Formal Verification of Neuro-symbolic Multi-agent Systems

Panagiotis Kouvaros

Verification of Autonomous Systems Group
Department of Computing, Imperial College London, UK

August 23, 2023

AI in the Last Decade

- Rapid progress in diverse domains (natural language processing, computer vision, autonomous vehicles)
- Mainly driven by deep learning (increased computational power, big data, improved statistical methods)
- Vast potential in revolutionising society
- Lack of interpretability and fragility to input perturbations hinders adoption in safety-critical applications
- Need for verification and neuro-symbolic approaches towards more transparent and trustworthy AI

Formal Verification of AI

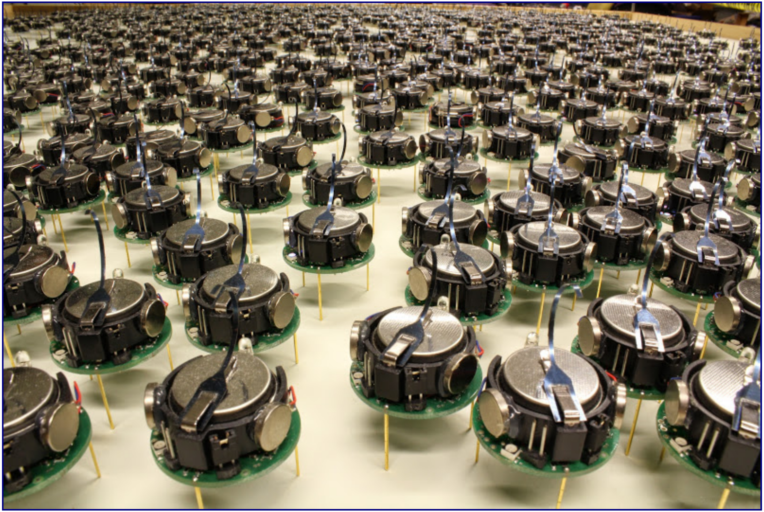
- Formal verification problem

$$\mathcal{S} \models \varphi \ ?$$

- $\mathcal{S}$ is a multi-agent system comprising **symbolic**, **neural** or **neuro-symbolic** agents

- φ is a specification expressing **temporal-epistemic**, **strategic** or **robustness** properties of agents

Verification of Unbounded Multi-agent Systems



Unbounded Multi-agent Systems

- Multi-agent systems composed of an **arbitrary number** of homogeneous agents
- As in multi-party negotiation protocols and auctions, voting protocols, swarm robotics
- Traditional techniques target verification for systems composed of a *known* number of agents.
- Need for methods for establishing properties of protocols **irrespective of the number of agents** in the system

Parameterised Verification

- **Parameterised Interleaved Interpreted Systems**: a *parameterised* semantical structure, where the parameter is the number of agents in the system

- **IACTLK**: an *indexed* version of the temporal-epistemic logic ACTLK that allows for quantifying over the agents
  - *Connectedness* property: $\forall i{:} AGAF$ connected(i) ("every robot is infinitely often close to another robot")

- **Parameterised model checking problem:**
  Input: A PIIS $\mathcal{S}$ and an IACTLK formula φ
  Output:
  - *Yes* if $\forall n \in \mathbb{N} : \mathcal{S}(n) \models \varphi$.
  - *No* otherwise

- The parameterised model checking problem for PIIS is in general **undecidable**

Cutoffs

- A natural number $c \in \mathbb{N}$ is said to be a **cutoff** for a PIIS $\mathcal{S}$ and a formula φ if

$$\forall n \in [1, c] : \mathcal{S}(c) \models \varphi \;\; \Leftrightarrow \;\; \forall n \geq 1 : \mathcal{S}(n) \models \varphi$$

- If a cutoff exists, then the PMCP can be solved by checking all the concrete systems up to the cutoff

- The existence and computability of cutoffs depends on the synchronisation primitives endowing the agents

Synchronisations and Cutoffs in PIIS

- **Fully synchronous** semantics always admit cutoffs (identification procedure in exponential time)

- **Asynchronous** semantics with **broadcast** communication primitives always admit cutoffs (identification procedure in linear time)

- **Asynchronous** semantics with **broadcast** and **agent-environment** communication primitives admit cutoffs under the condition that the environment can never block pairwise synchronisations for the system of one agent only (identification procedure in exponential time)

- Same condition holds for **multi-role systems** with pairwise communication between agents of different roles

