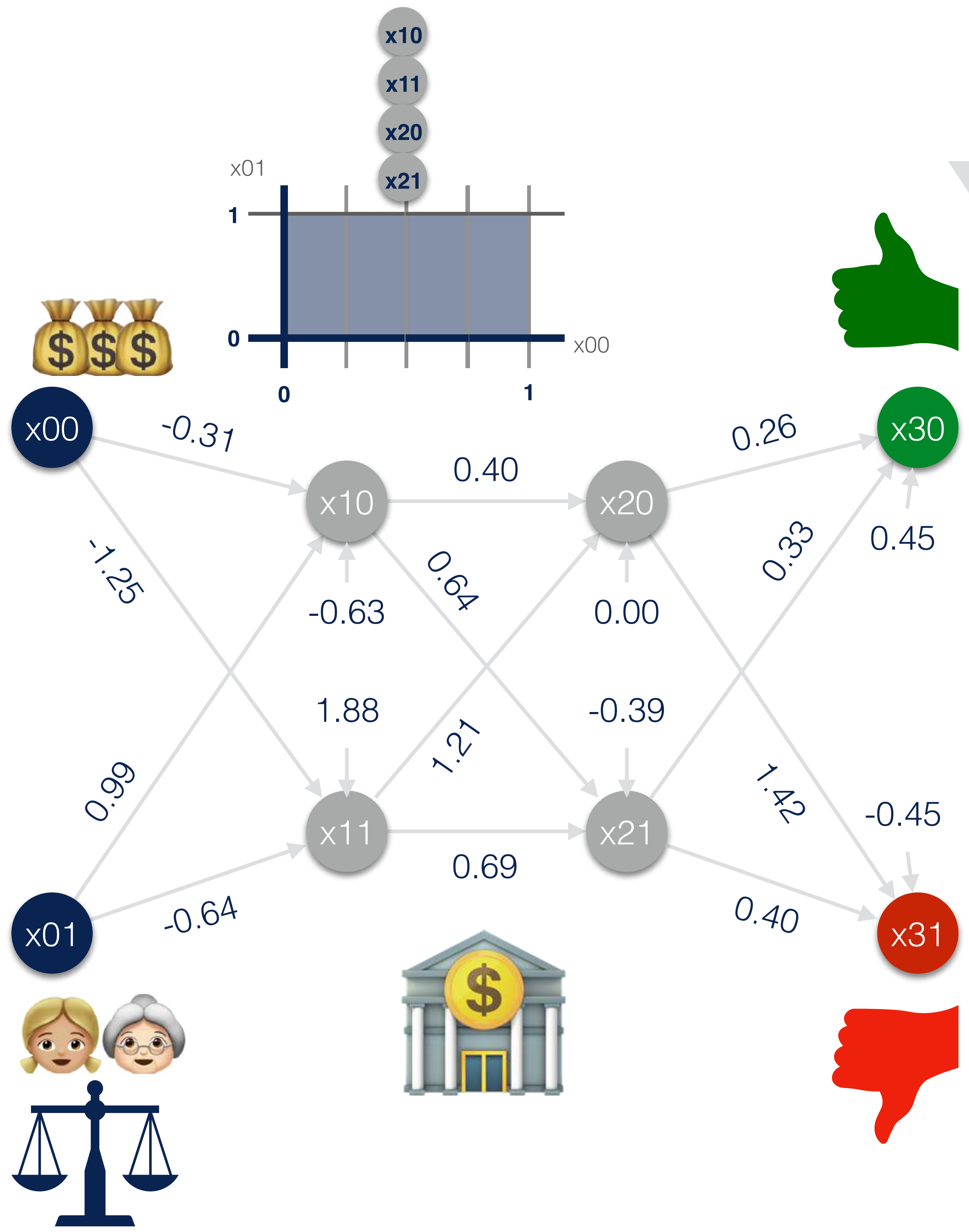




L = 0.25

U = 2

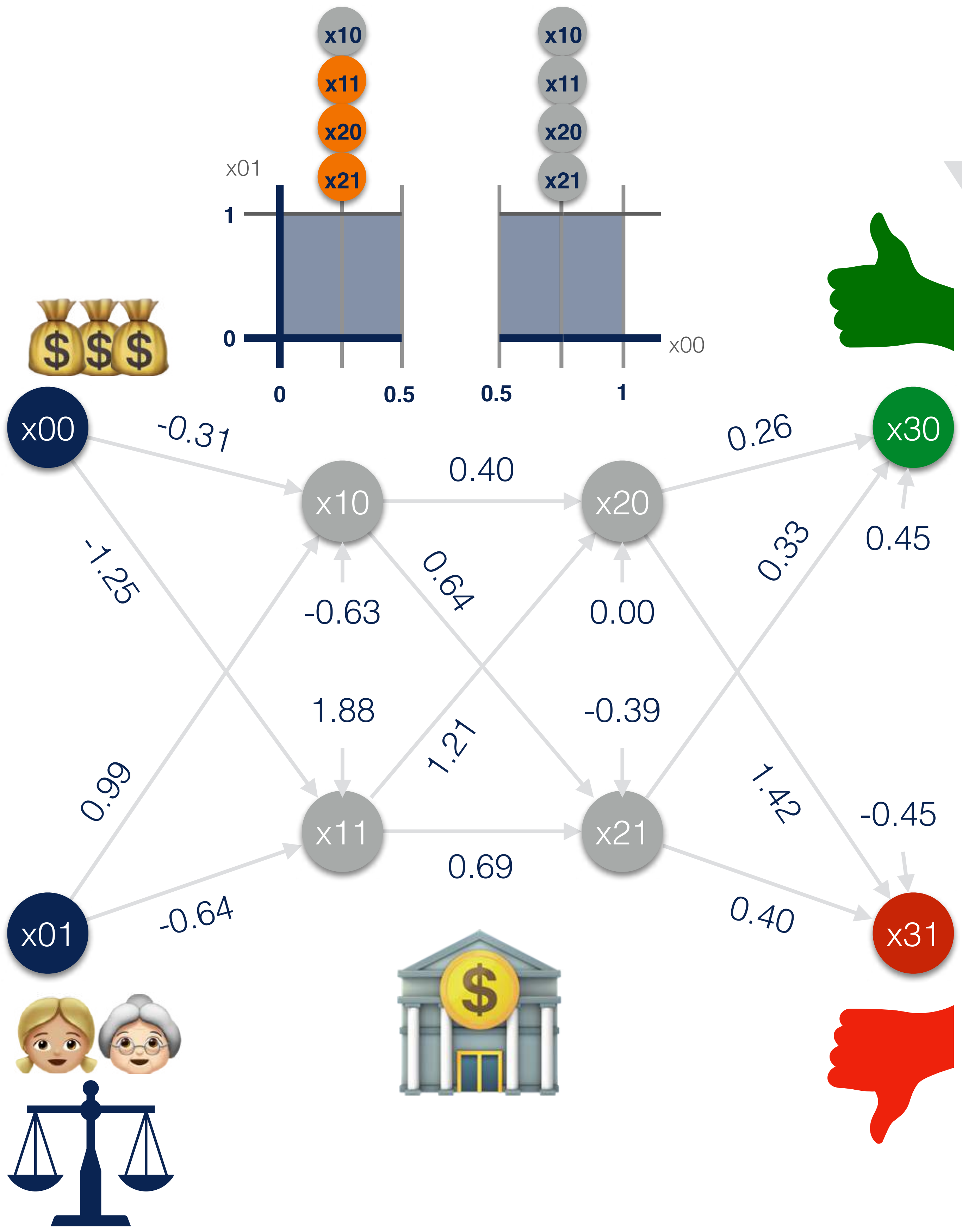
```
x00 = input()
x01 = input()

x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88

x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '👍' if x31 < 30 else '👎'
```



