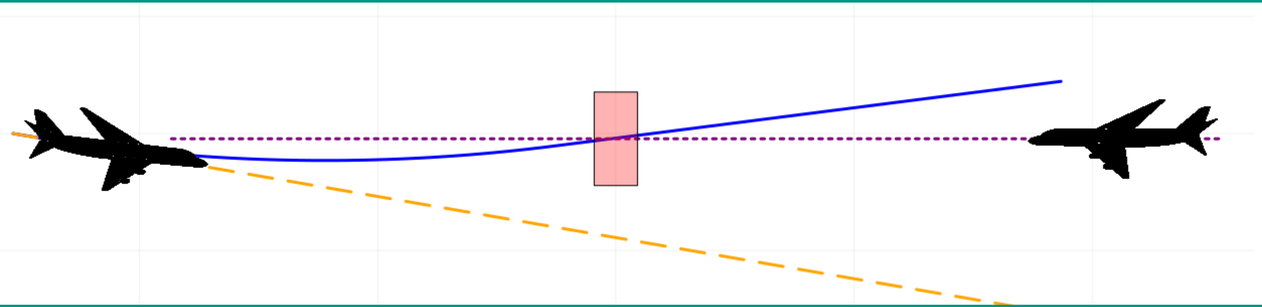
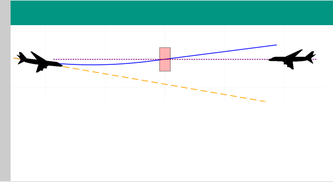


Of Good Demons and Bad Angels:

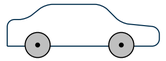
Guaranteeing Safe Control under Finite Precision

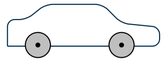
FMCAD 2025

Samuel Teuber, Debasmita Lohar, Bernhard Beckert | 2025



Motivation: Neural Network Control Systems





1. Cars
2. Constant Velocity
3. Differential Equations

Motivation: Neural Network Control Systems

1. Cars
2. Constant Velocity
3. Differential Equations

Motivation: Neural Network Control Systems

1. Cars
2. Constant Velocity
3. Differential Equations

Motivation: Neural Network Control Systems

$$p'_{rel} = v_{\text{rel}}$$
$$v'_{rel} = -a_{rel} = -g\,(p_{rel}, v_{rel})$$

How can we prove the **infinite-time horizon** safety of a strategy g?


How can we prove the **infinite-time horizon** safety of a strategy g?

1. Cars
2. Constant Velocity
3. Differential Equations

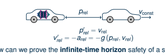
Motivation: Neural Network Control Systems

$$p'_{rel} = v_{\text{rel}}$$
$$v'_{rel} = -a_{rel} = -g\left(p_{rel}, v_{rel}\right)$$

How can we prove the **infinite-time horizon** safety of a strategy g?

$g\left(p_{rel}, v_{rel}\right) = -B < 0$

Safe if cars start far enough apart (depends on B).



How can we prove the **infinite-time horizon** safety of a strategy g?

$g\left(p_{rel}, v_{rel}\right) = -B < 0$

Safe if cars start far enough apart (depends on B).

1. Simple Case: Just brake

Motivation: Neural Network Control Systems

$$p'_{rel} = v_{\text{rel}}$$
$$v'_{rel} = -a_{rel} = -g\left(p_{rel}, v_{rel}\right)$$

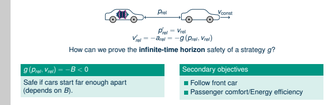
How can we prove the **infinite-time horizon** safety of a strategy g?

$g\left(p_{rel}, v_{rel}\right) = -B < 0$

Safe if cars start far enough apart (depends on B).

Secondary objectives

- Follow front car
- Passenger comfort/Energy efficiency



1. Continuous Time
2. Infinite-Time Safety

Motivation: Neural Network Control Systems

$$p'_{rel} = v_{\text{rel}}$$
$$v'_{rel} = -a_{rel} = -g\,(p_{rel}, v_{rel})$$

How can we prove the **infinite-time horizon** safety of a strategy g?

$g\,(p_{rel}, v_{rel}) = -B < 0$

Safe if cars start far enough apart (depends on B).

Secondary objectives

- Follow front car
- Passenger comfort/Energy efficiency

$\Rightarrow$ We leverage **Differential Dynamic Logic** to verify NN Control Systems

$p'_{rel} = v_{\text{rel}}$
$v'_{rel} = -a_{rel} = -g\,(p_{rel}, v_{rel})$

How can we prove the **infinite-time horizon** safety of a strategy g?

$g\,(p_{rel}, v_{rel}) = -B < 0$

Safe if cars start far enough apart (depends on B).

Secondary objectives

- Follow front car
- Passenger comfort/Energy efficiency

$\Rightarrow$ We leverage **Differential Dynamic Logic** to verify NN Control Systems

Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

[Platzer 2008]

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

[Platzer 2008]

1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

hybrid program

[Platzer 2008]

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

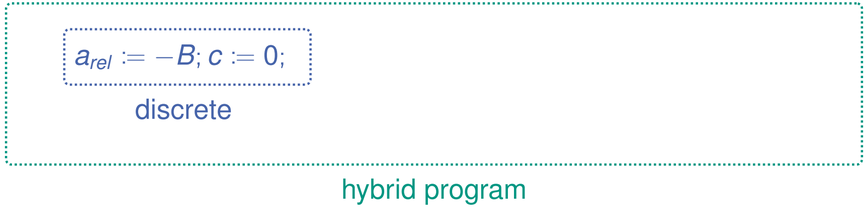
hybrid program

[Platzer 2008]

1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.



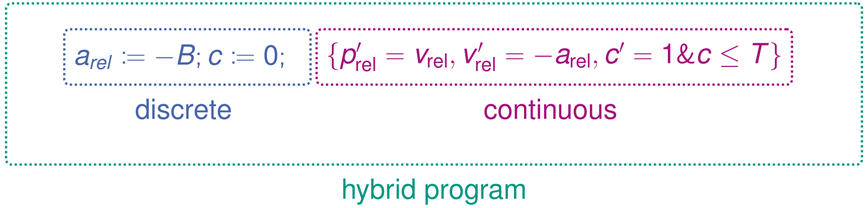
[Platzer 2008]

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.
$a_{rel} := -B; c := 0;$
discrete
hybrid program
[Platzer 2008]

1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.



[Platzer 2008]

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

$a_{rel} := -B; c := 0;$ | $\{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = -a_{\text{rel}}, c' = 1 \& c \leq T\}$

discrete continuous

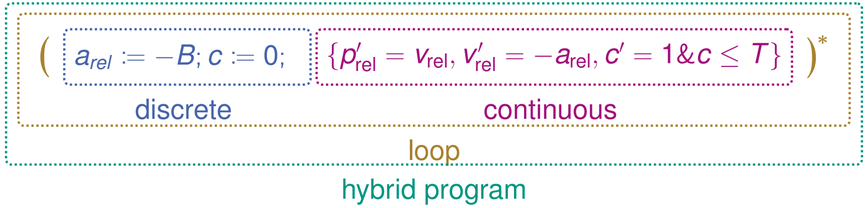
hybrid program

[Platzer 2008]

1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.



[Platzer 2008]

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

$(a_{rel} := -B; c := 0; \{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = -a_{\text{rel}}, c' = 1 \& c \leq T\})^*$

discrete continuous loop hybrid program

[Platzer 2008]

1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

$$\Big[\Big(\underbrace{a_{rel} := -B; c := 0;}_{\text{discrete}} \underbrace{\{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = -a_{\text{rel}}, c' = 1 \& c \leq T\}}_{\text{continuous}}\Big)^*\Big]$$

loop

hybrid program

after every program run

[Platzer 2008]

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

$$\Big[\Big(\underbrace{a_{rel} := -B; c := 0;}_{\text{discrete}} \underbrace{\{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = -a_{\text{rel}}, c' = 1 \& c \leq T\}}_{\text{continuous}}\Big)^*\Big]$$

loop

hybrid program

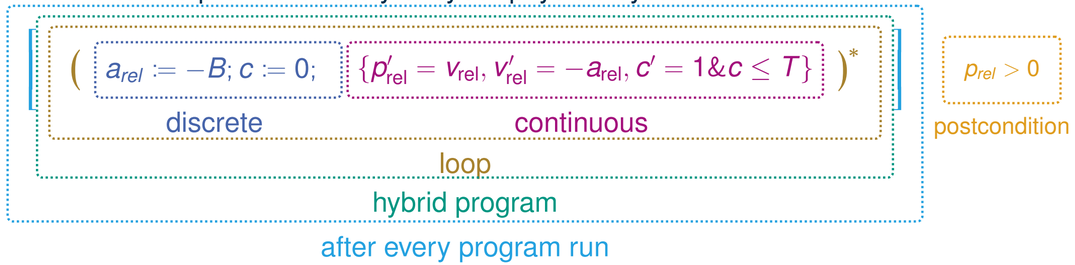
after every program run

[Platzer 2008]

1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.



[Platzer 2008]

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

[Platzer 2008]

1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.

$p_{rel} > \frac{v_{rel}^2}{2B} \rightarrow$ precondition

$[$ $($ $a_{rel} := -B; c := 0;$ discrete $\{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = -a_{\text{rel}}, c' = 1 \& c \leq T\}$ continuous $)^*$ $]$

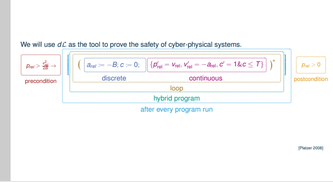
loop

hybrid program

after every program run

$p_{rel} > 0$ postcondition

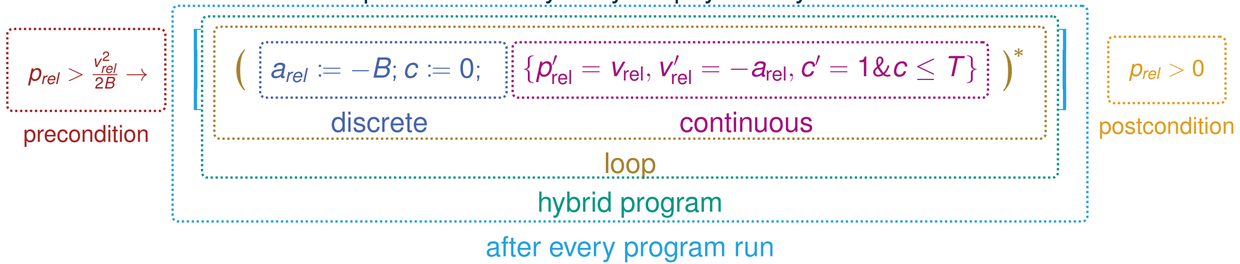
[Platzer 2008]



1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

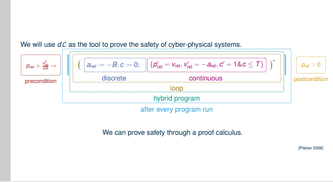
Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.



We can prove safety through a proof calculus.

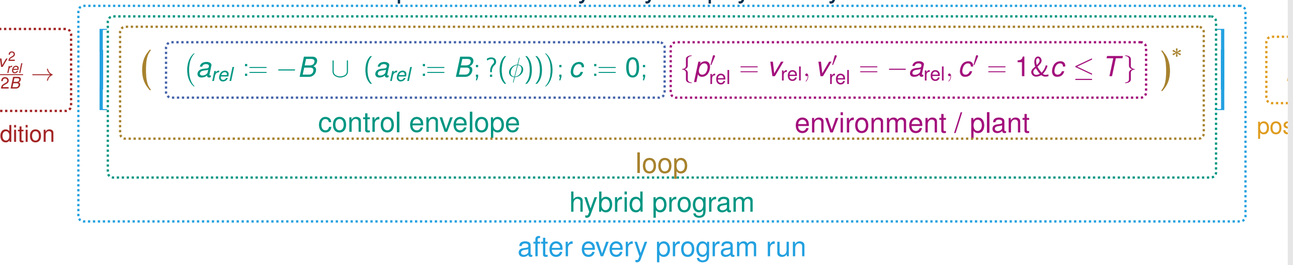
[Platzer 2008]



1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

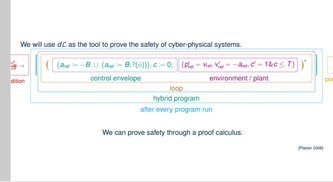
Differential Dynamic Logic by Example

We will use $d\mathcal{L}$ as the tool to prove the safety of cyber-physical systems.



We can prove safety through a proof calculus.

[Platzer 2008]



1. Real World programs $\neq$ hybrid programs
2. *Envelope* instead of concrete strategy
3. Need for monitoring

Verifiably Safe AI via Logically Linked Envelopes

$$\left(\alpha_{\text{ctrl}} \;;\alpha_{\text{plant}}\right)^{*} \quad \text{Safe}$$

g 

[NeurIPS'24]

1. Nondeterministic Mirror
2. VerSAILLE
3. NN Verification
4. Mirror: Refinement Proof
5. Safety guarantee

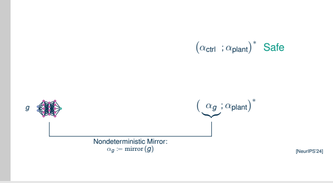
Verifiably Safe AI via Logically Linked Envelopes

$$\left(\alpha_{\text{ctrl}} \; ; \alpha_{\text{plant}}\right)^* \quad \text{Safe}$$

$$\left(\underbrace{\alpha_g}_{} \; ; \alpha_{\text{plant}}\right)^*$$

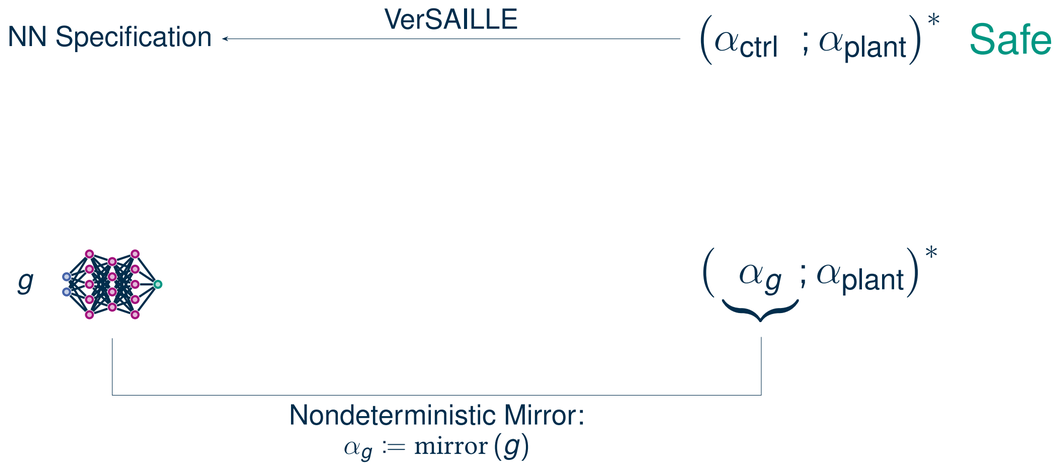
Nondeterministic Mirror:
$\alpha_g := \text{mirror}\,(g)$

[NeurIPS'24]

