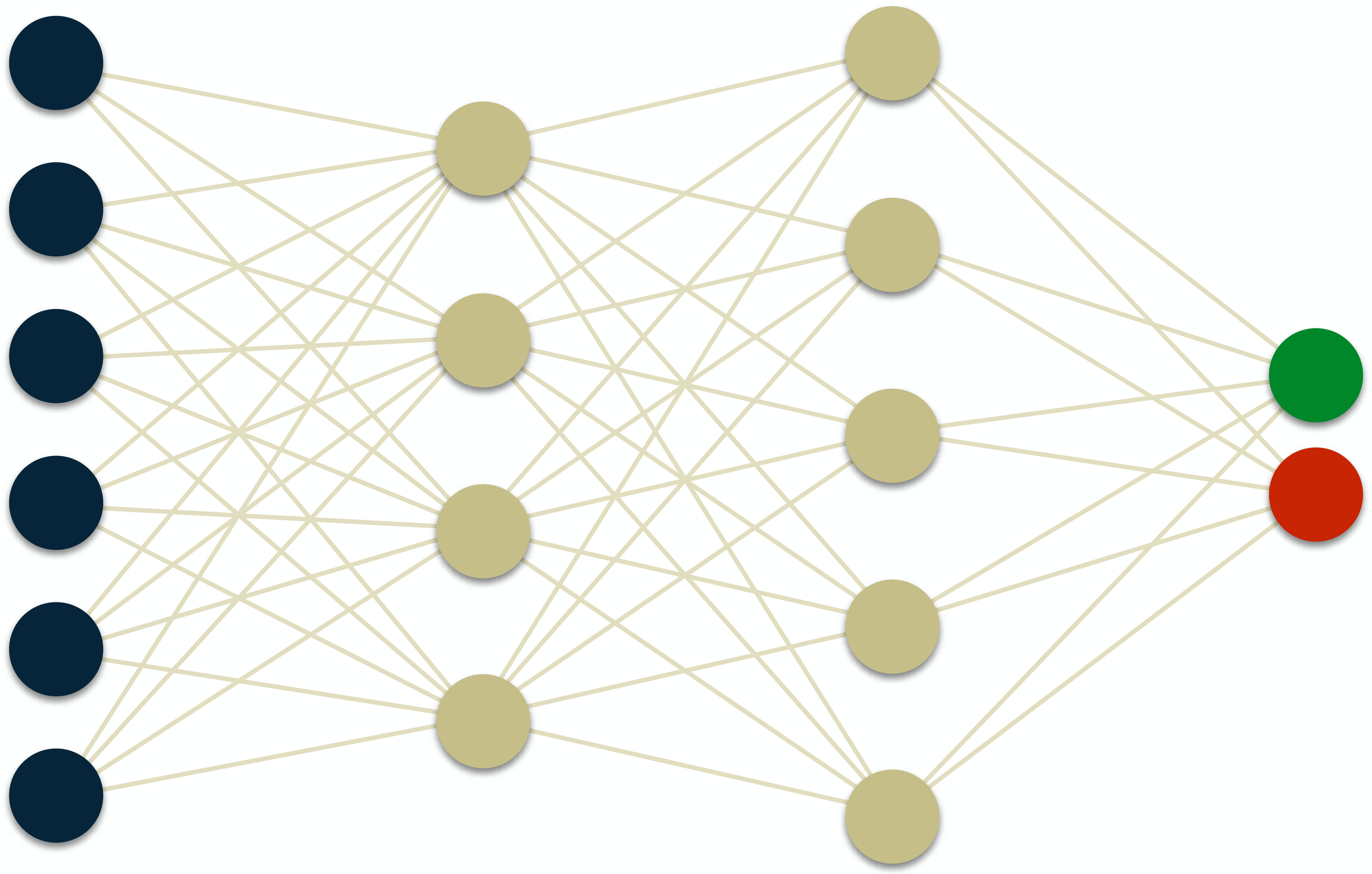
(Hyper)Safety Certification of Neural Network Surrogates for Aircraft Braking Distance Estimation

Tech Talk: Formal Verification of AI

Caterina Urban
Inria & École Normale Supérieure | Université PSL

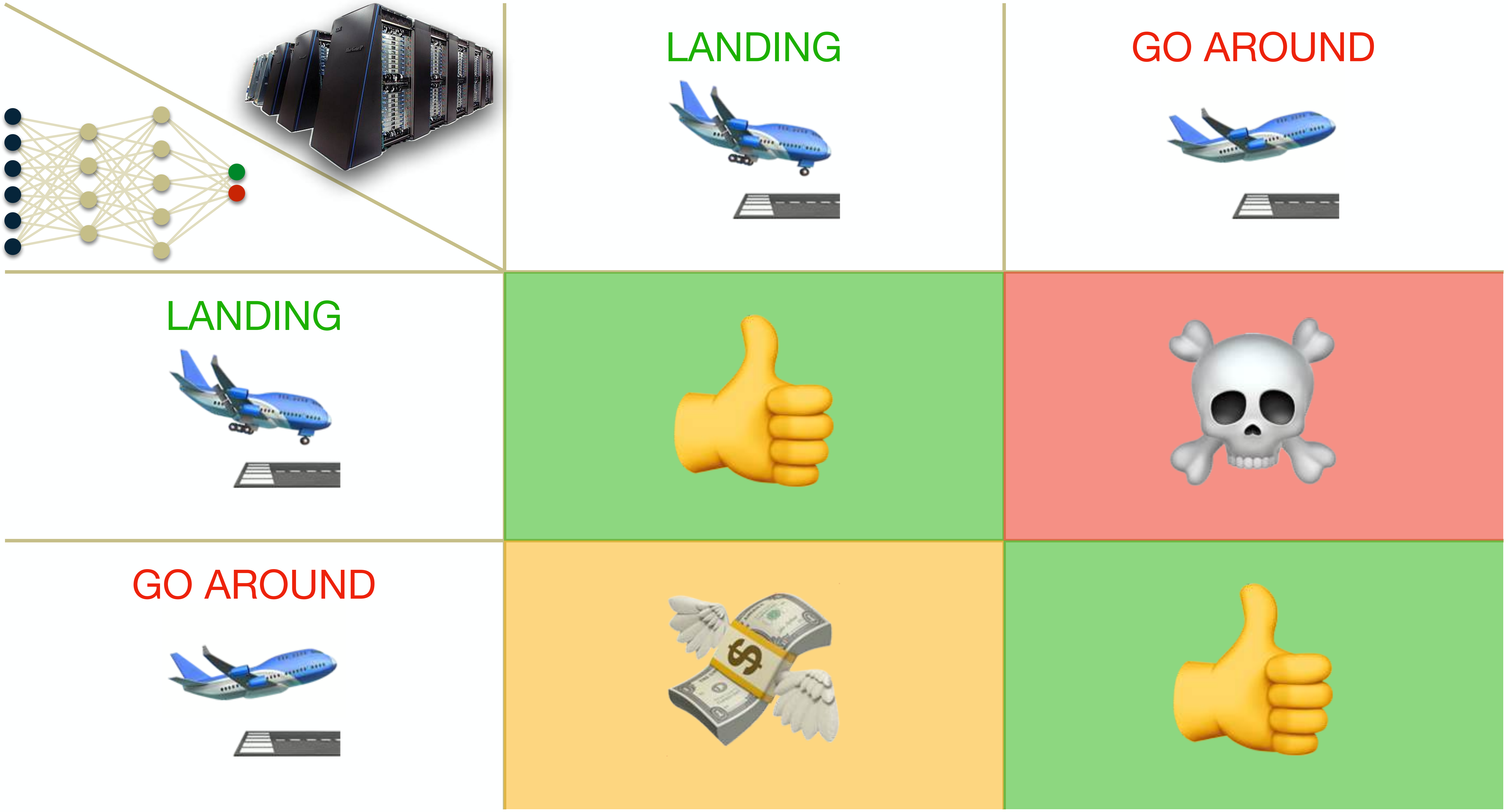
Neural Network Surrogates

Less Computing Power and Less Computing Time



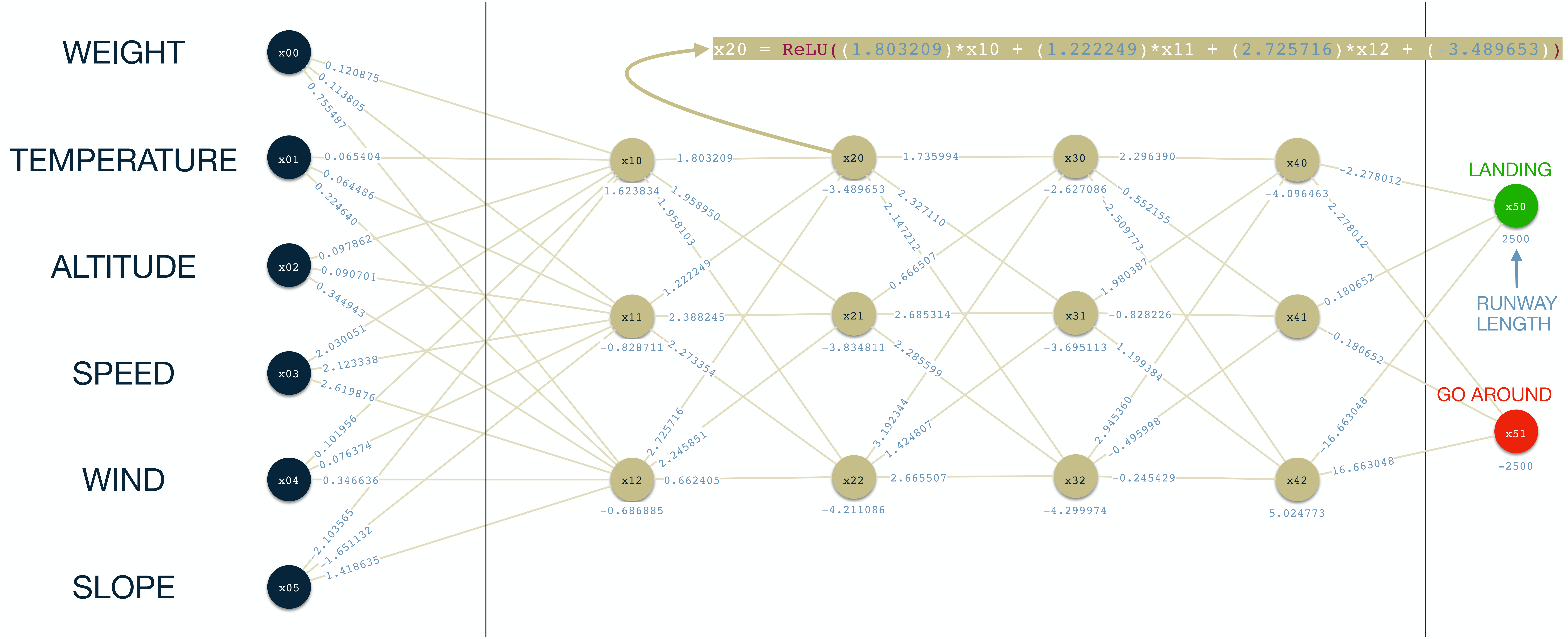
Runway Overrun Warning

Safety of Neural Network Surrogate



Runway Overrun Warning

Toy Example



Runway Overrun Warning

Toy Example

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

Neural Network Verification

Neural Network Explainability

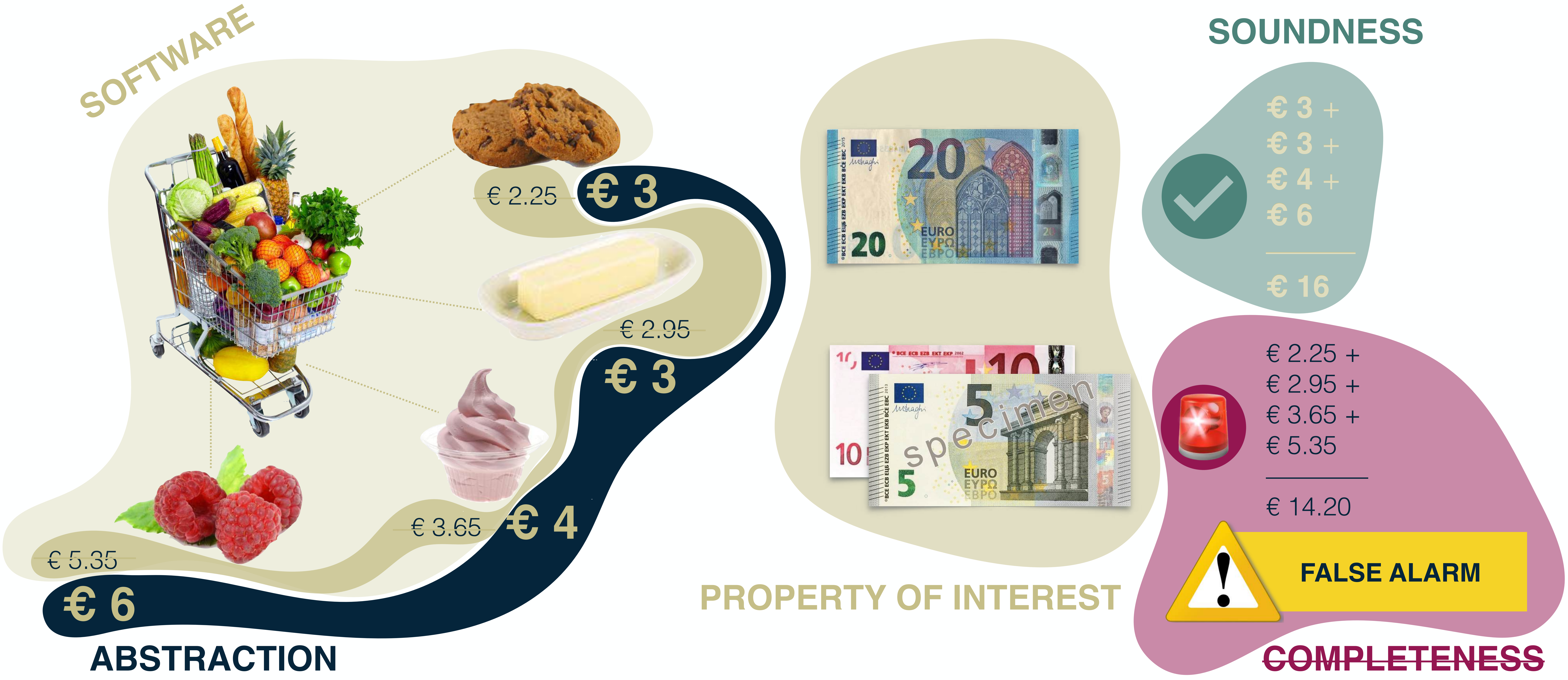
Neural Network Verification

Neural Network Explainability

(Hyper)Safety Certification of Neural Network Surrogates for Aircraft Braking Distance Estimation

= by means of Abstract Interpretation-Based Static Analysis

Abstract Interpretation

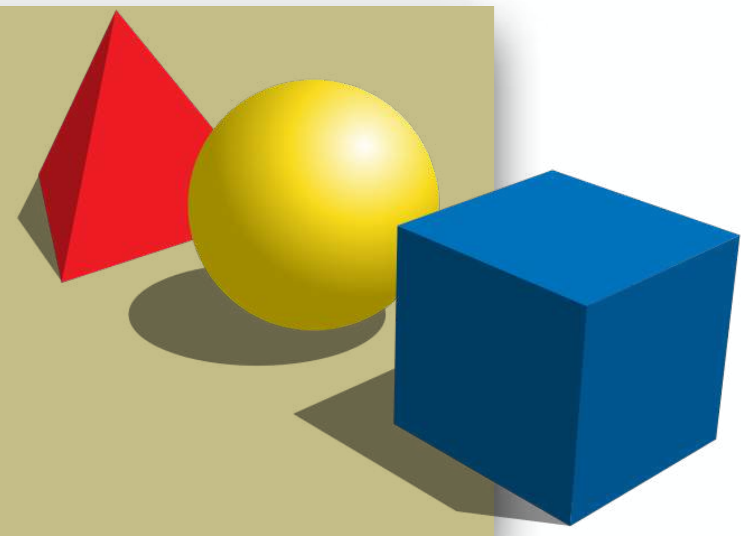


Abstract Interpretation

3-Step Recipe

practical tools
targeting specific programs

abstract semantics, abstract domains
algorithmic approaches to decide program properties



concrete semantics
mathematical models of the program behavior

Abstract Interpretation

3-Step Recipe

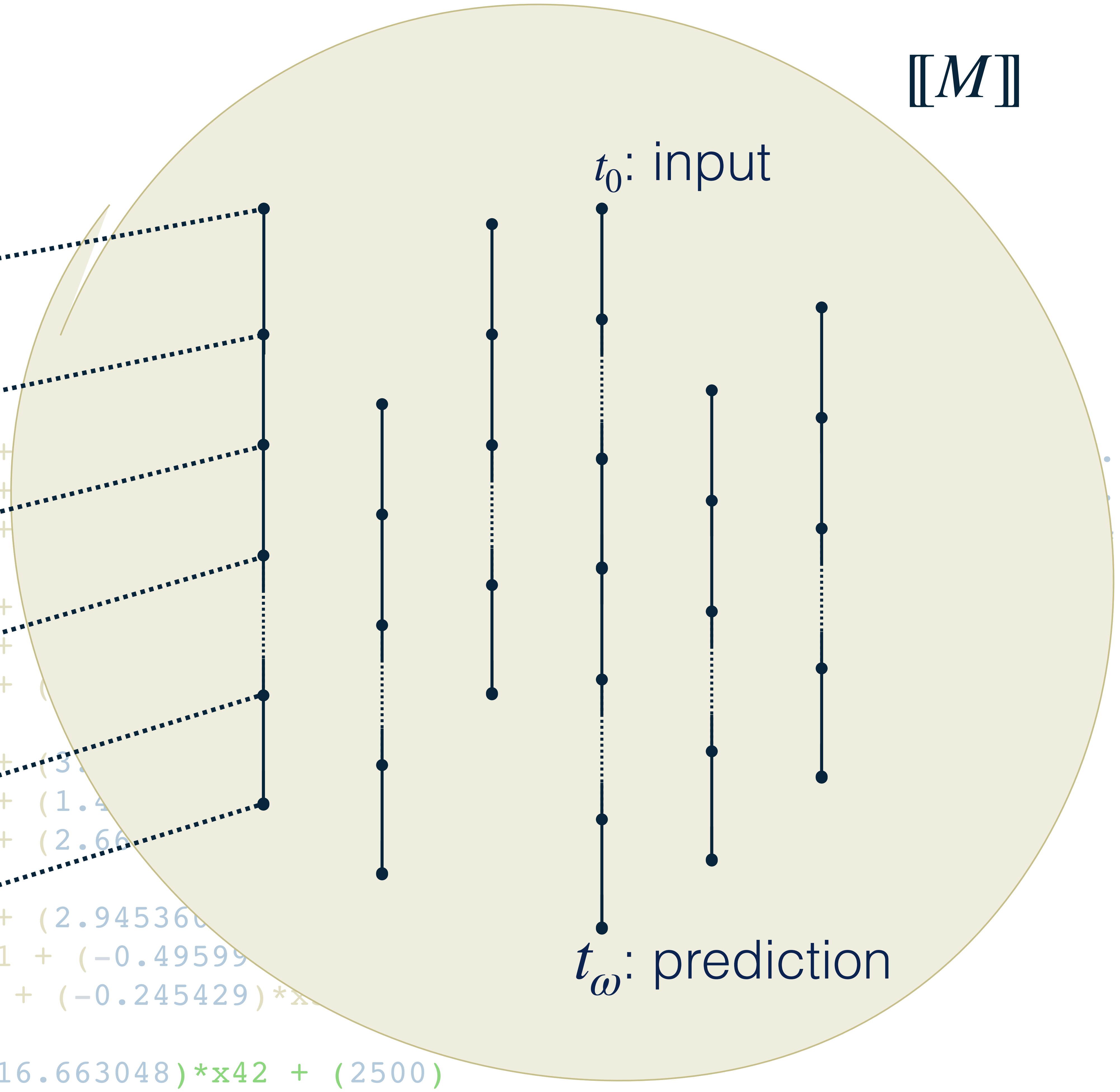
practical tools
targeting specific programs

abstract semantics, abstract domains
algorithmic approaches to decide program properties

