# Safety Verification of Neural Network Controlled Systems

A. Claviere[1] E. Asselin[1] C. Garion[2] C. Pagetti[3]

[1]Collins Aerospace, France

[2]ISAE-SUPAERO, France

[3]ONERA, France

Scientific meeting DGA AID, November 3, 2020

# Table of Contents

# Table of Contents

# The safety perspective

## Neural network controlled system

The combination of a continuous-time dynamical system with a discrete-time neural network based controller.

What if such a system is considered as *safety critical*?


One has to show evidence that the system fulfills a set of *safety requirements*.

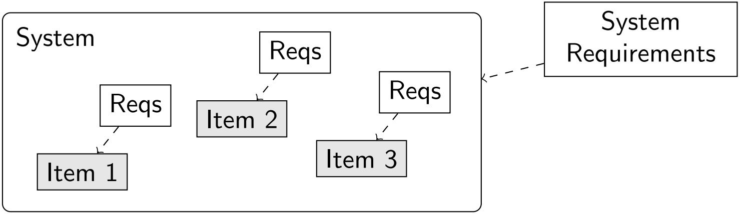
*e.g., "A catastrophic failure shall occur with a probability less than* $10^{-9}$ *per hour of flight."*

# Safety: the classical approach

The system has to be developped in accordance with stringent *standards*.

They involve:

- Refinement of the system requirements at the *item* level
  → each item must be allocated a *correct*, *comprehensive* specification

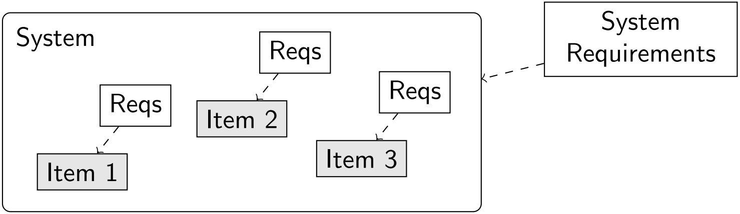


# Safety: the classical approach

The system has to be developped in accordance with stringent *standards*.

They involve:

- Refinement of the system requirements at the *item* level
  $\rightarrow$ each item must be allocated a *correct*, *comprehensive* specification

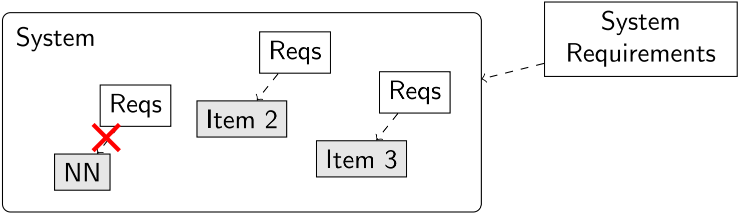



Each item must be developped in compliance with dedicated standards
$\rightarrow$ each item must be shown to fulfill its specification

# Safety: the case of NN controlled system

This approach is not applicable to neural network controlled systems:

- example data = *pointwise*, *non-comprehensive* specification
  → One cannot refine the system requirements at the network level



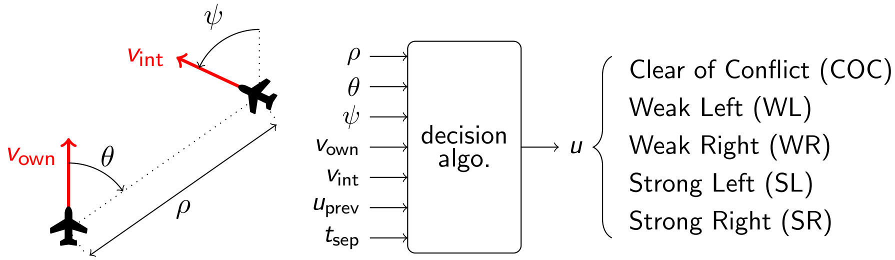

- The learning process does not guarantee the correctness of the network
  → It may be infeasible to show that a network fuflills its specification

# Motivating example: the ACAS Xu controller

Two aircraft:

- the *ownship*, equipped with the ACAS Xu
- the *intruder*, equipped or not with the ACAS Xu