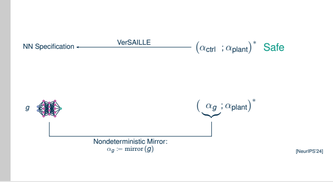


1. Nondeterministic Mirror
2. VerSAILLE
3. NN Verification
4. Mirror: Refinement Proof
5. Safety guarantee

Verifiably Safe AI via Logically Linked Envelopes

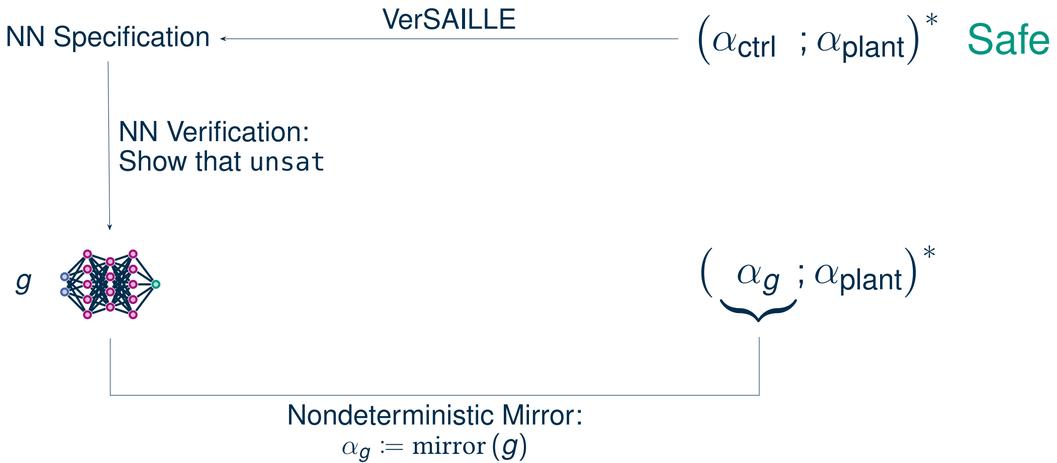


[NeurIPS'24]

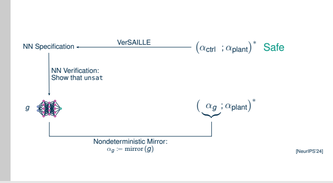


1. Nondeterministic Mirror
2. VerSAILLE
3. NN Verification
4. Mirror: Refinement Proof
5. Safety guarantee

Verifiably Safe AI via Logically Linked Envelopes

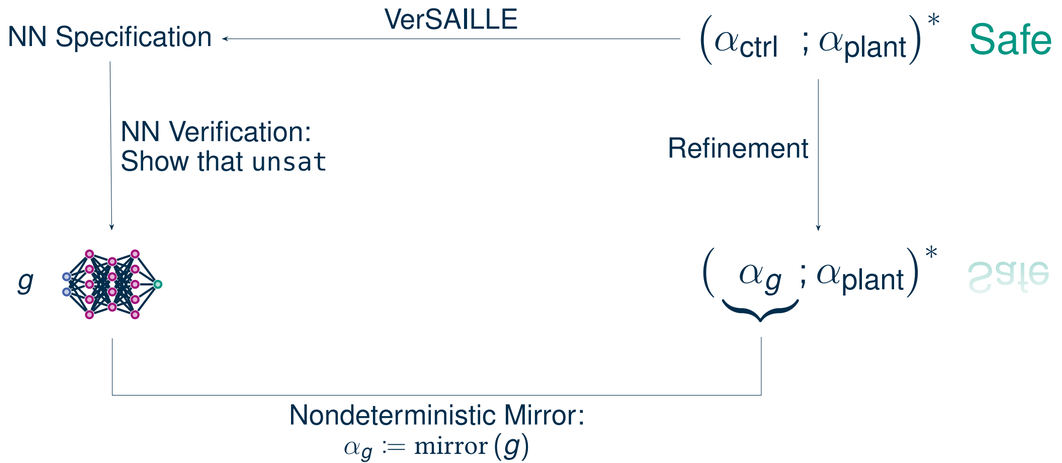


[NeurIPS'24]

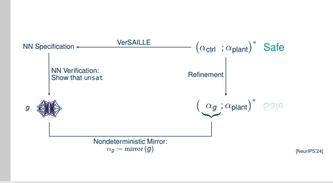


1. Nondeterministic Mirror
2. VerSAILLE
3. NN Verification
4. Mirror: Refinement Proof
5. Safety guarantee

Verifiably Safe AI via Logically Linked Envelopes



[NeurIPS'24]

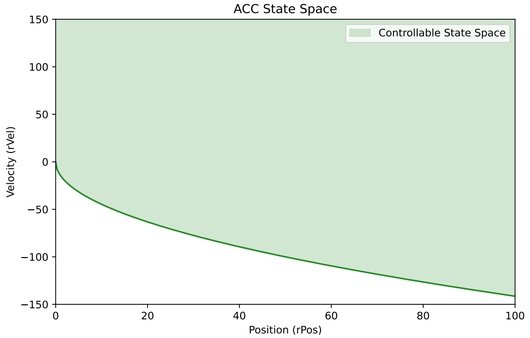


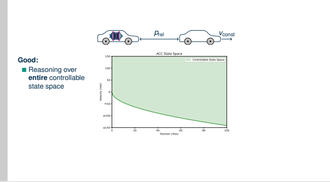
1. Nondeterministic Mirror
2. VerSAILLE
3. NN Verification
4. Mirror: Refinement Proof
5. Safety guarantee

VerSAILLE: The Good

Good:

- Reasoning over **entire** controllable state space

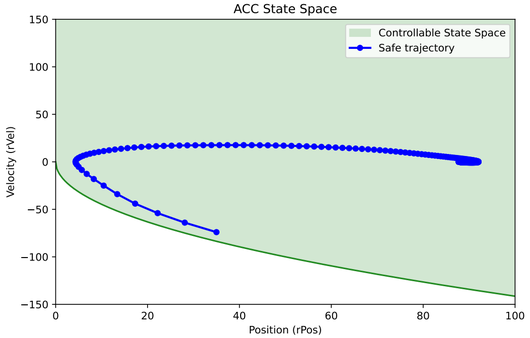


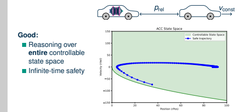


VerSAILLE: The Good

Good:

- Reasoning over **entire** controllable state space
- Infinite-time safety

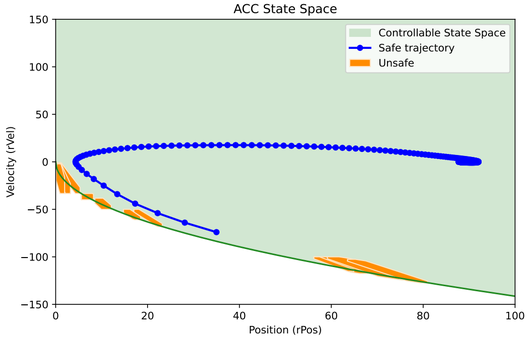


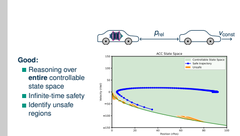


VerSAILLE: The Good

Good:

- Reasoning over **entire** controllable state space
- Infinite-time safety
- Identify unsafe regions

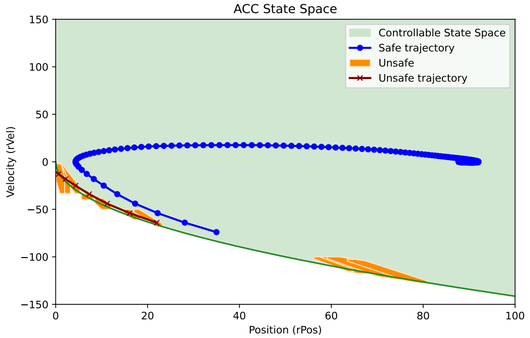


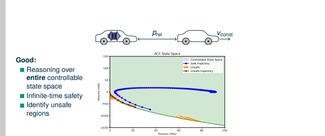


VerSAILLE: The Good

Good:

- Reasoning over **entire** controllable state space
- Infinite-time safety
- Identify unsafe regions

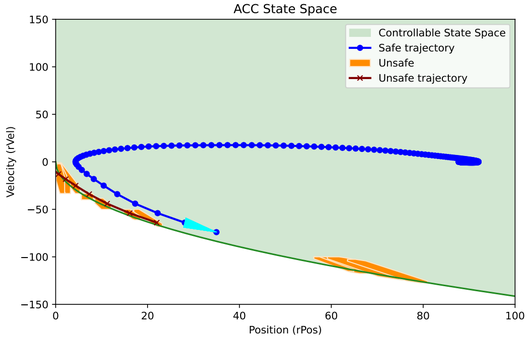




VerSAILLE: The Good and the Bad

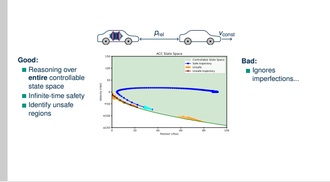
Good:

- Reasoning over **entire** controllable state space
- Infinite-time safety
- Identify unsafe regions



Bad:

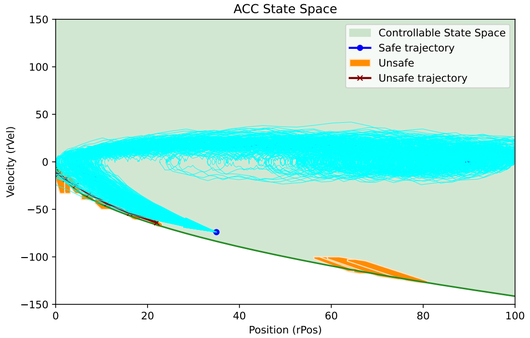
- Ignores imperfections...



VerSAILLE: The Good and the Bad

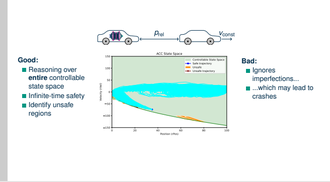
Good:

- Reasoning over **entire** controllable state space
- Infinite-time safety
- Identify unsafe regions



Bad:

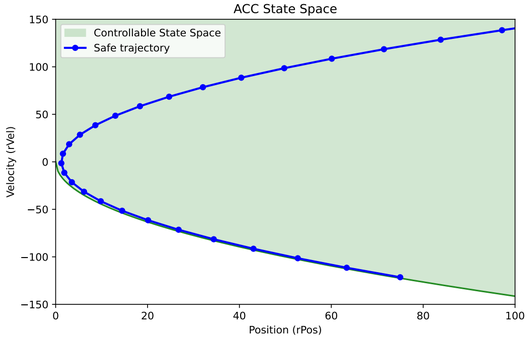
- Ignores imperfections...
- ...which may lead to crashes



VerSAILLE: The Good and the Bad

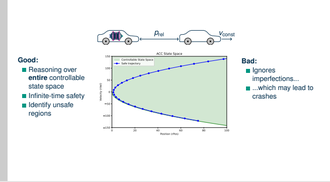
Good:

- Reasoning over **entire** controllable state space
- Infinite-time safety
- Identify unsafe regions



Bad:

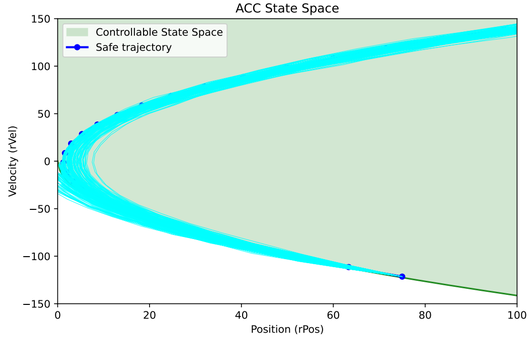
- Ignores imperfections...
- ...which may lead to crashes



VerSAILLE: The Good and the Bad

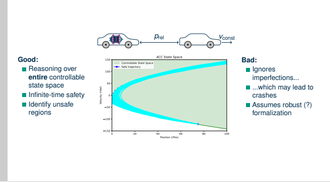
Good:

- Reasoning over **entire** controllable state space
- Infinite-time safety
- Identify unsafe regions



Bad:

- Ignores imperfections...
- ...which may lead to crashes
- Assumes robust (?) formalization

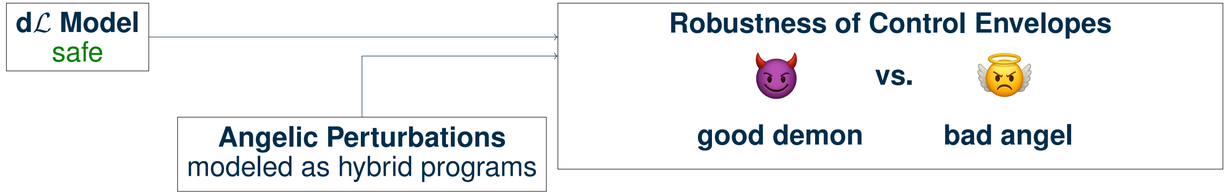


Contribution

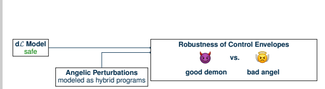
Angelic Perturbations
modeled as hybrid programs

