La confiance avec le contrôle : spécification et vérification d'hyperpropriétés sur réseaux de neurones

Journées Francophones des Langages Applicatifs, Oberbronn 2026

Guilhem Ardouin, Michele Alberti, Julien Girard-Satabin

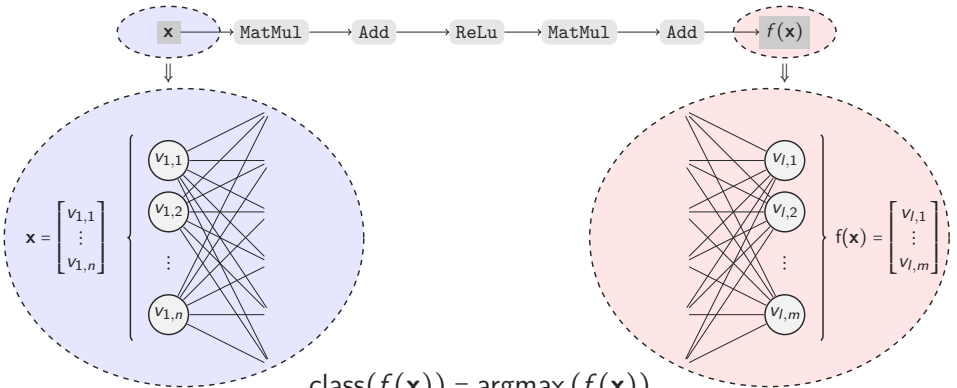
27 janvier 2026

Vérifier formellement des réseaux de neurones

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$$

Vérifier formellement des réseaux de neurones

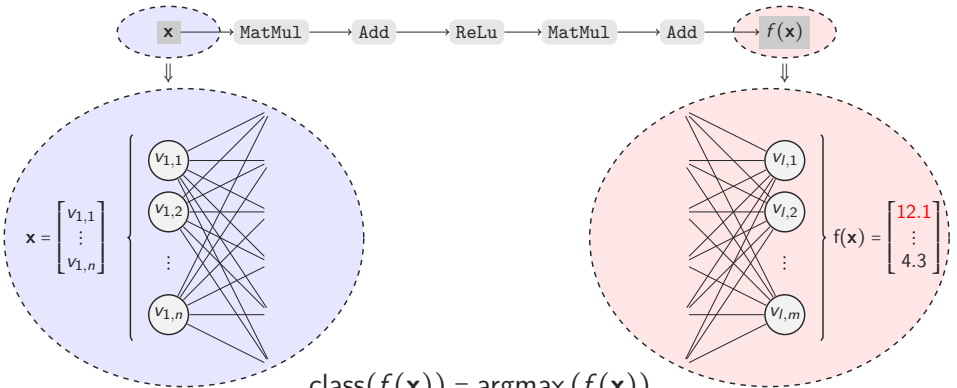
$$f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$$



$$\text{class}(f(\mathbf{x})) = \text{argmax}\,(f(\mathbf{x}))$$

Vérifier formellement des réseaux de neurones

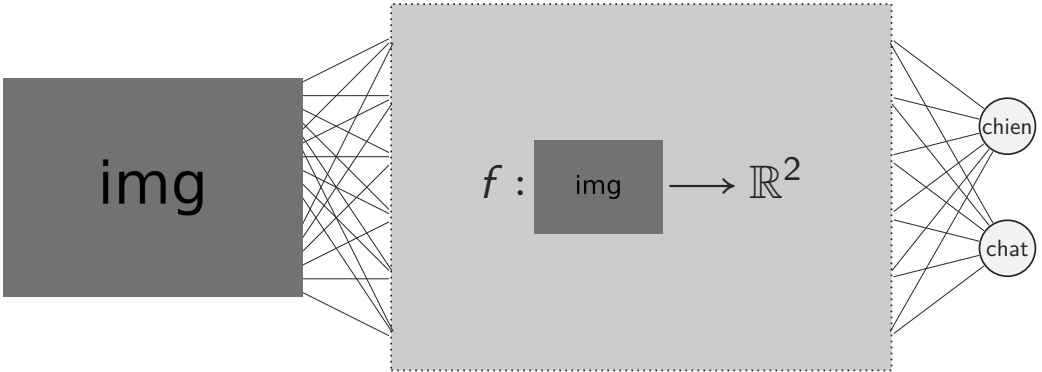
$$f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$$



$$\text{class}(f(\mathbf{x})) = \text{argmax}\,(f(\mathbf{x}))$$

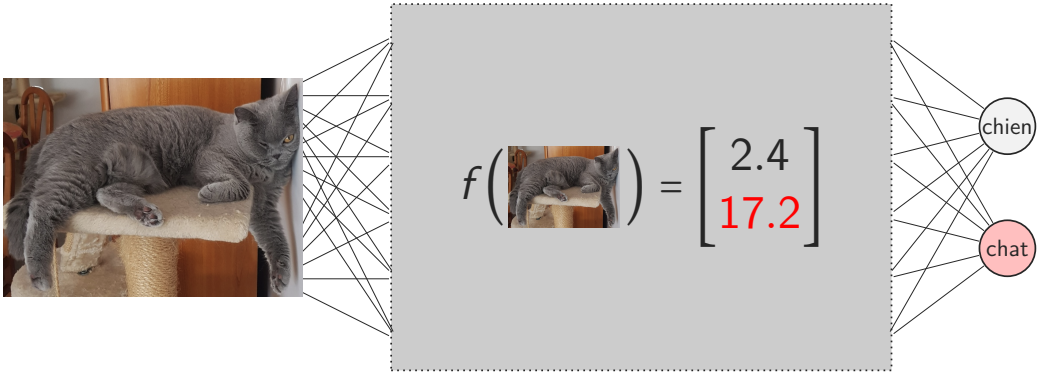
Quelles propriétés vérifier ?

Robustesse locale = la classification d'une entrée est la même pour une entrée « proche »



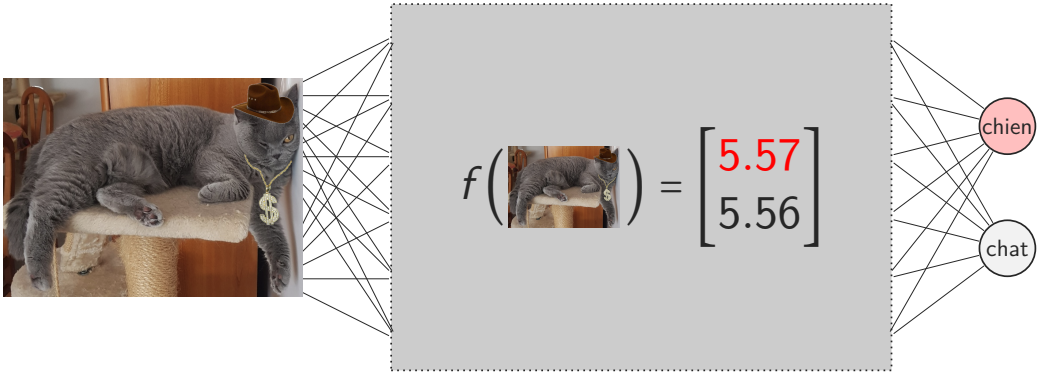
Quelles propriétés vérifier ?

Robustesse locale = la classification d'une entrée est la même pour une entrée « proche »



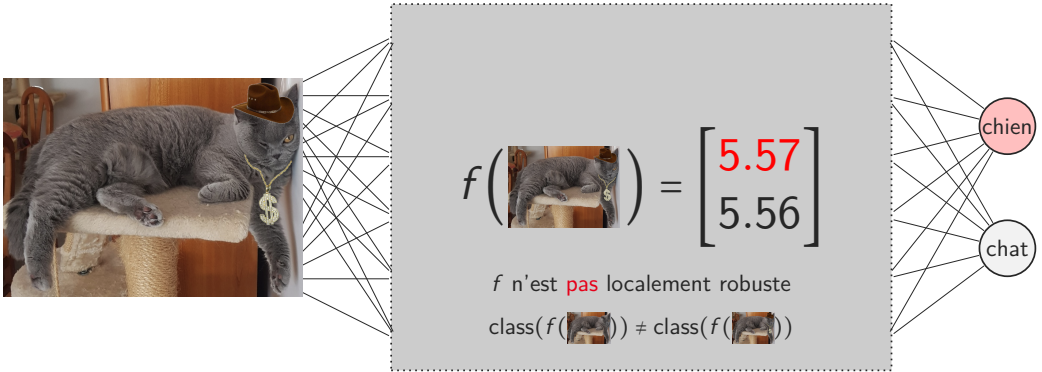
Quelles propriétés vérifier ?