**Objective**: avoid a near mid-air collision between the two aircraft



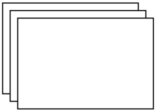
# Motivating example: the ACAS Xu controller

Two aircraft:

- the *ownship*, equipped with the ACAS Xu
- the *intruder*, equipped or not with the ACAS Xu

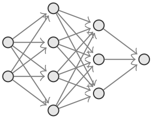
**Objective**: avoid a near mid-air collision between the two aircraft

*Original design: lookup tables*



2GB memory

*Neural network approximation*



2.4**MB memory** ($\times 0.001$)

**relative runtime:** $\times 0.97$

can be run on legacy avionics

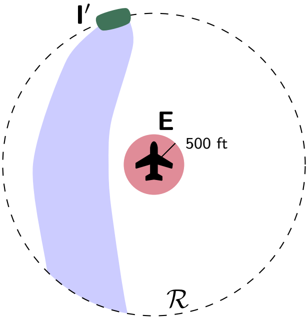
# Motivating example: the ACAS Xu controller

Two aircraft:

- the *ownship*, equipped with the ACAS Xu
- the *intruder*, equipped or not with the ACAS Xu

**Objective**: avoid a near mid-air collision between the two aircraft

How to prove that the neural network based ACAS Xu is safe?



# Safety: the case of NN controlled system

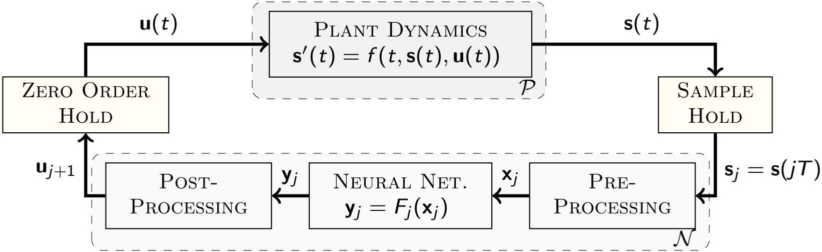
How to deal with these issues?

$\rightarrow$ Demonstrate safety without performing item-level refinement and analyses:

- construct a model of the overall system
- perform a reachability analysis on this model

# Table of Contents

# Closed-loop system




Closed-loop system $\mathcal{C}$:

- a *continuous-time* plant $\mathcal{P}$, with state $\mathbf{s}(t) \in \mathbb{R}^l$
- a *discrete-time* neural network based controller $\mathcal{N}$, with period $T$:
  - its $j^{th}$ execution occurs in $[jT, (j+1)T[$
  - $\mathbf{u}(t) = \mathbf{u}_{j+1} \ \forall t \in [(j+1)T, (j+2)T[$

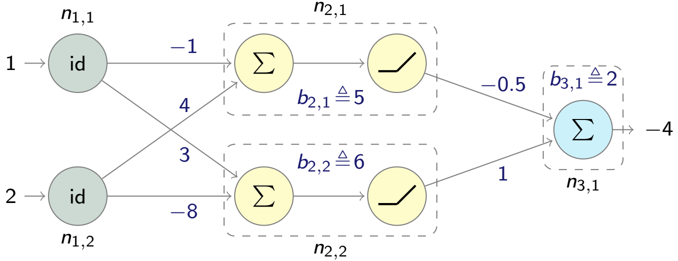
$\mathcal{N}$ is a *classifier i.e.*, $\mathbf{u}_{j+1} \in \mathbf{U} = \left\{\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(P)}\right\}$

# Neural Network based controller

Uses a *collection* of ReLU networks $\mathbf{N} = \{N^{(1)}, \ldots, N^{(D)}\}$
$\rightarrow$ only one network is executed at the $j^{th}$ control step, depending on the previous command: $N_j = \lambda(\mathbf{u}_j)$

## ReLU network

A (deterministic) function $F : \mathbb{R}^m \rightarrow \mathbb{R}^p$ that is the composition of affine transformations and non-linear ReLU units $\sigma : x \mapsto \max(0, x)$