concrete semantics
mathematical models of the program behavior

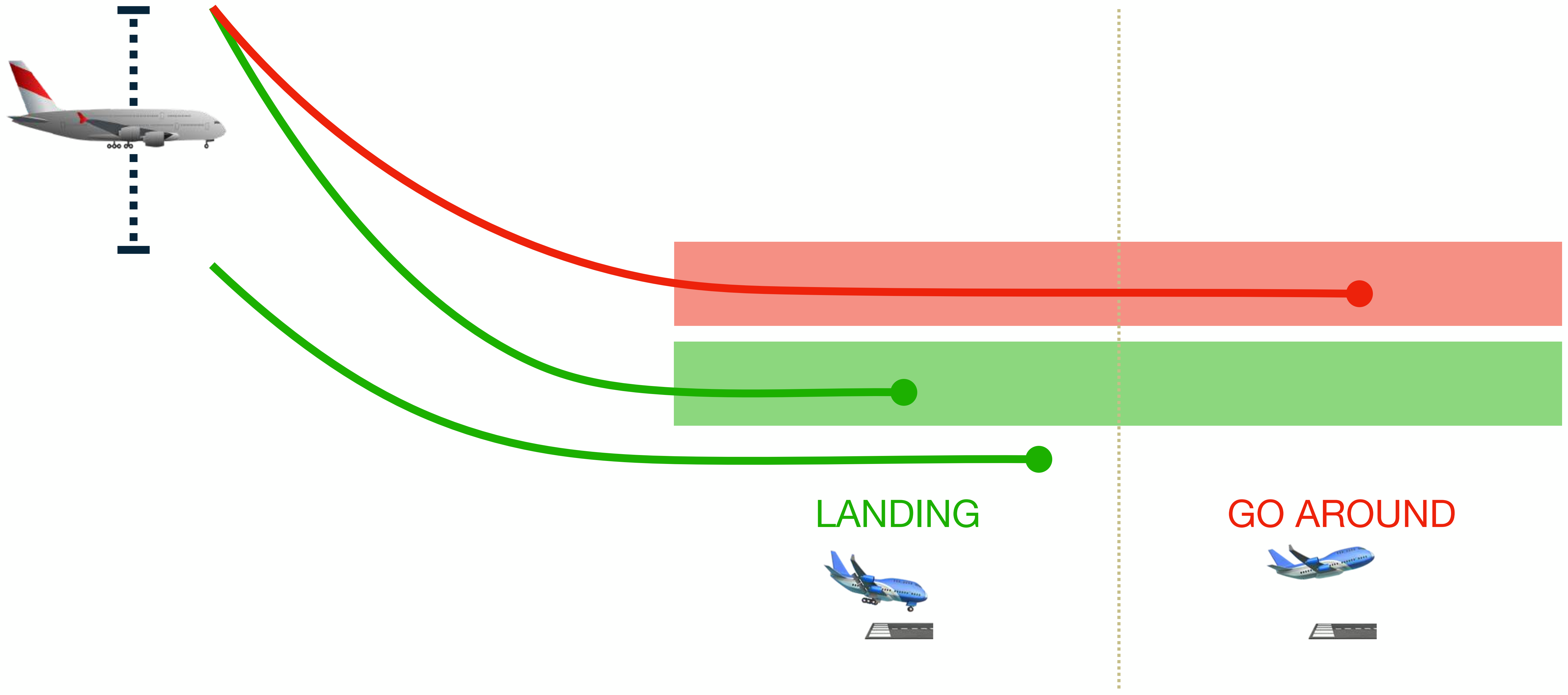
Forward Pass Trace Semantics

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + ... .103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + ... .651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + ... .18635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + ...
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + ...
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + ...

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3. ...
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.4 ...
x32 = ReLU((2.147212)*x20 + (2.203599)*x21 + (2.66 ...

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360 ...
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.49599 ...
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x ...

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```



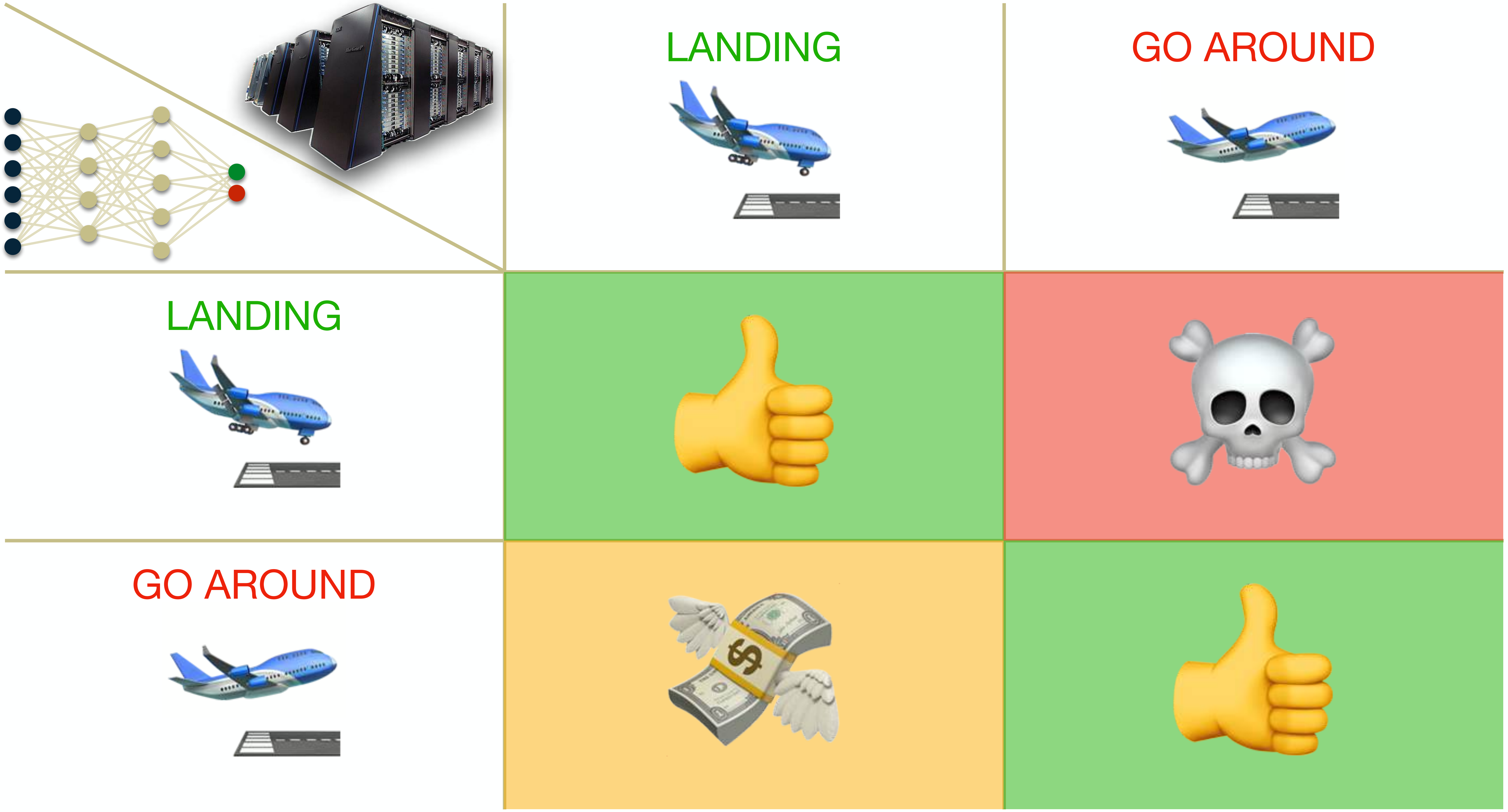
Safety

Hypersafety



Runway Overrun Warning

Safety of Neural Network Surrogate



Safety Verification

Extensional Properties

$\mathbf{I}$: input specification

$\mathbf{O}$: output specification

$$\mathcal{S} \stackrel{\text{def}}{=} \left\{ t \mid t_0 \models \mathbf{I} \Rightarrow t_\omega \models \mathbf{O} \right\}$$

$\mathcal{S}$ is the set of all forward pass traces that **satisfy** the specification

Theorem

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

Corollary

$$M \models \mathcal{S} \Leftarrow [\![M]\!] \subseteq [\![M]\!]^\natural \subseteq \mathcal{S}$$

Safety Verification

Example

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

I:

$-1 \leq x00 \leq 1$
$-1 \leq x01 \leq 1$
$-1 \leq x02 \leq 1$
$-1 \leq x03 \leq 1$
$-1 \leq x04 \leq 1$
$-1 \leq x05 \leq 1$

O:

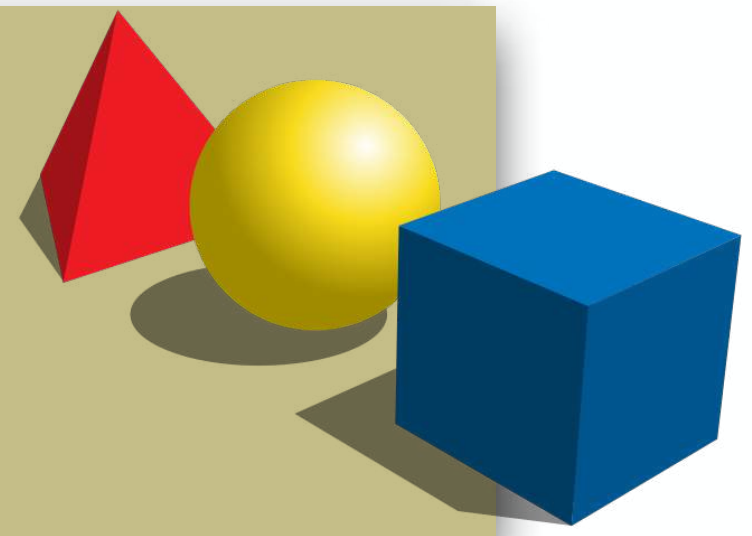
$x50 > x51$

Abstract Interpretation

3-Step Recipe

practical tools
targeting specific programs

abstract semantics, abstract domains
algorithmic approaches to decide program properties

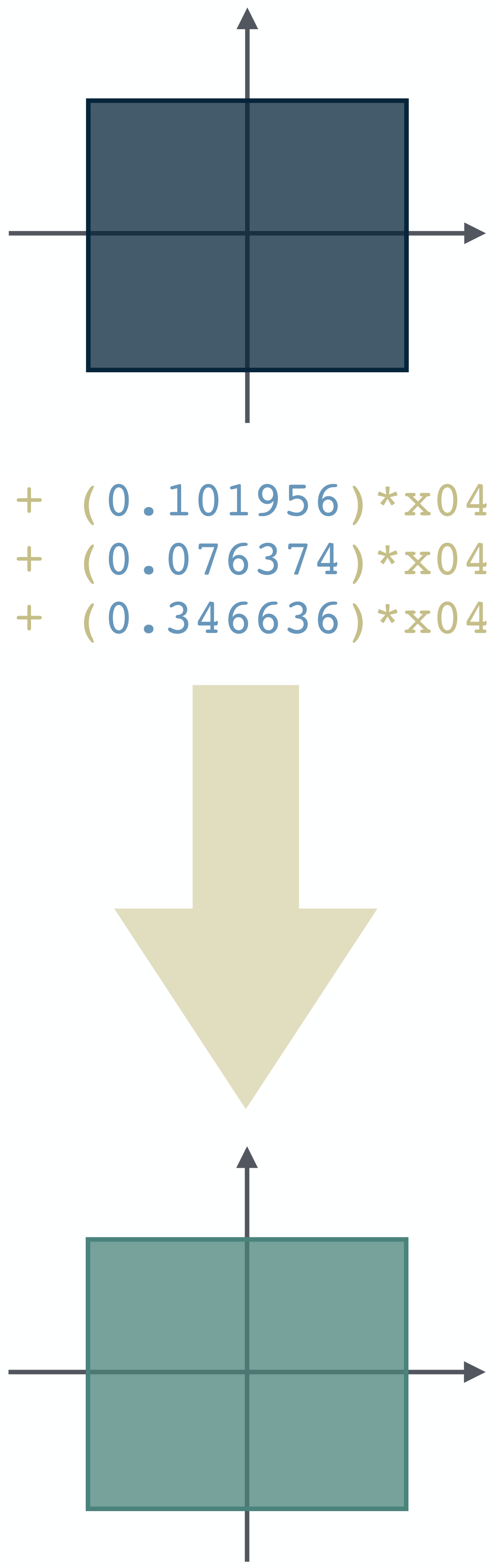


concrete semantics
mathematical models of the program behavior

Safety Verification

Static Forward Analysis

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```



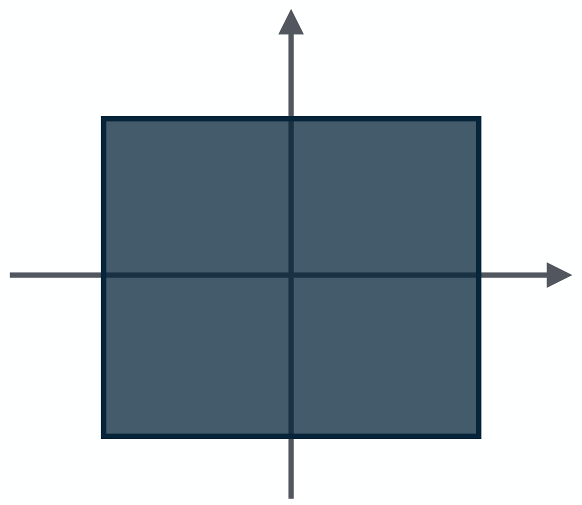
Safety Verification

Abstraction #1: Boxes Abstract Domain

$$x \mapsto [a, b]$$
$$a, b \in \mathbb{R}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

I:
x00: [-1, 1]
x01: [-1, 1]
x02: [-1, 1]
x03: [-1, 1]
x04: [-1, 1]
x05: [-1, 1]



① start from an **abstraction** of all possible inputs

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

Safety Verification

Abstraction #1: Boxes Abstract Domain

$$x \mapsto [a, b]$$
$$a, b \in \mathbb{R}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

I:
x00: [-1, 1]
x01: [-1, 1]
x02: [-1, 1]
x03: [-1, 1]
x04: [-1, 1]
x05: [-1, 1]

② proceed **forwards** **abstracting** the neural network computations

```
x10' = (0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834)
```

x10 -> [-2.90, 6.14]

```
x10 = ReLU(x10')
```

x10 -> [0, 6.14]

```
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
```

x11 -> [0, 3.29]

```
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))
```

x12 -> [0, 5.02]

⋮

```
x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

Safety Verification

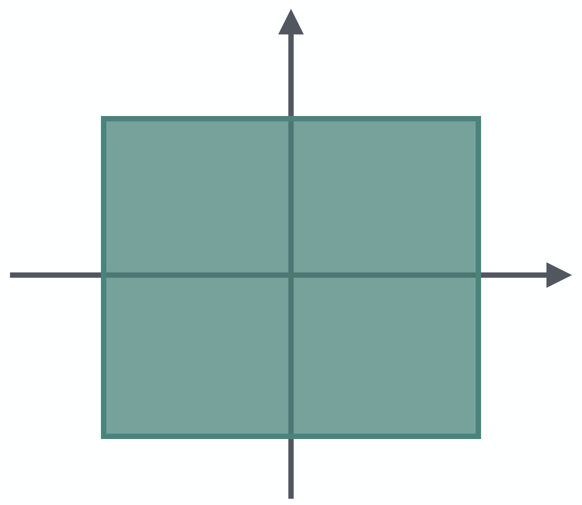
Abstraction #1: Boxes Abstract Domain

$x \mapsto [a, b]$

$a, b \in \mathbb{R}$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

I: x00: [-1, 1]
x01: [-1, 1]
x02: [-1, 1]
x03: [-1, 1]
x04: [-1, 1]
x05: [-1, 1]



③ check output for **inclusion** in **expected output**:
included → ✓ **safe**
otherwise → **alarm**

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))
```

x10 -> [0, 6.14]   x11 -> [0, 3.29]   x12 -> [0, 5.02]

```
x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))
```

x20 -> [0, 25.30]   x21 -> [0, 27.34]   x22 -> [0, 18.63]

⋮

x30 -> [0, 118.99]   x31 -> [0, 155.15]   x32 -> [0, 162.18]

```
x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))
```

x40 -> [0, 1054.08]   x41 -> [0, 0]   x42 -> [0, 191.11]

```
x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

O: x50 - x51 $\sqsubseteq$ [0, ∞]   x50 - x51 -> [-6171.35, 5000]

Abstract Interpretation

€ 2,5 +
€ 3 +
€ 4 +
€ 5,5

€ 15

€ 2.25 +
€ 2.95 +
€ 3.65 +
€ 5.35

€ 14.20

FALSE ALARM

Safety Verification

Abstraction #2: Symbolic Abstract Domain [Li19]

$$x \mapsto \begin{cases} E \\ [a,b] & a,b \in \mathbb{R} \end{cases}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

I: x00: $\begin{cases} x00 \\ [-1,1] \end{cases}$ x01: $\begin{cases} x01 \\ [-1,1] \end{cases}$ x02: $\begin{cases} x02 \\ [-1,1] \end{cases}$ x03: $\begin{cases} x03 \\ [-1,1] \end{cases}$ x04: $\begin{cases} x04 \\ [-1,1] \end{cases}$ x05: $\begin{cases} x05 \\ [-1,1] \end{cases}$

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

Safety Verification

Abstraction #2: Symbolic Abstract Domain [Li19]

$$x \mapsto \begin{cases} E \\ [a, b] \quad a, b \in \mathbb{R} \end{cases}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

I: x00: $\begin{cases} x00 \\ [-1,1] \end{cases}$ x01: $\begin{cases} x01 \\ [-1,1] \end{cases}$ x02: $\begin{cases} x02 \\ [-1,1] \end{cases}$ x03: $\begin{cases} x03 \\ [-1,1] \end{cases}$ x04: $\begin{cases} x04 \\ [-1,1] \end{cases}$ x05: $\begin{cases} x05 \\ [-1,1] \end{cases}$

```
x10' = (0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834)
```

x10': $\begin{cases} 0.12 * x00 + 0.07 * x01 + 0.10 * x02 + 2.03 * x03 + 0.10 * x04 - 2.10 * x05 + 1.62 \\ [-2.90, 6.14] \end{cases}$

$x_0 \mapsto \mathbf{E_0}$
$\dots$
$x_j \mapsto \mathbf{E_j}$
$\dots$

$$x = \sum_k w_k \cdot x_k + b$$

$$x \mapsto \sum_k w_k \cdot \mathbf{E_k} + b$$

```
⋮

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

Safety Verification

Abstraction #2: Symbolic Abstract Domain [Li19]

$$x \mapsto \begin{cases} E \\ [a,b] & a,b \in \mathbb{R} \end{cases}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