$\mathbf{L}$ = 0.25

$\mathbf{U}$ = 2

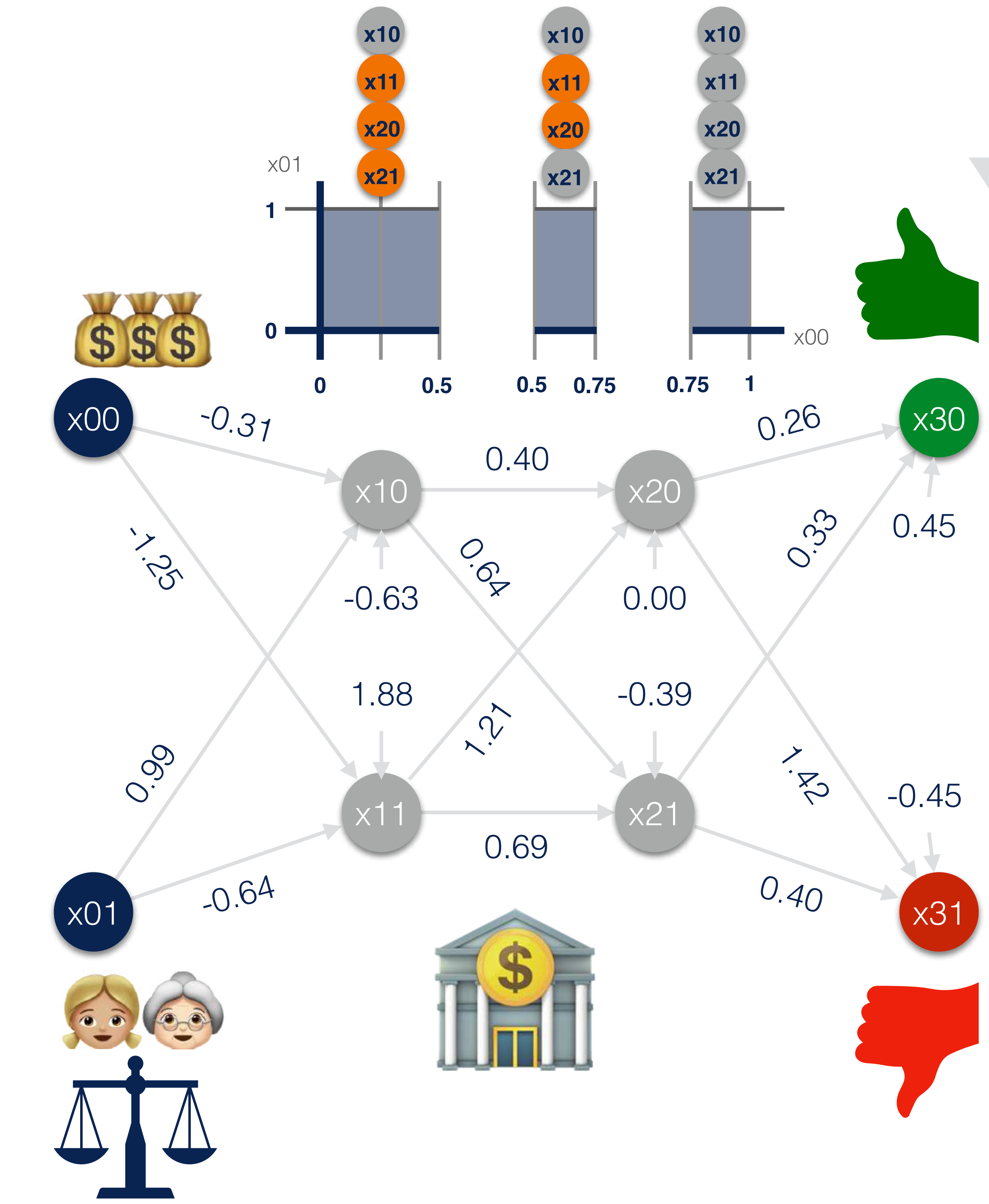
```
x00 = input()
x01 = input()

x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88

x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '👍' if x31 < 30 else '👎'
```