Robustesse locale = la classification d'une entrée est la même pour une entrée « proche »



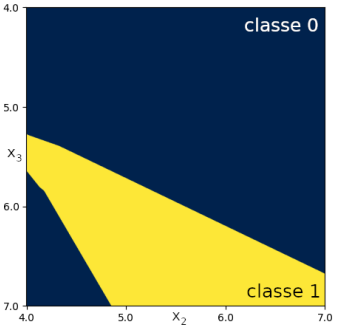
Quelles propriétés vérifier ?

Robustesse locale = la classification d'une entrée est la même pour une entrée « proche »



Quelles propriétés vérifier ?

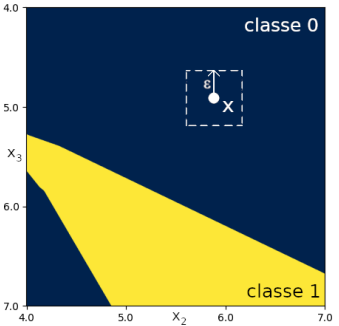
Robustesse locale $= \forall \mathbf{x}' \in \mathbb{R}^n . ||\mathbf{x} - \mathbf{x}'|| \leq \varepsilon \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$ (**x** fixé)



$$||\mathbf{x}|| = \max_i x_i$$

Quelles propriétés vérifier ?

Robustesse locale $= \forall \mathbf{x}' \in \mathbb{R}^n . ||\mathbf{x} - \mathbf{x}'|| \leq \varepsilon \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$ (**x** fixé)



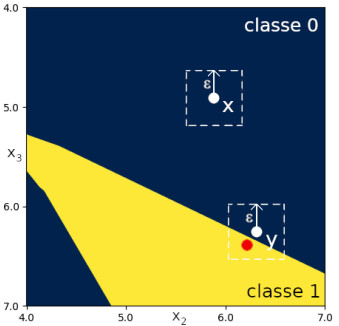
$$||\mathbf{x}|| = \max_i x_i$$

$$\forall \mathbf{x}' . ||\mathbf{x} - \mathbf{x}'|| \leq \varepsilon, \ \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

f robuste en $\mathbf{x}$

Quelles propriétés vérifier ?

Robustesse locale $= \forall \mathbf{x}' \in \mathbb{R}^n . ||\mathbf{x} - \mathbf{x}'|| \leq \varepsilon \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$ (**x** fixé)



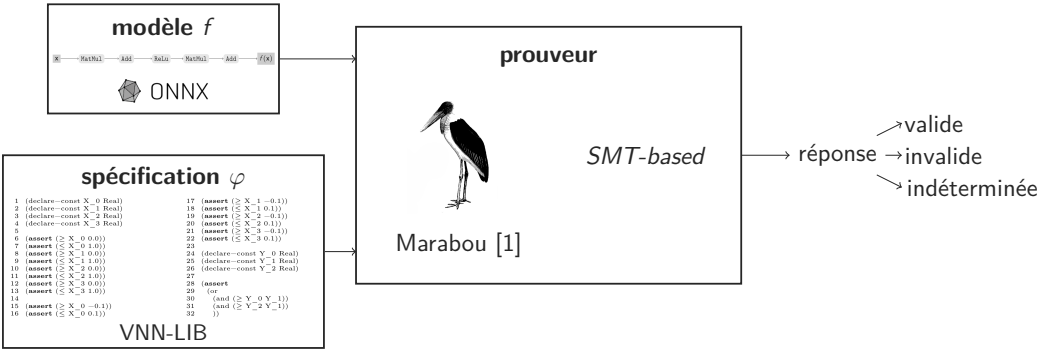
$$||\mathbf{x}|| = \max_i x_i$$

$$||\mathbf{y} - \bullet|| \leq \varepsilon \wedge \text{class}(f(\mathbf{y})) \neq \text{class}(f(\bullet))$$

contre-exemple : f non robuste en **y**

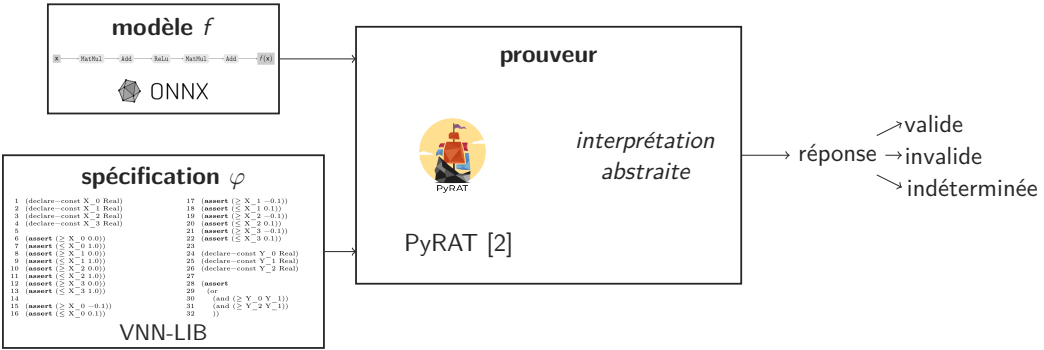
Comment vérifier cette propriété ?

Via différents prouveurs spécialisés :



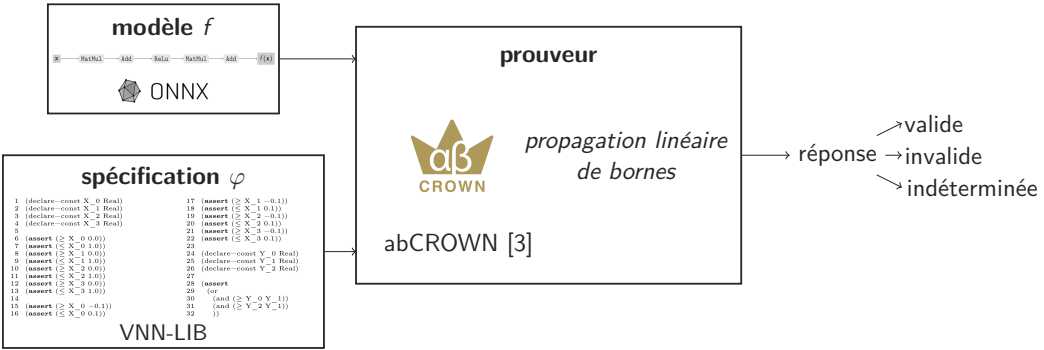
Comment vérifier cette propriété ?

Via différents prouveurs spécialisés :



Comment vérifier cette propriété ?

Via différents prouveurs spécialisés :



Vérifier formellement des réseaux de neurones

Quelles propriétés vérifier ?

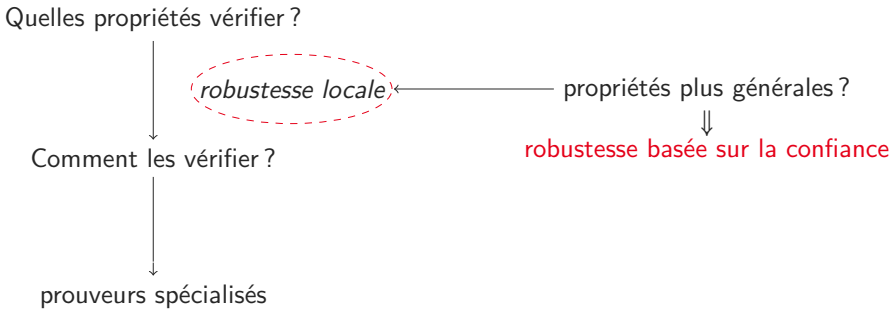
↓ *robustesse locale*

Comment les vérifier ?

↓

prouveurs spécialisés

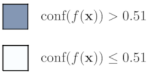
Vérifier formellement des réseaux de neurones