Angelic Perturbations
modeled as hybrid programs

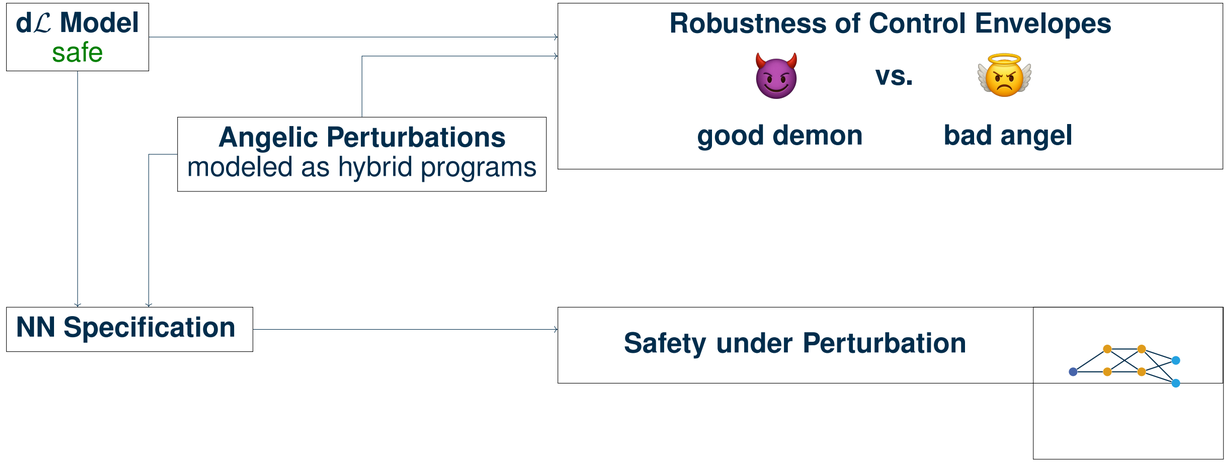
Contribution



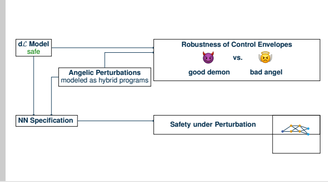
└Contribution



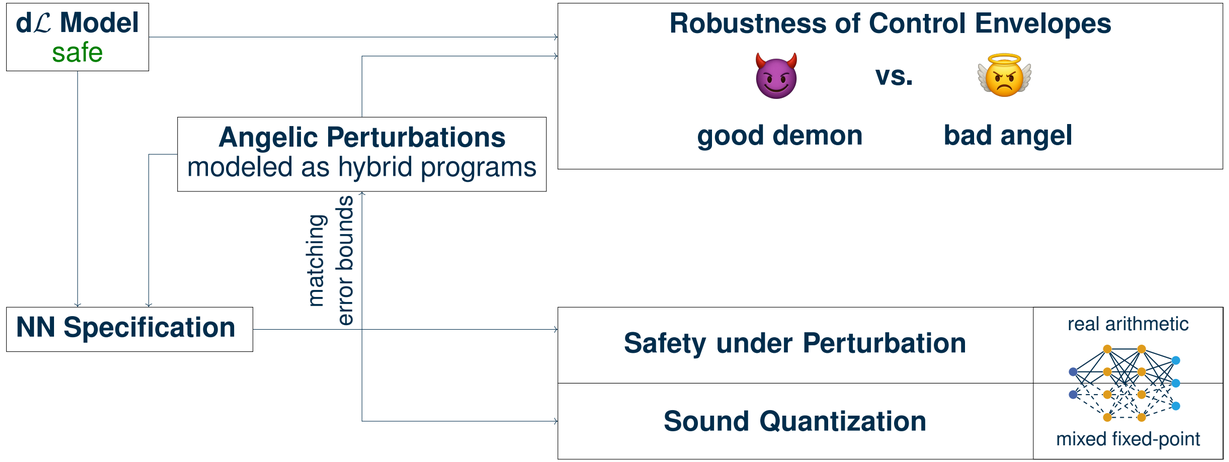
Contribution



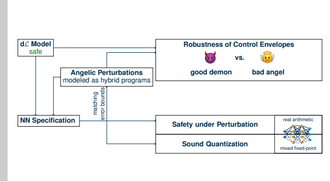
└Contribution



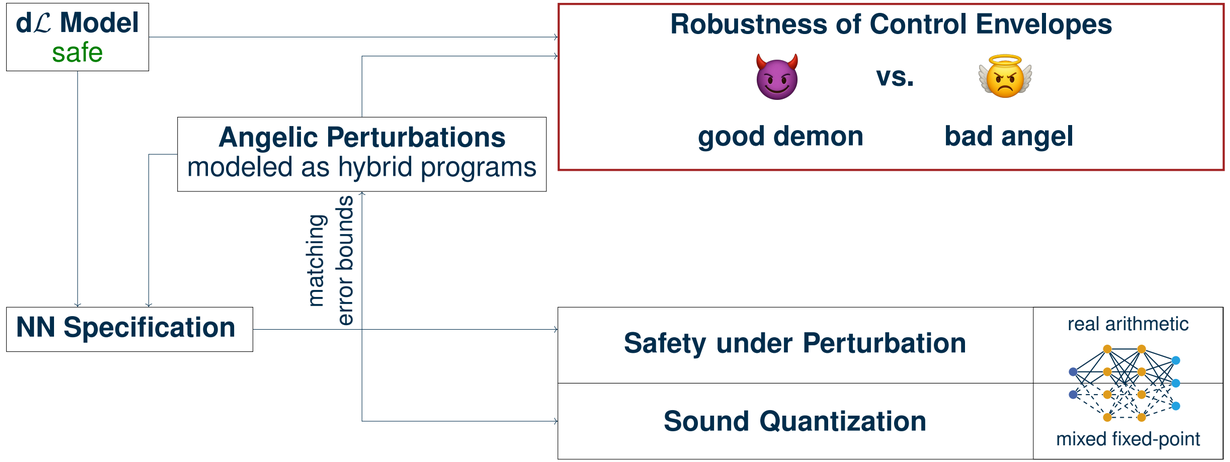
Contribution



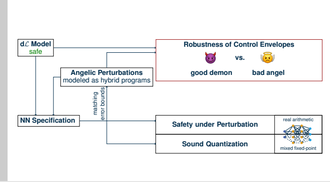
└Contribution



Contribution



└Contribution



Control Envelope Robustness

What we know:

$$\text{pre} \rightarrow \left[\left(\text{ctl}; \ \text{env} \right)^* \right] \text{post}$$

What we know:
$\text{pre} \rightarrow \left[\left(\text{ctl}; \ \text{env} \right)^* \right] \text{post}$

Control Envelope Robustness

What we know:

$$\text{pre} \rightarrow \left[\; (\; \text{ctl}; \;\; \text{env} \;)^* \; \right] \text{post}$$

What we want:

$$\text{pre} \rightarrow \left[\; (\; \text{sense}; \;\; \text{ctl} \;\; ; \;\; \text{act}; \;\; \text{env} \;)^* \; \right] \text{post}$$

What we know:
pre → [(ctl; env)*] post
What we want:
pre → [(sense; ctl ; act; env)*] post

Control Envelope Robustness

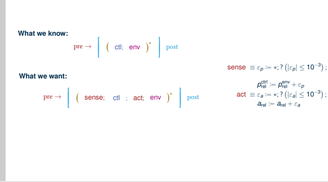
What we know:

$$\text{pre} \rightarrow \left[\left(\text{ctl}; \ \text{env} \right)^* \right] \text{post}$$

What we want:

$$\text{pre} \rightarrow \left[\left(\text{sense}; \ \text{ctl} \ ; \ \text{act}; \ \text{env} \right)^* \right] \text{post}$$

$$\begin{aligned}
\text{sense} &\equiv \varepsilon_p := *; ?\left(|\varepsilon_p| \leq 10^{-3}\right); \\
& \quad p_{\text{rel}}^{\text{ctrl}} := p_{\text{rel}}^{\text{env}} + \varepsilon_p \\
\text{act} &\equiv \varepsilon_a := *; ?\left(|\varepsilon_a| \leq 10^{-3}\right); \\
& \quad a_{\text{rel}} := a_{\text{rel}} + \varepsilon_a
\end{aligned}$$



1. Usually: **DEMON:** Bad env; **ANGEL:** Good control
2. HOWEVER: By tradition in game logics: **ANGEL** has control; **BOX:** Demon wins

Control Envelope Robustness

What we know:

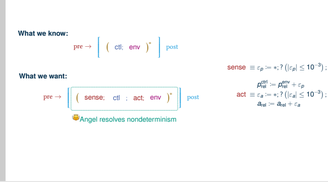
$$\text{pre} \rightarrow \left[\left(\text{ctl}; \ \text{env} \right)^* \right] \text{post}$$

What we want:

$$\text{pre} \rightarrow \left[\left(\text{sense}; \ \text{ctl} \ ; \ \text{act}; \ \text{env} \right)^* \right] \text{post}$$

😈 Angel resolves nondeterminism

$$\begin{aligned}
\text{sense} &\equiv \varepsilon_p := *; ?\left(|\varepsilon_p| \leq 10^{-3}\right); \\
& \quad p_{\text{rel}}^{\text{ctrl}} := p_{\text{rel}}^{\text{env}} + \varepsilon_p \\
\text{act} &\equiv \varepsilon_a := *; ?\left(|\varepsilon_a| \leq 10^{-3}\right); \\
& \quad a_{\text{rel}} := a_{\text{rel}} + \varepsilon_a
\end{aligned}$$



1. Usually: **DEMON:** Bad env; **ANGEL:** Good control
2. HOWEVER: By tradition in game logics: **ANGEL** has control; **BOX:** Demon wins

Control Envelope Robustness

What we know:

$$\text{pre} \rightarrow \left[\left(\text{ctl}; \ \text{env} \right)^* \right] \text{post}$$

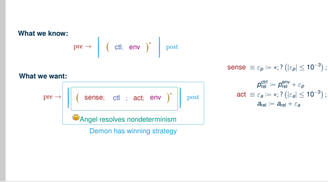
What we want:

$$\text{pre} \rightarrow \left[\left(\text{sense}; \ \text{ctl} \ ; \ \text{act}; \ \text{env} \right)^* \right] \text{post}$$

😈 Angel resolves nondeterminism

Demon has winning strategy

$$\begin{aligned}
\text{sense} &\equiv \varepsilon_p := *; ?\left(|\varepsilon_p| \leq 10^{-3}\right); \\
& \quad p_{\text{rel}}^{\text{ctrl}} := p_{\text{rel}}^{\text{env}} + \varepsilon_p \\
\text{act} &\equiv \varepsilon_a := *; ?\left(|\varepsilon_a| \leq 10^{-3}\right); \\
& \quad a_{\text{rel}} := a_{\text{rel}} + \varepsilon_a
\end{aligned}$$



1. Usually: **DEMON:** Bad env; **ANGEL:** Good control
2. HOWEVER: By tradition in game logics: **ANGEL** has control; **BOX:** Demon wins

Control Envelope Robustness

What we know:

$$\text{pre} \rightarrow \left[\left(\text{ctl}; \ \text{env} \right)^* \right] \text{post}$$

What we want:

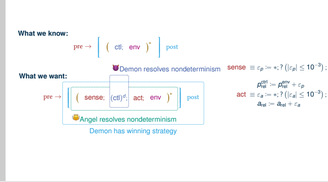
Demon resolves nondeterminism

$$\text{pre} \rightarrow \left[\left(\text{sense}; \ (\text{ctl})^d; \ \text{act}; \ \text{env} \right)^* \right] \text{post}$$

Angel resolves nondeterminism

Demon has winning strategy

$$\text{sense} \equiv \varepsilon_p := *; ?\left(|\varepsilon_p| \leq 10^{-3}\right);$$
$$p_{\text{rel}}^{\text{ctrl}} := p_{\text{rel}}^{\text{env}} + \varepsilon_p$$
$$\text{act} \equiv \varepsilon_a := *; ?\left(|\varepsilon_a| \leq 10^{-3}\right);$$
$$a_{\text{rel}} := a_{\text{rel}} + \varepsilon_a$$



1. Usually: **DEMON:** Bad env; **ANGEL:** Good control
2. HOWEVER: By tradition in game logics: **ANGEL** has control; **BOX:** Demon wins

Control Envelope Robustness

What we know:

$$\text{pre} \rightarrow \left[\left(\text{ctl}; \ \text{env} \right)^* \right] \text{post}$$

What we want:

$$\text{pre} \rightarrow \left[\left(\text{sense}; \ (\text{ctl})^d; \ \text{act}; \ \text{env} \right)^* \right] \text{post}$$

😈 Demon resolves nondeterminism

😇 Angel resolves nondeterminism

Demon has winning strategy

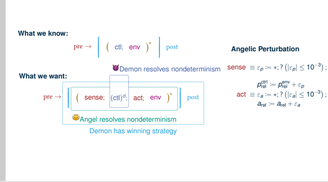
Angelic Perturbation

$$\text{sense} \equiv \varepsilon_p := *; ?\left(|\varepsilon_p| \leq 10^{-3}\right);$$

$$p_{\text{rel}}^{\text{ctrl}} := p_{\text{rel}}^{\text{env}} + \varepsilon_p$$

$$\text{act} \equiv \varepsilon_a := *; ?\left(|\varepsilon_a| \leq 10^{-3}\right);$$

$$a_{\text{rel}} := a_{\text{rel}} + \varepsilon_a$$



1. Usually: **DEMON:** Bad env; **ANGEL:** Good control
2. HOWEVER: By tradition in game logics: **ANGEL** has control; **BOX:** Demon wins

Control Envelope Robustness

What we know:

$$\text{pre} \rightarrow \left[\left(\text{ctl}; \ \text{env} \right)^* \right] \text{post}$$

What we want:

$$\text{pre} \rightarrow \left[\left(\text{sense}; \ (\text{ctl})^d; \ \text{act}; \ \text{env} \right)^* \right] \text{post}$$

😈 Demon resolves nondeterminism

😇 Angel resolves nondeterminism

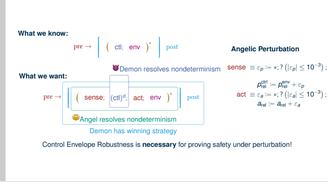
Demon has winning strategy

Angelic Perturbation

$$\text{sense} \equiv \varepsilon_p := *; ?\left(|\varepsilon_p| \leq 10^{-3}\right);$$

$$\text{act} \equiv \varepsilon_a := *; ?\left(|\varepsilon_a| \leq 10^{-3}\right); \begin{array}{l} p_{\text{rel}}^{\text{ctrl}} := p_{\text{rel}}^{\text{env}} + \varepsilon_p \\ a_{\text{rel}} := a_{\text{rel}} + \varepsilon_a \end{array}$$

Control Envelope Robustness is **necessary** for proving safety under perturbation!



1. Usually: **DEMON:** Bad env; **ANGEL:** Good control
2. HOWEVER: By tradition in game logics: **ANGEL** has control; **BOX:** Demon wins

Control Envelope Robustness: Observations

Liveness as Special Case

$$\text{sense} \equiv \textbf{skip}$$
$$\text{act} \equiv \textbf{skip}$$

Liveness as Special Case
sense ≡ skip
act ≡ skip

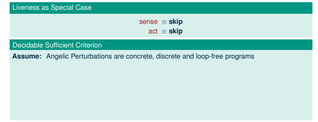
Control Envelope Robustness: Observations

Liveness as Special Case

$$\text{sense} \equiv \textbf{skip}$$
$$\text{act} \equiv \textbf{skip}$$

Decidable Sufficient Criterion

Assume: Angelic Perturbations are concrete, discrete and loop-free programs



Control Envelope Robustness: Observations

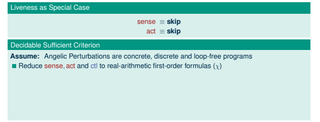
Liveness as Special Case

$$\text{sense} \equiv \textbf{skip}$$
$$\text{act} \equiv \textbf{skip}$$

Decidable Sufficient Criterion

Assume: Angelic Perturbations are concrete, discrete and loop-free programs

- Reduce sense, act and ctl to real-arithmetic first-order formulas (χ)



Control Envelope Robustness: Observations

Liveness as Special Case

$$\begin{aligned} \text{sense} &\equiv \textbf{skip} \\ \text{act} &\equiv \textbf{skip} \end{aligned}$$

Decidable Sufficient Criterion

Assume: Angelic Perturbations are concrete, discrete and loop-free programs

- Reduce sense, act and ctl to real-arithmetic first-order formulas (χ)
- For invariant inv check that: (based on insights from Prebet et al. 2024)

$$\begin{aligned} \forall \bar{x}_0\ \forall x_1\ \exists \bar{x}_2\ \forall \bar{x}_3\ \big(&\text{inv}\,(\bar{x}_0) \land \chi_{\text{sense}}\,(\bar{x}_0, \bar{x}_1) \rightarrow \\ &(\chi_{\text{ctl}}\,(\bar{x}_1, \bar{x}_2) \land (\chi_{\text{act}}\,(\bar{x}_2, \bar{x}_3) \rightarrow \chi_{\text{ctl}}\,(\bar{x}_0, \bar{x}_3)))\big) \end{aligned}$$

Liveness as Special Case
sense ≡ skip
act ≡ skip
Decidable Sufficient Criterion
Assume: Angelic Perturbations are concrete, discrete and loop-free programs
Reduce sense, act and ctl to real-arithmetic first-order formulas (χ)
For invariant inv check that: (based on insights from Prebet et al. 2024)

Control Envelope Robustness: Observations

Liveness as Special Case

$$\begin{aligned} \text{sense} &\equiv \textbf{skip} \\ \text{act} &\equiv \textbf{skip} \end{aligned}$$

Decidable Sufficient Criterion

Assume: Angelic Perturbations are concrete, discrete and loop-free programs

- Reduce sense, act and ctl to real-arithmetic first-order formulas (χ)
- For invariant inv check that: (based on insights from Prebet et al. 2024)

$$\forall \bar{x}_0\ \forall x_1\ \exists \bar{x}_2\ \forall \bar{x}_3\ \ \big(\text{inv}\,(\bar{x}_0) \wedge \chi_{\text{sense}}\,(\bar{x}_0, \bar{x}_1)\ \rightarrow \\ (\chi_{\text{ctl}}\,(\bar{x}_1, \bar{x}_2) \wedge (\chi_{\text{act}}\,(\bar{x}_2, \bar{x}_3) \rightarrow \chi_{\text{ctl}}\,(\bar{x}_0, \bar{x}_3)))\big)$$

Can be simplified for additive actuator perturbations

Liveness as Special Case
sense ≡ skip
act ≡ skip
Decidable Sufficient Criterion
Assume: Angelic Perturbations are concrete, discrete and loop-free programs
Reduce sense, act and ctl to real-arithmetic first-order formulas (χ)
For invariant inv check that: (based on insights from Prebet et al. 2024)
∀x̄0 ∀x1 ∃x̄2 ∀x̄3 (inv(x̄0) ∧ χsense(x̄0, x̄1) → (χctl(x̄1, x̄2) ∧ (χact(x̄2, x̄3) → χctl(x̄0, x̄3))))
Can be simplified for additive actuator perturbations