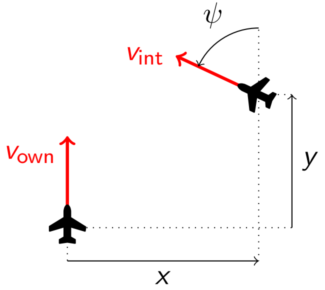


# Plant - Example (ACAS Xu)

$\mathcal{P} = \{\text{ownship}, \text{intruder}\}$ $\quad\quad$ $\mathbf{s}(t) = (x(t)\ y(t)\ \psi(t)\ v_{\text{own}}(t)\ v_{\text{int}}(t))^T$

simplified 2D kinematic model:

- the intruder has a uniform rectilinear displacement
- the ownship has a constant velocity



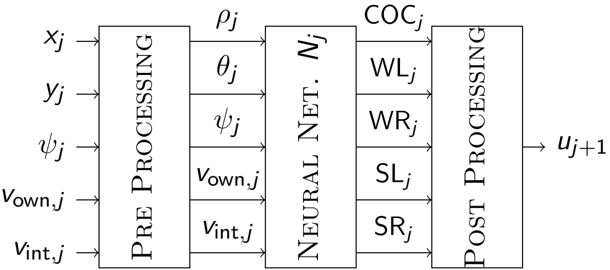

$$\begin{cases} x'(t) = -v_{\text{int}}(t) \cdot \sin(\psi(t)) \\ y'(t) = v_{\text{int}}(t) \cdot \cos(\psi(t)) - v_{\text{own}}(t) \\ \psi'(t) = -u(t) \\ v'_{\text{own}}(t) = 0 \\ v'_{\text{int}}(t) = 0 \end{cases}$$

# Neural Network based controller - Example (ACAS Xu)

Output = turn rate of ownship:

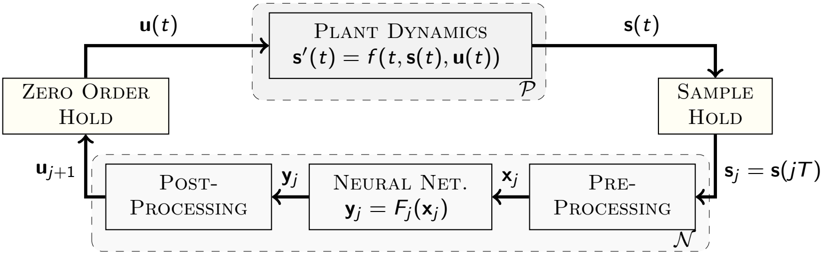
$$\mathbf{U} = \{0\ \text{deg/s}, 1.5\ \text{deg/s}, -1.5\ \text{deg/s}, 3\ \text{deg/s}, -3\ \text{deg/s}\}$$

Uses a collection of 5 ReLU networks $\mathbf{N} = \left\{N^{(1)}, \ldots, N^{(5)}\right\}$




The post-processing is a argmin function

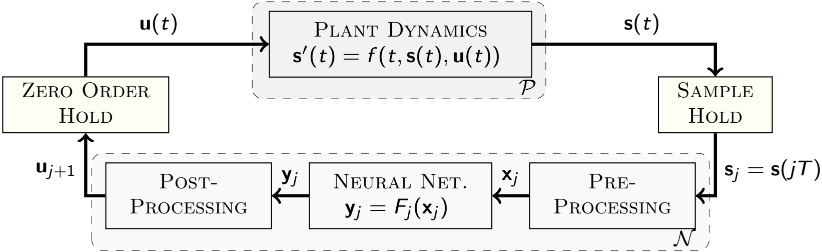
# Closed-loop system




The state of the closed-loop $\mathcal{C}$ is $\phi(t) = (\mathbf{s}(t), \mathbf{u}(t)) \in \mathbb{R}^l \times \mathbf{U}$

- $\mathbf{I} \subseteq \mathbb{R}^l \times \mathbf{U}$ is the set of the possible initial states
- $\mathbf{E} \subset \mathbb{R}^l \times \mathbf{U}$ is a set of *erroneous* states
- $\mathbf{T} \subset \mathbb{R}^l \times \mathbf{U}$ is a set of *target* states ($\mathbf{T} \cap \mathbf{E} = \emptyset$)

# Closed-loop system




The state of the closed-loop $\mathcal{C}$ is $\phi(t) = (\mathbf{s}(t), \mathbf{u}(t)) \in \mathbb{R}^l \times \mathbf{U}$