L = 0.25

U = 2

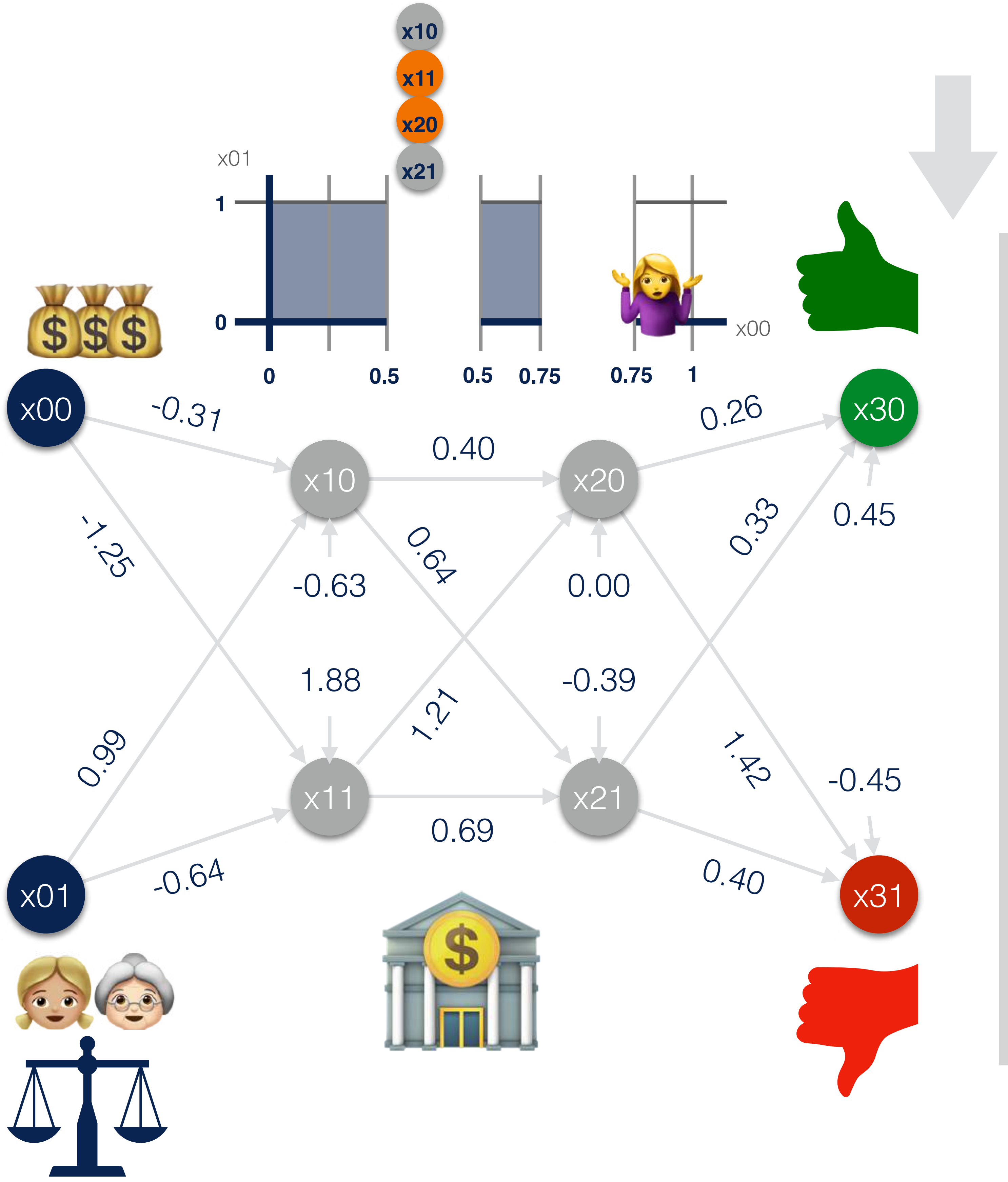
```
x00 = input()
x01 = input()

x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88

x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '👍' if x31 < 30 else '👎'
```