Control Envelope Robustness: Observations

Liveness as Special Case

$$\text{sense} \equiv \textbf{skip}$$
$$\text{act} \equiv \textbf{skip}$$

Decidable Sufficient Criterion

Assume: Angelic Perturbations are concrete, discrete and loop-free programs

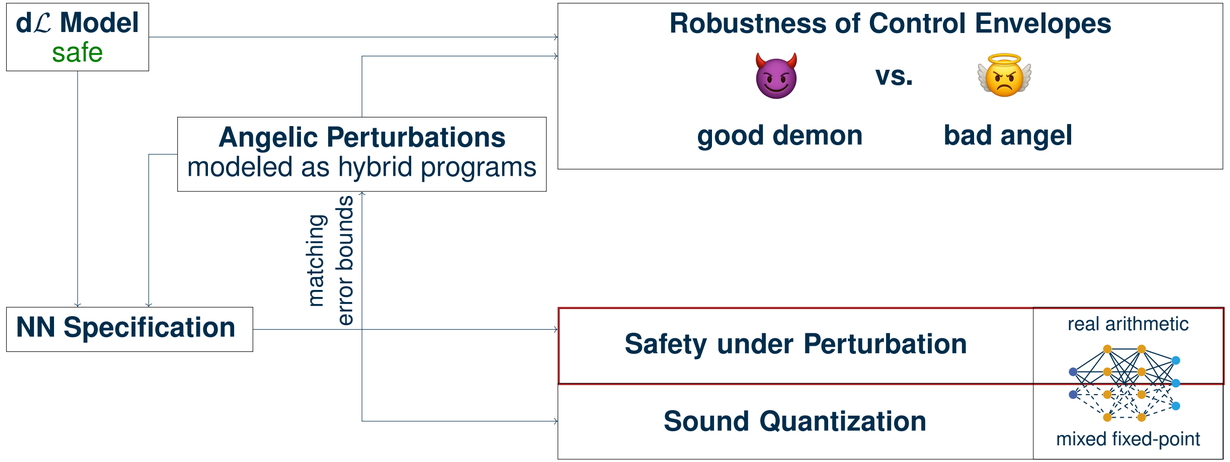
- Reduce sense, act and ctl to real-arithmetic first-order formulas (χ)
- For invariant inv check that: **Demon's winning action** on insights from Prebet et al. 2024)

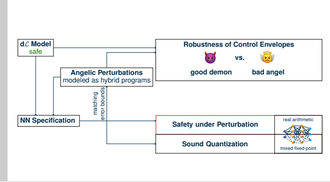
$$\forall \bar{x}_0\ \forall x_1\ \boxed{\exists \bar{x}_2}\ \forall \bar{x}_3\ \ \big(\text{inv}\,(\bar{x}_0) \wedge \chi_{\text{sense}}\,(\bar{x}_0, \bar{x}_1)\ \rightarrow \\ (\chi_{\text{ctl}}\,(\bar{x}_1, \bar{x}_2) \wedge (\chi_{\text{act}}\,(\bar{x}_2, \bar{x}_3) \rightarrow \chi_{\text{ctl}}\,(\bar{x}_0, \bar{x}_3)))\big)$$

Can be simplified for additive actuator perturbations

Liveness as Special Case
sense ≡ skip
act ≡ skip
Decidable Sufficient Criterion
Assume: Angelic Perturbations are concrete, discrete and loop-free programs
Reduce sense, act and ctl to real-arithmetic first-order formulas (χ)
For invariant inv check that: Demon's winning action on insights from Prebet et al. 2024)
Can be simplified for additive actuator perturbations

Contribution





Perturbed VerSAILLE

- Recall: VerSAILLE derives an NN specification from the dL model
- We extend this specification generation for **safety under angelic perturbations**

Recall: VerSAILLE derives an NN specification from the dL model
We extend this specification generation for
safety under angelic perturbations

Perturbed VerSAILLE

- Recall: VerSAILLE derives an NN specification from the dL model
- We extend this specification generation for **safety under angelic perturbations**

$$\forall \bar{x}_0\ \forall x_1\ \forall \bar{x}_2\ \forall \bar{x}_3\ \ \big(\mathsf{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \rightarrow \ \ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$$

Recall: VerSAILLE derives an NN specification from the dL model
We extend this specification generation for
safety under angelic perturbations
$\forall \bar{x}_0\ \forall x_1\ \forall \bar{x}_2\ \forall \bar{x}_3\ \ \big(\mathsf{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \rightarrow \ \ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$

Perturbed VerSAILLE

- Recall: VerSAILLE derives an NN specification from the dL model
- We extend this specification generation for **safety under angelic perturbations**

All Control Actions Safe

$$\forall \bar{x}_0\ \forall x_1\ \boxed{\forall \bar{x}_2}\ \forall \bar{x}_3\ \ \big(\mathsf{inv}\,(\bar{x}_0) \wedge \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \wedge \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \wedge \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \rightarrow\ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$$

Recall: VerSAILLE derives an NN specification from the dL model
We extend this specification generation for
safety under angelic perturbations
All Control Actions Safe
$\forall \bar{x}_0\ \forall x_1\ \boxed{\forall \bar{x}_2}\ \forall \bar{x}_3\ \ \big(\mathsf{inv}\,(\bar{x}_0) \wedge \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \wedge \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \wedge \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \rightarrow\ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$

Perturbed VerSAILLE

- Recall: VerSAILLE derives an NN specification from the dL model
- We extend this specification generation for **safety under angelic perturbations**

$$\forall \bar{x}_0\ \forall x_1\ \forall \bar{x}_2\ \forall \bar{x}_3\ \ \big(\mathsf{inv}\,(\bar{x}_0) \wedge \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \wedge \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \wedge \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \rightarrow \ \ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$$

Phrased as counterexample search:

$$\exists \bar{x}_0\ \exists x_1\ \exists \bar{x}_2\ \exists \bar{x}_3\ \ \big(\mathsf{inv}\,(\bar{x}_0) \wedge \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \wedge \mathit{impl}(\bar{x}_1, \bar{x}_2) \wedge \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \wedge \ \ \neg\chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$$

Recall: VerSAILLE derives an NN specification from the dL model
We extend this specification generation for
safety under angelic perturbations
$\forall \bar{x}_0\ \forall x_1\ \forall \bar{x}_2\ \forall \bar{x}_3\ \ \big(\mathsf{inv}\,(\bar{x}_0) \wedge \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \wedge \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \wedge \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \rightarrow \ \ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$
Phrased as counterexample search:
$\exists \bar{x}_0\ \exists x_1\ \exists \bar{x}_2\ \exists \bar{x}_3\ \ \big(\mathsf{inv}\,(\bar{x}_0) \wedge \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \wedge \mathit{impl}(\bar{x}_1, \bar{x}_2) \wedge \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \wedge \ \ \neg\chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$

Perturbed VerSAILLE

- Recall: VerSAILLE derives an NN specification from the dL model
- We extend this specification generation for **safety under angelic perturbations**

$$\forall \bar{x}_0\ \forall x_1\ \forall \bar{x}_2\ \forall \bar{x}_3\ \ \big(\mathit{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \rightarrow \ \ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$$

Phrased as counterexample search:

$$\exists \bar{x}_0\ \exists x_1\ \exists \bar{x}_2\ \exists \bar{x}_3\ \ \big(\mathit{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}(\underbrace{\bar{x}_1, \bar{x}_2}_{\text{NN input/output}}) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \land \ \ \neg\chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$$

Recall: VerSAILLE derives an NN specification from the dL model
We extend this specification generation for
safety under angelic perturbations
$\forall \bar{x}_0\ \forall x_1\ \forall \bar{x}_2\ \forall \bar{x}_3\ \ \big(\mathit{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \rightarrow \ \ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$
Phrased as counterexample search:
$\exists \bar{x}_0\ \exists x_1\ \exists \bar{x}_2\ \exists \bar{x}_3\ \ \big(\mathit{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}(\underbrace{\bar{x}_1, \bar{x}_2}_{\text{NN input/output}}) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \land \ \ \neg\chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$

Perturbed VerSAILLE

- Recall: VerSAILLE derives an NN specification from the dL model
- We extend this specification generation for **safety under angelic perturbations**

$$\forall \bar{x}_0\ \forall x_1\ \forall \bar{x}_2\ \forall \bar{x}_3\ \ \big(\mathit{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \rightarrow \ \ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$$

Phrased as counterexample search:

$$\exists \bar{x}_0\ \exists x_1\ \exists \bar{x}_2\ \exists \bar{x}_3\ \ \big(\mathit{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}(\underbrace{\bar{x}_1, \bar{x}_2}_{\text{NN input/output}}) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \land \ \ \neg\chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$$

Can be simplified for additive actuator perturbations

- Recall: VerSAILLE derives an NN specification from the dL model
- We extend this specification generation for **safety under angelic perturbations**

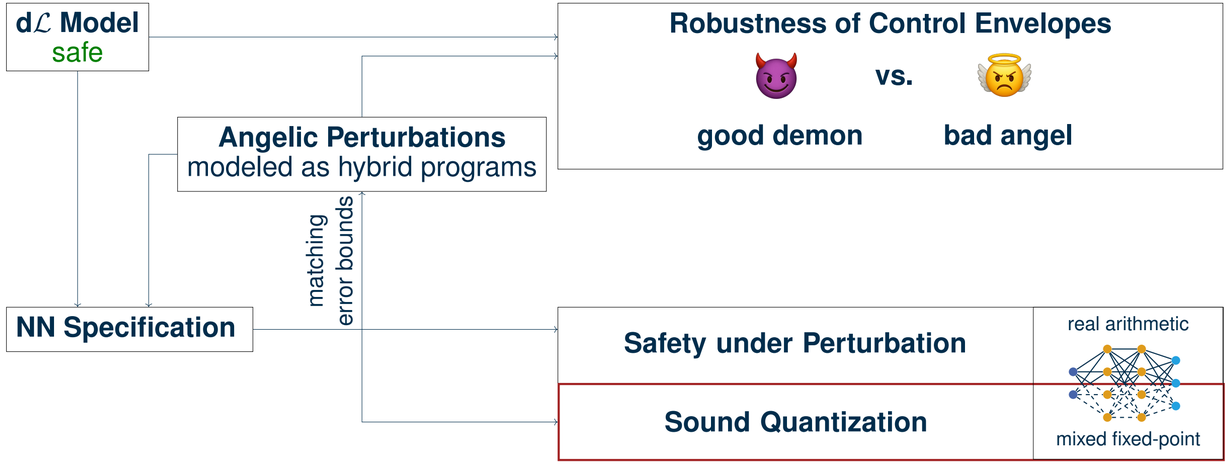
$\forall \bar{x}_0\ \forall x_1\ \forall \bar{x}_2\ \forall \bar{x}_3\ \ \big(\mathit{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}\,(\bar{x}_1, \bar{x}_2) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \rightarrow \ \ \chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$

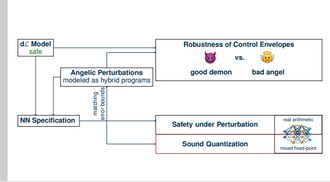
Phrased as counterexample search:

$\exists \bar{x}_0\ \exists x_1\ \exists \bar{x}_2\ \exists \bar{x}_3\ \ \big(\mathit{inv}\,(\bar{x}_0) \land \chi_{\mathsf{sense}}\,(\bar{x}_0, \bar{x}_1) \land \mathit{impl}(\underbrace{\bar{x}_1, \bar{x}_2}_{\text{NN input/output}}) \land \chi_{\mathsf{act}}\,(\bar{x}_2, \bar{x}_3)\big) \ \ \land \ \ \neg\chi_{\mathsf{ctl}}\,(\bar{x}_0, \bar{x}_3)$

Can be simplified for additive actuator perturbations

Contribution





NN Synthesis: Mixed Precision Tuning

Daisy Darulova et al. 2018

- Can assign mixed precision to variables given bound on output error
- Generates C++ implementation
- Compatible with Xilinx Vivado

Daisy Darulova et al. 2018
- Can assign mixed precision to variables given bound on output error
- Generates C++ implementation
- Compatible with Xilinx Vivado

NN Synthesis: Mixed Precision Tuning

Daisy Darulova et al. 2018

- Can assign mixed precision to variables given bound on output error
- Generates C++ implementation
- Compatible with Xilinx Vivado

Problem

No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)

Daisy Darulova et al. 2018
Can assign mixed precision to variables given bound on output error
Generates C++ implementation
Compatible with Xilinx Vivado
Problem
No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)

NN Synthesis: Mixed Precision Tuning

Daisy Darulova et al. 2018

- Can assign mixed precision to variables given bound on output error
- Generates C++ implementation
- Compatible with Xilinx Vivado

Problem

No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)

Real-Valued NN → **Preprocessing** Loop Unrolling

Daisy Darulova et al. 2018
- Can assign mixed precision to variables given bound on output error
- Generates C++ implementation
- Compatible with Xilinx Vivado

Problem
No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)

Real-Valued NN → **Preprocessing** Loop Unrolling

NN Synthesis: Mixed Precision Tuning

Daisy Darulova et al. 2018

- Can assign mixed precision to variables given bound on output error
- Generates C++ implementation
- Compatible with Xilinx Vivado

Problem

No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)

Daisy Darulova et al. 2018
Can assign mixed precision to variables given bound on output error
Generates C++ implementation
Compatible with Xilinx Vivado
Problem
No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)
Real-Valued NN Preprocessing Loop Unrolling Daisy (sound analysis)

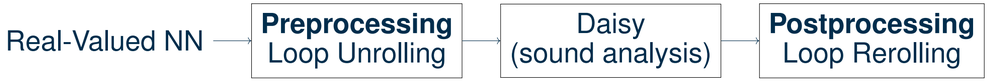
NN Synthesis: Mixed Precision Tuning

Daisy Darulova et al. 2018

- Can assign mixed precision to variables given bound on output error
- Generates C++ implementation
- Compatible with Xilinx Vivado

Problem

No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)



Daisy Darulova et al. 2018
Can assign mixed precision to variables given bound on output error
Generates C++ implementation
Compatible with Xilinx Vivado
Problem
No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)
Real-Valued NN Preprocessing Loop Unrolling Daisy (sound analysis) Postprocessing Loop Rerolling

NN Synthesis: Mixed Precision Tuning

Daisy Darulova et al. 2018

- Can assign mixed precision to variables given bound on output error
- Generates C++ implementation
- Compatible with Xilinx Vivado

Problem

No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)

Daisy Darulova et al. 2018
- Can assign mixed precision to variables given bound on output error
- Generates C++ implementation
- Compatible with Xilinx Vivado

Problem
No support for complex datastructures
(e.g. Matrix-Vector Product, Dot Product, etc.)