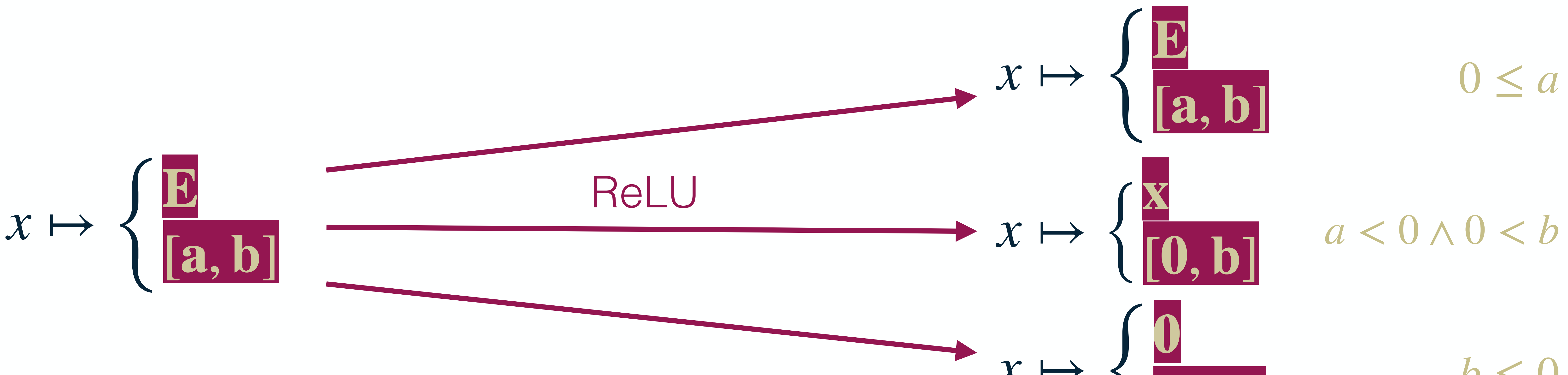
I: x00: $\begin{cases} x00 \\ [-1,1] \end{cases}$ x01: $\begin{cases} x01 \\ [-1,1] \end{cases}$ x02: $\begin{cases} x02 \\ [-1,1] \end{cases}$ x03: $\begin{cases} x03 \\ [-1,1] \end{cases}$ x04: $\begin{cases} x04 \\ [-1,1] \end{cases}$ x05: $\begin{cases} x05 \\ [-1,1] \end{cases}$

```
x10' = (0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834)
```

x10’: $\begin{cases} 0.12 * x00 + 0.07 * x01 + 0.10 * x02 + 2.03 * x03 + 0.10 * x04 - 2.10 * x05 + 1.62 \\ [-2.90, 6.14] \end{cases}$

```
x10 = ReLU(x10')
```

x10: $\begin{cases} x10 \\ [0, 6.14] \end{cases}$



⋮

```
x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

O: x50 - x51 ⊏ [0, ∞]

Safety Verification

Abstraction #2: Symbolic Abstract Domain [Li19]

$$x \mapsto \begin{cases} E \\ [a, b] & a, b \in \mathbb{R} \end{cases}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

I: x00: $\begin{cases} x00 \\ [-1,1] \end{cases}$ x01: $\begin{cases} x01 \\ [-1,1] \end{cases}$ x02: $\begin{cases} x02 \\ [-1,1] \end{cases}$ x03: $\begin{cases} x03 \\ [-1,1] \end{cases}$ x04: $\begin{cases} x04 \\ [-1,1] \end{cases}$ x05: $\begin{cases} x05 \\ [-1,1] \end{cases}$

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))
```

x10: $\begin{cases} x10 \\ [0, 6.14] \end{cases}$ x11: $\begin{cases} x11 \\ [0, 3.29] \end{cases}$ x11: $\begin{cases} x12 \\ [0, 5.02] \end{cases}$

⋮

```
x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))
```

x40: $\begin{cases} x40 \\ [0, 1054.08] \end{cases}$ x41: $\begin{cases} 0 \\ [0, 0] \end{cases}$ x42: $\begin{cases} x42 \\ [0, 191.11] \end{cases}$

```
x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

O: x50 - x51: $\begin{cases} -4.56 * x40 - 33.33 * x42 + 5000 \\ [-6171.35, 5000] \not\sqsubseteq [0, \infty] \end{cases}$

Safety Verification

Abstraction #3: DeepPoly Abstract Domain [Singh19]

$$x \mapsto \begin{cases} [L, U] \\ [a, b] & a, b \in \mathbb{R} \end{cases}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

I: x00: $\begin{cases}[x00, x00]\\ [-1,1]\end{cases}$ x01: $\begin{cases}[x01, x01]\\ [-1,1]\end{cases}$ x02: $\begin{cases}[x02, x02]\\ [-1,1]\end{cases}$ x03: $\begin{cases}[x03, x03]\\ [-1,1]\end{cases}$ x04: $\begin{cases}[x04, x04]\\ [-1,1]\end{cases}$ x05: $\begin{cases}[x05, x05]\\ [-1,1]\end{cases}$

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

Safety Verification

Abstraction #3: DeepPoly Abstract Domain [Singh19]

$$x \mapsto \begin{cases} [L, U] \\ [a, b] \quad a, b \in \mathbb{R} \end{cases}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

I: x00: $\begin{cases}[x00, x00] \\ [-1,1]\end{cases}$ x01: $\begin{cases}[x01, x01] \\ [-1,1]\end{cases}$ x02: $\begin{cases}[x02, x02] \\ [-1,1]\end{cases}$ x03: $\begin{cases}[x03, x03] \\ [-1,1]\end{cases}$ x04: $\begin{cases}[x04, x04] \\ [-1,1]\end{cases}$ x05: $\begin{cases}[x05, x05] \\ [-1,1]\end{cases}$

```
x10' = (0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834)
```

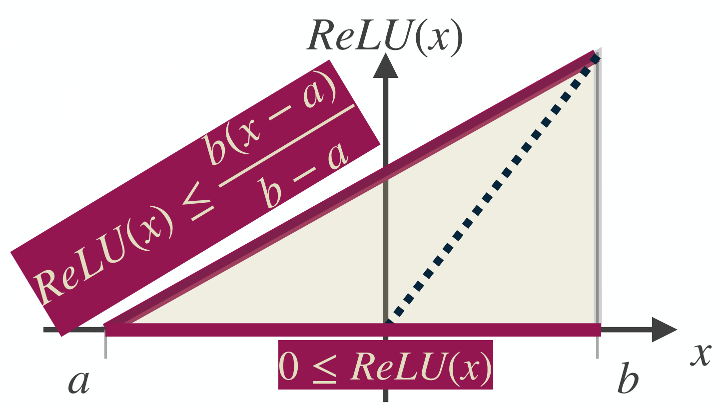
x10': $\begin{cases}\ldots [L, U] \\ [-2.90, 6.14]\end{cases}$

```
x10 = ReLU(x10')
```

x10: $\begin{cases}[x10', 0.68 * x10' + 1.97] \\ [-2.90, 6.14]\end{cases}$

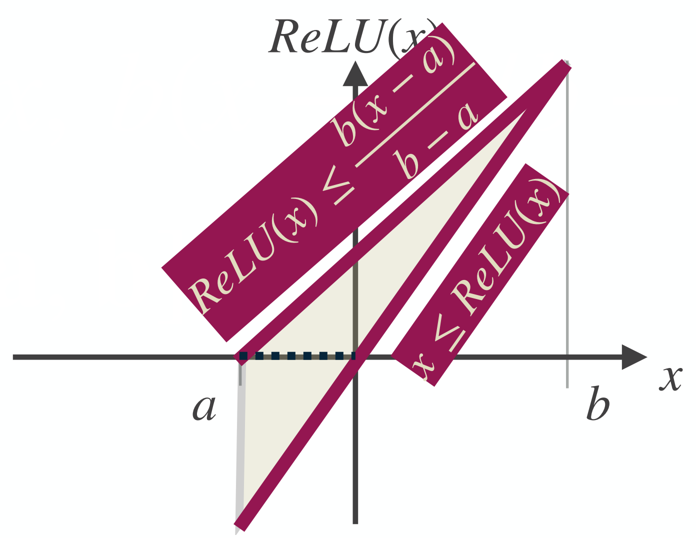
$x \mapsto \begin{cases}[\mathbf{L, U}] \\ [\mathbf{a, b}]\end{cases}$

$a < 0 < b \wedge -b \leq a$ → $x \mapsto \{$



ReLU

$a < 0 < b \wedge -a < b$ → $x \mapsto \{$



⋮

```
x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

Safety Verification

Abstraction #3: DeepPoly Abstract Domain [Singh19]

$$x \mapsto \begin{cases} [L, U] \\ [a, b] \quad a, b \in \mathbb{R} \end{cases}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

x04: $\begin{cases}[x04, x04] \\ [-1,1]\end{cases}$ x05: $\begin{cases}[x05, x05] \\ [-1,1]\end{cases}$

```
x10 = ReLU((0.120875)*x00 + ... 565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + ... 132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + ... 35)*x05 + (-0.686885))
```

x10: $\begin{cases}[x10', 0.68 * x10' + 1.9 \\ [-2.90, 6.14]\end{cases}$

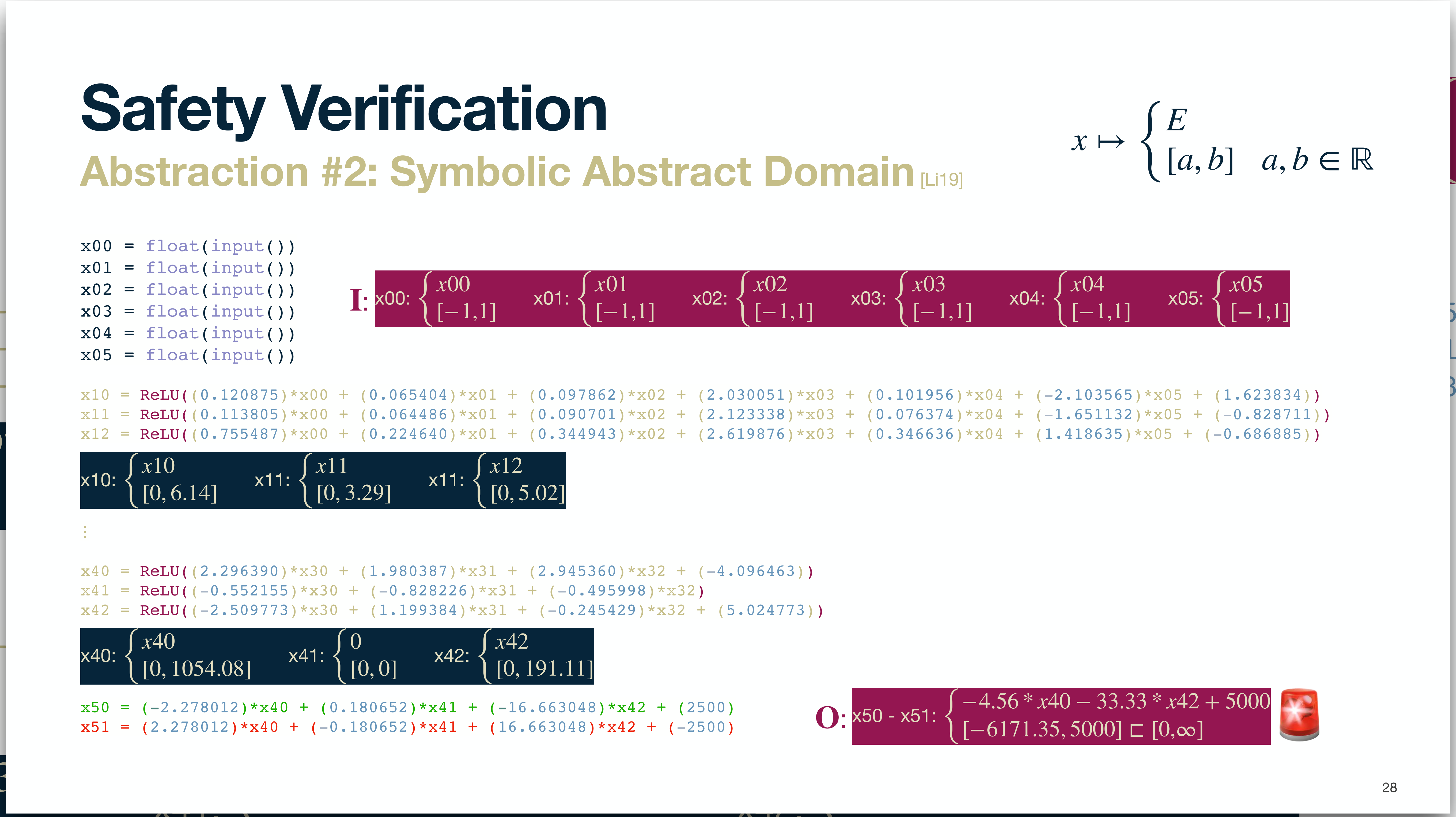
⋮

```
x40 = ReLU((2.296390)*x30 + ...
x41 = ReLU((-0.552155)*x30 ...
x42 = ReLU((-2.509773)*x30 ...
```

x40: $\begin{cases}[x40', 0.67 * x40' + 313 \\ [-467.10, 950.38]\end{cases}$ x41: $[0, 118.63]$ x42: $[-142.20, 162.09]$

```
x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

O: x50 - x51: $\begin{cases}\ldots \\ [-1424.80, 9072.12] \not\sqsubseteq [0,\infty]\end{cases}$



Safety Verification

Abstraction #2: Symbolic Abstract Domain [Li19]

$$x \mapsto \begin{cases} E \\ [a, b] \quad a, b \in \mathbb{R} \end{cases}$$

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

I: x00: $\begin{cases}x00 \\ [-1,1]\end{cases}$ x01: $\begin{cases}x01 \\ [-1,1]\end{cases}$ x02: $\begin{cases}x02 \\ [-1,1]\end{cases}$ x03: $\begin{cases}x03 \\ [-1,1]\end{cases}$ x04: $\begin{cases}x04 \\ [-1,1]\end{cases}$ x05: $\begin{cases}x05 \\ [-1,1]\end{cases}$

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))
```

x10: $\begin{cases}x10 \\ [0, 6.14]\end{cases}$ x11: $\begin{cases}x11 \\ [0, 3.29]\end{cases}$ x11: $\begin{cases}x12 \\ [0, 5.02]\end{cases}$

⋮

```
x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))
```

x40: $\begin{cases}x40 \\ [0, 1054.08]\end{cases}$ x41: $\begin{cases}0 \\ [0, 0]\end{cases}$ x42: $\begin{cases}x42 \\ [0, 191.11]\end{cases}$