Fault-tolerance

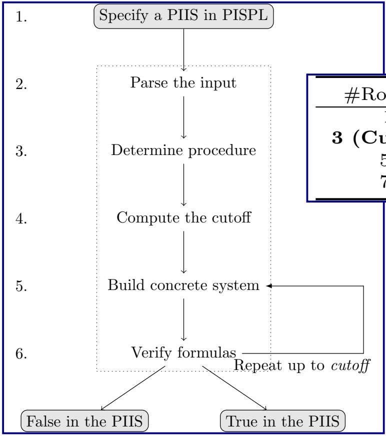
- Adverse functioning behaviour of some of the agents in the system
  - Could a faulty robot break a flying formation?

- Combined with fault-injection techniques parameterised verification can be used to establish the robustness of a UMAS to faults

- Additionally, adaptations of the labelling algorithm for CTLK can be used to compute the **maximum ratio of faulty to non-faulty agents** that the system can tolerate with respect to a specification

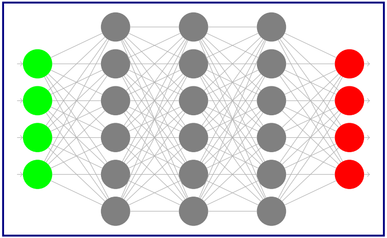
MCMAS-P

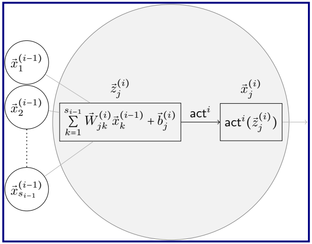
- Traditional verification can verify the Alpha swarm aggregation algorithm only up to 7 robots, whereas `MCMMAS-P` establishes the correctness of the protocol for any number of agents.



#Robots	#States	Time (s)	Memory (KiB)
1	423	1	12016112
3 (Cutoff)	**177243**	**15**	**29925984**
5	2.76×10^{34}	197	50101616
7	TIMEOUT	TIMEOUT	TIMEOUT

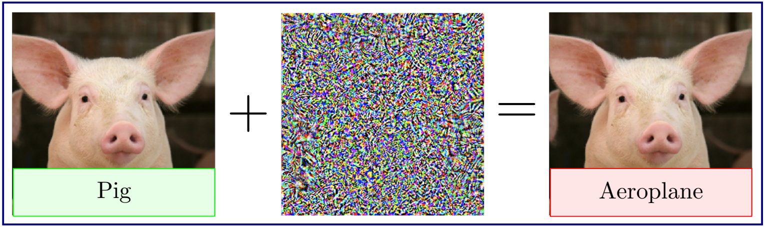
Verification of Neural Networks





Fragility of Neural Networks

- Imperceptible perturbations to the inputs often cause networks to miss-classify



Formal Verification Problem

- Aims to establish whether neural networks behave as intended in regions of the input space:

$$\forall \boldsymbol{x} \in \mathcal{X} \colon \boldsymbol{f}(\boldsymbol{x}) \in \mathcal{Y},$$

  where $\boldsymbol{f}$ is a network, $\mathcal{X}$ is a set of inputs and $\mathcal{Y}$ is a set of outputs of the network

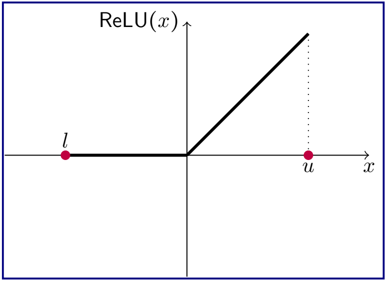
- **Reachability** and **local robustness** properties are instantiated from from above

- Robustness against certain input perturbations denoting $\mathcal{X}$ is particularly important in assuring safe behaviour in the operation domain

ReLU Networks

$$y(x) = \max(0, x)$$

- ReLU is **piecewise-linear**
- When $x \geq 0$ the node is said to be in the **active state** and outputs x
- When $x < 0$ the node is said to be in the **inactive state** and outputs 0