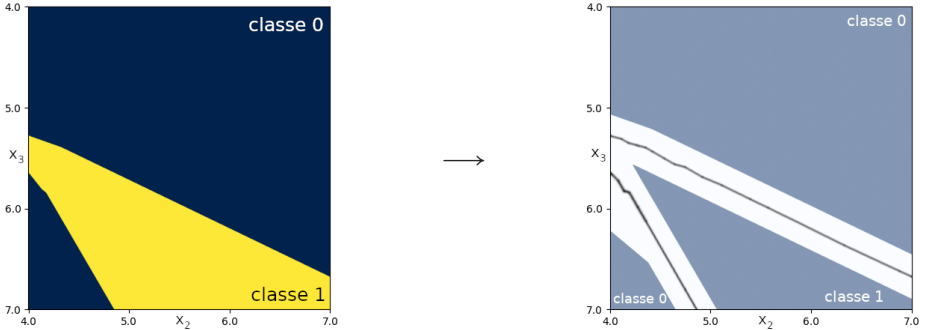


Robustesse basée sur la confiance

Notion de *confiance* : mesure de la confiance sur la prédiction du réseau

→ seules les entrées dont $\text{conf}(f(\mathbf{x})) > \kappa$ sont considérées

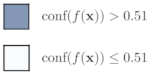


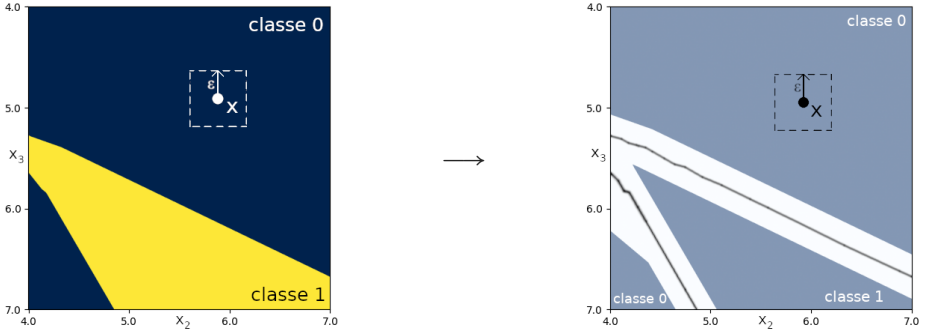


Robustesse basée sur la confiance

Notion de *confiance* : mesure de la confiance sur la prédiction du réseau

→ seules les entrées dont $\text{conf}(f(\mathbf{x})) > \kappa$ sont considérées

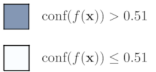


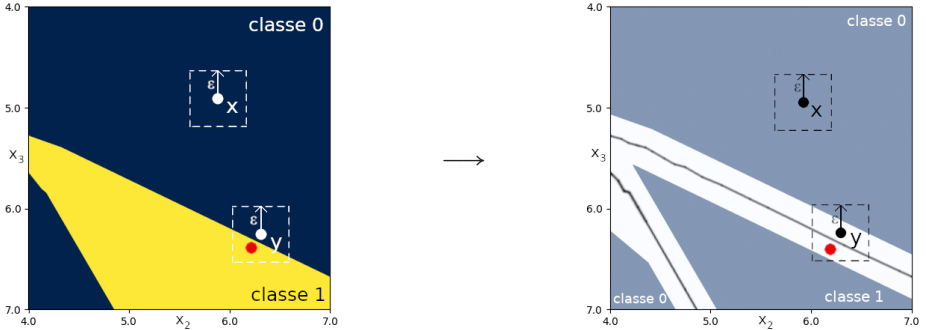


Robustesse basée sur la confiance

Notion de *confiance* : mesure de la confiance sur la prédiction du réseau

→ seules les entrées dont conf$(f(\mathbf{x})) > \kappa$ sont considérées

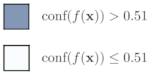


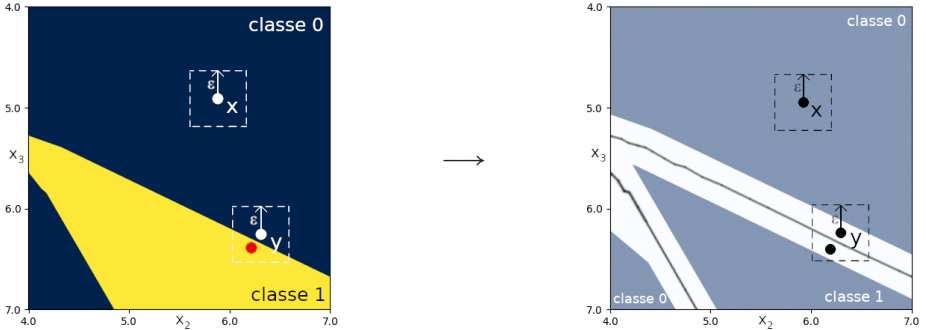


Robustesse basée sur la confiance

Notion de *confiance* : mesure de la confiance sur la prédiction du réseau

→ seules les entrées dont conf$(f(\mathbf{x})) > \kappa$ sont considérées

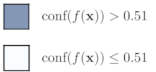


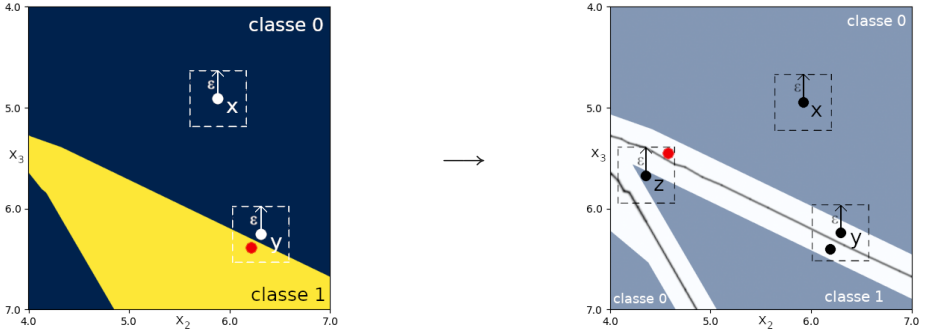


Robustesse basée sur la confiance

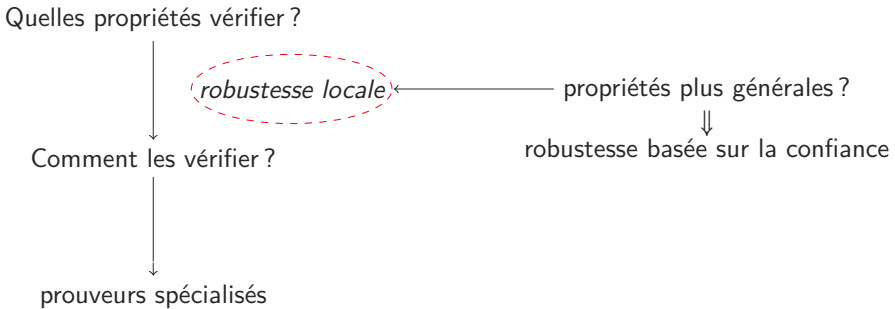
Notion de *confiance* : mesure de la confiance sur la prédiction du réseau

→ seules les entrées dont conf$(f(\mathbf{x})) > \kappa$ sont considérées

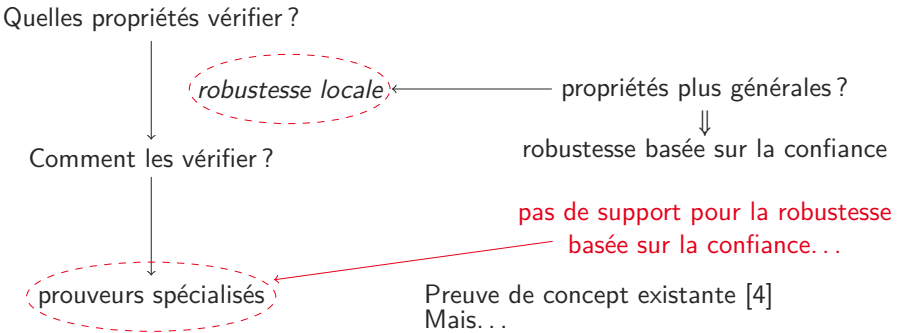




Vérifier la robustesse basée sur la confiance ?



Vérifier la robustesse basée sur la confiance ?



Vérifier la robustesse basée sur la confiance ?

Objectif : pouvoir *spécifier* et *vérifier* la robustesse basée sur la confiance

Vérifier la robustesse basée sur la confiance ?

Objectif : pouvoir *spécifier* et *vérifier* la robustesse basée sur la confiance

$$\Downarrow$$

CAISAR

Vérifier la robustesse basée sur la confiance ?

Objectif : pouvoir *spécifier* et *vérifier* la robustesse basée sur la confiance

$$\Downarrow$$

CAISAR

Questions de recherche :

- quels sont les verrous scientifiques pour le support de cette propriété dans les outils actuels ?
- quelles solutions peuvent-être mises en place ?