Real-Valued NN → Preprocessing Loop Unrolling → Daisy (sound analysis) → Postprocessing Loop Rerolling → Quantized NN (FPGA)

Evaluation

- Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)

■ Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)

Evaluation

- Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
- 3 Case Studies:

■ Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
■ 3 Case Studies:

Evaluation

- Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
- 3 Case Studies:
  - Adaptive Cruise Control (cont. actions)
    2 inputs / 256 ReLU nodes / approx. 12k parameters
  - Adaptive Cruise Control (discrete actions)
    (same architecture)

NeurIPS'24

- Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
- 3 Case Studies:
  - Adaptive Cruise Control (cont. actions)
    2 inputs / 256 ReLU nodes / approx. 12k parameters
  - Adaptive Cruise Control (discrete actions)
    (same architecture)

NeurIPS'24

Evaluation

- Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
- 3 Case Studies:
  - Adaptive Cruise Control (cont. actions) — NeurIPS'24
    2 inputs / 256 ReLU nodes / approx. 12k parameters
  - Adaptive Cruise Control (discrete actions)
    (same architecture)
  - Vertical Airborne Collision Avoidance — Julian et al. 2019
    5 inputs / 270 ReLU nodes / approx. 11k parameters

- Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
- 3 Case Studies:
  - Adaptive Cruise Control (cont. actions) — NeurIPS'24
    2 inputs / 256 ReLU nodes / approx. 12k parameters
  - Adaptive Cruise Control (discrete actions)
    (same architecture)
  - Vertical Airborne Collision Avoidance — Julian et al. 2019
    5 inputs / 270 ReLU nodes / approx. 11k parameters

Evaluation

- Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
- 3 Case Studies:
  - Adaptive Cruise Control (cont. actions) — NeurIPS'24
    2 inputs / 256 ReLU nodes / approx. 12k parameters
  - Adaptive Cruise Control (discrete actions)
    (same architecture)
  - Vertical Airborne Collision Avoidance — Julian et al. 2019
    5 inputs / 270 ReLU nodes / approx. 11k parameters

Verification & Synthesis Results

Case Study	δ	NN Verification		NN Synthesis	
		Time ($\mathbb{R}$)	Time (δ)	Time	# Cycles
ACC (cont.)	1.0	2.02m	9.3m	1.53h	567
ACC (disc.)	0.01	59s	1m	2.01h	579
VCAS DNC	25^{-3}	18.92m	27.45m	1.73h	655
DND	25^{-3}	15.75m	19.1m	1.82h	656

NN Verification:
NCubeV — NeurIPS'24

NN Synthesis:
Daisy — Darulova et al. 2018

- Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
- 3 Case Studies:
  - Adaptive Cruise Control (cont. actions) — NeurIPS'24
    2 inputs / 256 ReLU nodes / approx. 12k parameters
  - Adaptive Cruise Control (discrete actions)
    (same architecture)
  - Vertical Airborne Collision Avoidance — Julian et al. 2019
    5 inputs / 270 ReLU nodes / approx. 11k parameters

Verification & Synthesis Results

Case Study	δ	NN Verification		NN Synthesis	
		Time ($\mathbb{R}$)	Time (δ)	Time	# Cycles
ACC (cont.)	1.0	2.02m	9.3m	1.53h	567
ACC (disc.)	0.01	59s	1m	2.01h	579
VCAS DNC	25^{-3}	18.92m	27.45m	1.73h	655
DND	25^{-3}	15.75m	19.1m	1.82h	656

NN Verification: NCubeV — NeurIPS'24
NN Synthesis: Daisy — Darulova et al. 2018

Evaluation

- Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
- 3 Case Studies:
  - **Adaptive Cruise Control (cont. actions)** NeurIPS'24
    2 inputs / 256 ReLU nodes / approx. 12k parameters
  - Adaptive Cruise Control (discrete actions)
    (same architecture)
  - Vertical Airborne Collision Avoidance Julian et al. 2019
    5 inputs / 270 ReLU nodes / approx. 11k parameters

Verification & Synthesis Results

Case Study	δ	NN Verification		NN Synthesis	
		Time ($\mathbb{R}$)	Time (δ)	Time	# Cycles
ACC (cont.)	1.0	2.02m	9.3m	1.53h	567
ACC (disc.)	0.01	59s	1m	2.01h	579
VCAS DNC	25^{-3}	18.92m	27.45m	1.73h	655
DND	25^{-3}	15.75m	19.1m	1.82h	656

NN Verification:
NCubeV NeurIPS'24

NN Synthesis:
Daisy Darulova et al. 2018

Evaluation of full pipeline ($d\mathcal{L}$ down to Xilinx)
3 Case Studies:
Adaptive Cruise Control (cont. actions) NeurIPS'24
2 inputs / 256 ReLU nodes / approx. 12k parameters
Adaptive Cruise Control (discrete actions)
(same architecture)
Vertical Airborne Collision Avoidance Julian et al. 2019
5 inputs / 270 ReLU nodes / approx. 11k parameters
Verification & Synthesis Results
NN Verification: NCubeV NeurIPS'24
NN Synthesis: Daisy Darulova et al. 2018

Evaluation: Adaptive Cruise Control

- **Input:** Relative Position & Velocity
  Output: Relative Acceleration a_{rel}

■ **Input:** Relative Position & Velocity
Output: Relative Acceleration a_{rel}

Evaluation: Adaptive Cruise Control

- **Input:** Relative Position & Velocity
  Output: Relative Acceleration a_{rel}
- **Angelic Perturbation:**

$$\text{act} \equiv \left(\varepsilon_{a_{\text{rel}}} := *; ? \left(\left|\varepsilon_{a_{\text{rel}}}\right| \leq \delta_{a_{\text{rel}}}\right); a_{\text{rel}} := a_{\text{rel}} + \varepsilon_{a_{\text{rel}}}\right)$$

- **Input:** Relative Position & Velocity
  Output: Relative Acceleration a_{rel}
- **Angelic Perturbation:**

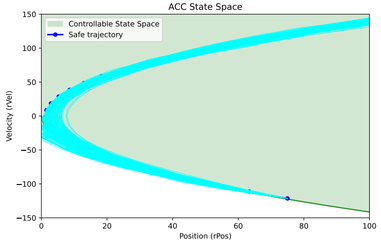
$$\text{act} \equiv \left(\varepsilon_{a_{\text{rel}}} := *; ? \left(\left|\varepsilon_{a_{\text{rel}}}\right| \leq \delta_{a_{\text{rel}}}\right); a_{\text{rel}} := a_{\text{rel}} + \varepsilon_{a_{\text{rel}}}\right)$$

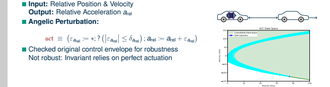
Evaluation: Adaptive Cruise Control

- **Input:** Relative Position & Velocity
  Output: Relative Acceleration a_{rel}
- **Angelic Perturbation:**

$$\text{act} \equiv \left(\varepsilon_{a_{\text{rel}}} := *; ? \left(\left|\varepsilon_{a_{\text{rel}}}\right| \leq \delta_{a_{\text{rel}}}\right); a_{\text{rel}} := a_{\text{rel}} + \varepsilon_{a_{\text{rel}}}\right)$$

- Checked original control envelope for robustness
  Not robust: Invariant relies on perfect actuation



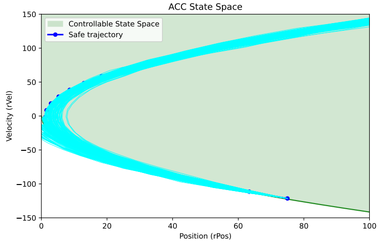


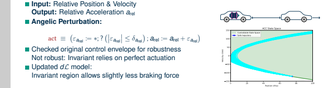
Evaluation: Adaptive Cruise Control

- **Input:** Relative Position & Velocity
  Output: Relative Acceleration a_{rel}
- **Angelic Perturbation:**

$$\text{act} \equiv \left(\varepsilon_{a_{\text{rel}}} := *; ?\left(\left|\varepsilon_{a_{\text{rel}}}\right| \leq \delta_{a_{\text{rel}}}\right); a_{\text{rel}} := a_{\text{rel}} + \varepsilon_{a_{\text{rel}}}\right)$$

- Checked original control envelope for robustness
  Not robust: Invariant relies on perfect actuation
- Updated $d\mathcal{L}$ model:
  Invariant region allows slightly less braking force



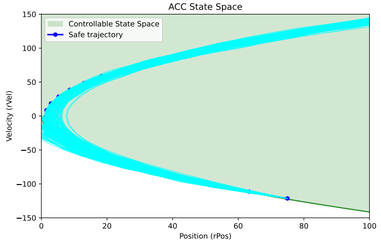


Evaluation: Adaptive Cruise Control

- **Input:** Relative Position & Velocity
  Output: Relative Acceleration a_{rel}
- **Angelic Perturbation:**

$$\text{act} \equiv \left(\varepsilon_{a_{\text{rel}}} := *; ? \left(\left|\varepsilon_{a_{\text{rel}}}\right| \leq \delta_{a_{\text{rel}}}\right); a_{\text{rel}} := a_{\text{rel}} + \varepsilon_{a_{\text{rel}}}\right)$$

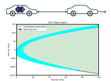
- Checked original control envelope for robustness
  Not robust: Invariant relies on perfect actuation
- Updated $d\mathcal{L}$ model:
  Invariant region allows slightly less braking force
- NN verification w.r.t. $\delta_{a_{\text{rel}}} = 1.0$



- **Input:** Relative Position & Velocity
  Output: Relative Acceleration a_{rel}
- **Angelic Perturbation:**

$$\text{act} \equiv \left(\varepsilon_{a_{\text{rel}}} := *; ? \left(\left|\varepsilon_{a_{\text{rel}}}\right| \leq \delta_{a_{\text{rel}}}\right); a_{\text{rel}} := a_{\text{rel}} + \varepsilon_{a_{\text{rel}}}\right)$$

- Checked original control envelope for robustness
  Not robust: Invariant relies on perfect actuation
- Updated $d\mathcal{L}$ model:
  Invariant region allows slightly less braking force
- NN verification w.r.t. $\delta_{a_{\text{rel}}} = 1.0$

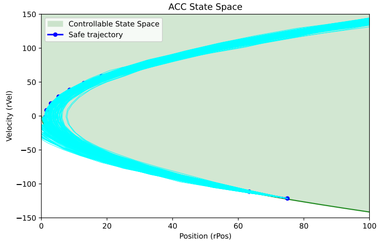


Evaluation: Adaptive Cruise Control

- **Input:** Relative Position & Velocity
  Output: Relative Acceleration a_{rel}
- **Angelic Perturbation:**

$$\text{act} \equiv \left(\varepsilon_{a_{\text{rel}}} := *; ?\left(\left|\varepsilon_{a_{\text{rel}}}\right| \leq \delta_{a_{\text{rel}}}\right); a_{\text{rel}} := a_{\text{rel}} + \varepsilon_{a_{\text{rel}}}\right)$$

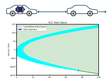
- Checked original control envelope for robustness
  Not robust: Invariant relies on perfect actuation
- Updated $d\mathcal{L}$ model:
  Invariant region allows slightly less braking force
- NN verification w.r.t. $\delta_{a_{\text{rel}}} = 1.0$
- Daisy: Synthesis of mixed-precision NN (1.53h)
  $\Rightarrow$ 3.66% bit savings over safe uniform implementation



- **Input:** Relative Position & Velocity
  Output: Relative Acceleration a_{rel}
- **Angelic Perturbation:**

$$\text{act} \equiv \left(\varepsilon_{a_{\text{rel}}} := *; ?\left(\left|\varepsilon_{a_{\text{rel}}}\right| \leq \delta_{a_{\text{rel}}}\right); a_{\text{rel}} := a_{\text{rel}} + \varepsilon_{a_{\text{rel}}}\right)$$

- Checked original control envelope for robustness
  Not robust: Invariant relies on perfect actuation
- Updated $d\mathcal{L}$ model:
  Invariant region allows slightly less braking force
- NN verification w.r.t. $\delta_{a_{\text{rel}}} = 1.0$
- Daisy: Synthesis of mixed-precision NN (1.53h)
  $\Rightarrow$ 3.66% bit savings over safe uniform implementation



Summary

Infinite-Time Horizon Safety:
From $d\mathcal{L}$ model down to the bit-wise implementation

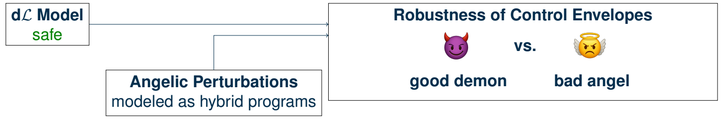
Angelic Perturbations
modeled as hybrid programs

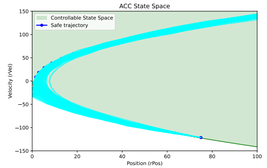
Infinite-Time Horizon Safety:
From $d\mathcal{L}$ model down to the bit-wise implementation

Angelic Perturbations
modeled as hybrid programs

Summary

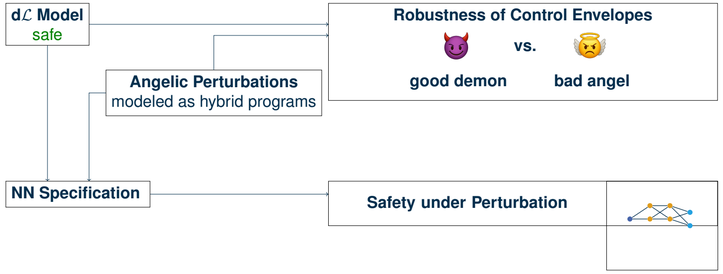
Infinite-Time Horizon Safety:
From $d\mathcal{L}$ model down to the bit-wise implementation

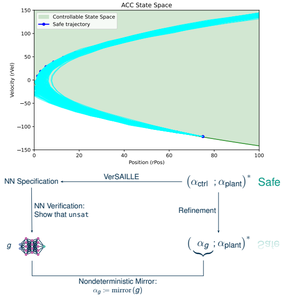


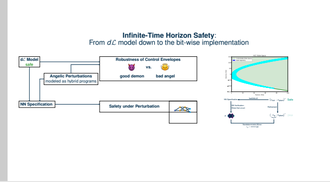


Summary

Infinite-Time Horizon Safety:
From $d\mathcal{L}$ model down to the bit-wise implementation

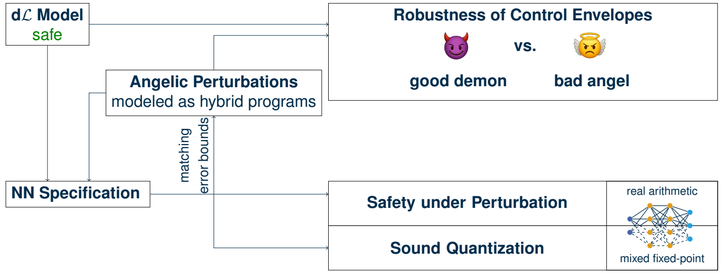


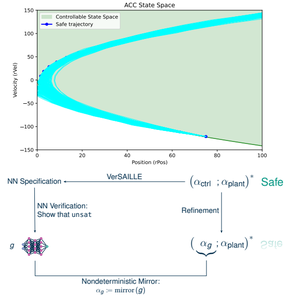


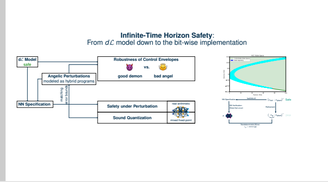


Summary

Infinite-Time Horizon Safety:
From $d\mathcal{L}$ model down to the bit-wise implementation

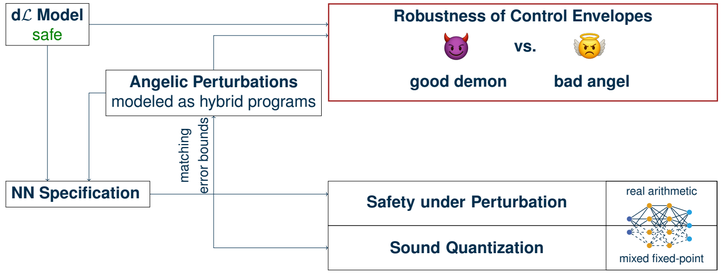


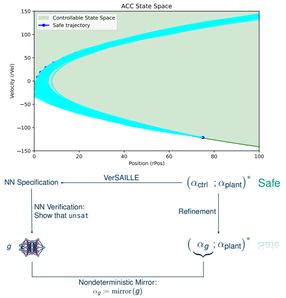


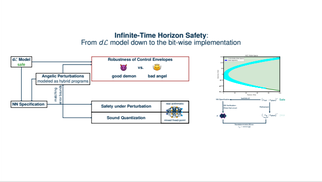


Summary

Infinite-Time Horizon Safety:
From $d\mathcal{L}$ model down to the bit-wise implementation

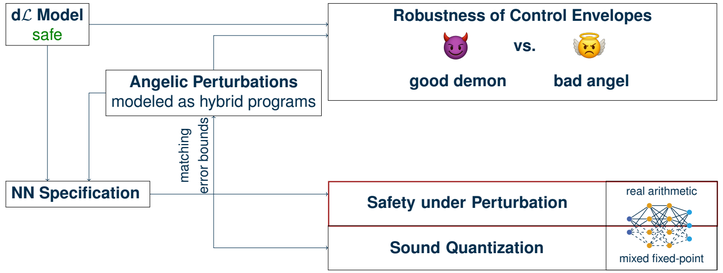


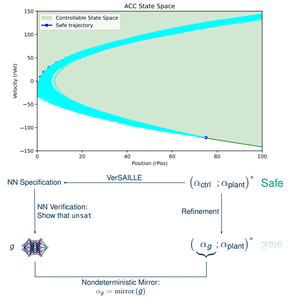


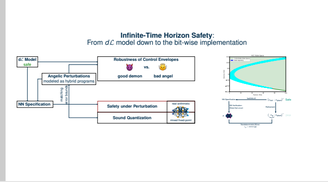


Summary

Infinite-Time Horizon Safety:
From $d\mathcal{L}$ model down to the bit-wise implementation

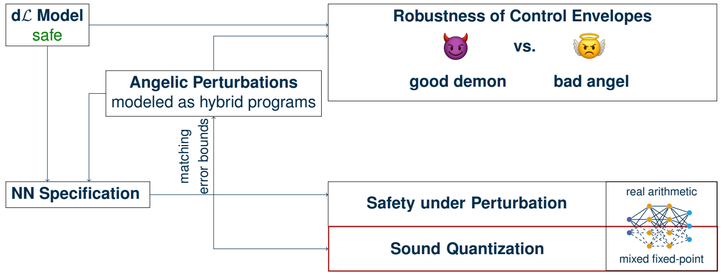


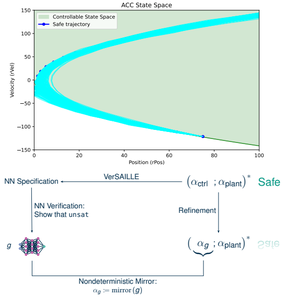




Summary

Infinite-Time Horizon Safety:
From $d\mathcal{L}$ model down to the bit-wise implementation





References I

[1] Eva Darulova, Einar Horn, and Saksham Sharma. “Sound mixed-precision optimization with rewriting”. In: *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2018, Porto, Portugal, April 11-13, 2018*. Ed. by Chris Gill et al. IEEE Computer Society / ACM, 2018, pp. 208–219. DOI: 10.1109/ICCPS.2018.00028. URL: https://doi.org/10.1109/ICCPS.2018.00028.