- $\mathbf{I} \subseteq \mathbb{R}^l \times \mathbf{U}$ is the set of the possible initial states
- $\mathbf{E} \subset \mathbb{R}^l \times \mathbf{U}$ is a set of *erroneous* states
- $\mathbf{T} \subset \mathbb{R}^l \times \mathbf{U}$ is a set of *target* states ($\mathbf{T} \cap \mathbf{E} = \emptyset$)

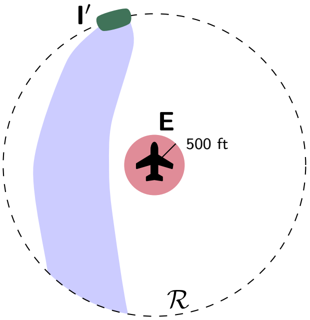
*Deterministic behaviour*: for a given initial state $\phi_0 \in \mathbf{I}$ and time horizon $\tau$, there is a unique function $\phi_{\phi_0}$ such that $\phi_{\phi_0}(t)$ is the state of $\mathcal{C}$ at instant $t \leq \tau$

# Closed-loop system - Example (ACAS Xu)

**I**: the ownship detects the intruder ($u = 0.0$deg/s *i.e.*, COC)

**E**: the intruder lies in the collision cylinder around ownship

**T**: the intruder is out of range



# Table of Contents

# Problem definition

### Definition

The *safety verification problem* $\mathcal{V}$ consists in deciding if:

$$\mathbf{R}_{[0,\tau]} \cap \mathbf{E} = \emptyset \quad (1)$$

where $\mathbf{R}_{[0,\tau]}$ are the reachable states over $[0, \tau]$

- $\mathcal{V}$ is undecidable when the plant $\mathcal{P}$ has a non-linear dynamics
- verifying pre/post-conditions on a network is a NP-hard problem
- the controller has a non-trivial logic

# Problem definition

**Definition**

The *safety verification problem* $\mathcal{V}$ consists in deciding if:

$$\mathbf{R}_{[0,\tau]} \cap \mathbf{E} = \emptyset \tag{1}$$

where $\mathbf{R}_{[0,\tau]}$ are the reachable states over $[0, \tau]$

**Definition**

The *safety verification problem* $\widetilde{\mathcal{V}}$ consists in finding a set $\widetilde{\mathbf{R}}_{[0,\tau]}$ satisfying $\widetilde{\mathbf{R}}_{[0,\tau]} \supset \mathbf{R}_{[0,\tau]}$ and $\widetilde{\mathbf{R}}_{[0,\tau]} \cap \mathbf{E} = \emptyset$

# Table of Contents

# Symbolic state and symbolic set

### Definition

A *symbolic state* is a 2-tuple $([\mathbf{s}], \mathbf{u})$ wherein $[\mathbf{s}] \subset \mathbb{R}^l$ is a $l$-dimensional box *i.e.*, the cartesian product of $l$ intervals, and $\mathbf{u} \in \mathbf{U}$. It symbolically represents the set $\{\phi(t) = (\mathbf{s}(t), \mathbf{u}(t)) \in \mathbb{R}^l \times \mathbf{U} \mid \mathbf{s}(t) \in [\mathbf{s}] \ \wedge \ \mathbf{u}(t) = \mathbf{u}\}$.

### Definition

A *symbolic set* is a collection of symbolic states defined by $\widetilde{\mathbf{\Phi}} = \{([\mathbf{s}]_k, \mathbf{u}_k)\}_{1 \leq k \leq K}$ wherein $K \in \mathbb{N}$. It corresponds to the union of the sets represented by each $([\mathbf{s}]_k, \mathbf{u}_k)$.