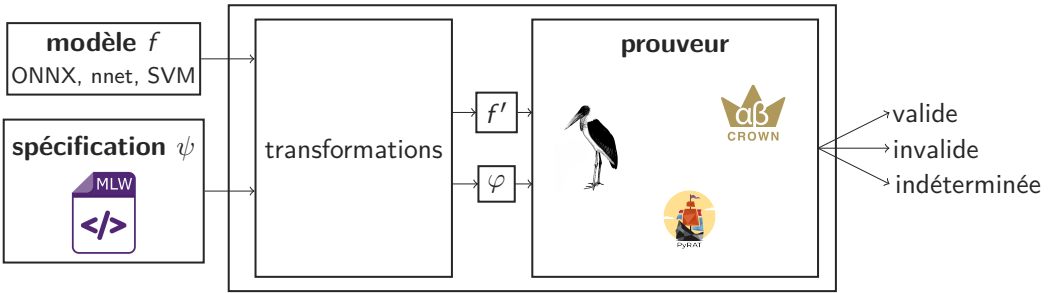
CAISAR [5]

CAISAR = **C**haracterizing **A**rtificial **I**ntelligence **S**afety **A**nd **R**obustness
Plateforme pour la spécification et vérification de modèles d'apprentissage automatique
→ utilise des prouveurs *externes* pour la vérification

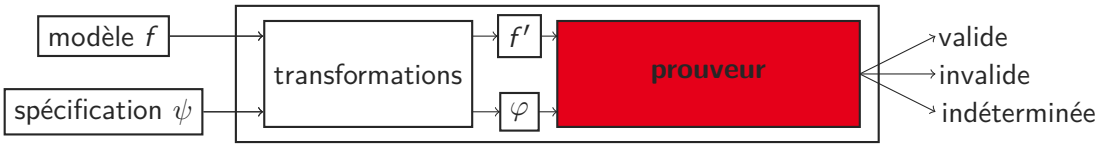
CAISAR

CAISAR [5]

CAISAR = **C**haracterizing **A**rtificial **I**ntelligence **S**afety **A**nd **R**obustness
Plateforme pour la spécification et vérification de modèles d'apprentissage automatique
→ utilise des prouveurs *externes* pour la vérification

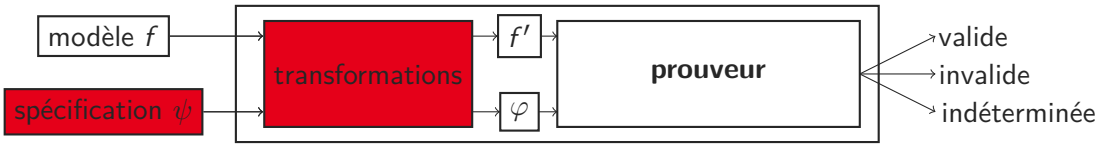


CAISAR : vérifier la robustesse basée sur la confiance ?



Rappel : la robustesse basée sur la confiance n'est pas supportée par les prouveurs

CAISAR : vérifier la robustesse basée sur la confiance ?



Rappel : la robustesse basée sur la confiance n'est pas supportée par les prouveurs

Enjeux :

1. spécifier la robustesse basée sur la confiance dans CAISAR
2. appliquer des transformations pour la rendre vérifiable (sans modifier les prouveurs)

Spécifier la robustesse (globale) basée sur la confiance ?

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

Spécifier la robustesse (globale) basée sur la confiance ?

contrainte arithmétique sur
plusieurs variables d'entrée

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \overbrace{\bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i} \wedge \text{conf}(f(\mathbf{x})) > \kappa \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

Spécifier la robustesse (globale) basée sur la confiance ?

contrainte arithmétique sur plusieurs variables d'entrée

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \overbrace{\bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i} \wedge \underbrace{\mathsf{conf}(f(\mathbf{x})) > \kappa} \Rightarrow \mathsf{class}(f(\mathbf{x})) = \mathsf{class}(f(\mathbf{x}'))$$

contrainte d'entrée sur une variable de sortie

Spécifier la robustesse (globale) basée sur la confiance ?

contrainte arithmétique sur plusieurs variables d'entrée

plusieurs exécutions

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i \wedge \underbrace{\text{conf}(f(\mathbf{x})) > \kappa}_{\text{contrainte d'entrée sur une variable de sortie}} \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

Spécifier la robustesse (globale) basée sur la confiance ?

contrainte arithmétique sur plusieurs variables d'entrée

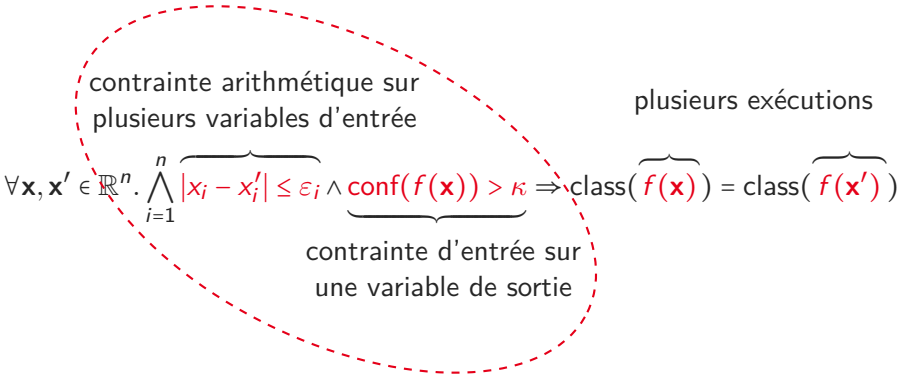
plusieurs exécutions

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \bigwedge_{i=1}^{n} \overbrace{|x_i - x_i'| \leq \varepsilon_i} \wedge \underbrace{\mathsf{conf}(f(\mathbf{x})) > \kappa} \Rightarrow \mathsf{class}(\overbrace{f(\mathbf{x})}) = \mathsf{class}(\overbrace{f(\mathbf{x}')})$$

contrainte d'entrée sur une variable de sortie

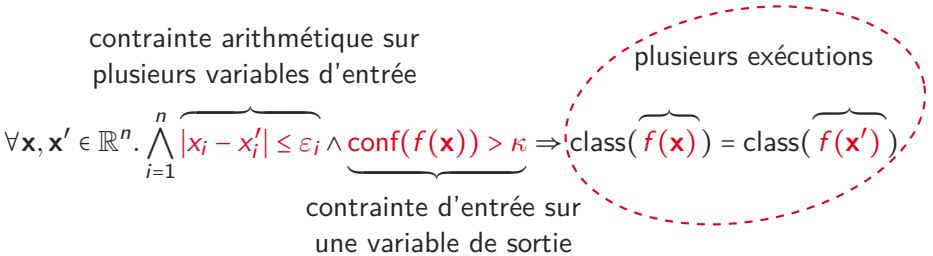
En quoi est-ce un problème ?

Spécifier la robustesse (globale) basée sur la confiance ?



non spécifiable dans les outils (VNN-LIB)

Spécifier la robustesse (globale) basée sur la confiance ?



hyperpropriétés non supportées (VNN-LIB)

Spécifier des hyperpropriétés ?

Hyperpropriété [6] = propriété qui concerne *plusieurs exécutions* du modèle

spécification :

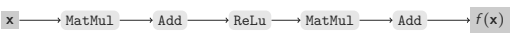
```
1  (declare-const X_0 Real)
2  (declare-const X_1 Real)
3  (declare-const X_2 Real)
4  (declare-const X_3 Real)
5
6  (assert (>= X_0 0.0))
7  (assert (<= X_0 1.0))
8  (assert (>= X_1 0.0))
9  (assert (<= X_1 1.0))
10 (assert (>= X_2 0.0))
11 (assert (<= X_2 1.0))
12 (assert (>= X_3 0.0))
13 (assert (<= X_3 1.0))
14
15 (assert (>= X_0 -0.1))
16 (assert (<= X_0 0.1))
```

x

$f(\mathbf{x})$

```
17 (assert (>= X_1 -0.1))
18 (assert (<= X_1 0.1))
19 (assert (>= X_2 -0.1))
20 (assert (<= X_2 0.1))
21 (assert (>= X_3 -0.1))
22 (assert (<= X_3 0.1))
23
24 (declare-const Y_0 Real)
25 (declare-const Y_1 Real)
26 (declare-const Y_2 Real)
27
28 (assert
29   (or
30     (and (>= Y_0 Y_1))
31     (and (>= Y_2 Y_1))
32     ))
```

modèle :



1 spécification VNN-LIB = 1 seul modèle

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

Spécifier des hyperpropriétés ?

Hyperpropriété [6] = propriété qui concerne *plusieurs exécutions* du modèle

spécification :

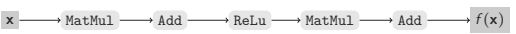
```
1  (declare-const X_0 Real)
2  (declare-const X_1 Real)
3  (declare-const X_2 Real)
4  (declare-const X_3 Real)
5
6  (assert (>= X_0 0.0))
7  (assert (<= X_0 1.0))
8  (assert (>= X_1 0.0))
9  (assert (<= X_1 1.0))
10 (assert (>= X_2 0.0))
11 (assert (<= X_2 1.0))
12 (assert (>= X_3 0.0))
13 (assert (<= X_3 1.0))
14
15 (assert (>= X_0 -0.1))
16 (assert (<= X_0 0.1))
```

x

$f(\mathbf{x})$

```
17 (assert (>= X_1 -0.1))
18 (assert (<= X_1 0.1))
19 (assert (>= X_2 -0.1))
20 (assert (<= X_2 0.1))
21 (assert (>= X_3 -0.1))
22 (assert (<= X_3 0.1))
23
24 (declare-const Y_0 Real)
25 (declare-const Y_1 Real)
26 (declare-const Y_2 Real)
27
28 (assert
29   (or
30     (and (>= Y_0 Y_1))
31     (and (>= Y_2 Y_1))
32     ))
```

modèle :