[2] Kyle D. Julian and Mykel J. Kochenderfer. “Guaranteeing Safety for Neural Network-Based Aircraft Collision Avoidance Systems”. In: *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*. 2019, pp. 1–10. DOI: 10.1109/DASC43569.2019.9081748.

[3] Stefan Mitsch and André Platzer. “ModelPlex: verified runtime validation of verified cyber-physical system models”. In: *Formal Methods Syst. Des.* 49.1-2 (2016), pp. 33–74. DOI: 10.1007/S10703-016-0241-Z. URL: https://doi.org/10.1007/s10703-016-0241-z.

2025-10-06 Of Good Demons and Bad Angels

└References

[1] Eva Darulova, Einar Horn, and Saksham Sharma. “Sound mixed-precision optimization with rewriting”. In: *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2018, Porto, Portugal, April 11-13, 2018*. Ed. by Chris Gill et al. IEEE Computer Society / ACM, 2018, pp. 208–219. DOI: 10.1109/ICCPS.2018.00028. URL: https://doi.org/10.1109/ICCPS.2018.00028.

[2] Kyle D. Julian and Mykel J. Kochenderfer. “Guaranteeing Safety for Neural Network-Based Aircraft Collision Avoidance Systems”. In: *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*. 2019, pp. 1–10. DOI: 10.1109/DASC43569.2019.9081748.

[3] Stefan Mitsch and André Platzer. “ModelPlex: verified runtime validation of verified cyber-physical system models”. In: *Formal Methods Syst. Des.* 49.1-2 (2016), pp. 33–74. DOI: 10.1007/S10703-016-0241-Z. URL: https://doi.org/10.1007/s10703-016-0241-z.

References II

[4] André Platzer. “Differential Dynamic Logic for Hybrid Systems”. In: *J. Autom. Reason.* 41.2 (2008), pp. 143–189. DOI: 10.1007/s10817-008-9103-8. URL: https://doi.org/10.1007/s10817-008-9103-8.

[5] Enguerrand Prebet and André Platzer. “Uniform Substitution for Differential Refinement Logic”. In: *Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part II*. Ed. by Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt. Vol. 14740. Lecture Notes in Computer Science. Springer, 2024, pp. 196–215. DOI: 10.1007/978-3-031-63501-4_11. URL: https://doi.org/10.1007/978-3-031-63501-4%5C_11.

[6] Chelsea Sidrane et al. “OVERT: An algorithm for safety verification of neural network control policies for nonlinear systems”. In: *Journal of Machine Learning Research* 23.117 (2022), pp. 1–45.

[NeurIPS’24] Samuel Teuber, Stefan Mitsch, and André Platzer. “Provably Safe Neural Network Controllers via Differential Dynamic Logic”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Curran Associates, Inc., 2024. URL: https://doi.org/10.48550/arXiv.2402.10998.

2025-10-06 Of Good Demons and Bad Angels

└References

[4] André Platzer. “Differential Dynamic Logic for Hybrid Systems”. In: *J. Autom. Reason.* 41.2 (2008), pp. 143–189. DOI: 10.1007/s10817-008-9103-8. URL: https://doi.org/10.1007/s10817-008-9103-8.

[5] Enguerrand Prebet and André Platzer. “Uniform Substitution for Differential Refinement Logic”. In: *Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part II*. Ed. by Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt. Vol. 14740. Lecture Notes in Computer Science. Springer, 2024, pp. 196–215. DOI: 10.1007/978-3-031-63501-4_11. URL: https://doi.org/10.1007/978-3-031-63501-4%5C_11.

[6] Chelsea Sidrane et al. “OVERT: An algorithm for safety verification of neural network control policies for nonlinear systems”. In: *Journal of Machine Learning Research* 23.117 (2022), pp. 1–45.

[NeurIPS’24] Samuel Teuber, Stefan Mitsch, and André Platzer. “Provably Safe Neural Network Controllers via Differential Dynamic Logic”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson et al. Curran Associates, Inc., 2024. URL: https://doi.org/10.48550/arXiv.2402.10998.

Backup

Backup

ModelPlex

Input: Safe Control Envelope in $d\mathcal{L}$

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[(a_{\text{rel}} := -B; c := 0; \{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = a_{\text{rel}}, c' = 1 \& c \leq T\})^*\right] p_{\text{rel}} > 0$$

Input: Safe Control Envelope in $d\mathcal{L}$

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[(a_{\text{rel}} := -B; c := 0; \{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = a_{\text{rel}}, c' = 1 \& c \leq T\})^*\right] p_{\text{rel}} > 0$$

1. Monitor Formula: If satisfied $\rightarrow$ implementation matches behavior of hybrid program
2. Controller: Usually abstract $\Rightarrow$ envelope
3. More complex programs

ModelPlex

Input: Safe Control Envelope in $d\mathcal{L}$ Controller/Envelope Plant

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[(a_{\text{rel}} := -B; t := 0; \{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = a_{\text{rel}}, t' = 1 \& t \leq T\})^*\right] p_{\text{rel}} > 0$$

Input: Safe Control Envelope in $d\mathcal{L}$ Controller/Envelope Plant

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[(a_{\text{rel}} := -B; t := 0; \{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = a_{\text{rel}}, t' = 1 \& t \leq T\})^*\right] p_{\text{rel}} > 0$$

1. Monitor Formula: If satisfied $\rightarrow$ implementation matches behavior of hybrid program
2. Controller: Usually abstract $\Rightarrow$ envelope
3. More complex programs

ModelPlex

Input: Safe Control Envelope in $d\mathcal{L}$ Controller/Envelope Plant

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[(a_{\text{rel}} := -B; t := 0; \{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = a_{\text{rel}}, t' = 1 \& t \leq T\})^*\right] p_{\text{rel}} > 0$$

ModelPlex creates a **controller monitor formula**:

$$a^+_{\text{rel}} = -B$$

Satisfaction during concrete run implies correct controller behavior.
Equally applicable for more complicated controllers (nondeterminism, conditions, ...)

[Mitsch et al. 2016]

Input: Safe Control Envelope in $d\mathcal{L}$ Controller/Envelope Plant

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[(a_{\text{rel}} := -B; t := 0; \{p'_{\text{rel}} = v_{\text{rel}}, v'_{\text{rel}} = a_{\text{rel}}, t' = 1 \& t \leq T\})^*\right] p_{\text{rel}} > 0$$

ModelPlex creates a **controller monitor formula**:

$$a^+_{\text{rel}} = -B$$

Satisfaction during concrete run implies correct controller behavior.
Equally applicable for more complicated controllers (nondeterminism, conditions, ...)

[Mitsch et al. 2016]

1. Monitor Formula: If satisfied → implementation matches behavior of hybrid program
2. Controller: Usually abstract ⇒ envelope
3. More complex programs

ModelPlex

Input: Safe Control Envelope in $d\mathcal{L}$

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[\left(a_{\text{rel}} := -B \cup \left(a_{\text{rel}} := *; ?\left(|a_{\text{rel}}| \leq B \wedge p_{\text{rel}} > 10^3\right)\right);\ \ldots\right)^*\right] p_{\text{rel}} > 0$$

ModelPlex creates a **controller monitor formula**:

Satisfaction during concrete run implies correct controller behavior.
Equally applicable for more complicated controllers (nondeterminism, conditions, ...)

[Mitsch et al. 2016]

Input: Safe Control Envelope in $d\mathcal{L}$

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[\left(a_{\text{rel}} := -B \cup \left(a_{\text{rel}} := *; ?\left(|a_{\text{rel}}| \leq B \wedge p_{\text{rel}} > 10^3\right)\right);\ \ldots\right)^*\right] p_{\text{rel}} > 0$$

ModelPlex creates a **controller monitor formula**:

Satisfaction during concrete run implies correct controller behavior.
Equally applicable for more complicated controllers (nondeterminism, conditions, ...)

[Mitsch et al. 2016]

1. Monitor Formula: If satisfied → implementation matches behavior of hybrid program
2. Controller: Usually abstract ⇒ envelope
3. More complex programs

ModelPlex

Input: Safe Control Envelope in $d\mathcal{L}$

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[\left(a_{\text{rel}} := -B \cup \left(a_{\text{rel}} := *; ?\left(|a_{\text{rel}}| \leq B \wedge p_{\text{rel}} > 10^3\right)\right); \ \ldots\right)^*\right] p_{\text{rel}} > 0$$

ModelPlex creates a **controller monitor formula**:

$$a_{\text{rel}}^+ = -B \vee \left|a_{\text{rel}}^+\right| \leq B \wedge p_{\text{rel}} > 10^3$$

Satisfaction during concrete run implies correct controller behavior.
Equally applicable for more complicated controllers (nondeterminism, conditions, ...)

[Mitsch et al. 2016]

Input: Safe Control Envelope in $d\mathcal{L}$

$$p_{\text{rel}} > \frac{v_{\text{rel}}^2}{2B} \rightarrow \left[\left(a_{\text{rel}} := -B \cup \left(a_{\text{rel}} := *; ?\left(|a_{\text{rel}}| \leq B \wedge p_{\text{rel}} > 10^3\right)\right); \ \ldots\right)^*\right] p_{\text{rel}} > 0$$

ModelPlex creates a **controller monitor formula**:

$$a_{\text{rel}}^+ = -B \vee \left|a_{\text{rel}}^+\right| \leq B \wedge p_{\text{rel}} > 10^3$$

Satisfaction during concrete run implies correct controller behavior.
Equally applicable for more complicated controllers (nondeterminism, conditions, ...)

[Mitsch et al. 2016]

1. Monitor Formula: If satisfied → implementation matches behavior of hybrid program
2. Controller: Usually abstract ⇒ envelope
3. More complex programs

Verifiably Safe AI via Logically Linked Envelopes

Theorem (Soundness)

Assume:

- *g is a (piece-wise Noetherian) neural network*
- *C* $\equiv \left(\phi \rightarrow \left[(\alpha_{ctrl}; \alpha_{plant})^*\right] \psi\right)$ *is a valid d$\mathcal{L}$ contract*
- $\mathrm{controller}$ *is a controller monitor (ModelPlex)*
- $\mathrm{invariant}$ *is an inductive invariant*

Theorem (Soundness)
Assume:
- *g is a (piece-wise Noetherian) neural network*
- *C* $\equiv \left(\phi \rightarrow \left[(\alpha_{ctrl};\alpha_{plant})^*\right] \psi\right)$ *is a valid d$\mathcal{L}$ contract*
- $\mathrm{controller}$ *is a controller monitor (ModelPlex)*
- $\mathrm{invariant}$ *is an inductive invariant*

1. Piece-Wise Noetherian: e.g. also sigmoid

Verifiably Safe AI via Logically Linked Envelopes

Theorem (Soundness)

Assume:

- *g is a (piece-wise Noetherian) neural network*
- *C* $\equiv \left(\phi \rightarrow \left[(\alpha_{ctrl}; \alpha_{plant})^*\right] \psi\right)$ *is a valid d$\mathcal{L}$ contract*
- $\mathrm{controller}$ *is a controller monitor (ModelPlex)*
- $\mathrm{invariant}$ *is an inductive invariant*

If an NN Verifier returns `unsat` *for the query* $p \equiv (\mathrm{invariant} \wedge \neg \mathrm{controller})$ *on g*

Theorem (Soundness)
Assume:
- g is a (piece-wise Noetherian) neural network
- C ≡ (φ → [(α_ctrl; α_plant)*] ψ) is a valid d$\mathcal{L}$ contract
- controller is a controller monitor (ModelPlex)
- invariant is an inductive invariant

If an NN Verifier returns unsat for the query p ≡ (invariant ∧ ¬controller) on g

1. Piece-Wise Noetherian: e.g. also sigmoid

Verifiably Safe AI via Logically Linked Envelopes

Theorem (Soundness)

Assume:

- *g is a (piece-wise Noetherian) neural network*
- *C* $\equiv \left(\phi \rightarrow \left[\left(\alpha_{ctrl}; \alpha_{plant}\right)^*\right] \psi\right)$ *is a valid d$\mathcal{L}$ contract*
- $\mathrm{controller}$ *is a controller monitor (ModelPlex)*
- $\mathrm{invariant}$ *is an inductive invariant*

If an NN Verifier returns `unsat` *for the query* $p \equiv (\mathrm{invariant} \wedge \neg \mathrm{controller})$ *on g*
Then $\phi \rightarrow \left[\left(\alpha_g; \alpha_{plant}\right)^*\right] \psi$ *is valid.*

Theorem (Soundness)
Assume:
- g is a (piece-wise Noetherian) neural network
- C ≡ (φ → [(α_ctrl; α_plant)*] ψ) is a valid d$\mathcal{L}$ contract
- controller is a controller monitor (ModelPlex)
- invariant is an inductive invariant

If an NN Verifier returns unsat for the query p ≡ (invariant ∧ ¬controller) on g
Then φ → [(α_g; α_plant)*] ψ is valid.

1. Piece-Wise Noetherian: e.g. also sigmoid

Verifiably Safe AI via Logically Linked Envelopes

Theorem (Soundness)

Assume:

- *g is a (piece-wise Noetherian) neural network*
- *C* $\equiv \left(\phi \rightarrow \left[\left(\alpha_{ctrl}; \alpha_{plant}\right)^*\right] \psi\right)$ *is a valid d$\mathcal{L}$ contract*
- $\mathrm{controller}$ *is a controller monitor (ModelPlex)*
- $\mathrm{invariant}$ *is an inductive invariant*

If an NN Verifier returns `unsat` *for the query* $p \equiv (\mathrm{invariant} \land \neg \mathrm{controller})$ *on g*
Then $\phi \rightarrow \left[\left(\alpha_g; \alpha_{plant}\right)^*\right] \psi$ *is valid.*

How can we verify the property $\mathrm{invariant} \land \neg \mathrm{controller}$ in practice?

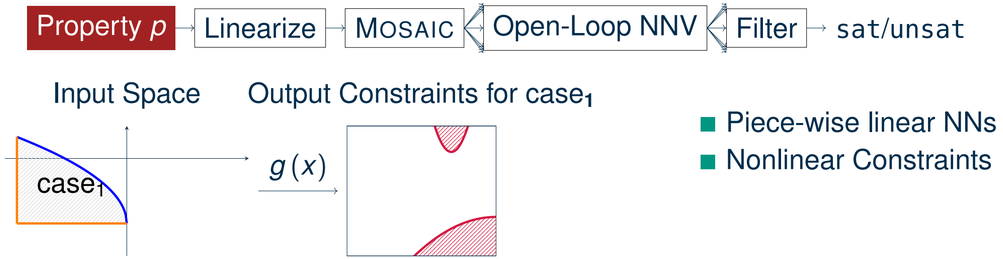
Theorem (Soundness)
Assume:
- g is a (piece-wise Noetherian) neural network
- C ≡ (φ → [(α_ctrl; α_plant)*] ψ) is a valid d$\mathcal{L}$ contract
- controller is a controller monitor (ModelPlex)
- invariant is an inductive invariant

If an NN Verifier returns unsat for the query p ≡ (invariant ∧ ¬controller) on g
Then φ → [(α_g; α_plant)*] ψ is valid.