MILP Compilation

- The verification problem is recast into a MILP program

$$
\begin{aligned}
\min_{\boldsymbol{x}^{(0)},\dots,\boldsymbol{x}^{(L)}} \quad & \boldsymbol{c}^T \boldsymbol{x}^{(L)} + \boldsymbol{c}_0 \\
\text{subject to} \quad & \boldsymbol{x}^{(0)} \in \mathcal{X}, \text{ for a set of perturbations } \mathcal{X} \\
& \boldsymbol{z}^{(i)} = \boldsymbol{W}^{(i)} \boldsymbol{x}^{(i-1)} + \boldsymbol{b}^{(i)}, \\
& \boldsymbol{x}_j^{(i)} \geq \boldsymbol{z}_j^{(i)},\ \boldsymbol{x}_j^{(i)} \leq \boldsymbol{z}_j^{(i)} - \boldsymbol{l}_i^{(j)} \cdot (1 - \boldsymbol{\delta}_j^{(i)}), \\
& \boldsymbol{x}_j^{(i)} \leq \boldsymbol{u}_i^{(j)} \cdot \boldsymbol{\delta}_j^{(i)},\ \boldsymbol{x}_j^{(i)} \geq 0,\ \boldsymbol{\delta}_j^{(i)} \in \{0, 1\}
\end{aligned}
$$

- The verification problem is satisfied iff its MILP encoding has no feasible solution

- Any feasible solution corresponds to an input violating the robustness of the network

- Perfomance greatly depends on the tightness of the pre-activation bounds and the number of binary variables encoding the ReLU non-linerarities

Improved Scalability

- **Symbolic interval propagation** enables fast, gpu-accelerated symbolic derivation of tights bounds through linear relaxations of the ReLU nodes

    - The bounds can be further tightned by acounting for intra-layer dependencies, instead of simply relying on local over- approximation areas, when choosing a relaxation

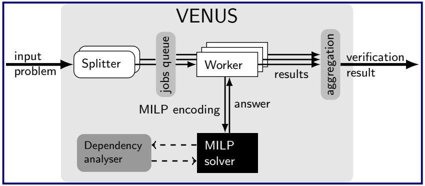
- **Dependency analysis** exploits relations whereby the operational states of the ReLUs for a set of inputs is connected by logical implication

    - Using MILP cuts

    - Using branching heuristics

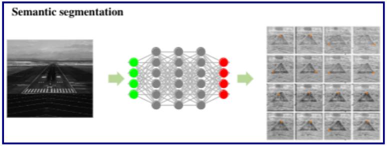
Venus

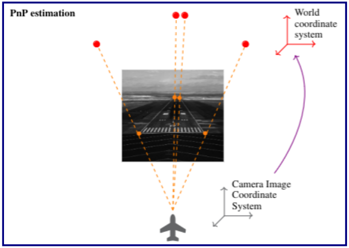
- Open-source neural network verification tool
- Extensive support for various types of layers and architectures present in applications
- Advances in symbolic bound propagation, dependency analysis and input domain splitting
- From checking networks of thoundands of nodes in 2019 it has progressed to the verification of networks of millions of nodes in 2023



Runway Detection

- **Challenge:** Key point prediction robustness: Is the key point prediction robust against input perturbations?
- **Models:** U-Nets with approximately 2 million non-linear nodes
- **Input perturbations**: white noise in (segments of) the input and photometric changes, including brightness and contrast changes



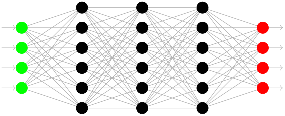


Runway Detection — Analysis Summary

- The models are robust with respect to minor perturbations but still **brittle to realistic alterations of the input**
- The models are more susceptible to some variabilities (brightness) than others (contrast)

Original Counterexample

Verification of Neuro-symbolic Multi-agent Systems



$\rightarrow$

```
import jason.*;
import ...;
public class robotEnv extends Environment {
  ...
  public robotEnv() {
    Literal gp =
           Literal.parseLiteral("green_patch(souffle)");
    addPercept(gp);
  }

  public boolean executeAction(String ag, Structure action) {
    if (action.equals(...)) {
      addPercept(ag,
           Literal.parseLiteral("location(souffle,c(3,4))");
    }
    ...
    return true;
} }
```

Neuro-symbolic Multi-agent Systems

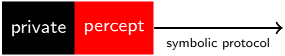
- Agents are equipped with a neural perception unit and a symbolic controller