1 spécification VNN-LIB = 1 seul modèle

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

Reformulation de la propriété (1)

Début de solution : reformuler la propriété

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

Reformulation de la propriété (1)

Début de solution : reformuler la propriété

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

$$\Leftrightarrow$$

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa$$
$$\Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\text{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

avec : $\text{clip}(\mathbf{x} + \boldsymbol{\delta})$ qui ramène $\mathbf{x} + \boldsymbol{\delta}$ dans le « bon intervalle » $(\mathbf{x} + \boldsymbol{\delta} \in \mathbb{R}^n)$

Reformulation de la propriété (1)

Début de solution : reformuler la propriété

$$\forall \mathbf{x}, \mathbf{x'} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

$$\Leftrightarrow$$

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa$$
$$\Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\text{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

avec : $\text{clip}(\mathbf{x} + \boldsymbol{\delta})$ qui ramène $\mathbf{x} + \boldsymbol{\delta}$ dans le « bon intervalle » $(\mathbf{x} + \boldsymbol{\delta} \in \mathbb{R}^n)$

Reformulation de la propriété (1)

Début de solution : reformuler la propriété

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

$$\Leftrightarrow$$

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa$$
$$\Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\text{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

avec : $\text{clip}(\mathbf{x} + \boldsymbol{\delta})$ qui ramène $\mathbf{x} + \boldsymbol{\delta}$ dans le « bon intervalle » $(\mathbf{x} + \boldsymbol{\delta} \in \mathbb{R}^n)$

Reformulation de la propriété (1)

Début de solution : reformuler la propriété

$$\forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n. \bigwedge_{i=1}^{n} |x_i - x_i'| \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa \Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$$

$$\Leftrightarrow$$

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa$$
$$\Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\text{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

avec : $\text{clip}(\mathbf{x} + \boldsymbol{\delta})$ qui ramène $\mathbf{x} + \boldsymbol{\delta}$ dans le « bon intervalle » $(\mathbf{x} + \boldsymbol{\delta} \in \mathbb{R}^n)$

Reformulation de la propriété (2)

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i \wedge \mathsf{conf}(f(\mathbf{x})) > \kappa$$

$$\Rightarrow \mathsf{class}(f(\mathbf{x})) = \mathsf{class}(f(\mathsf{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

Reformulation de la propriété (2)

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i \wedge \text{conf}(f(\mathbf{x})) > \kappa$$
$$\Rightarrow \text{class}(f(\mathbf{x})) = \text{class}(f(\text{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

$$\Leftrightarrow$$

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i$$
$$\Rightarrow \neg(\text{conf}(f(\mathbf{x})) > \kappa) \vee \text{class}(f(\mathbf{x})) = \text{class}(f(\text{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

Reformulation de la propriété (2)

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i \wedge \mathrm{conf}(f(\mathbf{x})) > \kappa$$
$$\Rightarrow \mathrm{class}(f(\mathbf{x})) = \mathrm{class}(f(\mathrm{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

$$\Leftrightarrow$$

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i$$
$$\Rightarrow \neg(\mathrm{conf}(f(\mathbf{x})) > \kappa) \vee \mathrm{class}(f(\mathbf{x})) = \mathrm{class}(f(\mathrm{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

Spécification dans CAISAR (WhyML)

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i$$
$$\Rightarrow \neg(\mathsf{conf}(f(\mathbf{x})) > \kappa) \vee \mathsf{class}(f(\mathbf{x})) = \mathsf{class}(f(\mathsf{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

Spécification dans CAISAR (WhyML)

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i$$
$$\Rightarrow \neg(\text{conf}(f(\mathbf{x})) > \kappa) \vee \text{class}(f(\mathbf{x})) = \text{class}(f(\text{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

WhyML

```
theory ConfidenceMeasure
  let function softmax_confidence (m: model) …
end

theory ClassGlobalRobustConf
  predicate unconfident_prediction (m: model) …
  predicate same_prediction (label_bounds: Label.bounds) …
  predicate confidence_globally_robust (valid_input_bounds: FeatureVector.t …
end
```

Spécification dans CAISAR (WhyML)

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i$$
$$\Rightarrow \neg(\text{conf}(f(\mathbf{x})) > \kappa) \vee \text{class}(f(\mathbf{x})) = \text{class}(f(\text{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

WhyML

```
theory ConfidenceMeasure
  let function softmax_confidence (m: model) …
end

theory ClassGlobalRobustConf
  predicate unconfident_prediction (m: model) …
  predicate same_prediction (label_bounds: Label.bounds) …
  predicate confidence_globally_robust (valid_input_bounds: FeatureVector.t …
end
```

φ

```
goal conf_global_robustness:
  let nn = read_model model_filename in
  let len = 5 in
  let bounds = (fun x -> valid_input_bounds x) in
  let conf_measure = (fun m x -> softmax_confidence m x) in
  confidence_globally_robust bounds
                             label_bounds
                             lbounds
                             ubounds
                             len
                             nn
                             eps
                             threshold
                             conf_measure
```

Spécification dans CAISAR (WhyML)

$$\forall \mathbf{x} \in \mathbb{R}^n, \forall \boldsymbol{\delta} \in \mathbb{R}^n. \bigwedge_{i=1}^{n} -\varepsilon_i \leq \delta_i \leq \varepsilon_i$$
$$\Rightarrow \neg(\text{conf}(f(\mathbf{x})) > \kappa) \vee \text{class}(f(\mathbf{x})) = \text{class}(f(\text{clip}(\mathbf{x} + \boldsymbol{\delta})))$$

WhyML

```
theory ConfidenceMeasure
  let function softmax_confidence (m: model) …
end

theory ClassGlobalRobustConf
  predicate unconfident_prediction (m: model) …
  predicate same_prediction (label_bounds: Label.bounds) …
  predicate confidence_globally_robust (valid_input_bounds: FeatureVector.t …
end
```

φ

```
goal conf_global_robustness:
  let nn = read_model model_filename in
  let len = 5 in
  let bounds = (fun x -> valid_input_bounds x) in
  let conf_measure = (fun m x -> softmax_confidence m x) in
  confidence_globally_robust bounds
                             label_bounds
                             lbounds
                             ubounds
                             len
                             nn
                             eps
                             threshold
                             conf_measure
```

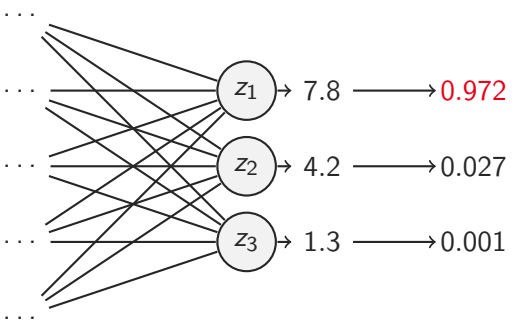
Lever les derniers verrous ? → interprétation

Interprétation de la spécification : confiance ?

$$\text{conf}(f(\mathbf{x})) = \max(\text{softmax}(f(\mathbf{x})))$$

Interprétation de la spécification : confiance ?

$$\text{conf}(f(\mathbf{x})) = \max(\text{softmax}(f(\mathbf{x})))$$



Interprétation de la spécification : confiance

$$\text{conf}(f(\mathbf{x})) = \max(\text{softmax}(f(\mathbf{x})))$$

Interprétation de la spécification : confiance

$$\text{conf}(f(\mathbf{x})) = \max(\text{softmax}(f(\mathbf{x})))$$

Spécification :

$$\neg(\text{conf}(f(\mathbf{x})) > \kappa)$$

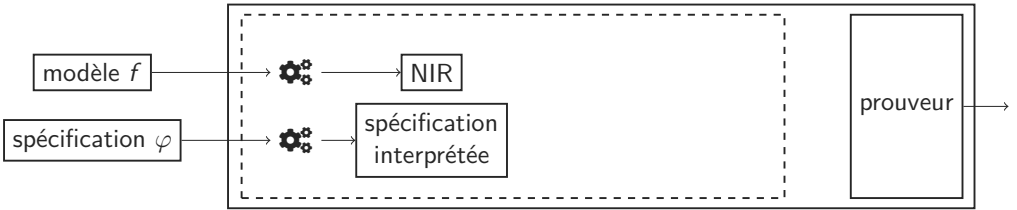
$$\downarrow \text{ spécification (interprétée)}$$

$$\forall i \in \{1, \ldots, m\},\ \text{softmax}(f(\mathbf{x}))[i] \leq \kappa$$

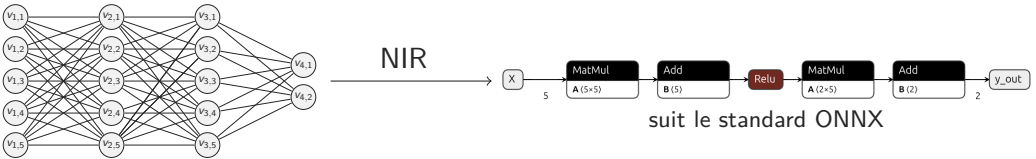
softmax : vector $\longrightarrow$ vector