$\mathbf{L}$ = 0.25

$\mathbf{U}$ = 2

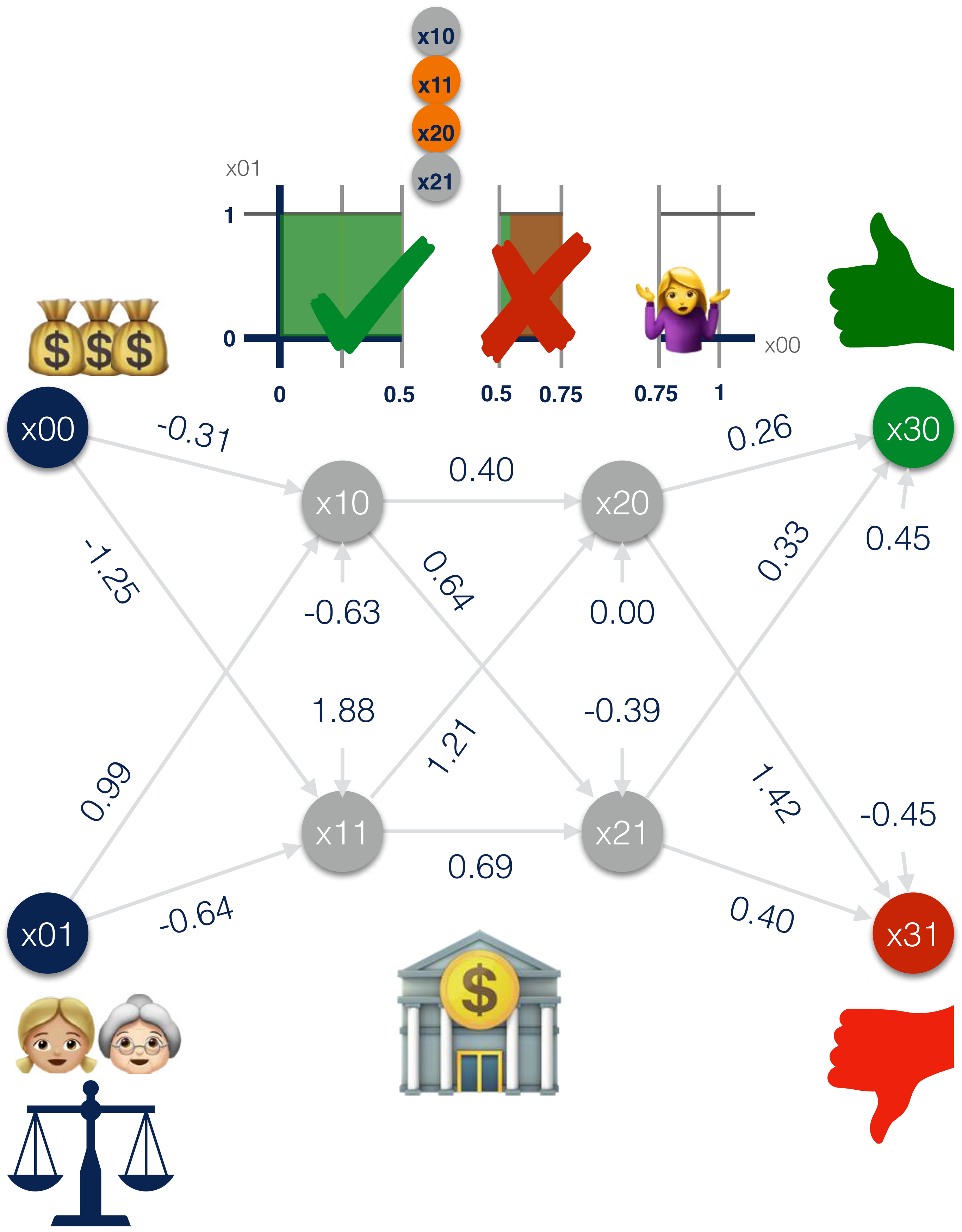
```
x00 = input()
x01 = input()

x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88

x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '👍' if x31 < 30 else '👎'
```



L = 0.25

U = 2

```
x00 = input()
x01 = input()

x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88

x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11

x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)

x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21

x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)

return '👍' if x31 < 30 else '👎'
```