- A **neuro-symbolic agent** is a tuple $(L_a, obs_a, Act_a, \mathrm{prot}_a, tr_a)$, where
  - $L_a = Prv_a \times Per_a$ is a *set of local states* comprising tuples of *private states* and *percepts*
  - $obs_a\colon L_a \times L_E \to Per_a$ is an *observation function* implemented by a neural network
  - Act_a is a nonempty finite *set of actions*
  - $\mathrm{prot}_a\colon L_a \to 2^{Act_a} \setminus \varnothing$ is a *local protocol function*
  - $tr_a\colon L_a \times Act_1 \times \cdots \times Act_K \times Act_E \to Prv_a$ is a *local transition function*

private	percept

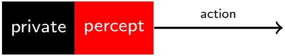
Neuro-symbolic Multi-agent Systems

- Agents are equipped with a neural perception unit and a symbolic controller
- A **neuro-symbolic agent** is a tuple $(L_a, obs_a, Act_a, \mathrm{prot}_a, tr_a)$, where
  - $L_a = Prv_a \times Per_a$ is a *set of local states* comprising tuples of *private states* and *percepts*
  - $obs_a\colon L_a \times L_E \to Per_a$ is an *observation function* implemented by a neural network
  - Act_a is a nonempty finite *set of actions*
  - $\mathrm{prot}_a\colon L_a \to 2^{Act_a} \setminus \varnothing$ is a *local protocol function*
  - $tr_a\colon L_a \times Act_1 \times \cdots \times Act_K \times Act_E \to Prv_a$ is a *local transition function*



Neuro-symbolic Multi-agent Systems

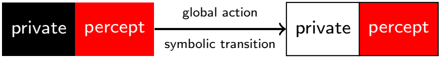
- Agents are equipped with a neural perception unit and a symbolic controller
- A **neuro-symbolic agent** is a tuple $(L_a, obs_a, Act_a, \mathrm{prot}_a, tr_a)$, where
  - $L_a = Prv_a \times Per_a$ is a *set of local states* comprising tuples of *private states* and *percepts*
  - $obs_a \colon L_a \times L_E \to Per_a$ is an *observation function* implemented by a neural network
  - Act_a is a nonempty finite *set of actions*
  - $\mathrm{prot}_a \colon L_a \to 2^{Act_a} \setminus \varnothing$ is a *local protocol function*
  - $tr_a \colon L_a \times Act_1 \times \cdots \times Act_K \times Act_E \to Prv_a$ is a *local transition function*



Neuro-symbolic Multi-agent Systems

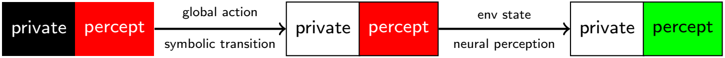
- Agents are equipped with a neural perception unit and a symbolic controller

- A **neuro-symbolic agent** is a tuple $(L_a, obs_a, Act_a, \mathrm{prot}_a, tr_a)$, where
  - $L_a = Prv_a \times Per_a$ is a *set of local states* comprising tuples of *private states* and *percepts*
  - $obs_a \colon L_a \times L_E \to Per_a$ is an *observation function* implemented by a neural network
  - Act_a is a nonempty finite *set of actions*
  - $\mathrm{prot}_a \colon L_a \to 2^{Act_a} \setminus \varnothing$ is a *local protocol function*
  - $tr_a \colon L_a \times Act_1 \times \cdots \times Act_K \times Act_E \to Prv_a$ is a *local transition function*



Neuro-symbolic Multi-agent Systems

- Agents are equipped with a neural perception unit and a symbolic controller
- A **neuro-symbolic agent** is a tuple $(L_a, obs_a, Act_a, \text{prot}_a, tr_a)$, where
  - $L_a = Prv_a \times Per_a$ is a *set of local states* comprising tuples of *private states* and *percepts*
  - $obs_a \colon L_a \times L_E \to Per_a$ is an *observation function* implemented by a neural network
  - Act_a is a nonempty finite *set of actions*
  - $\text{prot}_a \colon L_a \to 2^{Act_a} \setminus \varnothing$ is a *local protocol function*
  - $tr_a \colon L_a \times Act_1 \times \cdots \times Act_K \times Act_E \to Prv_a$ is a *local transition function*



Neuro-symbolic Multi-agent Systems

- Agents are equipped with a neural perception unit and a symbolic controller
- A **neuro-symbolic agent** is a tuple $(L_a, obs_a, Act_a, \mathrm{prot}_a, tr_a)$, where
  - $L_a = Prv_a \times Per_a$ is a *set of local states* comprising tuples of *private states* and *percepts*
  - $obs_a \colon L_a \times L_E \to Per_a$ is an *observation function* implemented by a neural network
  - Act_a is a nonempty finite *set of actions*
  - $\mathrm{prot}_a \colon L_a \to 2^{Act_a} \setminus \varnothing$ is a *local protocol function*
  - $tr_a \colon L_a \times Act_1 \times \cdots \times Act_K \times Act_E \to Prv_a$ is a *local transition function*
- **Environments** are similarly defined (but no percepts)
- **Neuro-symbolic MAS** compose neuro-symbolic agents with the environment for a set I of initial states