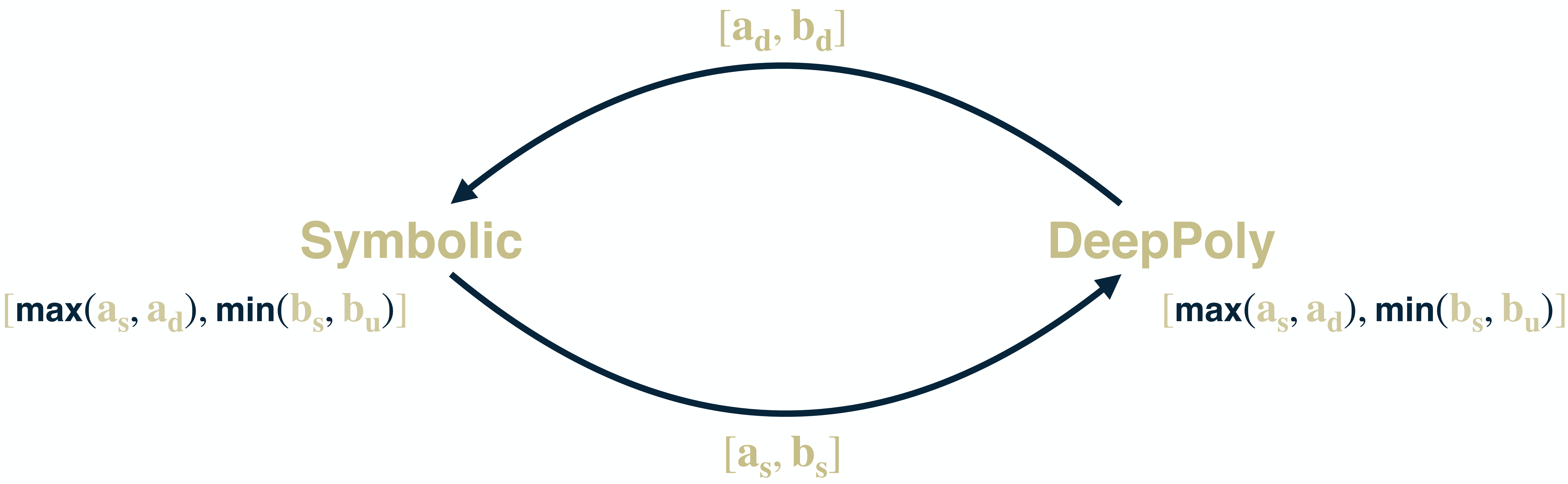
```
x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

O: x50 - x51: $\begin{cases}-4.56 * x40 - 33.33 * x42 + 5000 \\ [-6171.35, 5000] \not\sqsubseteq [0,\infty]\end{cases}$

28

Reduced Product Domain

Symbolic Abstract Domain & DeepPoly Abstract Domain



D. Mazzucato and CU. Reduced Products of Abstract Domains for Fairness Certification of Neural Networks. In SAS, 2021

Safety Verification

Abstraction #4: Symbolic & DeepPoly Product Abstract Domain

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

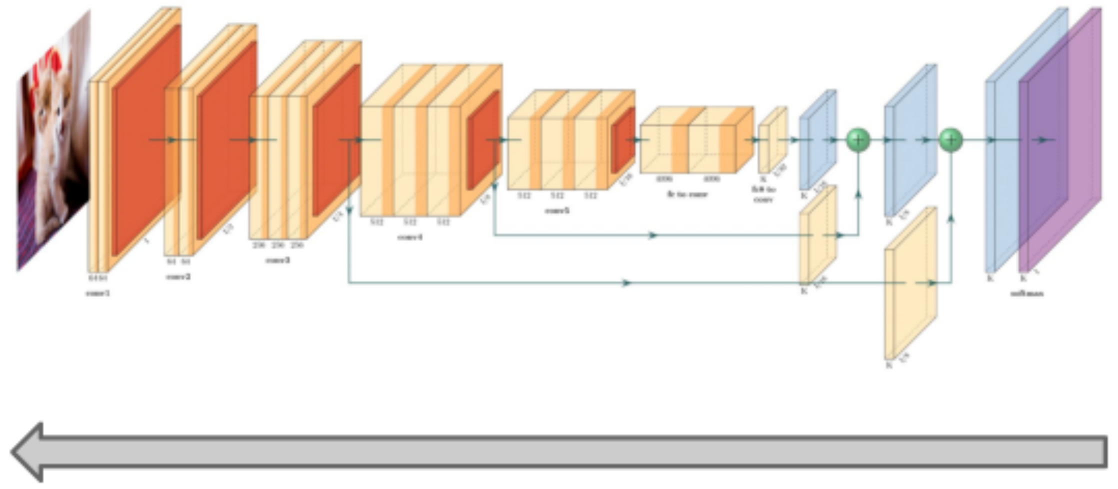
I: x00: $\begin{cases} x00 \\ [x00, x00] \\ [-1,1] \end{cases}$ x01: $\begin{cases} x01 \\ [x01, x01] \\ [-1,1] \end{cases}$ x02: $\begin{cases} x02 \\ [x02, x02] \\ [-1,1] \end{cases}$ x03: $\begin{cases} x03 \\ [x03, x03] \\ [-1,1] \end{cases}$ x04: $\begin{cases} x04 \\ [x04, x04] \\ [-1,1] \end{cases}$ x05: $\begin{cases} x05 \\ [x05, x05] \\ [-1,1] \end{cases}$

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (2500)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

O: x50 - x51: $\begin{cases} \vdots \\ [670.04,\ 5000.0] \sqsubseteq [0,\infty] \end{cases}$ ✓

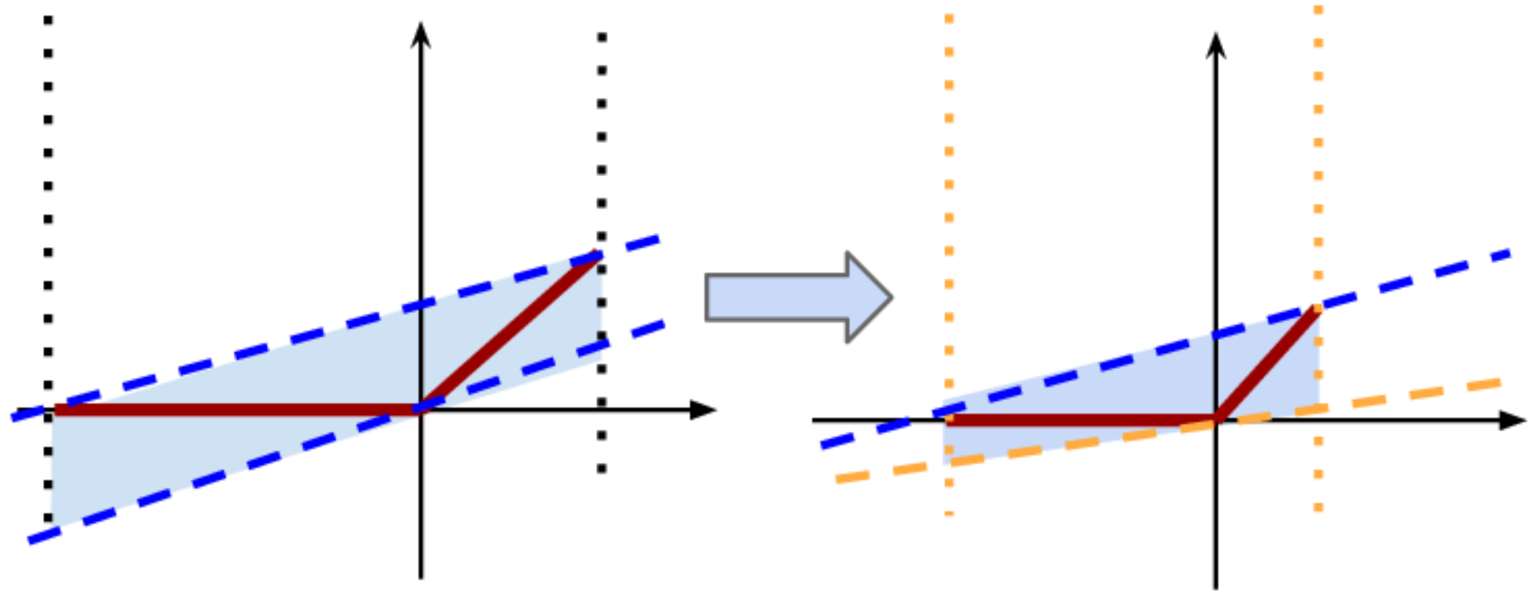
Safety Verification

Going Farther: $\alpha\beta$-CROWN

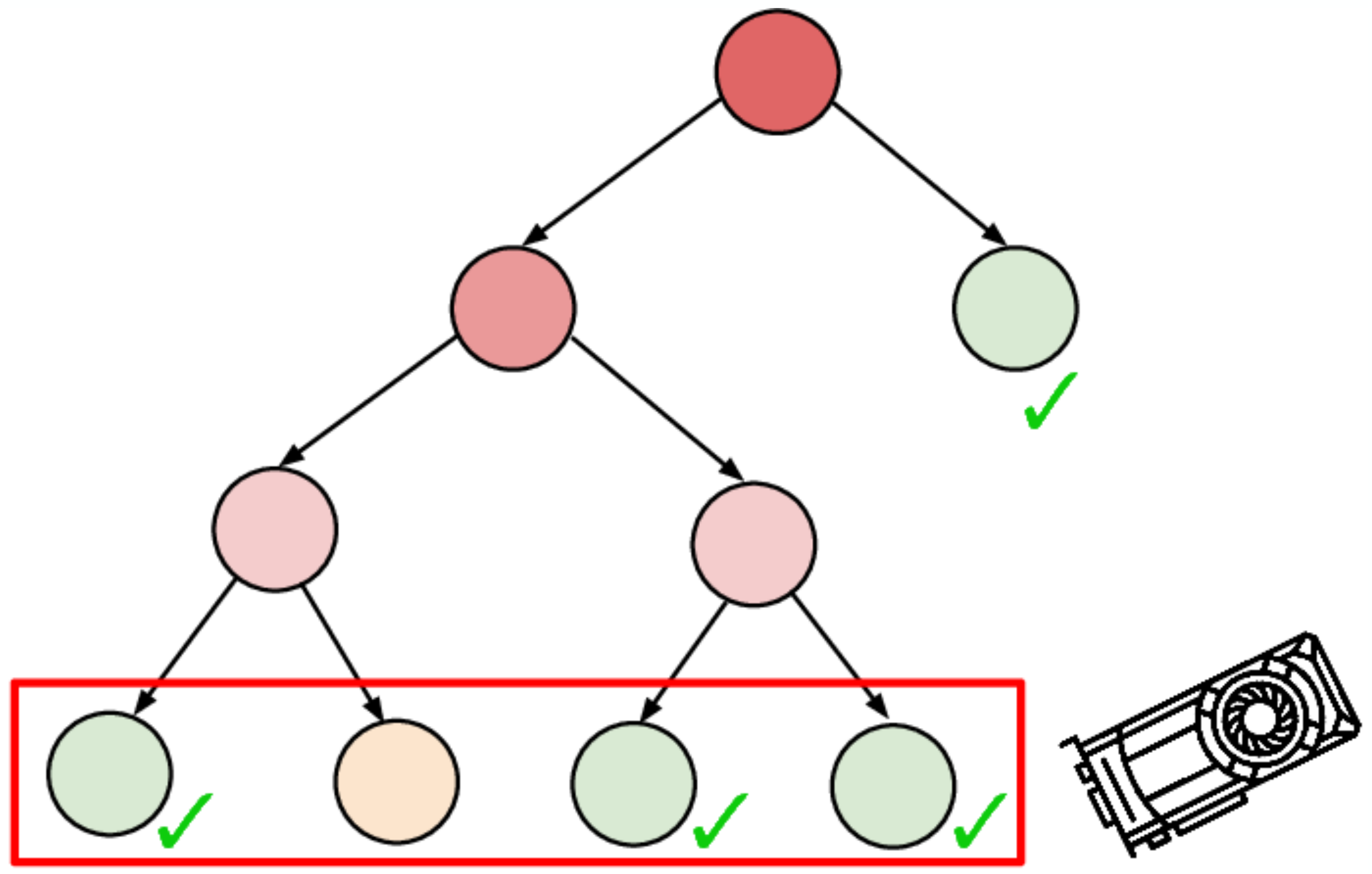


$$\min_{x\in\mathcal{C}} f(x) \geq \min_{x\in\mathcal{C}} \boldsymbol{a}^\top x + c$$

Efficient bound propagation (**CROWN**)



GPU optimized relaxation (***α*-CROWN**)



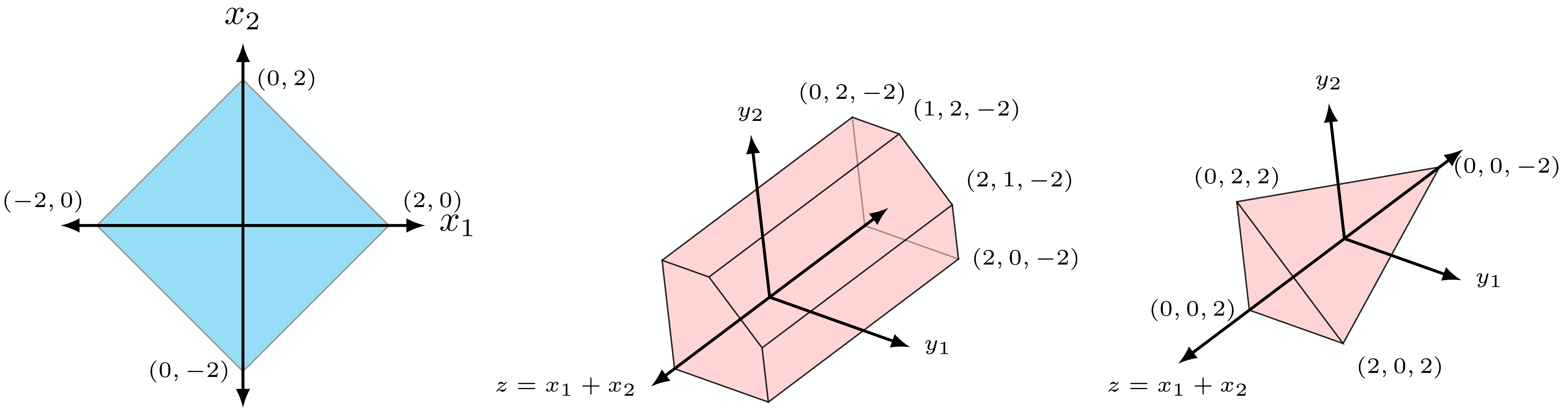
Parallel branch and bound (***β*-CROWN**)

Winner of the International Verification of Neural Networks Competition since 2021

https://github.com/Verified-Intelligence/alpha-beta-CROWN

Safety Verification

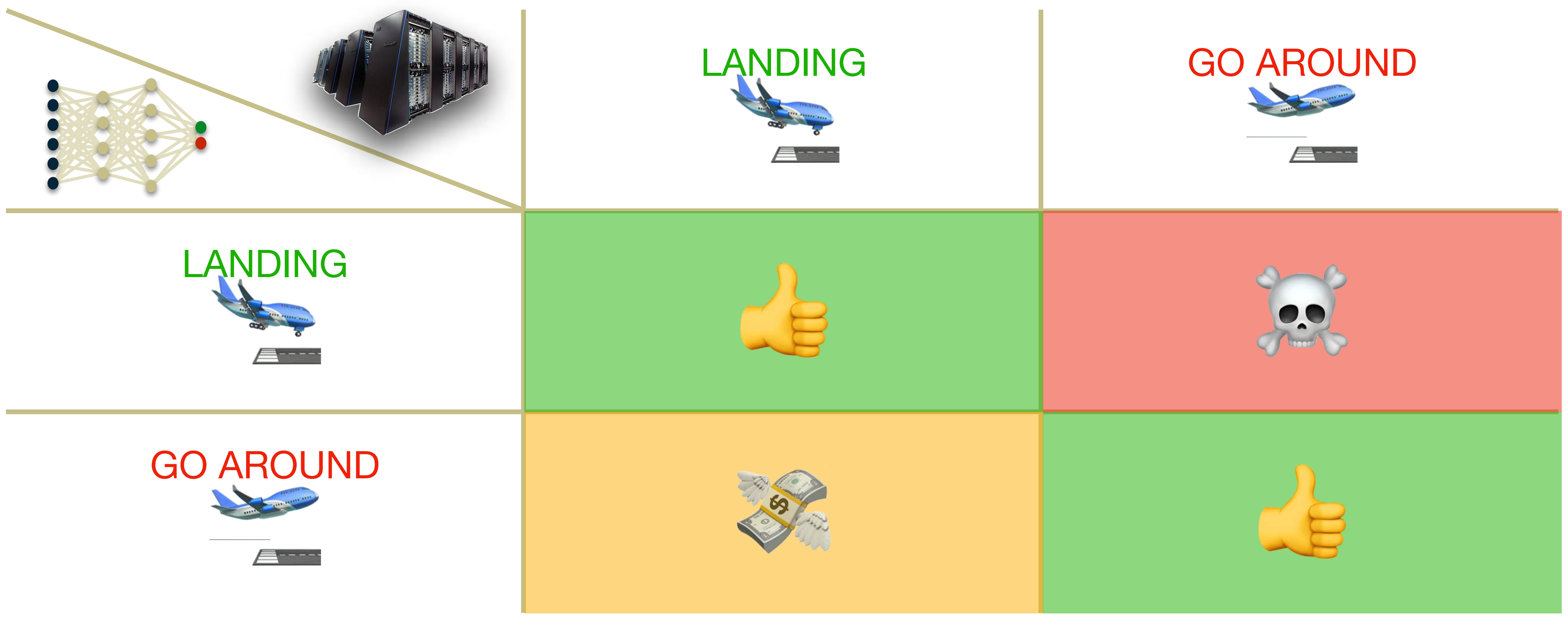
Going Farther: Multi-Neuron Abstractions



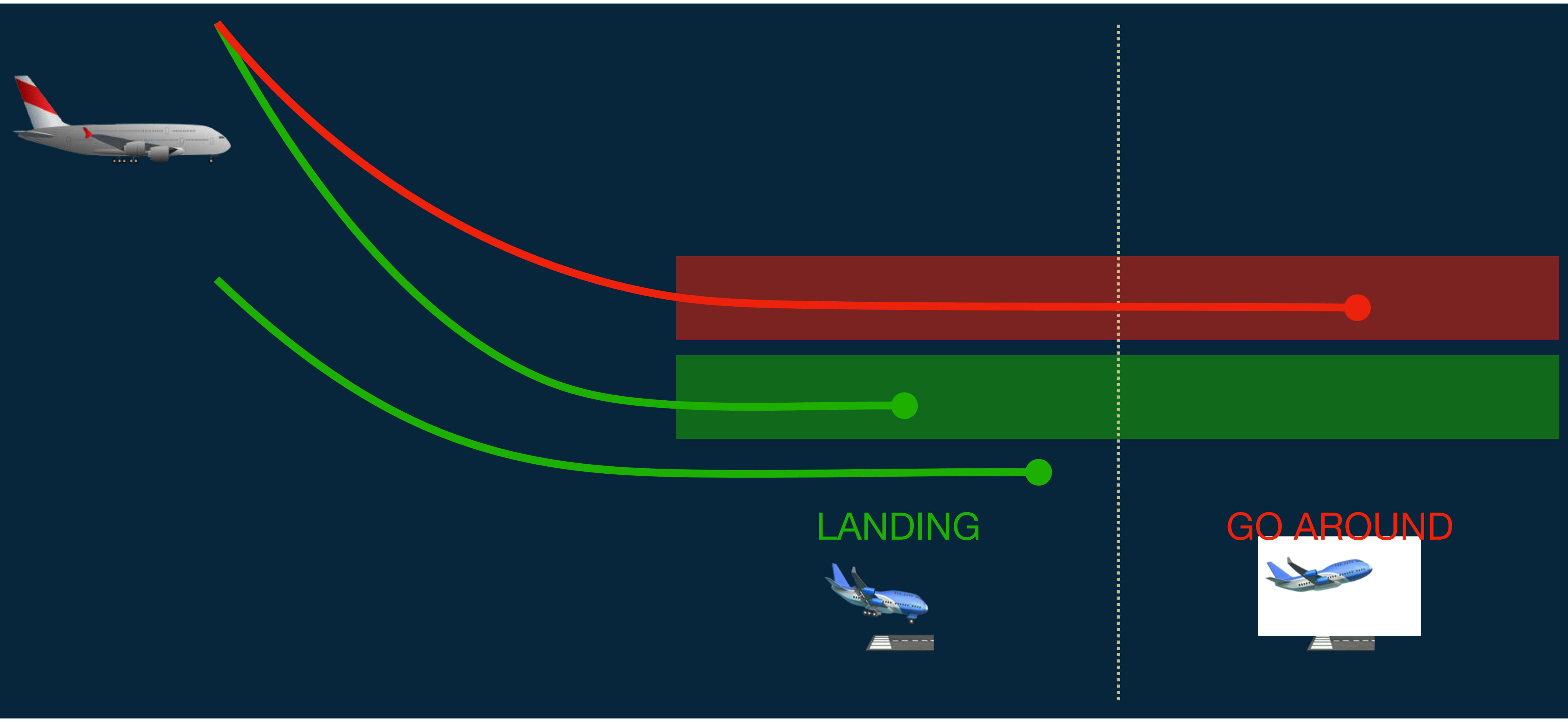
(a) Input shape (b) 1-ReLU (c) 2-ReLU

Singh, Ganvir, Püschel and Vechev. Beyond the Single Neuron Convex Barrier for Neural Network Certification. In NeurIPS, 2019

Safety

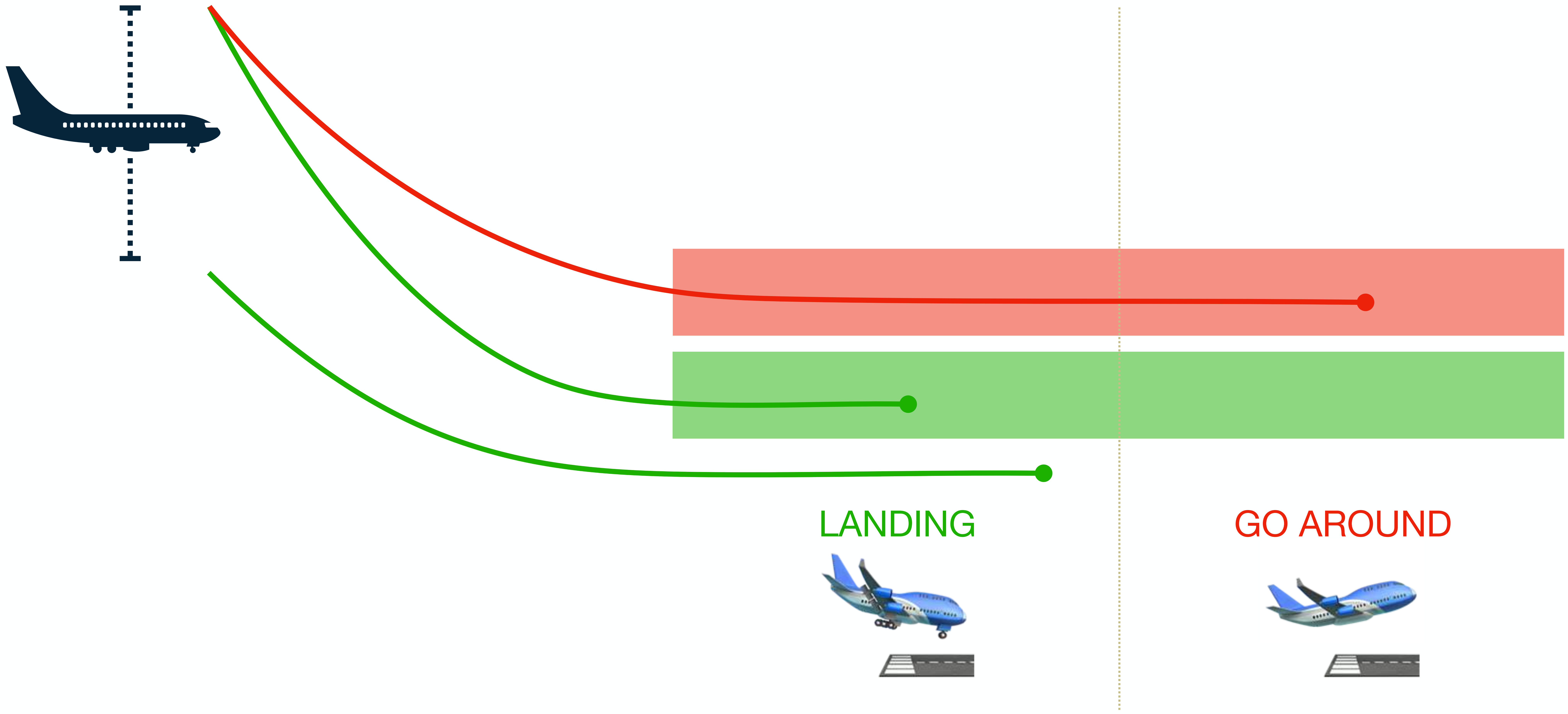


Hypersafety



Runway Overrun Warning

HyperSafety of Neural Network Surrogate



Hyperproperty Verification

Abstract Non-Interference Properties

η: input abstraction

ρ: output abstraction

$$\mathscr{H} \stackrel{\text{def}}{=} \left\{ T \mid \forall t, t' \in T \colon \eta(t_0) = \eta(t'_0) \Rightarrow \rho(t_\omega) = \rho(t'_\omega) \right\}$$

$\mathscr{H}$ is the set of all forward pass traces that **satisfy** abstract non-interference with respect to η and ρ

Theorem

$$M \models \mathscr{H} \Leftrightarrow [\![M]\!] \in \mathscr{H} \Leftrightarrow \{[\![M]\!]\} \subseteq \mathscr{H}$$

Corollary

$$M \models \mathscr{H} \Leftarrow \{[\![M]\!]\} \subseteq [\![M]\!]^\natural \subseteq \mathscr{H}$$

Giacobazzi and Mastroeni. Abstract Non-Interference: A Unifying Framework for Weakening Information-Flow. In TOPS, 2018.

Abstract Non-Interference Verification

Example

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```

η:

$\eta(x00) = x00$

$\eta(x01) = x01$

ALTITUDE $\eta(x02) = \top$

$\eta(x03) = x03$

$\eta(x04) = x04$

$\eta(x05) = x05$

"**the risk of a runway overrun does not change when only varying the altitude** at which it is measured (in the expected range) and nothing else"

ρ:

$\rho(x50) = 1$ if $x50 > x51$ else 0

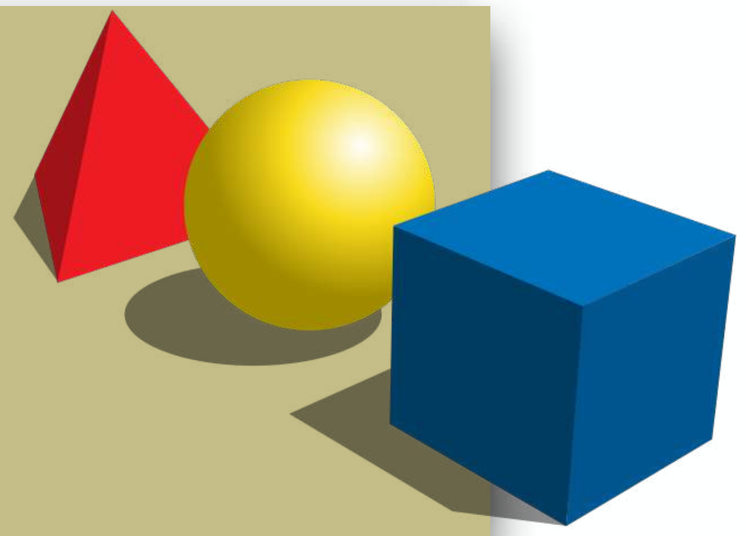
$\rho(x51) = 1$ if $x51 > x50$ else 0

Abstract Interpretation

3-Step Recipe

practical tools
targeting specific programs

abstract semantics, abstract domains
algorithmic approaches to decide program properties



concrete semantics
mathematical models of the program behavior

Abstract Interpretation

3-Step Recipe

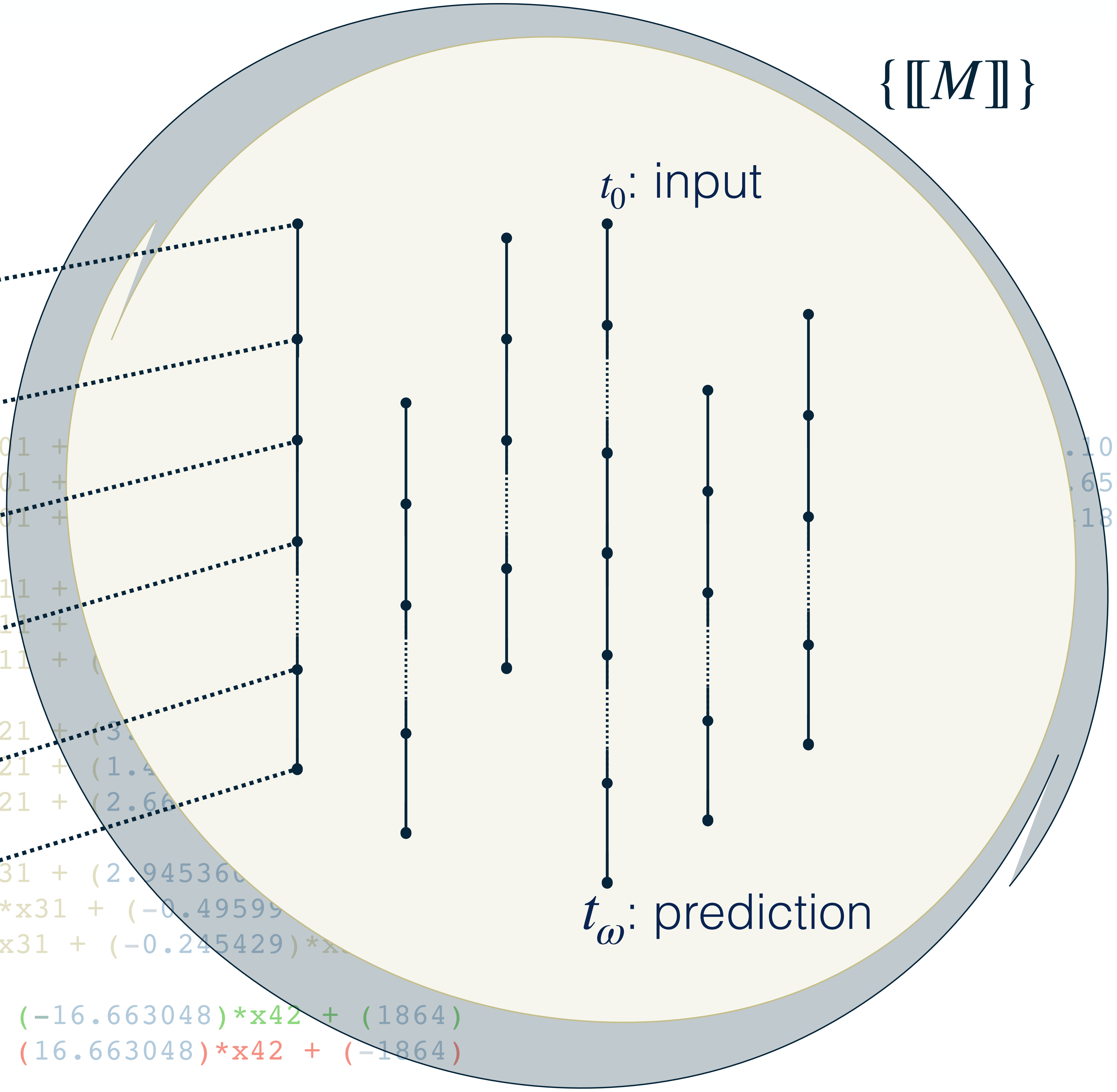
practical tools
targeting specific programs

abstract semantics, abstract domains
algorithmic approaches to decide program properties

concrete semantics
mathematical models of the program behavior

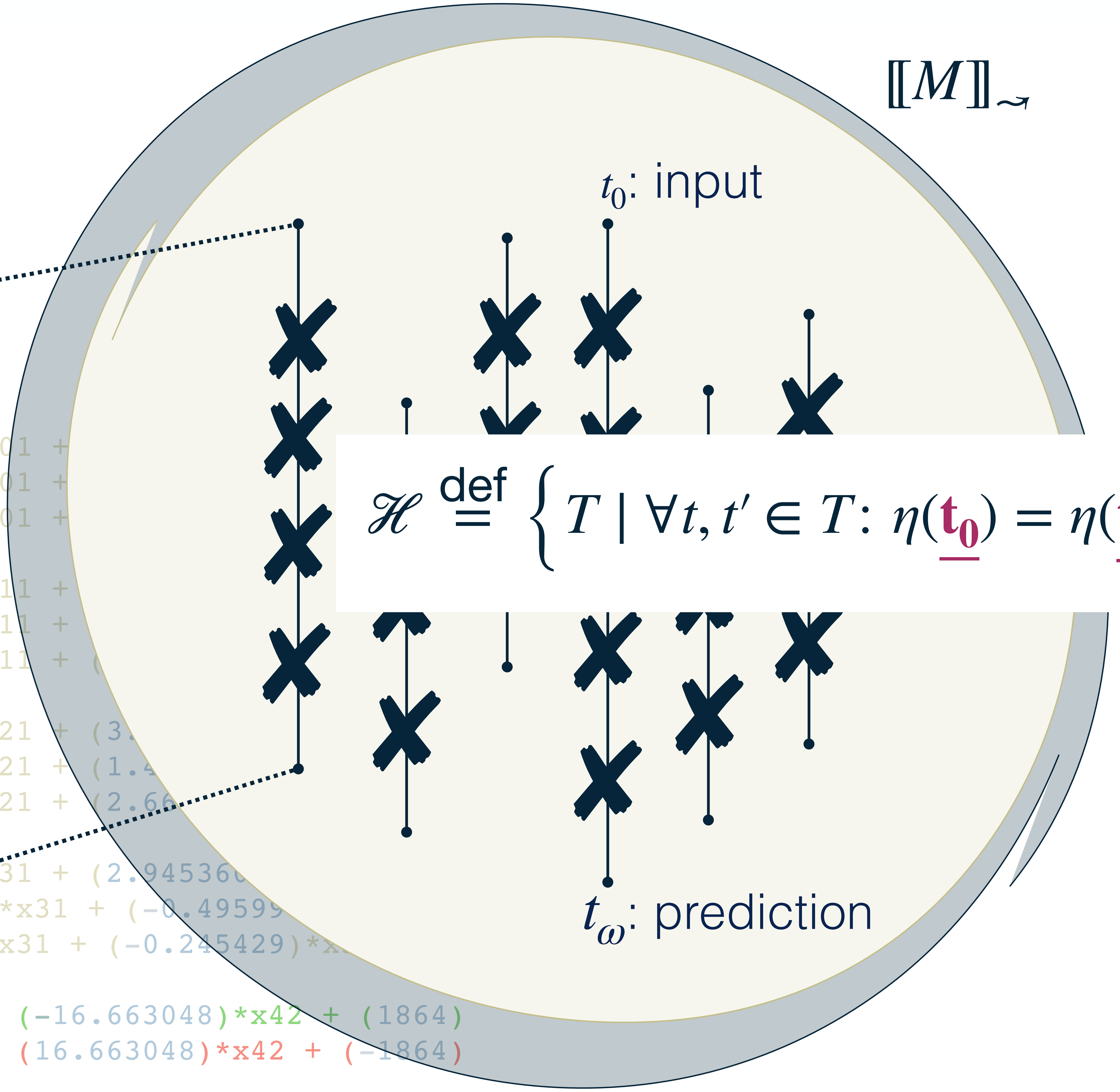
Collecting Semantics

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 +                                  .103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 +                                  .651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 +                                  .18635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 +
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 +
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.4
x32 = ReLU((2.147212)*x20 + (2.205599)*x21 + (2.66

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.94536
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.49599
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```



Dependency Semantics

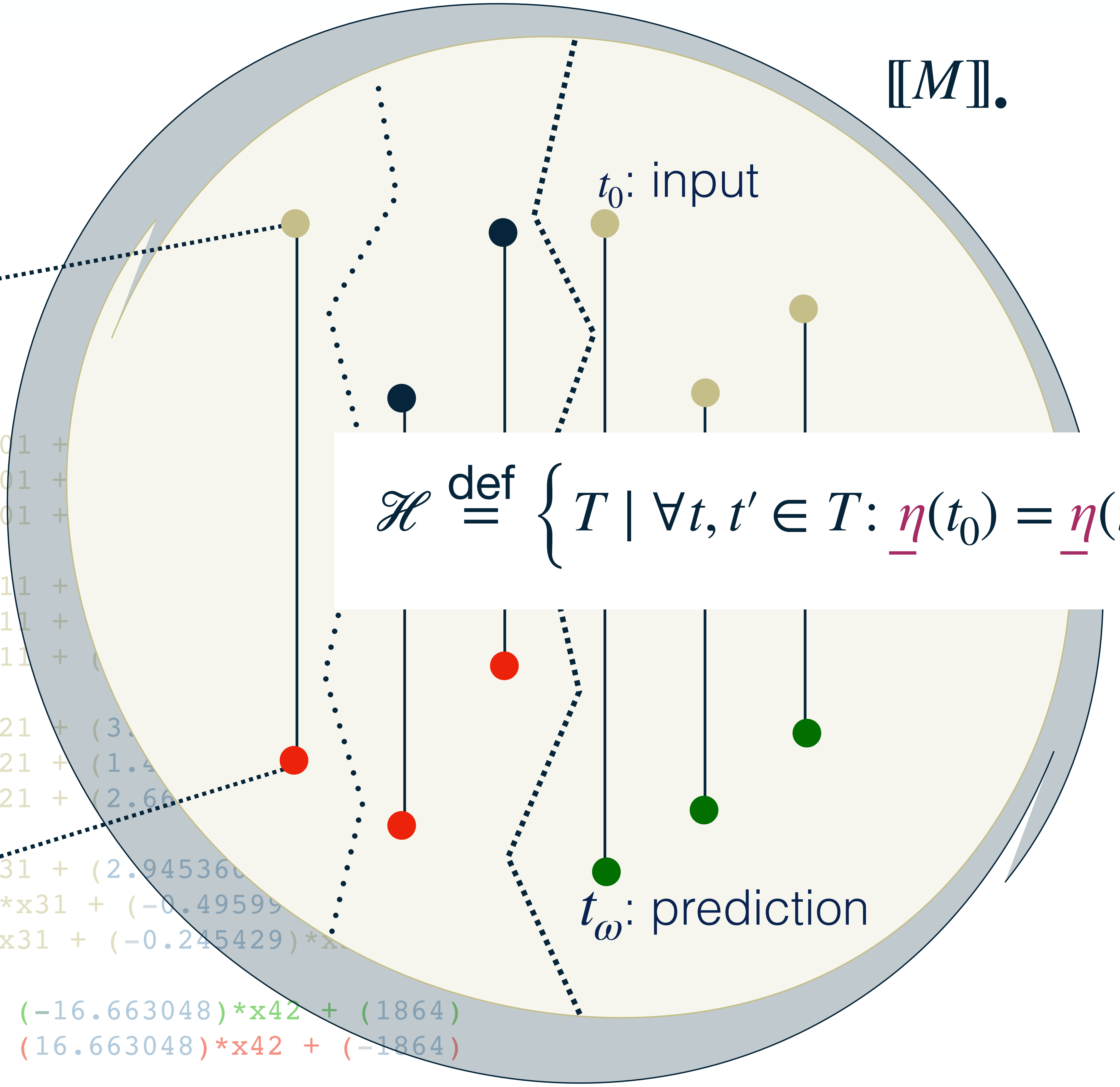
```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 +
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 +
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 +