Interprétation de la spécification : confiance

$$\text{conf}(f(\mathbf{x})) = \max(\text{softmax}(f(\mathbf{x})))$$

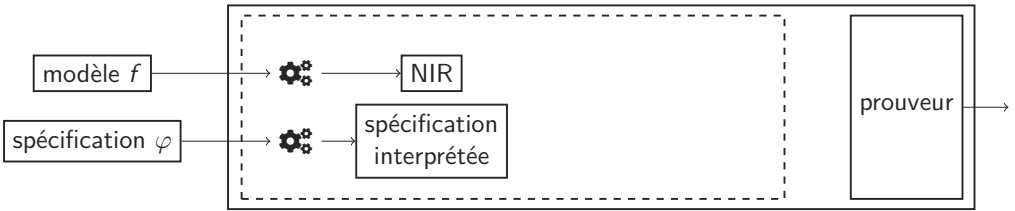


Modèle :



Interprétation de la spécification : confiance

$$\text{conf}(f(\mathbf{x})) = \max(\text{softmax}(f(\mathbf{x})))$$



Spécification :

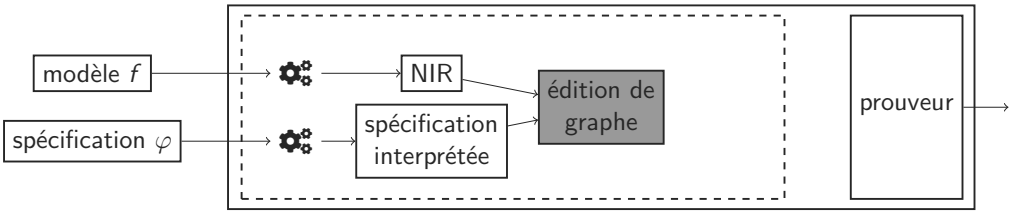
$$\forall i \in \{1, \ldots, m\},\ \text{softmax}(f(\mathbf{x}))[i] \leq \kappa$$ ✗

Modèle :

f :

Interprétation de la spécification : confiance

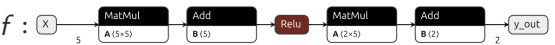
$$\text{conf}(f(\mathbf{x})) = \max(\text{softmax}(f(\mathbf{x})))$$



Spécification :

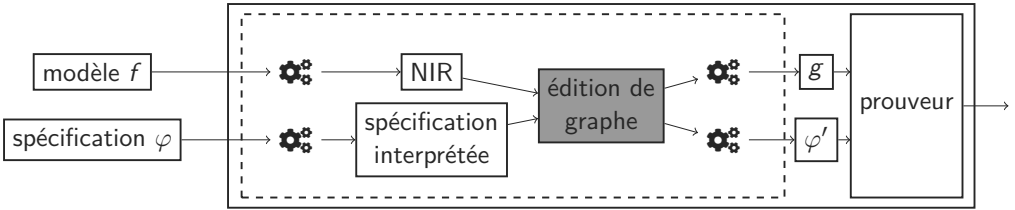
$$\forall i \in \{1, \ldots, m\},\ \text{softmax}(f(\mathbf{x}))[i] \leq \kappa \ \text{✗}$$

Modèle :



Interprétation de la spécification : confiance

$$\text{conf}(f(\mathbf{x})) = \max(\text{softmax}(f(\mathbf{x})))$$



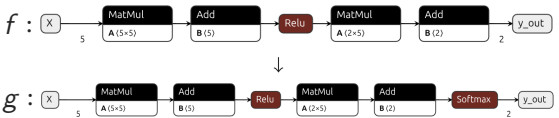
Spécification :

$$\forall i \in \{1, \ldots, m\},\ \text{softmax}(f(\mathbf{x}))[i] \leq \kappa \ \text{✗}$$

$$\downarrow$$

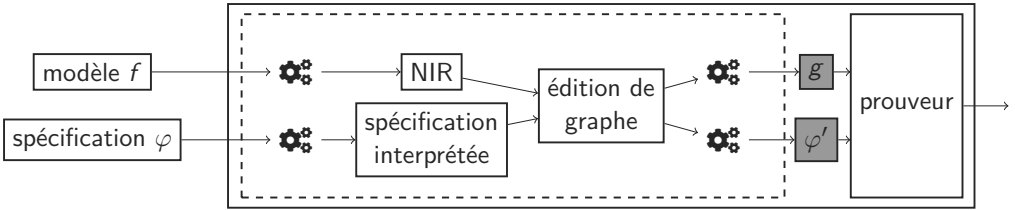
$$\varphi' : \forall i \in \{1, \ldots, m\},\ g(\mathbf{x})[i] \leq \kappa \ \text{✓}$$

Modèle :



Interprétation de la spécification : confiance

$$\text{conf}(f(\mathbf{x})) = \max(\text{softmax}(f(\mathbf{x})))$$



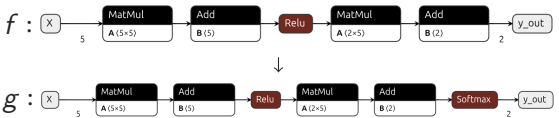
Spécification :

$$\forall i \in \{1, \ldots, m\},\ \text{softmax}(f(\mathbf{x}))[i] \leq \kappa \ \text{✗}$$

$$\downarrow$$

$$\varphi' : \forall i \in \{1, \ldots, m\},\ g(\mathbf{x})[i] \leq \kappa \ \text{✓}$$

Modèle :



Interprétation de la spécification : édition de graphe

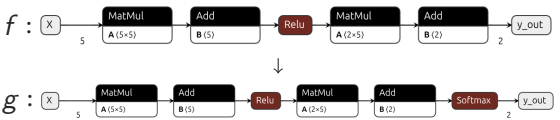
Spécification :

$$\forall i \in \{1, \ldots, m\},\ \text{softmax}(f(\mathbf{x}))[i] \leq \kappa \ \text{✗}$$

$$\downarrow$$

$$\forall i \in \{1, \ldots, m\},\ g(\mathbf{x})[i] \leq \kappa \ \text{✓}$$

Modèle :



Interprétation de la spécification : édition de graphe

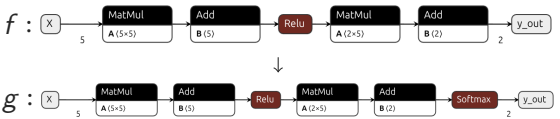
Spécification :

$$\forall i \in \{1, \ldots, m\},\ \mathrm{softmax}(f(\mathbf{x}))[i] \leq \kappa \ \text{✗}$$

$$\downarrow$$

$$\forall i \in \{1, \ldots, m\},\ g(\mathbf{x})[i] \leq \kappa \ \text{✓}$$

Modèle :



Édition de graphe → *réduire* le problème d'expressivité (spécification ↦ modèle)

Interprétation de la spécification : édition de graphe

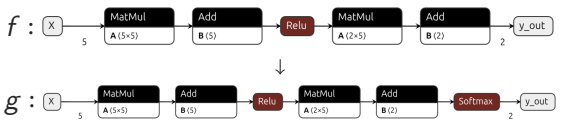
Spécification :

$$\forall i \in \{1, \ldots, m\},\ \text{softmax}(f(\mathbf{x}))[i] \leq \kappa \ \text{✗}$$

$$\downarrow$$

$$\forall i \in \{1, \ldots, m\},\ g(\mathbf{x})[i] \leq \kappa \ \text{✓}$$

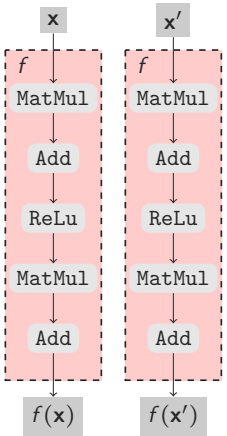
Modèle :



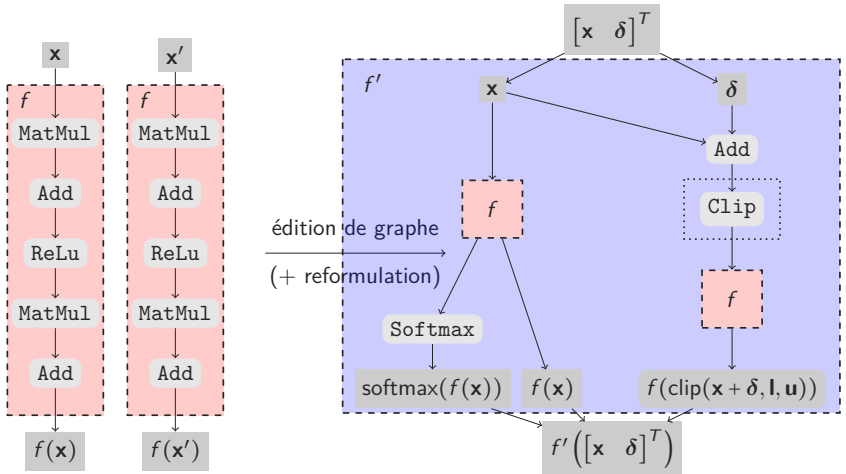
Édition de graphe → *réduire* le problème d'expressivité (spécification ↦ modèle)