Formal Verification of NMAS

- **Verification problem**: Given a NMAS $\mathcal{S}$ and a logic formula φ, check whether every initial state in $\mathcal{S}$ satisfies φ
- Verification of NMAS against unbounded reachability properties (i.e., the system eventually reaches an unsafe state) is **undecidable**
- Bounded fragment of **ATL*** (alternating-time temporal logic):

$$\begin{array}{rcl} \varphi & ::= & \alpha \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle\langle \Gamma \rangle\rangle X^k \varphi, \mid [[\Gamma]]^k \varphi \\ \alpha & ::= & c_1(1) + \cdots + c_m(m)\, op\, c \end{array}$$

where α is a linear inequality over the outputs of the network and Γ is a subset of agents.

- $\langle\langle ag \rangle\rangle X^k(\text{safe})$: "agent *ag* has a strategy to enter a safe configuration after k steps irrespective of what the intruder does."

MILP Encodings

- The verification problem is recast to a **MILP feasibility problem**
- **Monolithic** encoding
  - $\mathcal{S} \models \varphi$ iff $\pi_{\mathcal{S},\neg\varphi\wedge\varphi_I}$ is infeasible
  - Single (exponentially large) program $\pi_{\mathcal{S},\neg\varphi\wedge\varphi_I}$

$$
\begin{aligned}
\pi_{\mathcal{S},l}(\boldsymbol{x}) &= \mathcal{V}(l,\boldsymbol{x})\\
\pi_{\mathcal{S},\varphi_1\vee\varphi_2}(\boldsymbol{x}) &= (\delta=1)\Rightarrow \pi_{\mathcal{S},\varphi_1}(\boldsymbol{x}) \,\cup\, (\delta=0)\Rightarrow \pi_{\mathcal{S},\varphi_2}(\boldsymbol{x})\\
\pi_{\mathcal{S},\varphi_1\wedge\varphi_2}(\boldsymbol{x}) &= \pi_{\mathcal{S},\varphi_1}(\boldsymbol{x}) \cup \pi_{\mathcal{S},\varphi_2}(\boldsymbol{x})\\
\pi_{\mathcal{S},\langle\!\langle\Gamma\rangle\!\rangle X\varphi}(\boldsymbol{x}) &= \bigcup_{i=1}^{|Ind_\Gamma|}(\delta_i=1)\Rightarrow\left(\bigcup_{j=1}^{|Ind_{\overline{\Gamma}}|} C_{\Sigma^i_\Gamma\cup\Sigma^j_{\overline{\Gamma}}}(\boldsymbol{x},\boldsymbol{y}_j)\right) \cup \sum_{i=1}^{|Ind_\Gamma|}\delta_i=1 \,\cup\, \bigcup_{j=1}^{|Ind_{\overline{\Gamma}}|}\pi_{\mathcal{S},\varphi}(\boldsymbol{y}_j)\\
\pi_{\mathcal{S},[\![\Gamma]\!]X\varphi}(\boldsymbol{x}) &= \bigcup_{i=1}^{|Ind_\Gamma|}\left(\bigcup_{j=1}^{|Ind_{\overline{\Gamma}}|}(\delta_{ij}=1)\Rightarrow C_{\Sigma^i_\Gamma\cup\Sigma^j_{\overline{\Gamma}}}(\boldsymbol{x},\boldsymbol{y}_i) \,\cup\, \sum_{j=1}^{|Ind_{\overline{\Gamma}}|}\delta_{ij}=1 \,\cup\, \pi_{\mathcal{S},\varphi}(\boldsymbol{y}_i)\right)
\end{aligned}
$$

where the binary variables $\delta, \delta_i, \delta_{ij}$, the state variables $\boldsymbol{y_j}, \boldsymbol{y}_i$ and all auxiliary variables in $C_{\Sigma^i_\Gamma\cup\Sigma^j_{\overline{\Gamma}}}$ are fresh