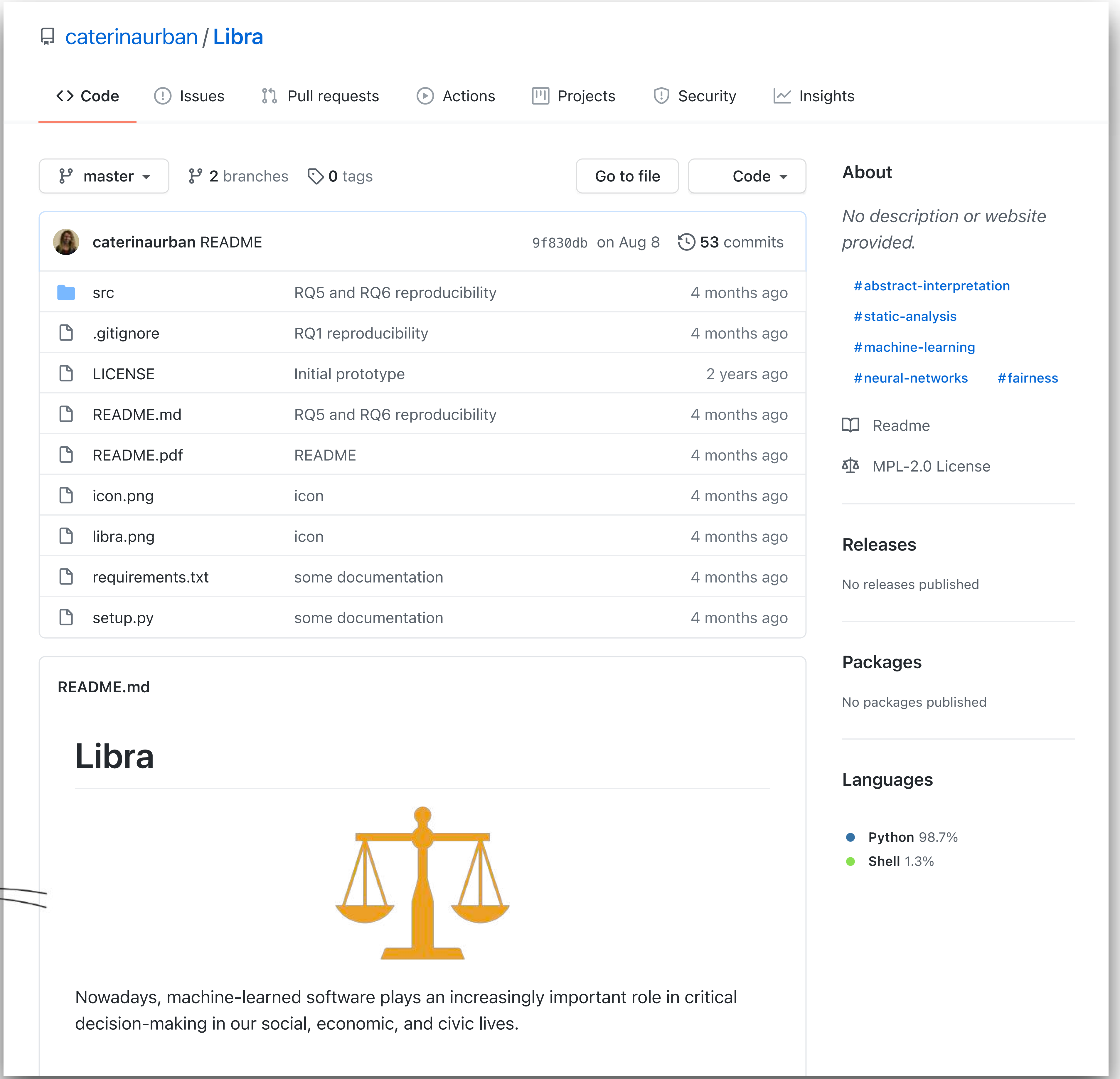


L = 0.25

U = 2

```
x00 = input()
x01 = input()
x10 = -0.31 * x00 + 0.99 * x01 + (-0.63)
x11 = -1.25 * x00 + (-0.64) * x01 + 1.88
x10 = 0 if x10 < 0 else x10
x11 = 0 if x11 < 0 else x11
x20 = 0.40 * x10 + 1.21 * x11 + 0.00
x21 = 0.64 * x10 + 0.69 * x11 + (-0.39)
x20 = 0 if x20 < 0 else x20
x21 = 0 if x21 < 0 else x21
1.16 * x20 + 0.07 * x21 ≤ 0.90        1.16 * x20 + 0.07 * x21 ≥ 0.90
x30 = 0.26 * x20 + 0.33 * x21 + 0.45
X31 = 1.42 * x20 + 0.40 * x21 + (-0.45)
x30 ≥ x31        x31 ≥ x30
return '👍' if x31 < 30 else '👎'
```

Libra



caterinaurban / Libra

Code | Issues | Pull requests | Actions | Projects | Security | Insights

master | 2 branches | 0 tags | Go to file | Code

caterinaurban README 9f830db on Aug 8 | 53 commits

src	RQ5 and RQ6 reproducibility	4 months ago
.gitignore	RQ1 reproducibility	4 months ago
LICENSE	Initial prototype	2 years ago
README.md	RQ5 and RQ6 reproducibility	4 months ago
README.pdf	README	4 months ago
icon.png	icon	4 months ago
libra.png	icon	4 months ago
requirements.txt	some documentation	4 months ago
setup.py	some documentation	4 months ago

README.md

Libra

Nowadays, machine-learned software plays an increasingly important role in critical decision-making in our social, economic, and civic lives.

About

No description or website provided.