Limitation : support des opérateurs ONNX pour un prouveur
→ seulement PyRAT et Maraboupy supportent `Softmax`

Interprétation de la spécification : édition de graphe



Interprétation de la spécification : édition de graphe



Évaluation

Différents cas d'usage testés (dont reproduction des résultats de [4])

Évaluation

Différents cas d'usage testés (dont reproduction des résultats de [4])

Cas d'usage *industriel* : $f : [0,1]^7 \longrightarrow \mathbb{R}^5$ ($\varepsilon = 0.001$)
1630 paramètres (justesse : 98%)

	$\kappa = 0.999$	$\kappa = 0.998$	$\kappa = 0.9$	$\kappa = 0.8$	$\kappa = 0.7$	$\kappa = 0.33$	$\kappa = 0.32$	$\kappa = 0.31$	$\kappa = 0.25$
Maraboupy	🕐	🕐	🕐	🕐	🕐	🕐	🕐	🕐	🕐
PyRAT	✓(38s)	✓(42s)	✓(25min15)	✓(71min6)	🕐	🕐	✗(26s)	✗(26s)	✗(24s)

✓ = "Valide", ✗ = "Invalide", 🕐 = Hors temps (2h)

Évaluation

Différents cas d'usage testés (dont reproduction des résultats de [4])

Cas d'usage *industriel* : $f : [0,1]^7 \longrightarrow \mathbb{R}^5$ ($\varepsilon = 0.001$)
1630 paramètres (justesse : 98%)

	$\kappa = 0.999$	$\kappa = 0.998$	$\kappa = 0.9$	$\kappa = 0.8$	$\kappa = 0.7$	$\kappa = 0.33$	$\kappa = 0.32$	$\kappa = 0.31$	$\kappa = 0.25$
Maraboupy	🕐	🕐	🕐	🕐	🕐	🕐	🕐	🕐	🕐
PyRAT	✓(38s)	✓(42s)	✓(25min15)	✓(71min6)	🕐	🕐	✗(26s)	✗(26s)	✗(24s)

✓ = "Valide", ✗ = "Invalide", 🕐 = Hors temps (2h)

Observations :

- preuve de validité difficile pour petits κ
- contre-exemples trouvés pour petits κ
- passage à l'échelle difficile → $\#\vee$ quadratique dans φ'

Conclusion

Contributions :

- mise en évidence des limitations des outils actuels pour la spécification et vérification de la robustesse basée sur la confiance
- méthodes pour contourner ces limitations
- support pour la spécification et vérification de cette propriété dans CAISAR (Maraboupy et PyRAT)

Conclusion

Contributions :

- mise en évidence des limitations des outils actuels pour la spécification et vérification de la robustesse basée sur la confiance
- méthodes pour contourner ces limitations
- support pour la spécification et vérification de cette propriété dans CAISAR (Maraboupy et PyRAT)

Futurs travaux :

- améliorations pour le passage à l'échelle : encoder l'égalité des classes via `Argmax`
- support de nouvelles propriétés basées sur la confiance (*fairness*, équivalence)

CAISAR

Site web de CAISAR : `https://caisar-platform.com/`
Code open-source : `https://git.frama-c.com/pub/caisar/`

Nous recrutons : `https://caisar-platform.com/positions/`

INSTITUT CARNOT CEA LIST

université PARIS-SACLAY

References I

[1] H. Wu, O. Isac, A. Zeljic, T. Tagomori, M. L. Daggitt, W. Kokke, I. Refaeli, G. Amir, K. Julian, S. Bassan, P. Huang, O. Lahav, M. Wu, M. Zhang, E. Komendantskaya, G. Katz, and C. W. Barrett, "Marabou 2.0 : A versatile formal analyzer of neural networks," in *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part II* (A. Gurfinkel and V. Ganesh, eds.), vol. 14682 of *Lecture Notes in Computer Science*, pp. 249–264, Springer, 2024.

[2] A. Lemesle, J. Lehmann, T. L. Gall, and Z. Chihani, "Verifying neural networks with pyrat," in *Static Analysis - 32nd International Symposium, SAS 2025, Singapore, October 13-14, 2025, Proceedings* (H. Oh and Y. Sui, eds.), vol. 16100 of *Lecture Notes in Computer Science*, pp. 11–33, Springer, 2025.

References II

[3] S. Wang, H. Zhang, K. Xu, X. Lin, S. Jana, C. Hsieh, and J. Z. Kolter, "Beta-crown : Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification," *CoRR*, vol. abs/2103.06624, 2021.

[4] A. Athavale, E. Bartocci, M. Christakis, M. Maffei, D. Nickovic, and G. Weissenbacher, "Verifying global two-safety properties in neural networks with confidence," in *Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part II* (A. Gurfinkel and V. Ganesh, eds.), vol. 14682 of *Lecture Notes in Computer Science*, pp. 329–351, Springer, 2024.

References III

[5] M. Alberti, F. Bobot, J. Girard-Satabin, A. Grastien, A. Varasse, and Z. Chihani, "The CAISAR platform : Extending the reach of machine learning specification and verification," in *Integrated Formal Methods - 20th International Conference, iFM 2025, Paris, France, November 19-21, 2025, Proceedings* (F. Damiani and M. Farrell, eds.), vol. 16194 of *Lecture Notes in Computer Science*, pp. 290–309, Springer, 2025.

[6] M. R. Clarkson and F. B. Schneider, "Hyperproperties," *J. Comput. Secur.*, vol. 18, no. 6, pp. 1157–1210, 2010.

Annexes

Spécification WhyML (1/4)

Encoder $\neg(\text{conf}(f(\mathbf{x})) > \kappa)$

```
1  let function softmax_confidence (m: model)
2                                  (x: FeatureVector.t) =
3    let output = (m @@ x) in
4    softmax output
```

Spécification WhyML (2/4)

```
1  predicate unconfident_prediction (m: model)
2                                   (conf_measure: model
3                                                → FeatureVector.t
4                                                → FeatureVector.t)
5                                   (x: FeatureVector.t)
6                                   (threshold: t)
7                                   (label_bounds: Label.bounds) =
8    let conf = conf_measure m x in
9    forall i: Label.t. Label.valid label_bounds i →
10   conf[i] .≤ threshold
```

Spécification WhyML (3/4)

Encoder $\text{class}(f(\mathbf{x})) = \text{class}(f(\mathbf{x}'))$

```
1   predicate same_prediction (label_bounds: Label.bounds)
2                             (lbounds: α)
3                             (ubounds: α)
4                             (m: model)
5                             (x: FeatureVector.t)
6                             (delta: FeatureVector.t) =
7     forall l: Label.t. Label.valid label_bounds l →
8     let y = clip (x + delta) lbounds ubounds in
9     let pred_x = advises label_bounds m l x in
10    let pred_y = advises label_bounds m l y in
11    (¬ pred_x  ∨  pred_y) ∧ (pred_x  ∨  ¬ pred_y)
```

Spécification WhyML (4/4)

```
1  predicate confidence_globally_robust (valid_input_bounds: FeatureVector.t
2                                        → bool)
3                                    (label_bounds: Label.bounds)
4                                    (lbounds: α)
5                                    (ubounds: α)
6                                    (len: int)
7                                    (m: model)
8                                    (eps: t)
9                                    (threshold: t)
10                                   (conf_measure: model
11                                           → FeatureVector.t
12                                           → FeatureVector.t) =
13   forall x: FeatureVector.t.
14   has_length x len →
15   forall delta: FeatureVector.t.
16   has_length delta len →
17   valid_input_bounds x →
18   bounded_by_epsilon delta eps →
19   unconfident_prediction m conf_measure x threshold label_bounds  ∨
20   same_prediction label_bounds lbounds ubounds m x delta
```