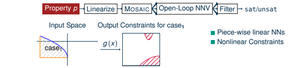
How can we verify the property invariant ∧ ¬controller in practice?

1. Piece-Wise Noetherian: e.g. also sigmoid

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

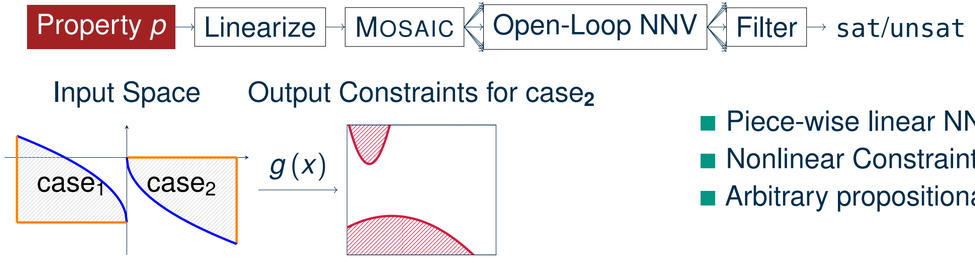


- Piece-wise linear NNs
- Nonlinear Constraints

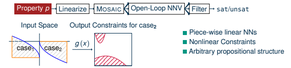


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

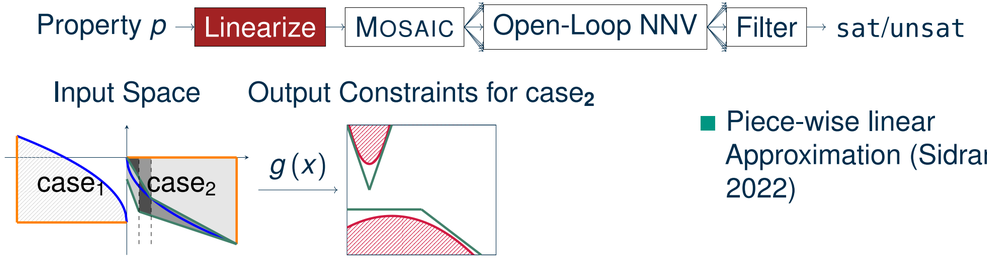


- Piece-wise linear NNs
- Nonlinear Constraints
- Arbitrary propositional structure

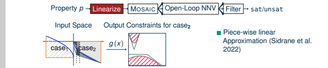


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

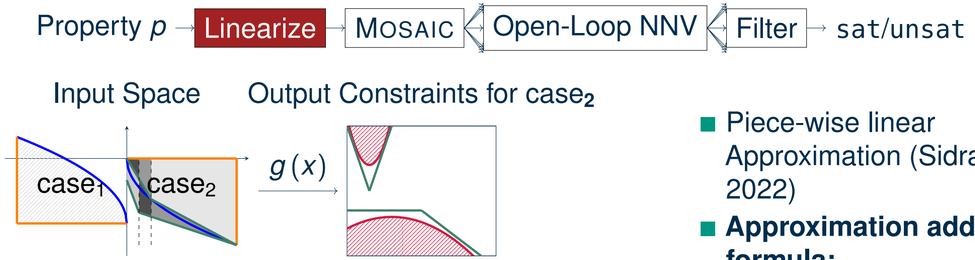


- Piece-wise linear Approximation (Sidrane et al. 2022)



1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

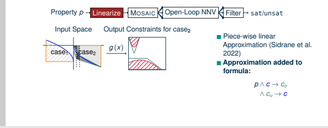


- Piece-wise linear Approximation (Sidrane et al. 2022)
- **Approximation added to formula:**

$$p \wedge c \rightarrow c_o$$
$$\wedge\, c_u \rightarrow c$$

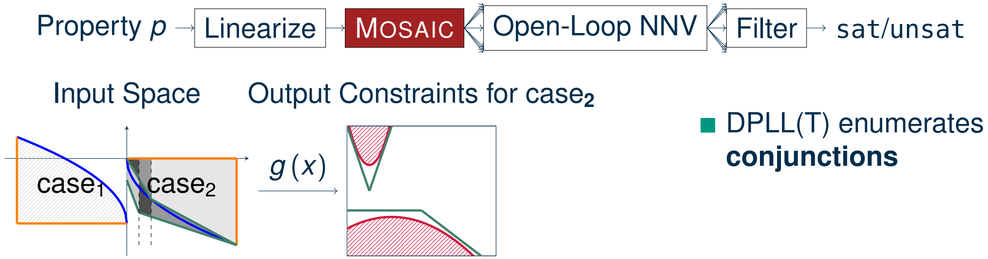
References

Backup

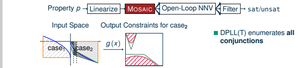


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

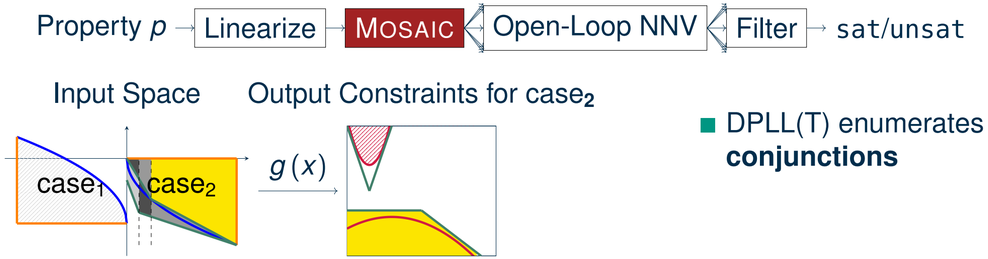


- DPLL(T) enumerates **all conjunctions**

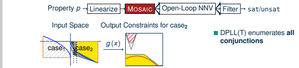


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

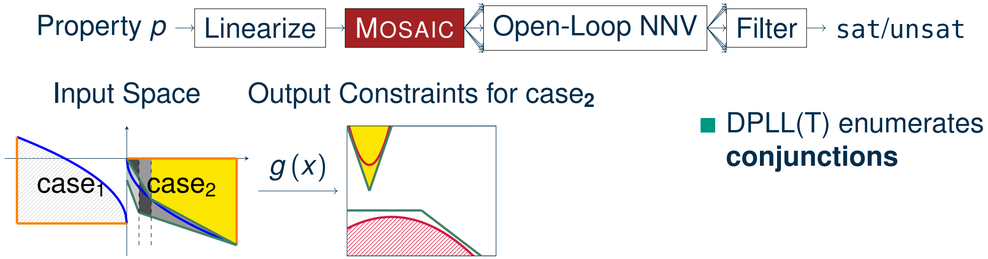


- DPLL(T) enumerates **all conjunctions**

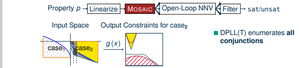


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

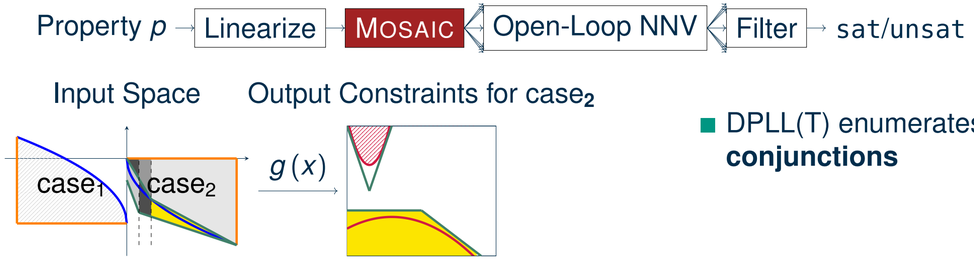


- DPLL(T) enumerates **all conjunctions**

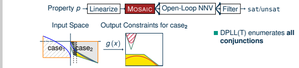


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

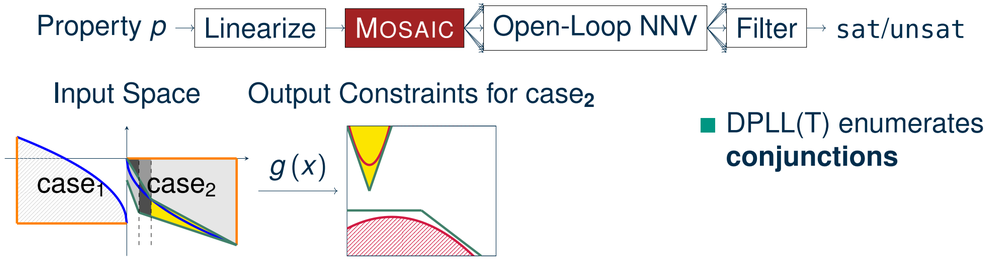


- DPLL(T) enumerates **all conjunctions**



1. Piece-wise linear NNs!

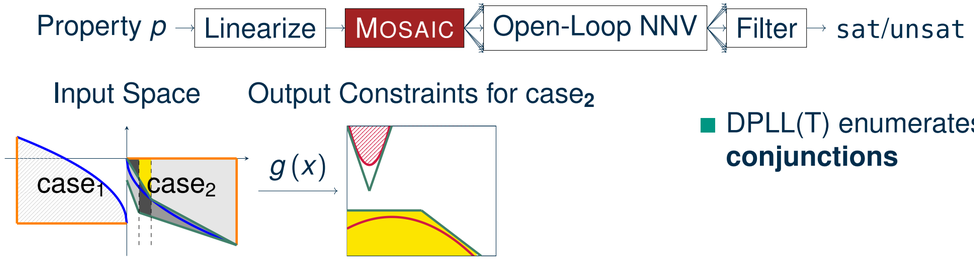
Mosaic: Efficient and Complete NN Verification for Nonlinear Properties



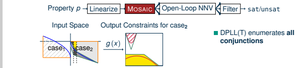
- DPLL(T) enumerates **all conjunctions**

1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

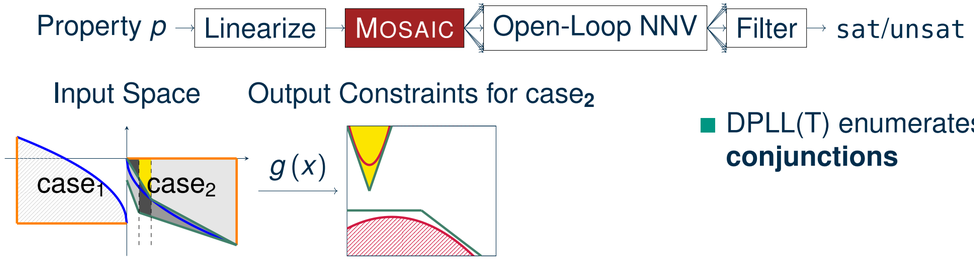


- DPLL(T) enumerates **all conjunctions**



1. Piece-wise linear NNs!

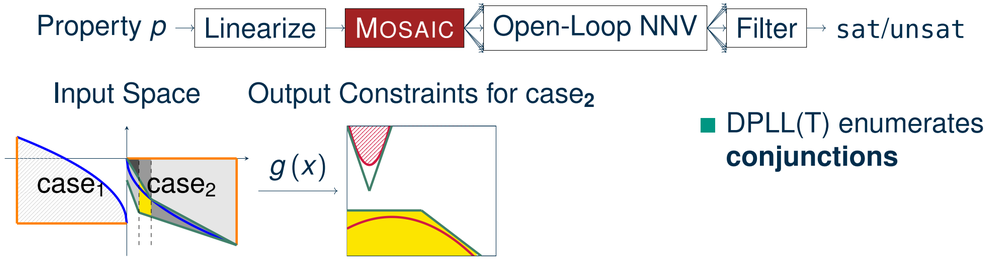
Mosaic: Efficient and Complete NN Verification for Nonlinear Properties



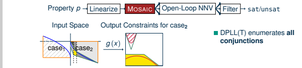
- DPLL(T) enumerates **all conjunctions**

1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

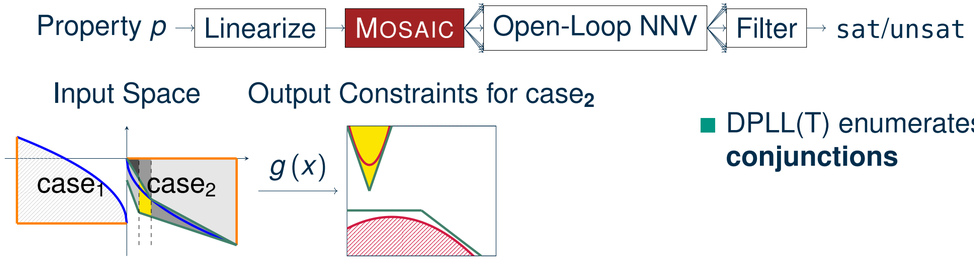


- DPLL(T) enumerates **all conjunctions**

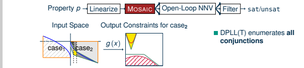


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

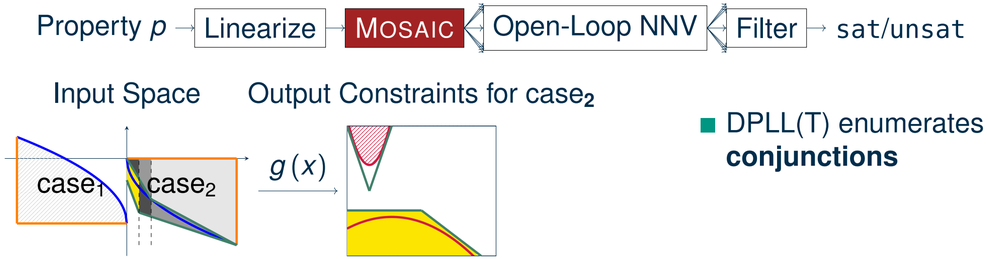


- DPLL(T) enumerates **all conjunctions**

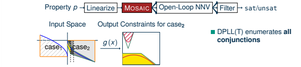


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

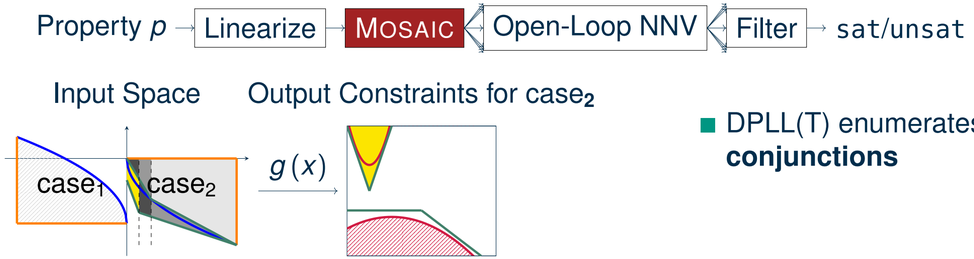


- DPLL(T) enumerates **all conjunctions**

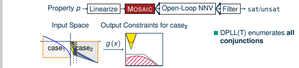


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

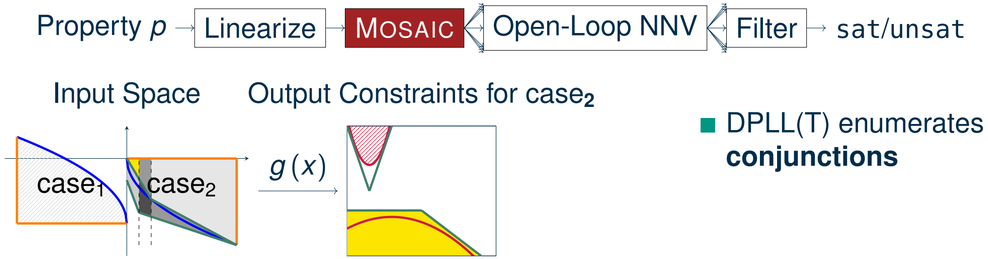


- DPLL(T) enumerates **all conjunctions**

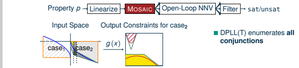


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

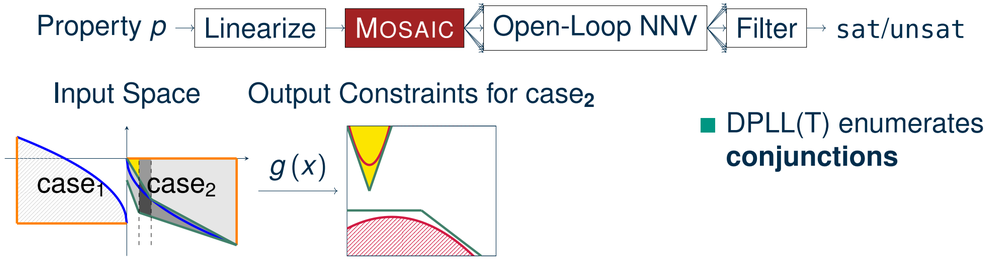


- DPLL(T) enumerates **all conjunctions**

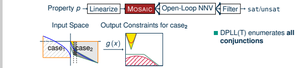


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

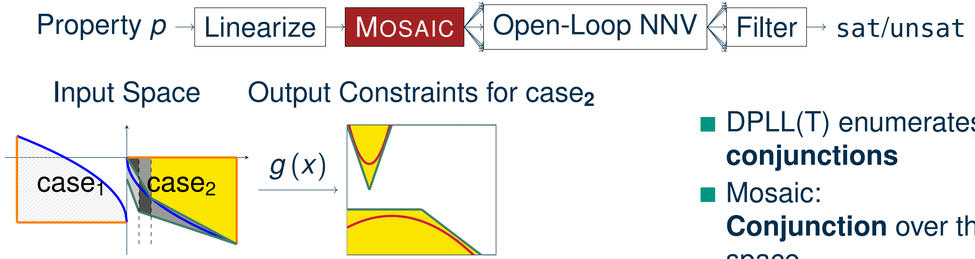


- DPLL(T) enumerates **all conjunctions**



1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

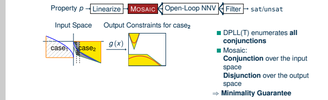


- DPLL(T) enumerates **all conjunctions**
- Mosaic:
  Conjunction over the input space
  Disjunction over the output space