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 +
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 +
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.4
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.66

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.49599
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```



$$\mathscr{H} \overset{\text{def}}{=} \left\{ T \mid \forall t, t' \in T \colon \eta(\underline{\mathbf{t_0}}) = \eta(\underline{\mathbf{t'_0}}) \Rightarrow \rho(\underline{\mathbf{t_\omega}}) = \rho(\underline{\mathbf{t'_\omega}}) \right\}$$

Parallel Semantics

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 +
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 +
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 +

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 +
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 +
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.4
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.66

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.49599
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```



$$\mathcal{H} \stackrel{\text{def}}{=} \left\{ T \mid \forall t, t' \in T \colon \underline{\eta}(t_0) = \underline{\eta}(t'_0) \Rightarrow \underline{\rho}(t_\omega) = \underline{\rho}(t'_\omega) \right\}$$

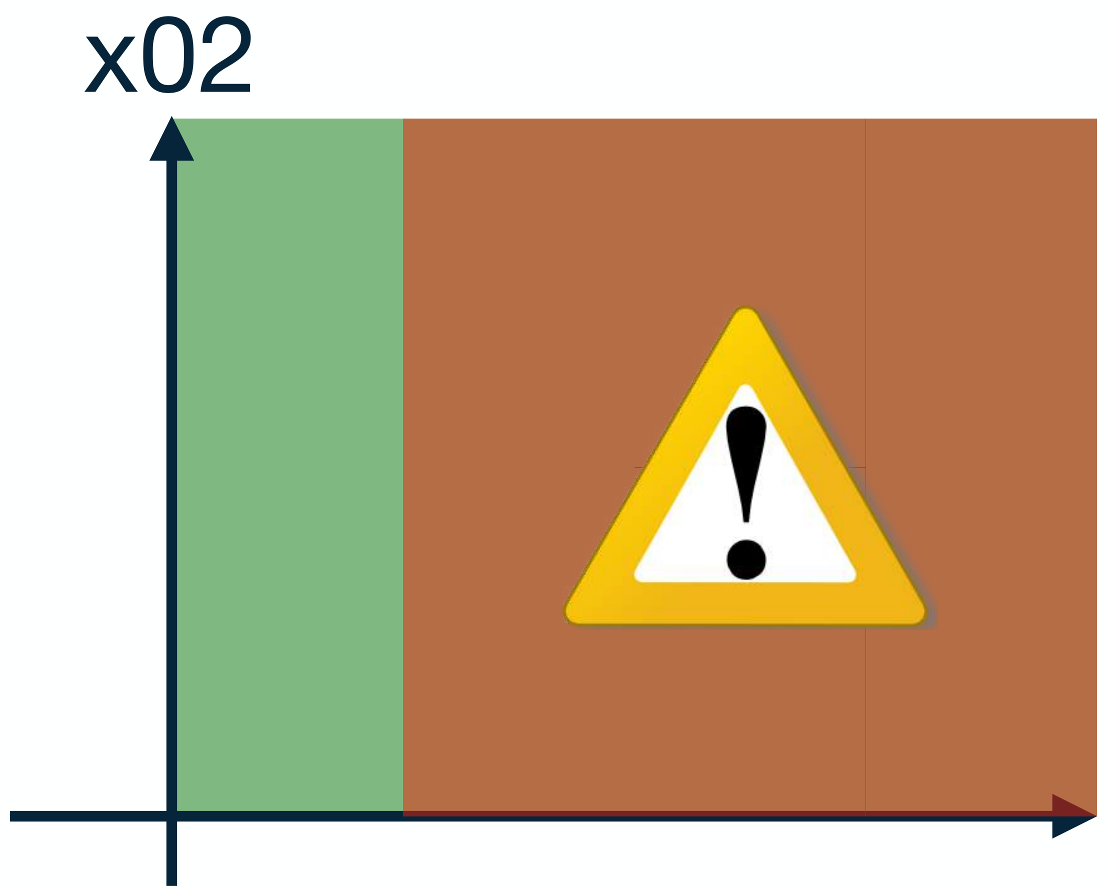
Hyperproperty Verification

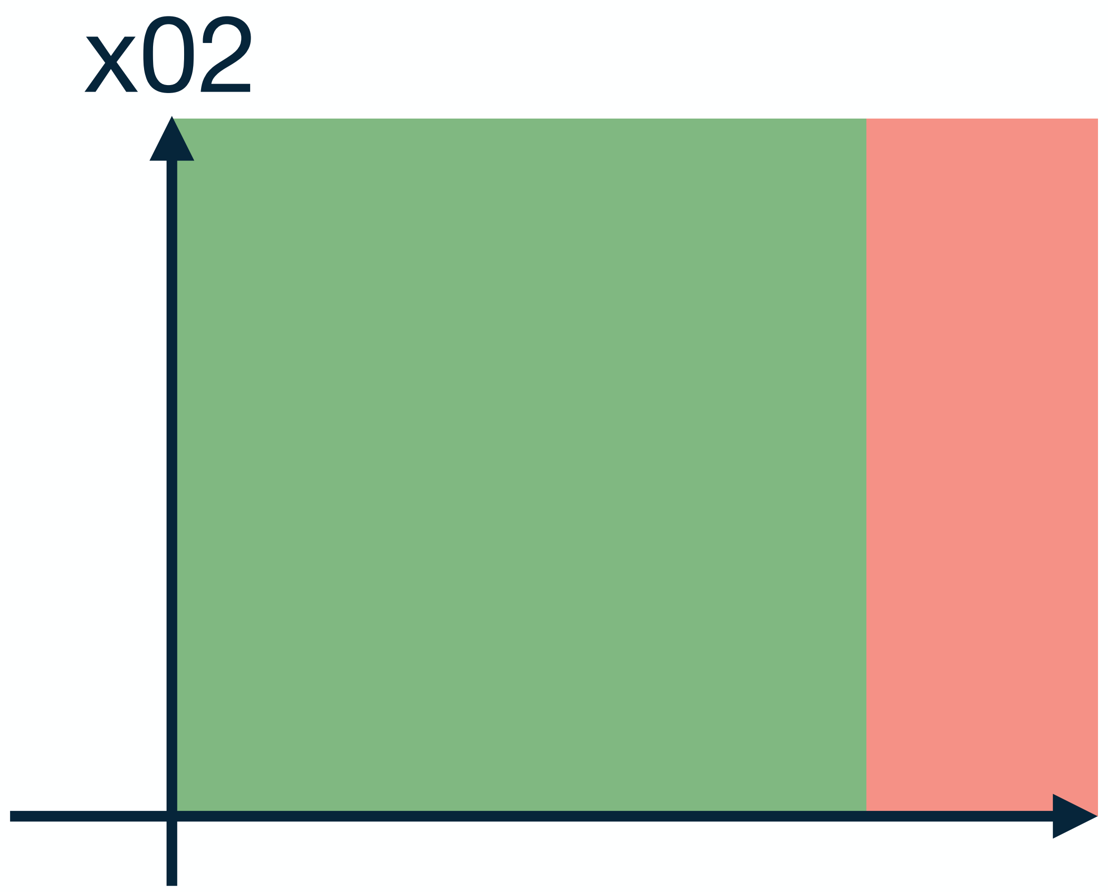
Abstract Non-Interference Properties

$$\mathscr{H} \stackrel{\text{def}}{=} \left\{ T \mid \forall t, t' \in T \colon \eta(t_0) = \eta(t'_0) \Rightarrow \rho(t_\omega) = \rho(t'_\omega) \right\}$$

Lemma

$$M \models \mathscr{H} \Leftrightarrow \forall A, B \in [\![M]\!]_{\bullet} \colon \rho(A_\omega) \neq \rho(B_\omega) \Rightarrow \eta(A_0) \neq \eta(B_0)$$