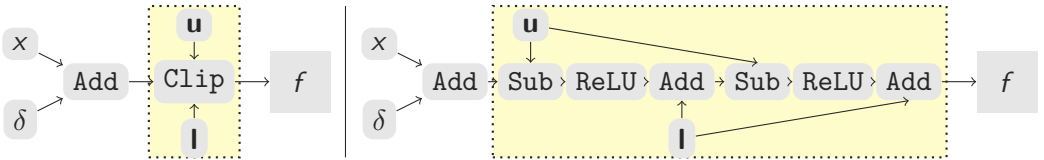
Spécification cas d'usage

```
1  theory GERMAN_CREDIT
2
3    use ieee_float.Float64
4    use caisar.model.Model
5    use caisar.types.IntWithBounds as Label
6    use caisar.types.Vector
7    use caisar.types.VectorFloat64 as FeatureVector
8    use caisar.confidence.ClassGlobalRobustConf
9    use caisar.confidence.ConfidenceMeasure
10
11   constant model_filename: string
12   constant eps: Float64.t
13   constant threshold: Float64.t
14
15   constant label_bounds: Label.bounds = Label.{ lower = 0;
16                                                  upper = 1 }
17
18   type bounds = {
19     b0: Float64.t;
20     b1: Float64.t;
21     b2: Float64.t;
22     b3: Float64.t;
23     b4: Float64.t;
24   }
25   constant lbounds: bounds = {
26     b0 = (4.0:t); b1 = (250.0:t); b2 = (1.0:t); b3 = (1.0:t); b4 = (19.0:t);
27   }
28
29   constant ubounds: bounds = {
30     b0 = (72.0:t); b1 = (18424.0:t); b2 = (4.0:t); b3 = (4.0:t); b4 = (75.0:t);
31   }
32
33   predicate valid_input_bounds (x: FeatureVector.t) =
34     (x[0] .≥ lbounds.b0 ∧ x[0] .≤ ubounds.b0) ∧
35     (x[1] .≥ lbounds.b1 ∧ x[1] .≤ ubounds.b1) ∧
36     (x[2] .≥ lbounds.b2 ∧ x[2] .≤ ubounds.b2) ∧
37     (x[3] .≥ lbounds.b3 ∧ x[3] .≤ ubounds.b3) ∧
38     (x[4] .≥ lbounds.b4 ∧ x[4] .≤ ubounds.b4)
39
40   goal conf_global_robustness:
41     let nn = read_model model_filename in
42     let len = 5 in
43     let bounds = (fun x → valid_input_bounds x) in
44     let conf_measure = (fun m x → softmax_confidence m x) in
45     confidence_globally_robust bounds label_bounds lbounds ubounds len
46                                nn eps threshold conf_measure
47
48 end
```

Édition du `Clip`

$$\text{clip}(\mathbf{x},\mathbf{l},\mathbf{u})_i = \begin{cases} x_i & \text{si } x_i \in [l_i, u_i] \\ l_i & \text{si } x_i < l_i \\ u_i & \text{si } x_i > u_i \end{cases} \quad \rightarrow \quad \text{clip}(\mathbf{x},\mathbf{l},\mathbf{u})_i = \max(l_i, \min(x_i, u_i))$$

$$\text{clip}(\mathbf{x},\mathbf{l},\mathbf{u}) = \text{ReLU}(\mathbf{u} - (\mathbf{l} + \text{ReLU}(\mathbf{u} - \mathbf{x}))) + \mathbf{l}$$

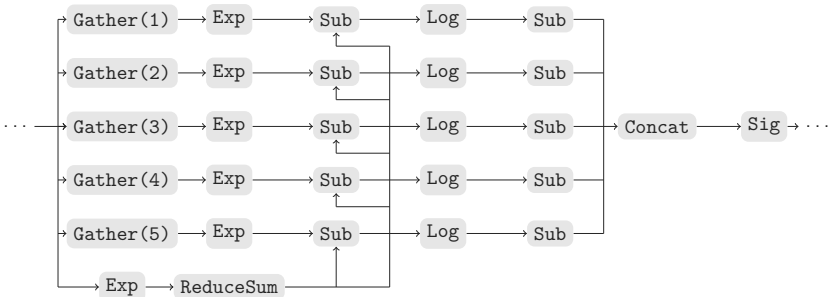


Réécriture softmax

Pour tout $\mathbf{z} \in \mathbb{R}^n$, i$\in \{1, \ldots, n\}$,

$$\text{softmax}(\mathbf{z})_i = \text{Sig}\left(z_i - \log\left(\sum_{j=1, j\neq i}^{n} e^{z_j}\right)\right)$$

où $\text{Sig}(x) = \frac{1}{1+e^{-x}}$.



Support opérateurs (softmax)

Prover	Gather	Concat	Sub	Exp	Log	ReduceSum	Sigmoid
Marabou	✗	✗	✓	✗	✗	✗	✗
Maraboupy	✓	✓	✓	✗	✗	✗	✓
PyRAT	✓	✓	✓	✓	✓	✓	✓
nnenum	✓	✓	✓	✗	✗	✗	✗
abcrown	✗	✓	✓	✗	✗	✗	✗

VNN-LIB (confiance)

```
(declare-const X_0 Real)
(declare-const X_1 Real)
(declare-const X_2 Real)
(declare-const X_3 Real)
(declare-const X_4 Real)
(declare-const X_5 Real)
(declare-const X_6 Real)
(declare-const X_7 Real)
(declare-const X_8 Real)
(declare-const X_9 Real)

(declare-const Y_0 Real)
(declare-const Y_1 Real)
(declare-const Y_2 Real)
(declare-const Y_3 Real)
(declare-const Y_4 Real)
(declare-const Y_5 Real)

(assert (≥ X_0 4.0))
(assert (≤ X_0 72.0))
(assert (≥ X_1 250.0))
(assert (≤ X_1 18424.0))
(assert (≥ X_2 1.0))
(assert (≤ X_2 4.0))
(assert (≥ X_3 1.0))
(assert (≤ X_3 4.0))
(assert (≥ X_4 19.0))
(assert (≤ X_4 75.0))

(assert (≥ X_5 −0.00100000004749745130538940429687 5))
(assert (≤ X_5 0.00100000004749745130538940429687 5))
(assert (≥ X_6 −0.00100000004749745130538940429687 5))
(assert (≤ X_6 0.00100000004749745130538940429687 5))
(assert (≥ X_7 −0.00100000004749745130538940429687 5))
(assert (≤ X_7 0.00100000004749745130538940429687 5))
(assert (≥ X_8 −0.00100000004749745130538940429687 5))
(assert (≤ X_8 0.00100000004749745130538940429687 5))
(assert (≥ X_9 −0.00100000004749745130538940429687 5))
(assert (≤ X_9 0.00100000004749745130538940429687 5))

(assert (or (and (≥ Y_0 0.5))
            (and (≥ Y_1 0.5))
            ))
(assert
  (or
    (and (≤ Y_5 Y_4)
    (≥ Y_3 Y_2))
    (and (≥ Y_5 Y_4)
    (≤ Y_3 Y_2))
    (and (≤ Y_4 Y_5)
    (≥ Y_2 Y_3))
    (and (≥ Y_4 Y_5)
    (≤ Y_2 Y_3))
))
```

Évaluation (Athavale)

COMPAS	$\varepsilon = 0.001$	$\varepsilon = 0.01$	$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.5$	$\varepsilon = 0.6$	$\varepsilon = 0.7$	$\varepsilon = 0.8$	$\varepsilon = 0.9$	$\varepsilon = 1.0$
$\kappa = 0.5$	?/ ✓	?/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓
$\kappa = 0.6$	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓
$\kappa = 0.7$	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	✗/ ✓	✗/ ✓	✗/ ✓
$\kappa = 0.8$	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	✗/ ✓
$\kappa = 0.9$	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓
German Credit	$\varepsilon = 0.001$	$\varepsilon = 0.01$	$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.5$	$\varepsilon = 0.6$	$\varepsilon = 0.7$	$\varepsilon = 0.8$	$\varepsilon = 0.9$	$\varepsilon = 1.0$
$\kappa = 0.5$	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗
$\kappa = 0.6$	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗
$\kappa = 0.7$	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗	✓/ ✗
$\kappa = 0.8$	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓
$\kappa = 0.9$	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓
Law School	$\varepsilon = 0.001$	$\varepsilon = 0.01$	$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.5$	$\varepsilon = 0.6$	$\varepsilon = 0.7$	$\varepsilon = 0.8$	$\varepsilon = 0.9$	$\varepsilon = 1.0$
$\kappa = 0.5$	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓
$\kappa = 0.6$	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓
$\kappa = 0.7$	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓
$\kappa = 0.8$	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓
$\kappa = 0.9$	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓
Adult	$\varepsilon = 0.001$	$\varepsilon = 0.01$	$\varepsilon = 0.1$	$\varepsilon = 0.2$	$\varepsilon = 0.3$	$\varepsilon = 0.4$	$\varepsilon = 0.5$	$\varepsilon = 0.6$	$\varepsilon = 0.7$	$\varepsilon = 0.8$	$\varepsilon = 0.9$	$\varepsilon = 1.0$
$\kappa = 0.5$	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓	✗/ ✓
$\kappa = 0.6$	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓
$\kappa = 0.7$	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓
$\kappa = 0.8$	✓/ ✓	✓/ ✓	✓/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓	?/ ✓
$\kappa = 0.9$	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓	✓/ ✓