#abstract-interpretation #static-analysis #machine-learning #neural-networks #fairness

Readme

MPL-2.0 License

Releases

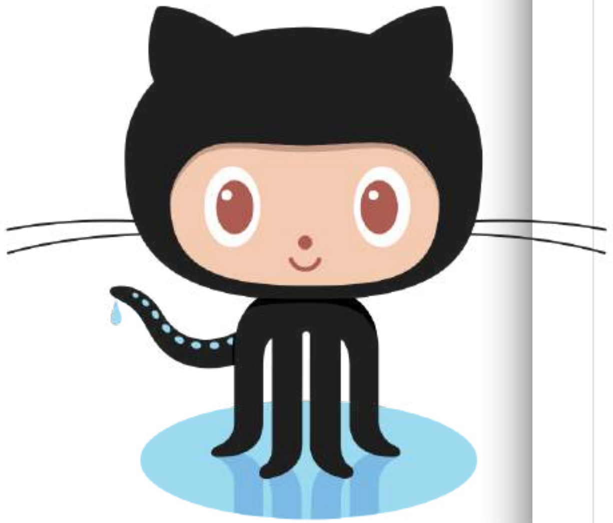
No releases published

Packages

No packages published

Languages

● **Python** 98.7%
● **Shell** 1.3%



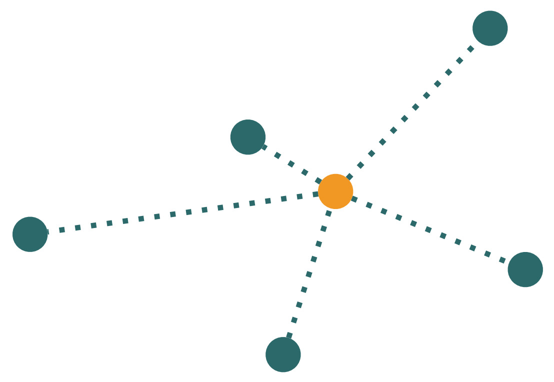
Formal Methods for Model Training

Robust Training

Minimizing the Worst-Case Loss for Each Input

Adversarial Training

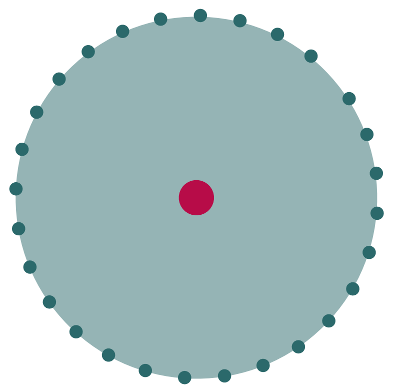
Minimizing a Lower Bound on the Worst-Case Loss for Each Input



💡 generate adversarial inputs and use them as training data

Certified Training

Minimizing an Upper Bound on the Worst-Case Loss for Each Input



💡 use upper bound as regularizer to encourage robustness

Bibliography

[Kurd03] **Zeshan Kurd, Tim Kelly**. Establishing Safety Criteria for Artificial Neural Networks. In KES, pages 63-169, 2003.

[Li19] **Jianlin Li, Jiangchao Liu, Pengfei Yang, Liqian Chen, Xiaowei Huang, and Lijun Zhang**. Analyzing Deep Neural Networks with Symbolic Propagation: Towards Higher Precision and Faster Verification. In SAS, page 296–319, 2019.

[Singh19] **Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin T. Vechev**. An Abstract Domain for Certifying Neural Networks. In POPL, pages 41:1 - 41:30, 2019.

[Mazzucato21] **Denis Mazzucato and Caterina Urban**. Reduced Products of Abstract Domains for Fairness Certification of Neural Networks. In SAS, 2021.

Bibliography

[Julian16] **Kyle D. Julian, Jessica Lopez, Jeffrey S. Brush, Michael P. Owen, Mykel J. Kochenderfer**. Policy Compression for Aircraft Collision Avoidance Systems. In DASC, pages 1–10, 2016.

[Katz17] **Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, Mykel J. Kochenderfer**. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In CAV, pages 97–117, 2017.

[Galhotra17] **Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou**. Fairness Testing: Testing Software for Discrimination. In FSE, pages 498–510, 2017.

[Urban20] **Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang**. Perfectly Parallel Fairness Certification of Neural Networks. In OOPSLA, pages 185:1–185:30, 2020.

[Urban21] **Caterina Urban and Antoine Miné**. A Review of Formal Methods applied to Machine Learning. https://arxiv.org/abs/2104.02466, 2021.