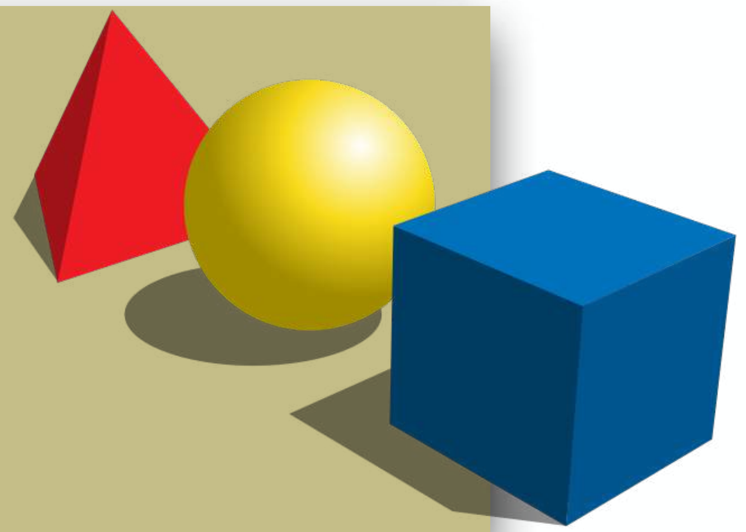


Abstract Interpretation

3-Step Recipe

practical tools
targeting specific programs

abstract semantics, abstract domains
algorithmic approaches to decide program properties

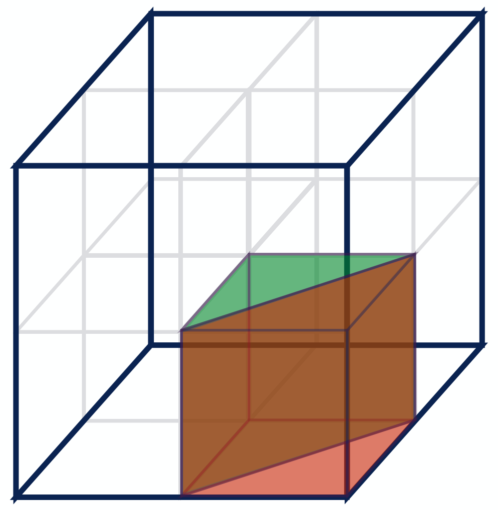


concrete semantics
mathematical models of the program behavior

Hyperproperty Verification [Urban20]

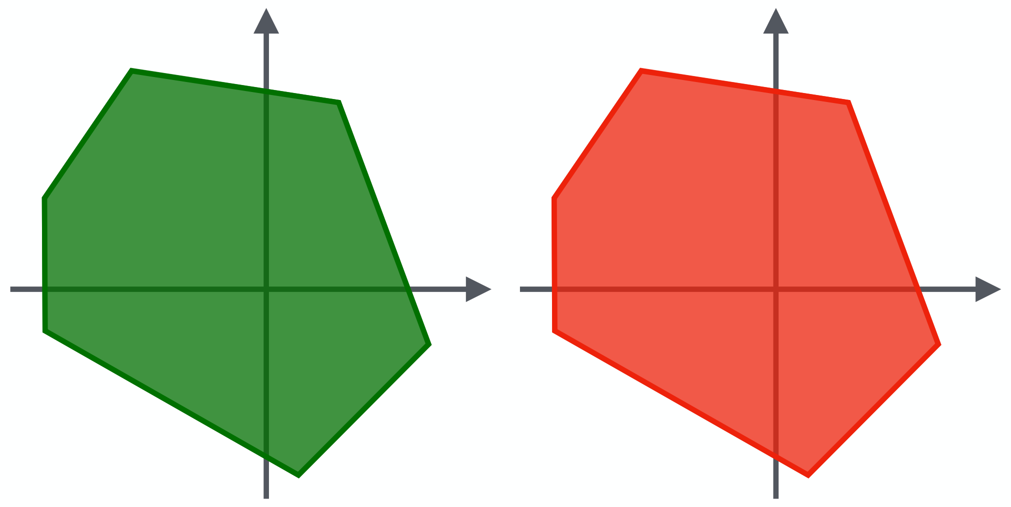
Naïve Static Backward Analysis

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```



① check for **disjunction** in corresponding **input partitions**:
disjoint → ✓ **safe**
otherwise → 🚨 **alarm**

① start from an **abstraction** for each possible classification outcome



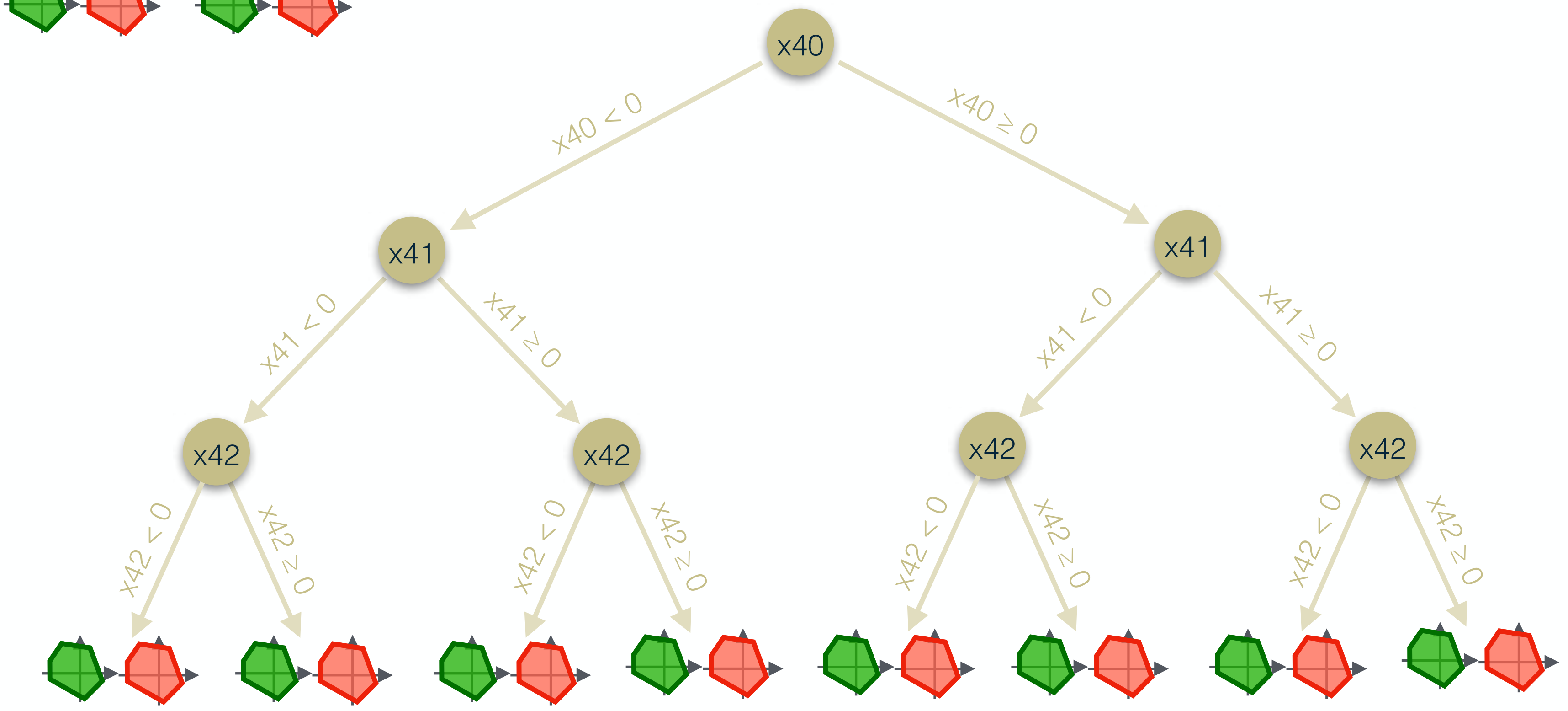
Naïve Static Backward Analysis

Disjunctive Polyhedra Abstract Domain

too many disjunctions!

```
x00 = float(input())
x0  = float(input())
      float(input())
x05 = float(input())
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))
x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))
x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))
x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```

4096 / 4096 (before x10)
512 / 512 (before x20)
64 / 64 (before x30)
8 / 8 (before x40)



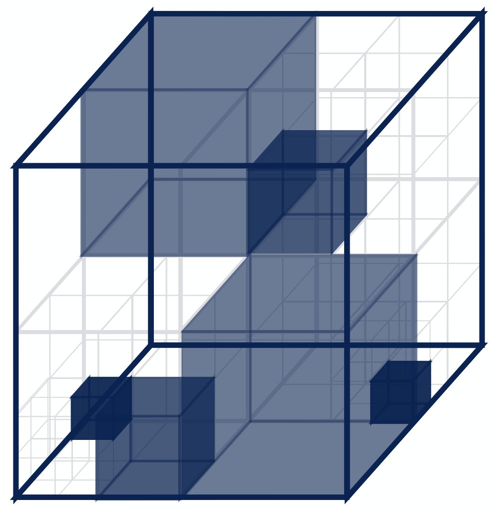
$(-4.556024) * x40 + (0.361304) * x41 + (-33.326096) * x42 + (3728) > 0$

$(4.556024) * x40 + (33.326096) * x42 - 3728 > 0$

Hyperproperty Verification [Urban20]

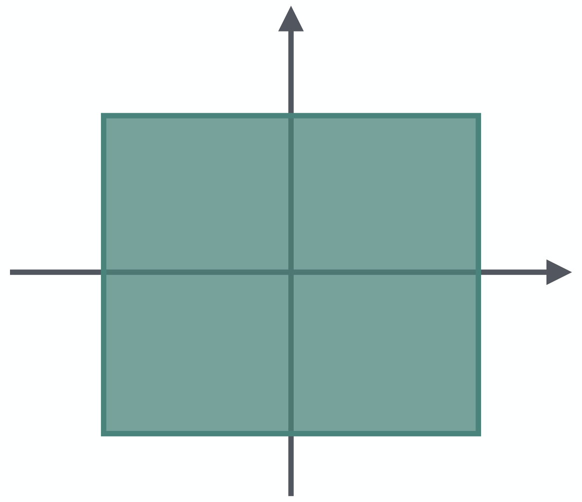
Static Forward Analysis

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```



① start from a **partition** of the input space

② proceed **forwards** **in parallel** from all partitions



③ check output for:
- **unique classification outcome** → ✓ **safe**
- **abstract activation pattern**

Static Forward Analysis

Symbolic & DeepPoly Product Abstract Domain

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

η: x00: [-1, 1] · x01: [-1, 1] · x02: T · x03: [-1, 0] · x04: [-1, 1] · x05: [-1, 1]

η: x00: [0, 1] · x01: [-1, 0] · x02: T · x03: [0.5, 1] · x04: [0, 1] · x05: [-1, 0]

η: x00: [0, 1] · x01: [0, 1] · x02: T · x03: [0.5, 1] · x04: [0, 1] · x05: [-1, 0]

…

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.09…62)*x02 + (2.03…51)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.09…01)*x02 + (2.12…38)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.34…43)*x02 + (2.61…76)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.72…16)*x12 + (-3.4…653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.24…51)*x12 + (-3.8…811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.66…05)*x12 + (-4.2…086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.19…44)*x22 + (-2.6…086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.42…07)*x22 + (-3.6…113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.66…07)*x22 + (-4.2…974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.94…60)*x32 + (-4.0…463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0…95998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.…5429)*x32 + (5.…4773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```

Activation pattern (x10–x42), shared by the second and third partitions: 1, 1, ?, 1, 1, 1, 1, 1, 1, 1, 0, ?

ρ: x50: [1, 1] · x51: [0, 0] ✔

ρ: x50: [0, 1] · x51: [0, 1]

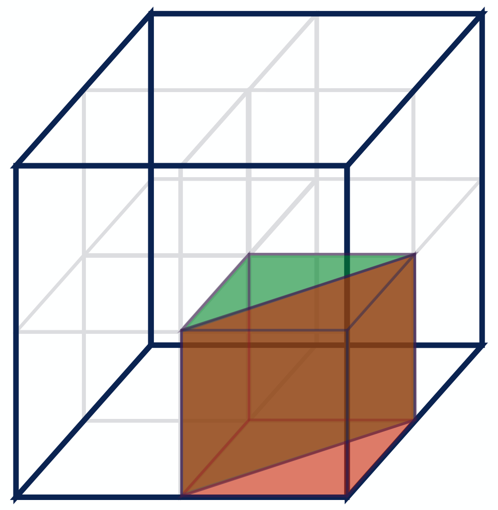
ρ: x50: [0, 1] · x51: [0, 1]

several partitions share the same abstract activation pattern

Hyperproperty Verification [Urban20]

Static Backward Analysis

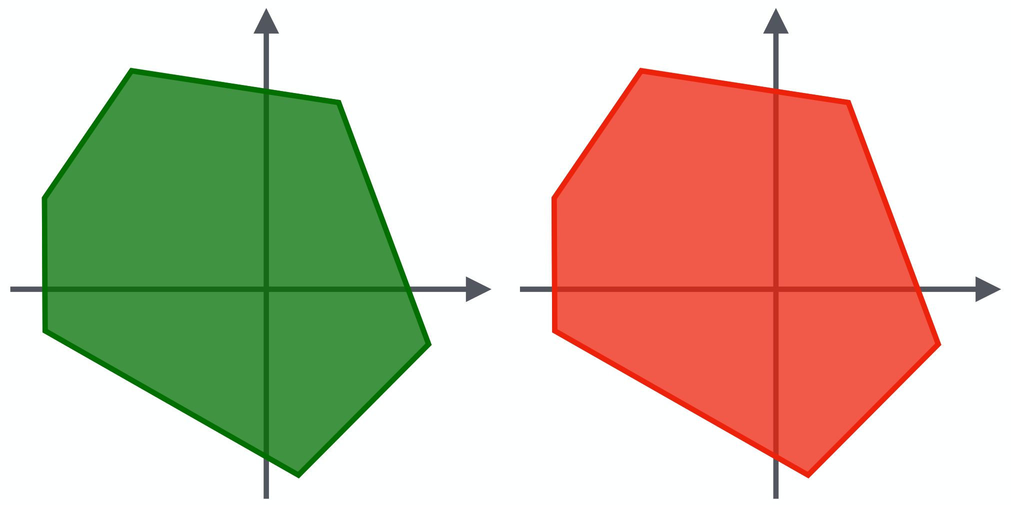
```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```



① check for **disjunction** in corresponding **input partitions**:
disjoint → ✔ **safe**
otherwise → 🚨 **alarm**

② proceed **backwards** in parallel **for each abstract activation pattern**

① start from an **abstraction** for each possible classification outcome



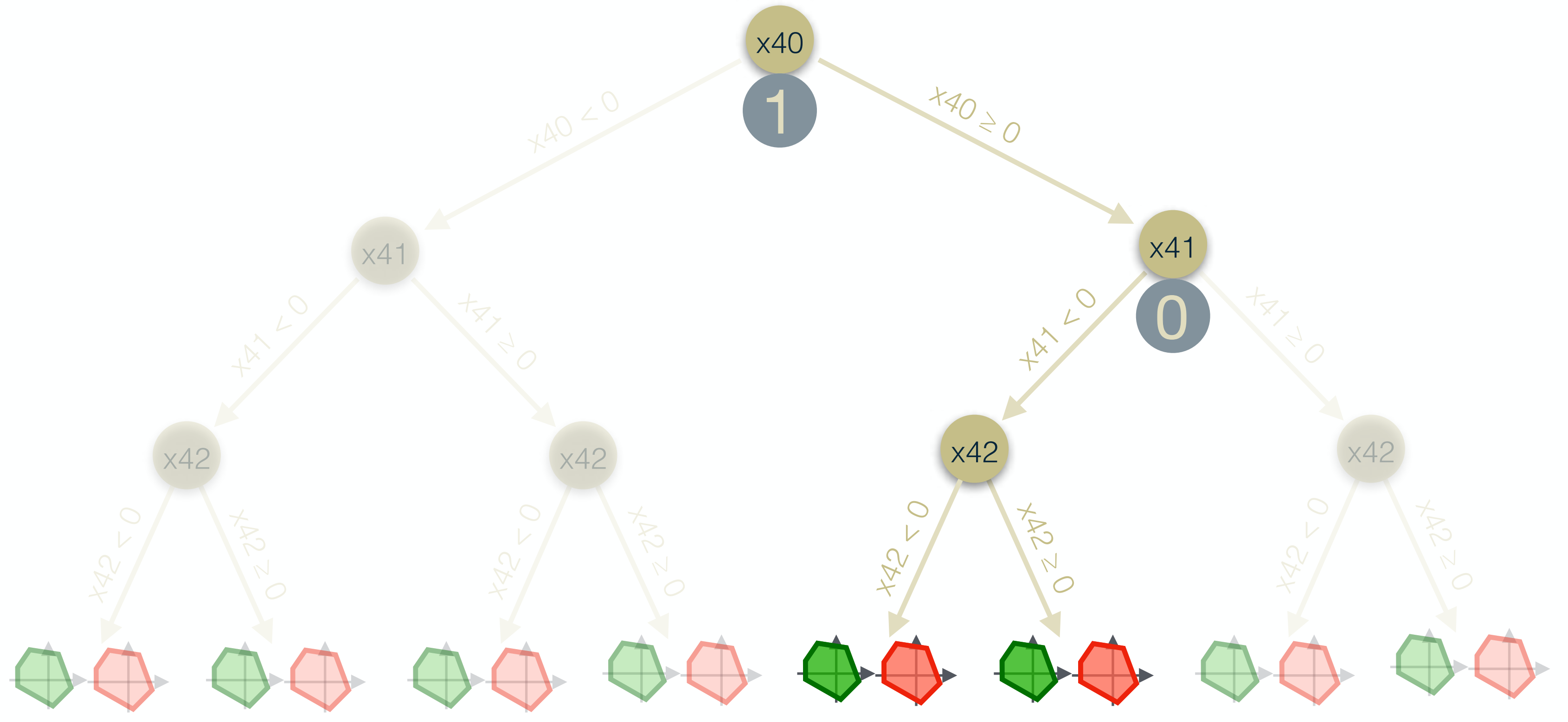
Static Backward Analysis

Symbolic & DeepPoly Product Abstract Domain

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

η:	η:
x00: [0, 1]	x00: [0, 1]
x01: [-1, 0]	x01: [0, 1]
x02: ⊤	x02: ⊤
x03: [0.5, 1]	x03: [0.5, 1]
x04: [0, 1]	x04: [0, 1]
x05: [-1, 0]	x05: [-1, 0]

```
1 x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
1 x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
? x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

1 x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
1 x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
1 x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

1 x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
1 x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
1 x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

1 x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
0 x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
? x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```



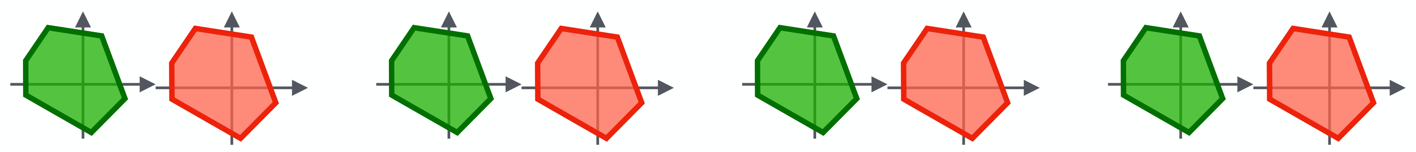
$(-4.556024) * x40 + (0.361304) * x41 + (-33.326096) * x42 + (3728) > 0$

$(4.556024) * x40 + (33.326096) * x42 - 3728 > 0$

Static Backward Analysis

Symbolic & DeepPoly Product Abstract Domain

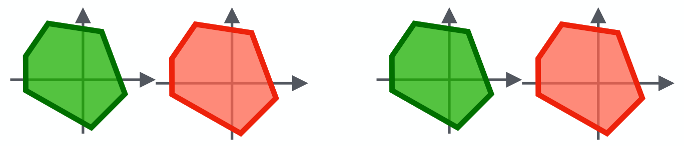
```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

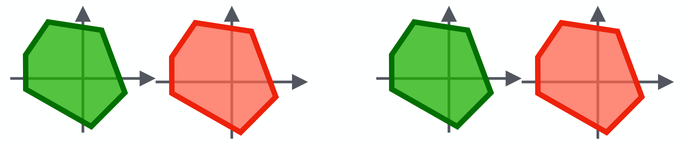


η:	η:	counterexample	
x00: [0, 1]	x00: [0, 1]	x00: 1	x00: 1
x01: [-1, 0]	x01: [0, 1]	x01: 1	x01: 1
x02: T	x02: T	x02: -1	x02: 1
x03: [0.5, 1]	x03: [0.5, 1]	x03: 1	x03: 1
x04: [0, 1]	x04: [0, 1]	x04: 1	x04: 1
x05: [-1, 0]	x05: [-1, 0]	x05: -1	x05: -1