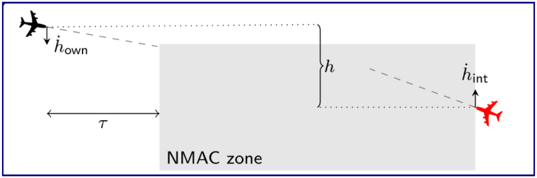
- **Compositional** encoding
  - $\mathcal{S} \models \varphi$ iff every program in $\Pi_{\mathcal{S},\neg\varphi\wedge\varphi_I}$ is infeasible
  - Set of (smaller) program with parallelisable infeasibility checks

$$
\begin{aligned}
\Pi_{\mathcal{S},l}(\boldsymbol{x}) &= \{[\mathcal{V}(l,\boldsymbol{x})]\},\\
\Pi_{\mathcal{S},\varphi_1\vee\varphi_2}(\boldsymbol{x}) &= \Pi_{\mathcal{S},\varphi_1}(\boldsymbol{x}) \cup \Pi_{\mathcal{S},\varphi_2}(\boldsymbol{x}),\\
\Pi_{\mathcal{S},\varphi_1\wedge\varphi_2}(\boldsymbol{x}) &= \Pi_{\mathcal{S},\varphi_1}(\boldsymbol{x}) \times \Pi_{\mathcal{S},\varphi_2}(\boldsymbol{x}),
\end{aligned}
\qquad
\begin{aligned}
\Pi_{\mathcal{S},\langle\!\langle\Gamma\rangle\!\rangle X\varphi}(\boldsymbol{x}) &= \bigcup_{i=1}^{|Ind_\Gamma|}\left(\bigtimes_{j=1}^{|Ind_{\overline{\Gamma}}|}\left\{\left[C_{\Sigma^i_\Gamma\cup\Sigma^j_{\overline{\Gamma}}}(\boldsymbol{x},\boldsymbol{y}_j)\right]\right\}\times\Pi_{\mathcal{S},\varphi}(\boldsymbol{y}_j)\right),\\
\Pi_{\mathcal{S},[\![\Gamma]\!]X\varphi}(\boldsymbol{x}) &= \bigtimes_{i=1}^{|Ind_\Gamma|}\left(\bigcup_{j=1}^{|Ind_{\overline{\Gamma}}|}\left\{\left[C_{\Sigma^i_\Gamma\cup\Sigma^j_{\overline{\Gamma}}}(\boldsymbol{x},\boldsymbol{y}_i)\right]\right\}\right)\times\Pi_{\mathcal{S},\varphi}(\boldsymbol{y}_i).
\end{aligned}
$$

Two-agent Aircraft Collision Avoidance

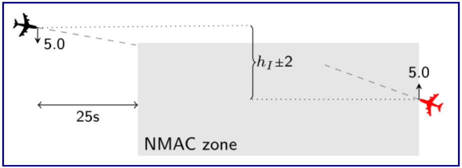
- Two agents, **ownship** and **intruder**, each equipped with a **collision avoidance system** producing vertical climbrate **advisories** to the pilot (perception unit).
- Given an advisory, the pilot chooses the **acceleration** from an appropriate interval (action selection unit).
- The goal is to avoid a **near mid-air collision** (NMAC): when the ownship and intruder are separated by less than 100ft vertically and 500ft horizontally.

h: Vertical separation (ft).
$\dot{h}_{\text{own}}$: Ownship vertical climbrate (ft/s).
$\dot{h}_{\text{int}}$: Intruder vertical climbrate (ft/s).
τ: Time to loss of horizontal separation (s).
ad_{own}: Previous advisory issued to the ownship.
ad_{int}: Previous advisory issued to the intruder.



Two-agent Aircraft Collision Avoidance

- $\varphi = \langle\langle ag \rangle\rangle X^k(\text{safe})$ (*the ownship has a strategy to enter a safe configuration after k time steps*)
- $I = [h_I - 2, h_I + 2] \times \{-5.0\} \times \{5.0\} \times \{25\} \times \{\text{COC}\} \times \{\text{COC}\}$



	Parallel			Monolithic		
h_I	95	100	105	95	100	105
φ^1	2.636s	2.698s	3.042s	0.253s	0.212s	83.93s
φ^2	9.602s	17.66s	24.04s	84.45s	47.65s	–
φ^3	414.1s	384.7s	393.9s	–	–	–

Conclusions and Future Work

- Significant progress in the formal verification for AI
- Scalability remains an issue
- Focus mostly on a limited repertoire of specifications
- Robust training and neuro-symbolic learning
- Verification for unbounded systems of neuro-symbolic agents