# Procedure

The procedure involves two types of sets:

- The symbolic set $\widetilde{\mathbf{R}}_j$ approximates the reachable states at $t = jT$
- The symbolic set $\widetilde{\mathbf{R}}_{[j[}$ approximates the reachable states for $t \in [jT, (j+1)T[$

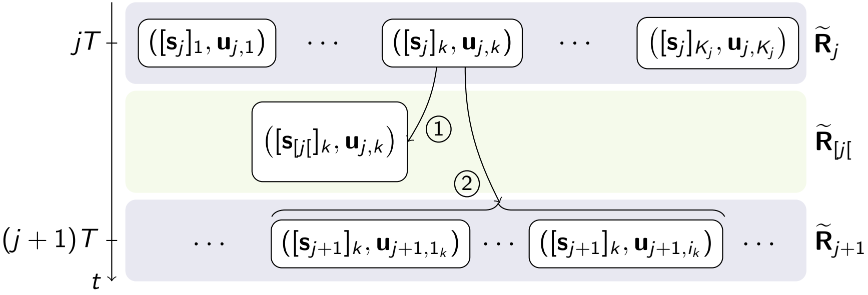
It works iteratively:

- starts with the symbolic set $\widetilde{\mathbf{R}}_0 \supset \mathbf{I}$ enclosing the possible initial states.
- at control step $j$, builds $\widetilde{\mathbf{R}}_{[j[}$ and $\widetilde{\mathbf{R}}_{j+1}$ based on $\widetilde{\mathbf{R}}_j$.

Finally, $\widetilde{\mathbf{R}}_{[0,\tau]}$ is taken as the union of the $\widetilde{\mathbf{R}}_{[j[}$.

# Procedure

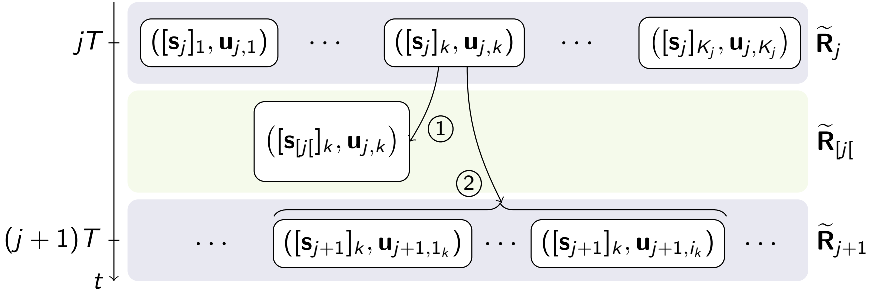
Approximation of the $j^{th}$ control step:




- 1 involves validated simulation (DynIBEX)
- 2 involves both validated simulation (DynIBEX) and abstract interpretation (with a specialized solver ReluVal)

# Procedure

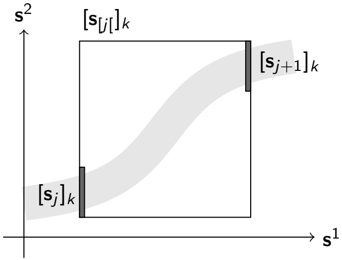
Approximation of the $j^{th}$ control step:

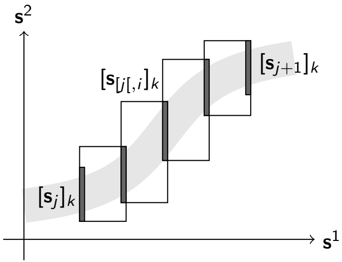



To take account of a potential termination of $\mathcal{C}$ ($\phi \in \mathbf{T}$), the symbolic states $([\mathbf{s}_j]_k, \mathbf{u}_{j,k}) \subset \mathbf{T}$ are not further propagated

# Optimizations

- Improving precision



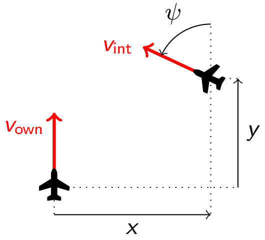

- Improving time complexity

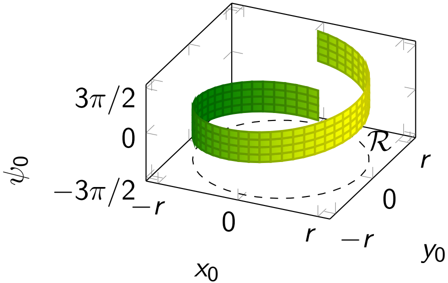
  Avoid an exponential blow up of the number of symbolic states in $\widetilde{\mathbf{R}}_j$
  $\rightarrow$ merge the *"closest"* symbolic states

# Table of Contents

# Experimental setup

**Partitioning**: $\widetilde{\mathbf{R}}_0 = \{([\mathbf{s}_0]_k, 0.0 \text{ deg/s})\}_{1 \le k \le K_0}$ with $K_0 = 198,764$