✓ ✗

```
1 x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
1 x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
? x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

1 x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
1 x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
1 x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

⋮

1 x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
0 x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
? x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```





$(-4.556024) * x40 + (0.361304) * x41 + (-33.326096) * x42 + (3728) > 0$

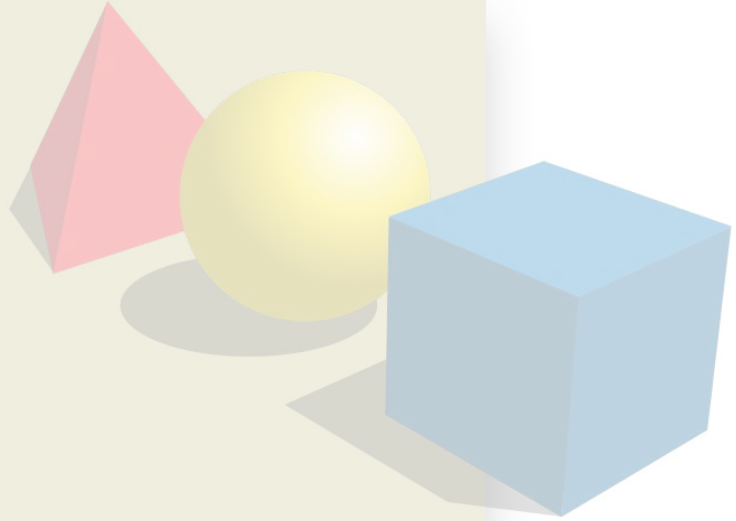
$(4.556024) * x40 + (33.326096) * x42 - 3728 > 0$

Abstract Interpretation

3-Step Recipe

practical tools
targeting specific programs

abstract semantics, abstract domains
algorithmic approaches to decide program properties

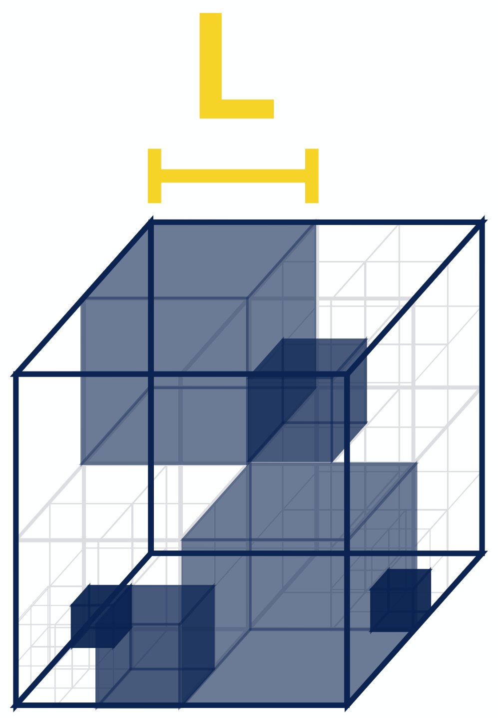


concrete semantics
mathematical models of the program behavior

Hyperproperty Verification [Urban20]

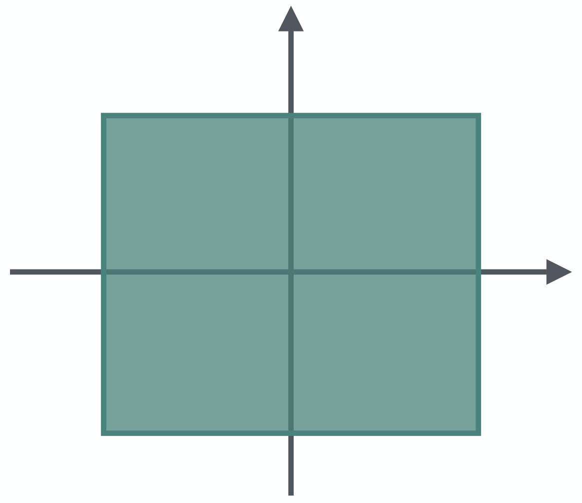
Static Forward Analysis

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

1 x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
1 x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
? x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

? x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
? x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
? x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

? x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
1 x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
0 x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

1 x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
0 x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
0 x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```



① **iteratively** partition the input space

② proceed **forwards** **in parallel** from all partitions



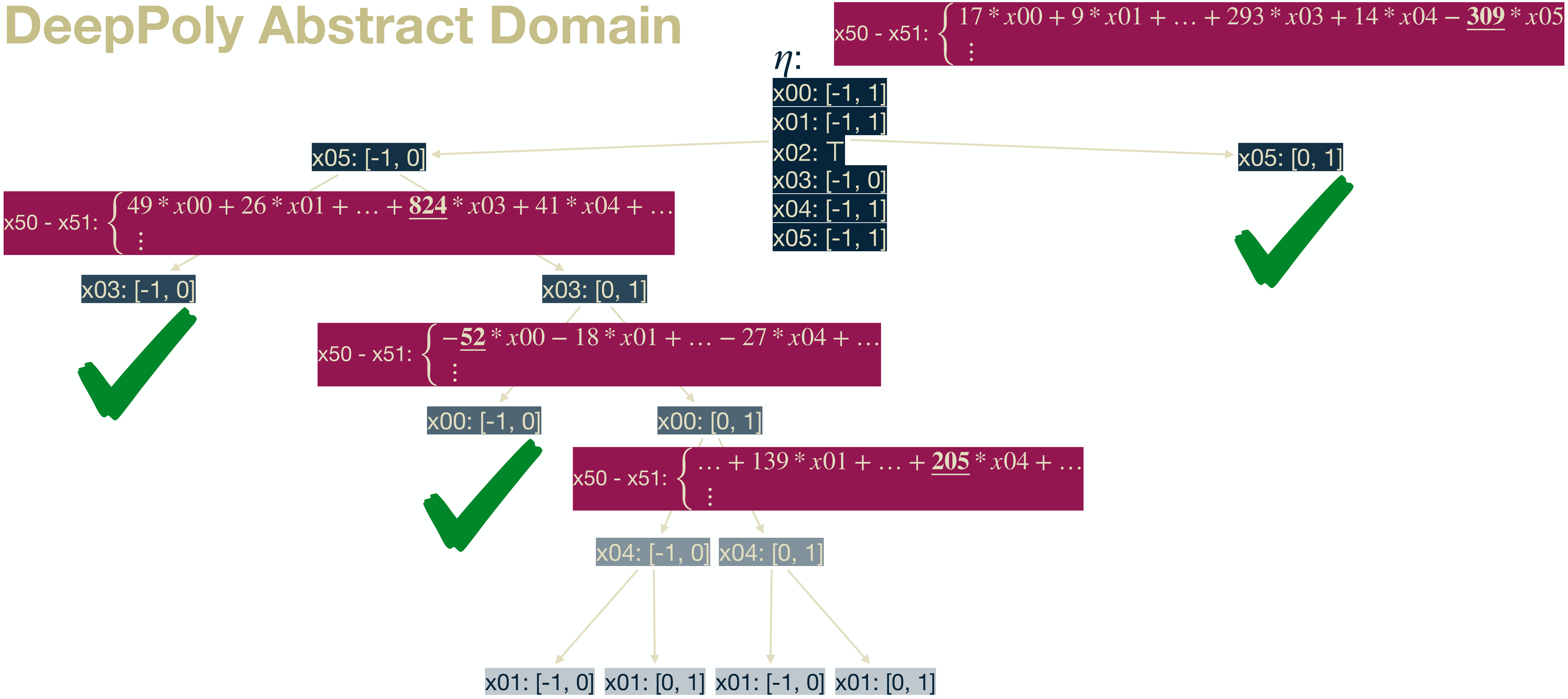
③ check output for:
- **unique classification outcome** → ✓ **safe**
- **abstract activation pattern** U

Partitioning Strategies: Interval Range

DeepPoly Abstract Domain

Partitioning Strategies: ReCIPH

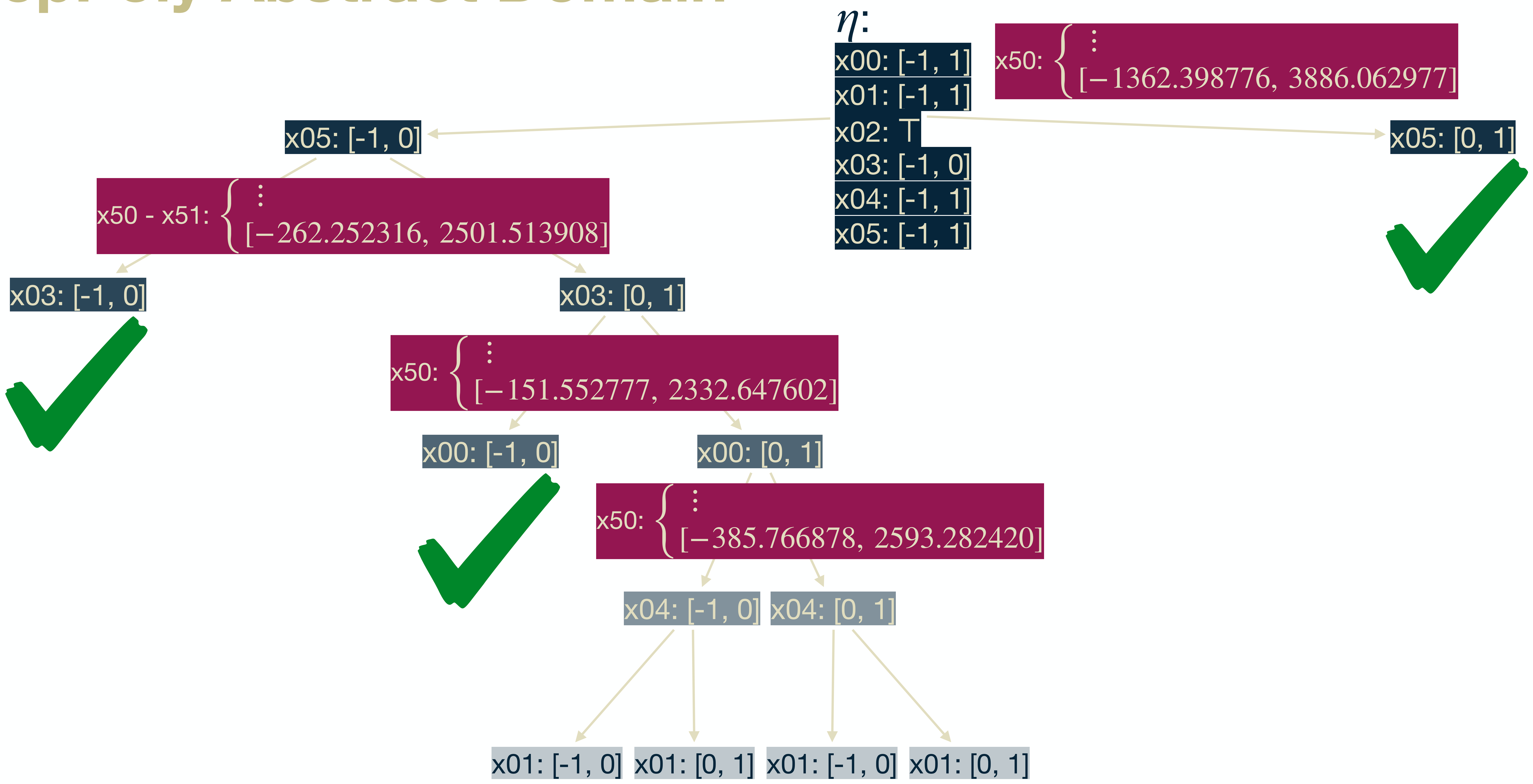
DeepPoly Abstract Domain



Durand, Lemesle, Chihani, CU, and Terrier. ReCIPH: Relational Coefficients for Input Partitioning Heuristic. In WFVML, 2022

Input Refinement $\not\Rightarrow$ Output Refinement

DeepPoly Abstract Domain



Input Refinement ⇏ Output Refinement

DeepPoly with Input Range Partitioning

Scalability-vs-Precision Tradeoff

Analyzed Input Space Percentage

L	U	Boxes	Symbolic	DeepPoly		Product	
				Input Range Partitioning	ReCIPH	Input Range Partitioning	ReCIPH
1	2	46,9 %	46,9 %	68,8 %	87,5 %	90,6 %	90,6 %
	6	46,9 %	46,9 %	68,8 %	87,5 %	90,6 %	90,6 %
0.5	2	76,9 %	89,2 %	100,0 %	100,0 %	100,0 %	100,0 %
	6	84,4 %	89,9 %	100,0 %	100,0 %	100,0 %	100,0 %

Execution Time

L	U	Boxes	Symbolic	DeepPoly		Product	
				Input Range Partitioning	ReCIPH	Input Range Partitioning	ReCIPH
1	2	0,08s	0,14s	0,26s	0,11s	0,26s	0,12s
	6	0,16s	0,31s	0,51s	0,20s	0,35s	0,20s
0.5	2	8,88s	5,76s	2,60s	1,61s	2,10s	1,61s
	6	64,67s	40,90s	2,65s	1,63s	2,10s	1,62s

Neural Network Verification

Neural Network Explainability

Abductive Explanations (AXp) [Marques-Silva21]

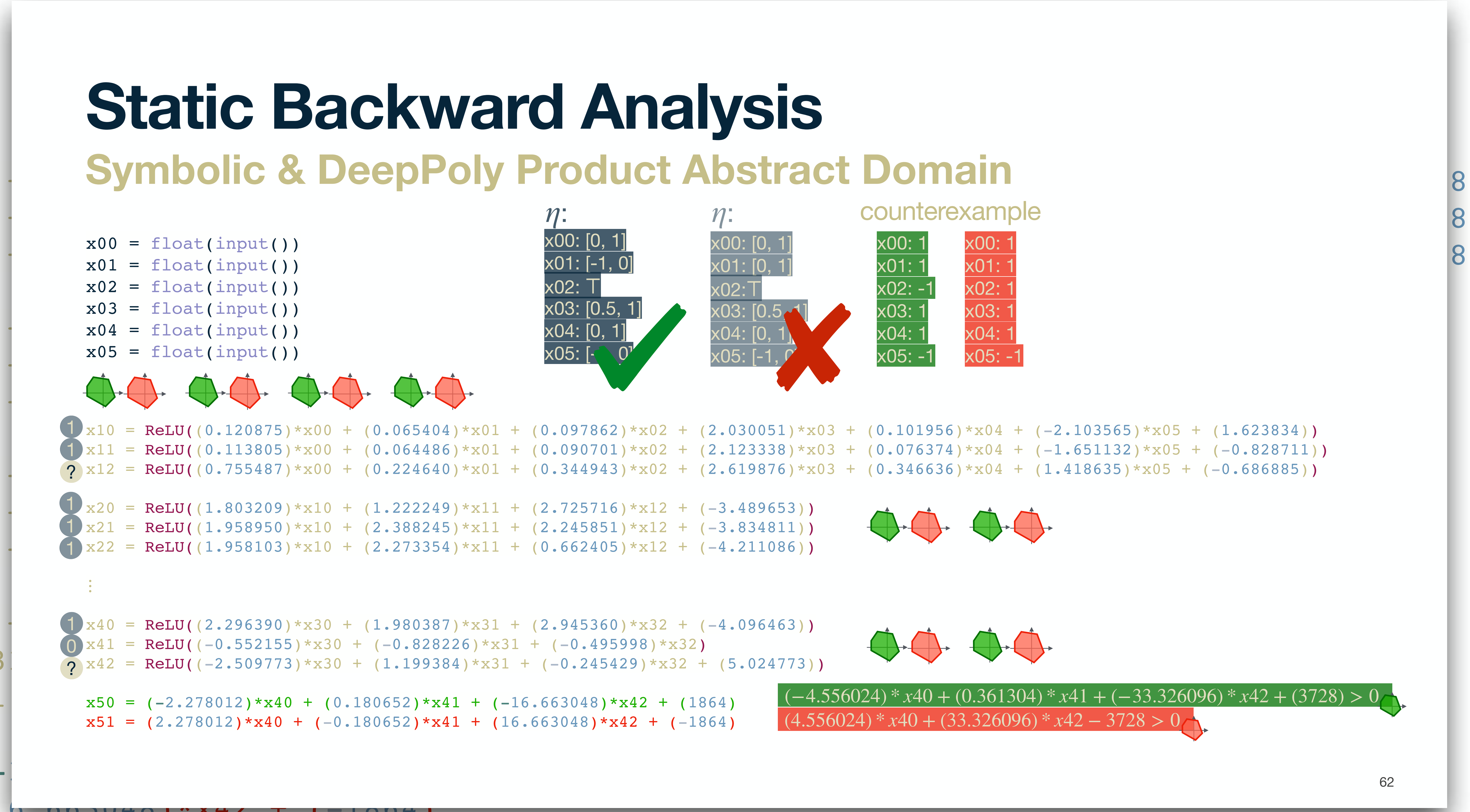
Subset-Minimal Set of Input Features Sufficient for Ensuring Prediction

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 ...834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 ...8711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 ...885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 ...
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 ...
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 ...

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 ...
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 ...
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 ...

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 ...
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x3...
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31...

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-...
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```

x:

x00: 1
x01: 1
x02: -1
x03: 1
x04: 1
x05: -1

AXp = { x02 }



Over-Approximating One AXp

Drop (i.e., Free) Input Features While AXp Condition Holds

SAME PREDICTION

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

x:
x00: [-1, 1]
x01: [-1, 1]
x02: -1
x03: [-1, 1]
x04: [-1, 1]
x05: [-1, 1]

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + ... + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + ... + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + ... + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + ...
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + ...
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + ...

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + ...
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + ...
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + ...

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + ...
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + ...
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + ...

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-2500)
```

{ x00, x01, x02, x03, x04, x05 } → x51

Free x00: { x01, x02, x03, x04, x05 } → x51

Free x01: { x02, x03, x04, x05 } → x51

Free x02: { x03, x04, x05 } → ✗

Free x03: { x02, x04, x05 } → x51

Free x04: { x02, x05 } → x51

Free x05: { x02 } → x51

AXp = { x02 }

Over-Approximating One AXp

Drop (i.e., Free) Input Features While AXp Condition Holds

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

x: x00: 1, x01: 1, x02: -1, x03: 1, x04: 1, x05: -1

BOXES

oAXp = { x02, x03, x05 }

SYMBOLIC

oAXp = { x00, x02, x03 }
oAXp = { x02, x03, x05 }

DEEPPOLY

oAXp = { x02, x03 }
oAXp = { x02, x05 }

= PRODUCT

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```

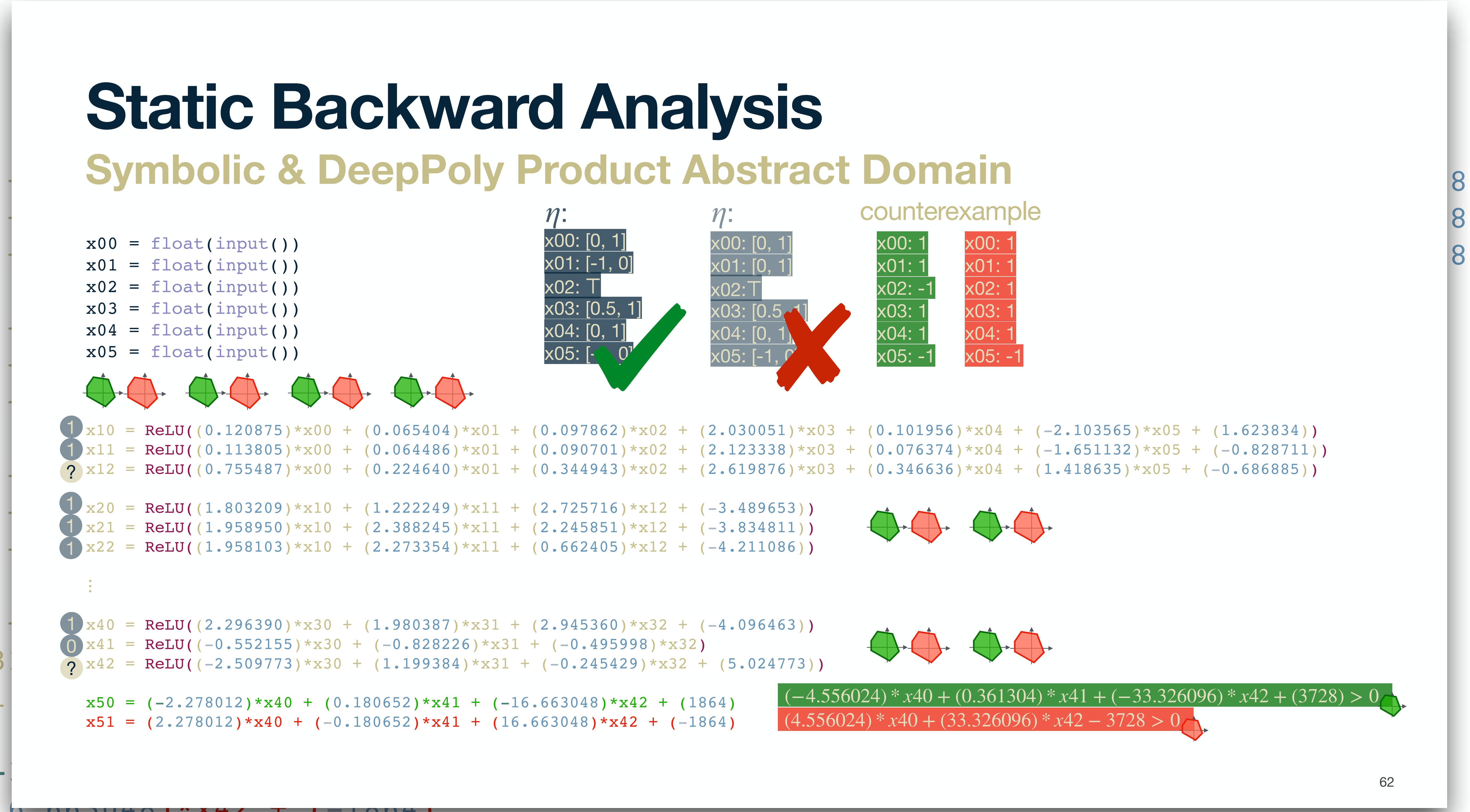
Contrastive Explanations (CXp) [Marques-Silva21]

Subset-Minimal Set of Input Features Sufficient for Changing Prediction

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())

x10 = ReLU((0.120875)*x00 + (0.065404)*x01 ...834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 ...8711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 ...885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 ...
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 ...
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 ...