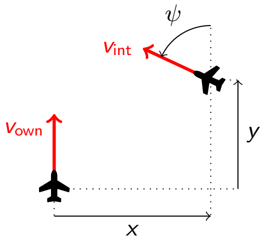


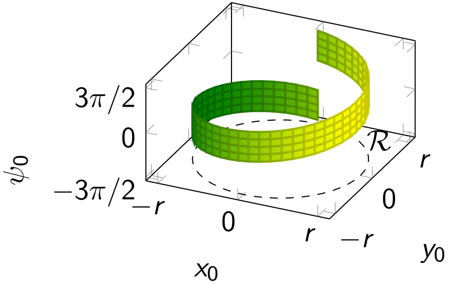




- a single initial symbolic state $([\mathbf{s}_0], 0.0 \text{ deg/s})$ approximating $\mathbf{I}$ is unsafe
- the $K_0$ initial symbolic states composing $\widetilde{\mathbf{R}}_0$ can be seen as $K_0$ *independent* verification problems (parallelization)
- the smaller the box $[\mathbf{s}_0]_k$, the more precise the reachability analysis

# Experimental setup

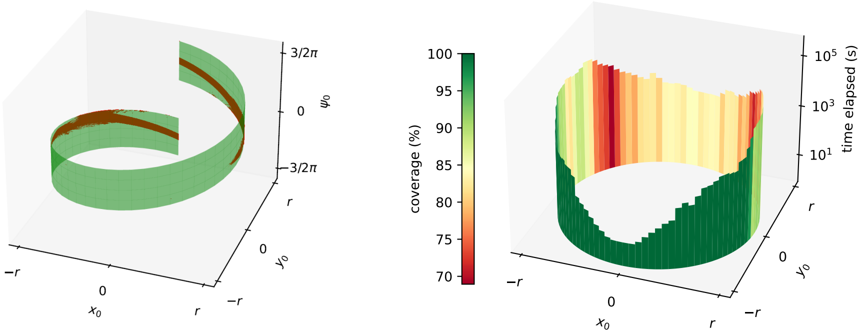
**Partitioning**: $\widetilde{\mathbf{R}}_0 = \{([\mathbf{s}_0]_k, 0.0 \text{ deg/s})\}_{1 \leq k \leq K_0}$ with $K_0 = 198,764$



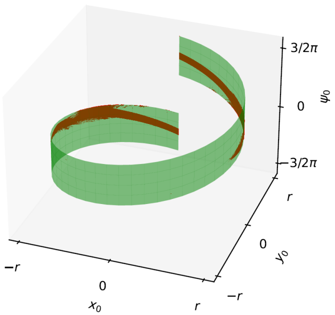

**Split refinement**: the initial symbolic states $([\mathbf{s}_0]_k, 0.0 \text{ deg/s})$ for which the system cannot be proved safe are bisected $\rightarrow$ new reachability analysis

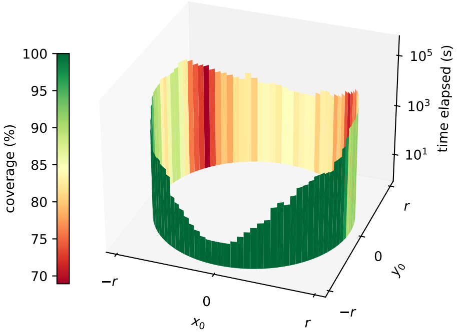
# Results




- coverage of 90.3%
- capability to identify the initial states for which the system could not be proved safe

# Results




- most critical situations: the intruder is approaching from the left
- symmetry *w.r.t.* the $x_0 = 0$ axis: captures the symmetry of the collision avoidance problem

# Table of Contents

# Conclusion

- system-level approach for verifying the safety of neural network controlled systems
  $\rightarrow$ realistic model together with reachability analysis
- applicable to real-world systems
- provide valuable information from a practical point of view

# Future work

- more efficient partitioning strategy (*e.g.*, CFD)
- more efficient heuristics for splitting the initial symbolic states
- combine the approach with an efficient falsification strategy
- ACAS Xu: consider multiple UAVs, each one being equipped with a collision avoidance controller