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 ...
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 ...
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 ...

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 ...
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x3...
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31...

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-...
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```

x:
x00: 1
x01: 1
x02: -1
x03: 1
x04: 1
x05: -1

CXp = { x02 }



Under-Approximating One CXp

Drop (i.e., Fix) Input Features While CXp Condition Holds

DIFFERENT PREDICTIONS

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

x:
- x00: 1
- x01: 1
- x02: [-1, 1]
- x03: 1
- x04: 1
- x05: -1

{ x00, x01, x02, x03, x04, x05 } → x50 x51

Fix x00: { x01, x02, x03, x04, x05 } → x50 x51

Fix x01: { x02, x03, x04, x05 } → x50 x51

Fix x02: { x03, x04, x05 } → x51

Fix x03: { x02, x04, x05 } → x50 x51

Fix x04: { x02, x05 } → x50 x51

Fix x05: { x02 } → x50 x51

CXp = { x02 }

```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + ... (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + ... x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + ... (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + ...
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + ...
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + ...

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + ...
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + ...
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + ...

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + ...)
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 ...)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + ...))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```

Under-Approximating One CXp

Drop (i.e., Fix) Input Features While CXp Condition Holds

```
x00 = float(input())
x01 = float(input())
x02 = float(input())
x03 = float(input())
x04 = float(input())
x05 = float(input())
```

x:
x00: 1
x01: 1
x02: -1
x03: 1
x04: 1
x05: -1

BOXES
WAXp = { x02 }
WAXp = { x03 }
WAXp = { x05 }

SYMBOLIC
WAXp = { x02 }
WAXp = { x03 }
WAXp = { x00, x05 }

DEEPPOLY
WAXp = { x02 }
WAXp = { x03, x05 }

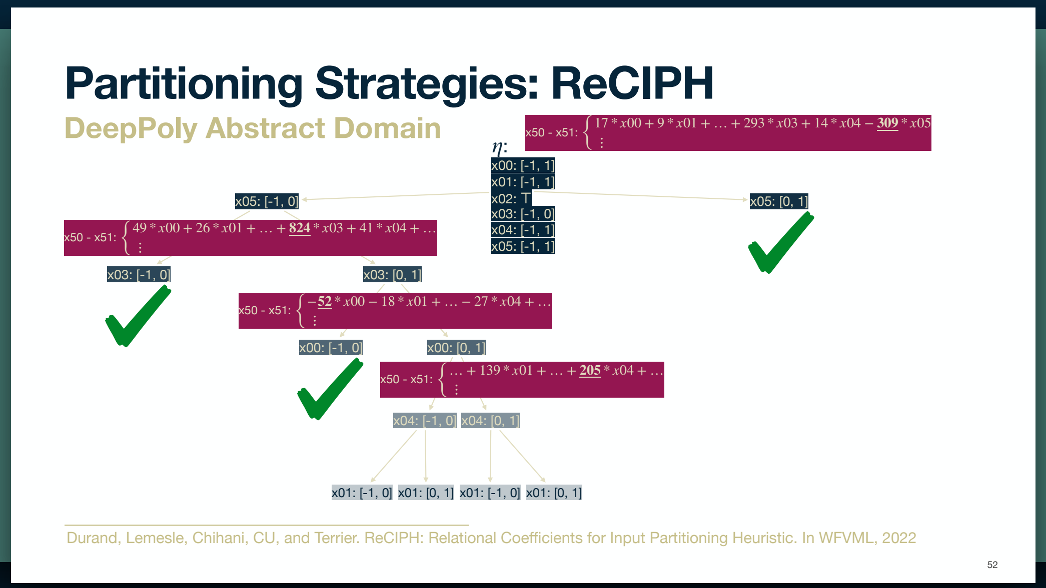
= PRODUCT

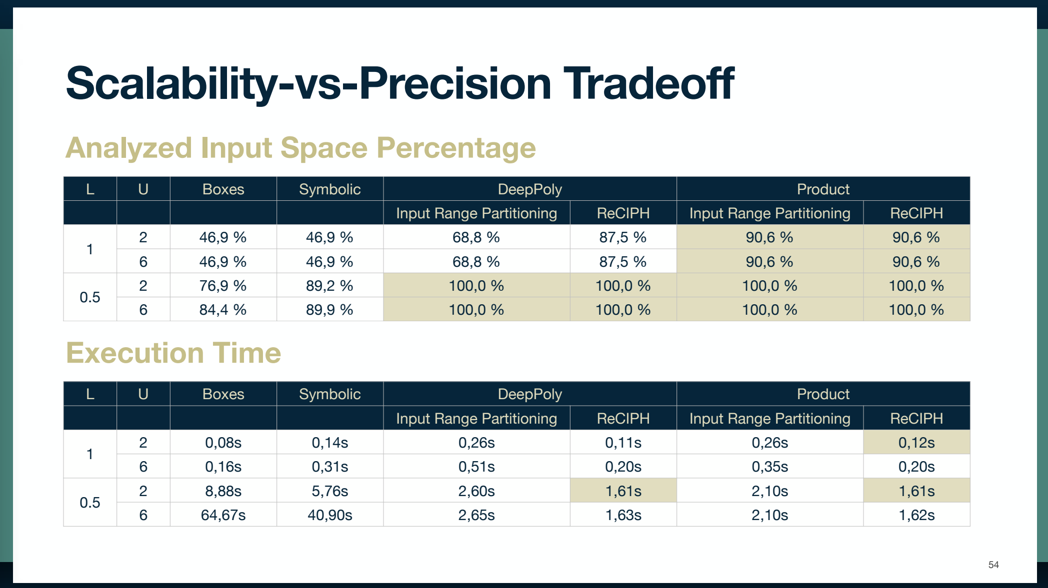
```
x10 = ReLU((0.120875)*x00 + (0.065404)*x01 + (0.097862)*x02 + (2.030051)*x03 + (0.101956)*x04 + (-2.103565)*x05 + (1.623834))
x11 = ReLU((0.113805)*x00 + (0.064486)*x01 + (0.090701)*x02 + (2.123338)*x03 + (0.076374)*x04 + (-1.651132)*x05 + (-0.828711))
x12 = ReLU((0.755487)*x00 + (0.224640)*x01 + (0.344943)*x02 + (2.619876)*x03 + (0.346636)*x04 + (1.418635)*x05 + (-0.686885))

x20 = ReLU((1.803209)*x10 + (1.222249)*x11 + (2.725716)*x12 + (-3.489653))
x21 = ReLU((1.958950)*x10 + (2.388245)*x11 + (2.245851)*x12 + (-3.834811))
x22 = ReLU((1.958103)*x10 + (2.273354)*x11 + (0.662405)*x12 + (-4.211086))

x30 = ReLU((1.735994)*x20 + (0.666507)*x21 + (3.192344)*x22 + (-2.627086))
x31 = ReLU((2.327110)*x20 + (2.685314)*x21 + (1.424807)*x22 + (-3.695113))
x32 = ReLU((2.147212)*x20 + (2.285599)*x21 + (2.665507)*x22 + (-4.299974))

x40 = ReLU((2.296390)*x30 + (1.980387)*x31 + (2.945360)*x32 + (-4.096463))
x41 = ReLU((-0.552155)*x30 + (-0.828226)*x31 + (-0.495998)*x32)
x42 = ReLU((-2.509773)*x30 + (1.199384)*x31 + (-0.245429)*x32 + (5.024773))

x50 = (-2.278012)*x40 + (0.180652)*x41 + (-16.663048)*x42 + (1864)
x51 = (2.278012)*x40 + (-0.180652)*x41 + (16.663048)*x42 + (-1864)
```

Verification and Explainability

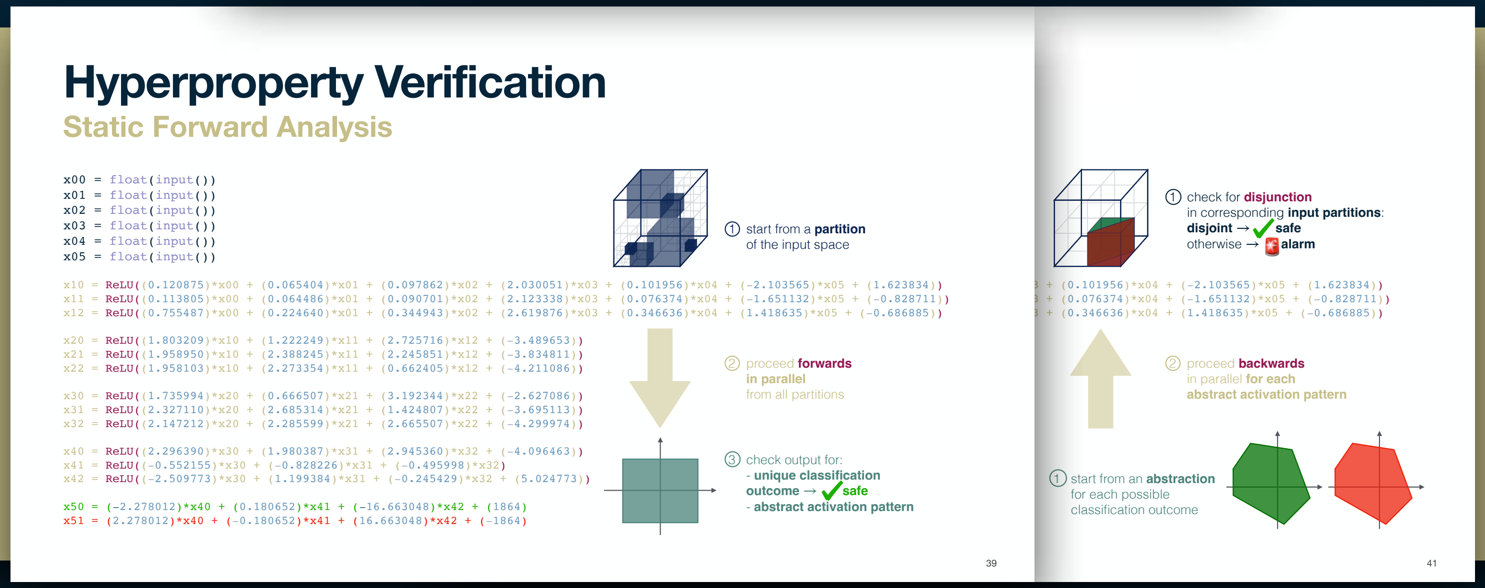
Safety-Critical Neural Networks

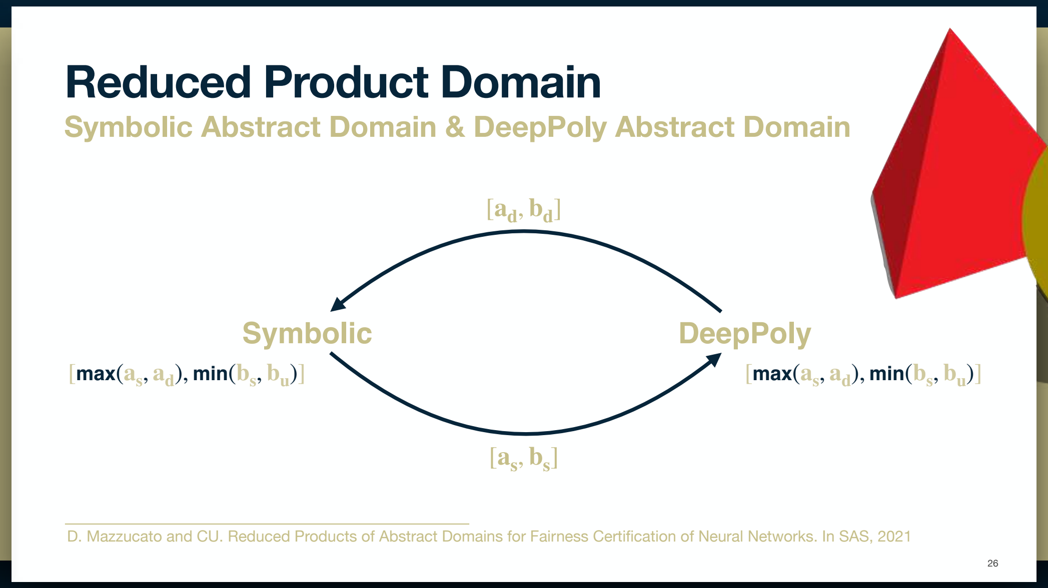
practical tools
targeting specific programs

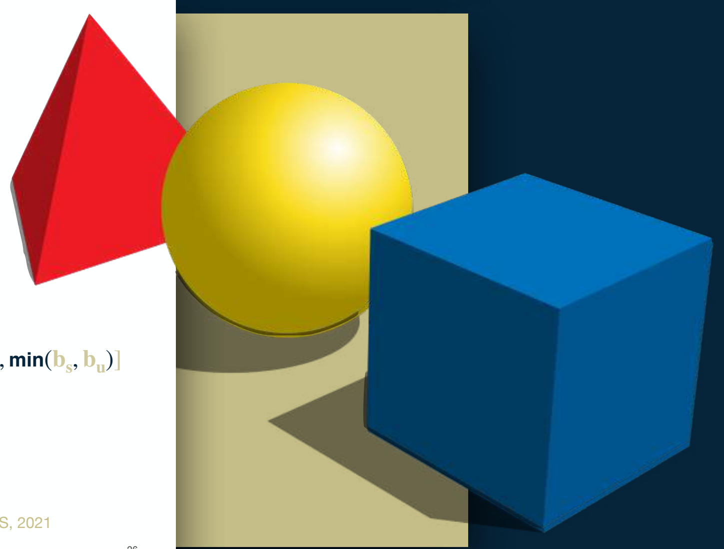




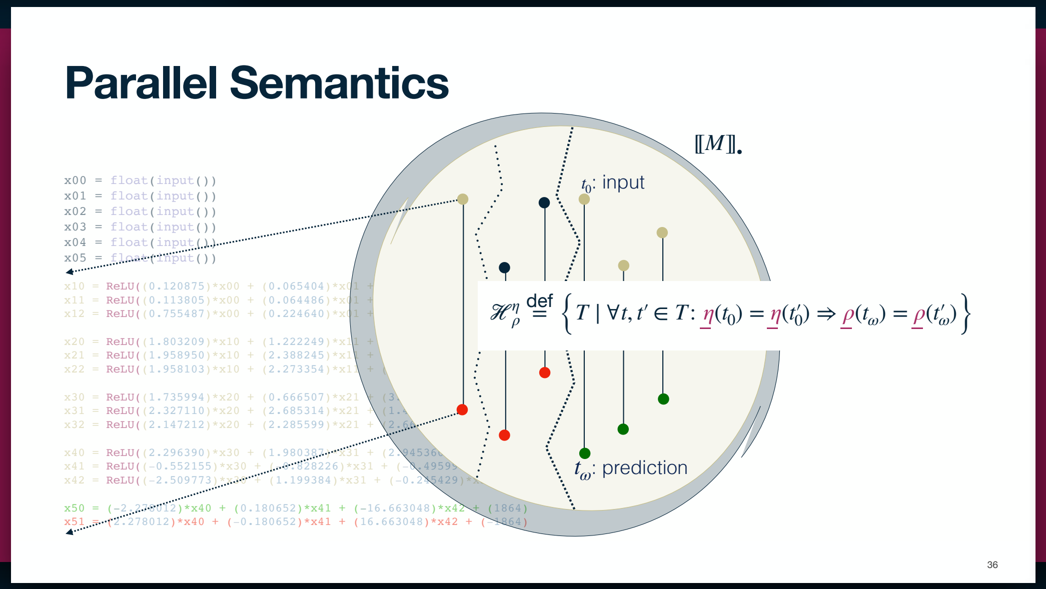
algorithmic approaches
to decide program properties

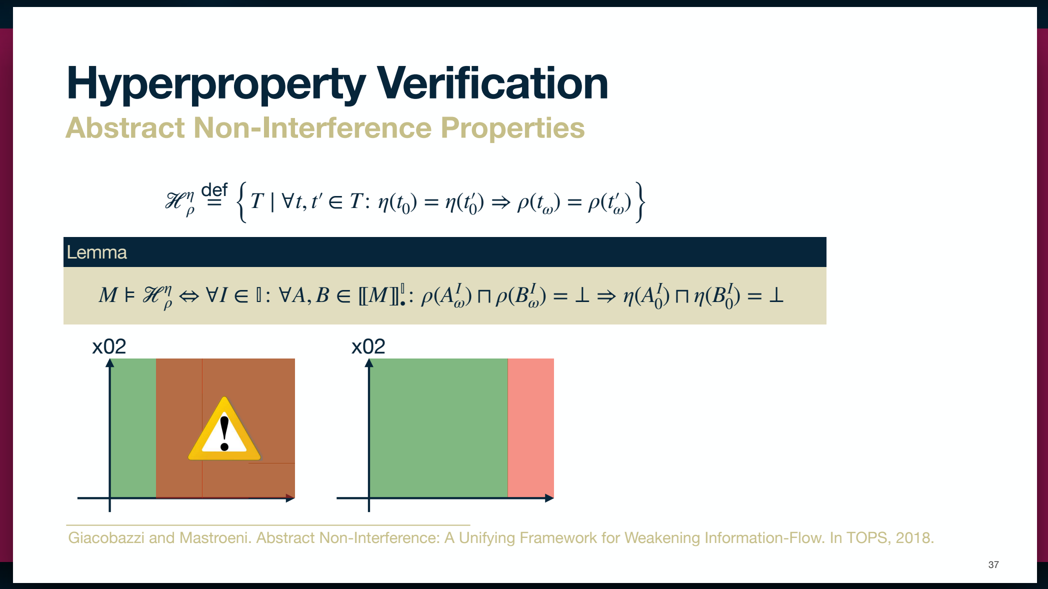






mathematical models
of the program behavior





THANKS!

References

[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.
symbolic abstraction

[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.
deeppoly abstraction

[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.
hypersafety verification

[Marques-Silva21] **João Marques-Silva, Thomas Gerspacher, Martin C. Cooper, Alexey Ignatiev, and Nina Narodytska**. Explanations for Monotonic Classifiers. In ICML, pages 7469-7479, 2021.
[Wu23] **Min Wu, Haoze Wu, Clark W. Barrett**. VeriX: Towards Verified Explainability of Deep Neural Networks. In NeurIPS, 2023.
logic-based explanations