⇒ **Minimality Guarantee**

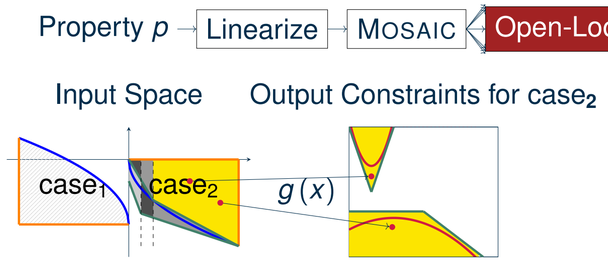
References

Backup

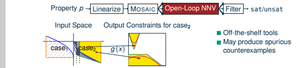


1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties

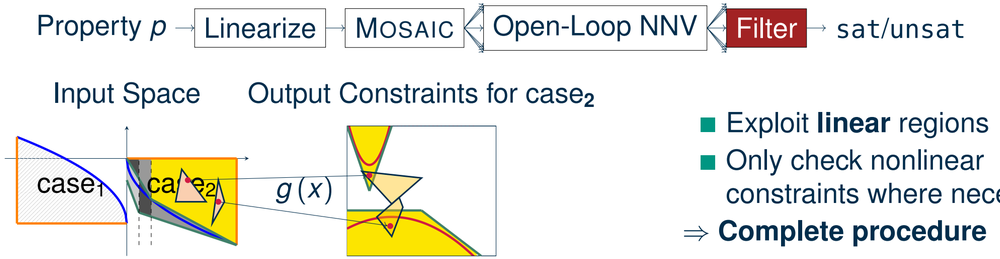


- Off-the-shelf tools
- May produce spurious counterexamples



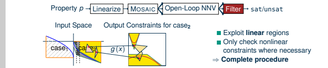
1. Piece-wise linear NNs!

Mosaic: Efficient and Complete NN Verification for Nonlinear Properties



- Exploit **linear** regions
- Only check nonlinear constraints where necessary

$\Rightarrow$ **Complete procedure**



1. Piece-wise linear NNs!

Complexity of Mosaic

We assume:
- An NN with N ReLU nodes
- A property with M atomic constraints and I input variables

Then we get the following complexities:
- M atomic constraints: $\mathcal{O}\left(2^M\right)$ NNV queries
- Naive encoding via SMT for N ReLU nodes: $\mathcal{O}\left(2^{2^{N+I}}\right)$
- With Mosaic: $\mathcal{O}\left(2^{N+2^I}\right)$

Overall:

$$\mathcal{O}\left(2^{M+2^{N+I}}\right) \text{ vs. } \mathcal{O}\left(2^{M+N+2^I}\right)$$

We assume:
- An NN with N ReLU nodes
- A property with M atomic constraints and I input variables

Then we get the following complexities:
- M atomic constraints: $\mathcal{O}\left(2^M\right)$ NNV queries
- Naive encoding via SMT for N ReLU nodes: $\mathcal{O}\left(2^{2^{N+I}}\right)$
- With Mosaic: $\mathcal{O}\left(2^{N+2^I}\right)$

Overall:

$\mathcal{O}\left(2^{M+2^{N+I}}\right)$ vs. $\mathcal{O}\left(2^{M+N+2^I}\right)$

Evaluation: Vertical Airborne Collision Avoidance

Prev. Adv.	Status	Time	CE regions	First CE
DNC	**safe**	0.35 h	—	—
DND	**safe**	0.28 h	—	—
DES1500	**unsafe**	5.45 h	49,428	0.04 h
CL1500	**unsafe**	5.18 h	34,658	0.08 h
SDES1500	**unsafe**	4.05 h	5,360	0.97 h
SCL1500	**unsafe**	4.89 h	11,323	0.36 h
SDES2500	**unsafe**	3.66 h	5,259	1.39 h
SCL2500	**unsafe**	4.45 h	7,846	0.53 h

Prev. Adv.	Status	Time	CE regions	First CE
DNC	**safe**	0.35 h	—	—
DND	**safe**	0.28 h	—	—
DES1500	**unsafe**	5.45 h	49,428	0.04 h
CL1500	**unsafe**	5.18 h	34,658	0.08 h
SDES1500	**unsafe**	4.05 h	5,360	0.97 h
SCL1500	**unsafe**	4.89 h	11,323	0.36 h
SDES2500	**unsafe**	3.66 h	5,259	1.39 h
SCL2500	**unsafe**	4.45 h	7,846	0.53 h

Evaluation: Comparison with SMT

Evaluated on two Adaptive Cruise Control properties (one satisfiable, one unsatisfiable).
NNs with 256 ReLU nodes.

Tool	`ACC_Large`		`ACC_Large` retrained	
	Status	Time	Status	Time
Mathematica	MO	—	MO	—
dReal	TO	—	TO	—
Z3	unknown	510s	unknown	1793s
Z3++	unknown	2550s	unknown	2269s
cvc5	TO	—	TO	—
MathSAT	TO	—	TO	—
ours	**sat**	**87s**	**unsat**	**124s**

Evaluated on two Adaptive Cruise Control properties (one satisfiable, one unsatisfiable).
NNs with 256 ReLU nodes.

Tool	`ACC_Large`		`ACC_Large` retrained	
	Status	Time	Status	Time
Mathematica	MO	—	MO	—
dReal	TO	—	TO	—
Z3	unknown	510s	unknown	1793s
Z3++	unknown	2550s	unknown	2269s
cvc5	TO	—	TO	—
MathSAT	TO	—	TO	—
ours	**sat**	**87s**	**unsat**	**124s**

Evaluation: Comparison with Closed-Loop Techniques

Attempt to prove bounded safety on part of Adaptive Cruise Control NN:

Tool	Nonlinearities	Evaluated Configurations	Time (s)	Share of State Space	Result
NNV	no	4	711	0.009%	**safe for 0.1s**
JuliaReach	no	4	—	0.009%	unknown
CORA	yes	10	—	0.009%	unknown
POLAR	poly. Zono.	12	—	0.009%	unknown
ours	polynomial	1	**124**	100.000%	**safe for** ∞

Attempt to prove bounded safety on part of Adaptive Cruise Control NN:

Tool	Nonlinearities	Evaluated Configurations	Time (s)	Share of State Space	Result
NNV	no	4	711	0.009%	**safe for 0.1s**
JuliaReach	no	4	—	0.009%	unknown
CORA	yes	10	—	0.009%	unknown
POLAR	poly. Zono.	12	—	0.009%	unknown
ours	polynomial	1	**124**	100.000%	**safe for** ∞

Evaluation: Comparison with Closed-Loop Techniques

Attempt to prove bounded safety on part of Adaptive Cruise Control NN:

Tool	Nonlinearities	Evaluated Configurations	Time (s)	Share of State Space	Result
NNV	no	4	711	0.009%	**safe for 0.1s**
JuliaReach	no	4	—	0.009%	unknown
CORA	yes	10	—	0.009%	unknown
POLAR	poly. Zono.	12	—	0.009%	unknown
ours	polynomial	1	**124**	100.000%	**safe for** ∞

Infinte-time horizon: *k*-induction?

Conceptual comparison to NNV

- Attempted to show invariance w.r.t. nonlinear loop invariant
- Overapproximation **can lead to wrong results!**

Attempt to prove bounded safety on part of Adaptive Cruise Control NN:
Tool | Nonlinearities | Evaluated Configurations | Time (s) | Share of State Space | Result
NNV | no | 4 | 711 | 0.009% | safe for 0.1s
JuliaReach | no | 4 | — | 0.009% | unknown
CORA | yes | 10 | — | 0.009% | unknown
POLAR | poly. Zono. | 12 | — | 0.009% | unknown
ours | polynomial | 1 | 124 | 100.000% | safe for ∞
Infinte-time horizon: k-induction?
Conceptual comparison to NNV
Attempted to show invariance w.r.t. nonlinear loop invariant
Overapproximation can lead to wrong results!

Evaluation: Performance of Mosaic

Comparison to DNNV

- DNNV: No support for nonlinear properties
  Enumerates all **propositionally feasible assumptions**

Comparison to DNNV
- DNNV: No support for nonlinear properties
  Enumerates all **propositionally feasible assumptions**

Evaluation: Performance of Mosaic

Comparison to DNNV

- DNNV: No support for nonlinear properties
  Enumerates all **propositionally feasible assumptions**
- Compare efficiency by counting

Comparison to DNNV
- DNNV: No support for nonlinear properties
  Enumerates all **propositionally feasible assumptions**
- Compare efficiency by counting

Evaluation: Performance of Mosaic

Comparison to DNNV

- DNNV: No support for nonlinear properties
  Enumerates all **propositionally feasible assumptions**
- Compare efficiency by counting

Property	# Conjunctions	**# Queries**	# Feasible Conjunctions	# SMT calls
ACC	2.4k	20	86	261
ACC (Fallback)	5.1k	15	72	235
ACAS (DNC)	117.5M	1.7k	9.9k	11.4k
ACAS (DND)	88.9M	1.8k	10.4k	12.0k
ACAS (DES1500)	451.3B	12.5k	58.8k	66.4k
ACAS (CLI1500)	374.4B	13.1k	62.5k	70.4k
ACAS (SDES1500)	9.1T	18.6k	64.1k	75.8k
ACAS (SCLI1500)	18.2T	21.8k	76.0k	88.5k
ACAS (SDES2500)	39.0T	19.0k	66.7k	78.5k
ACAS (SCLI2500)	19.4T	18.6k	67.7k	79.8k

References

Backup

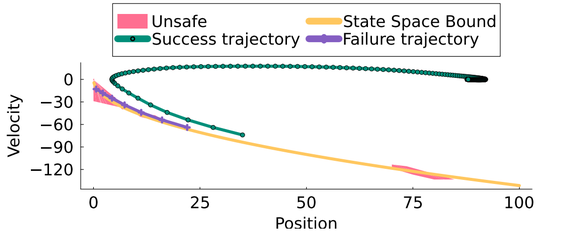
Comparison to DNNV
DNNV: No support for nonlinear properties
Enumerates all propositionally feasible assumptions
Compare efficiency by counting
Property | # Conjunctions | # Queries | # Feasible Conjunctions | # SMT calls
ACC | 2.4k | 20 | 86 | 261
ACC (Fallback) | 5.1k | 15 | 72 | 235
ACAS (DNC) | 117.5M | 1.7k | 9.9k | 11.4k
ACAS (DND) | 88.9M | 1.8k | 10.4k | 12.0k
ACAS (DES1500) | 451.3B | 12.5k | 58.8k | 66.4k
ACAS (CLI1500) | 374.4B | 13.1k | 62.5k | 70.4k
ACAS (SDES1500) | 9.1T | 18.6k | 64.1k | 75.8k
ACAS (SCLI1500) | 18.2T | 21.8k | 76.0k | 88.5k
ACAS (SDES2500) | 39.0T | 19.0k | 66.7k | 78.5k
ACAS (SCLI2500) | 19.4T | 18.6k | 67.7k | 79.8k

Evaluation: Adaptive Cruise Control

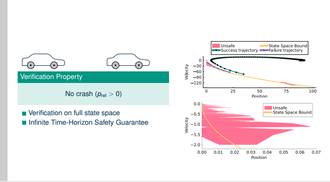
Verification Property

No crash ($p_{\text{rel}} > 0$)

- Verification on full state space
- Infinite Time-Horizon Safety Guarantee



References

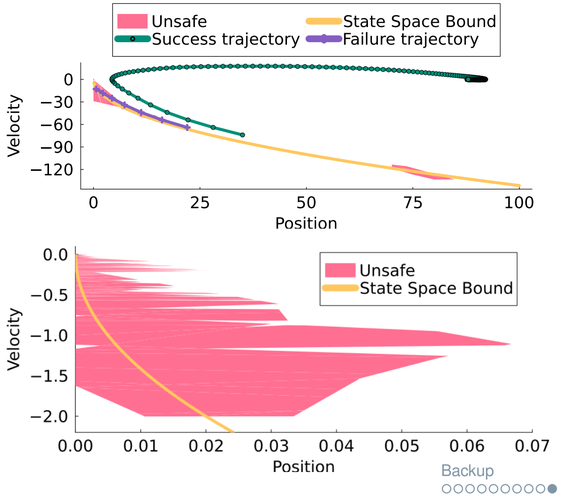


Evaluation: Adaptive Cruise Control

Verification Property

No crash ($p_{\text{rel}} > 0$)

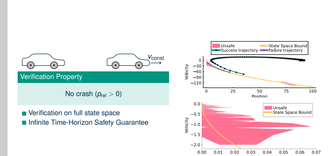
- Verification on full state space
- Infinite Time-Horizon Safety Guarantee



References

KIT

Backup

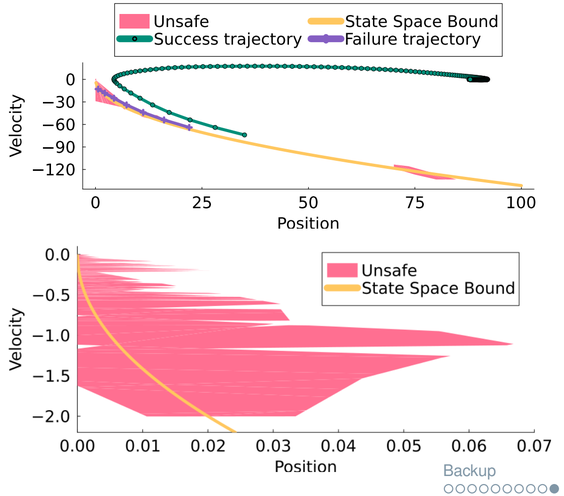


Evaluation: Adaptive Cruise Control

Verification Property

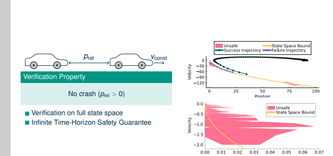
No crash ($p_{\text{rel}} > 0$)

- Verification on full state space
- Infinite Time-Horizon Safety Guarantee



References

KIT

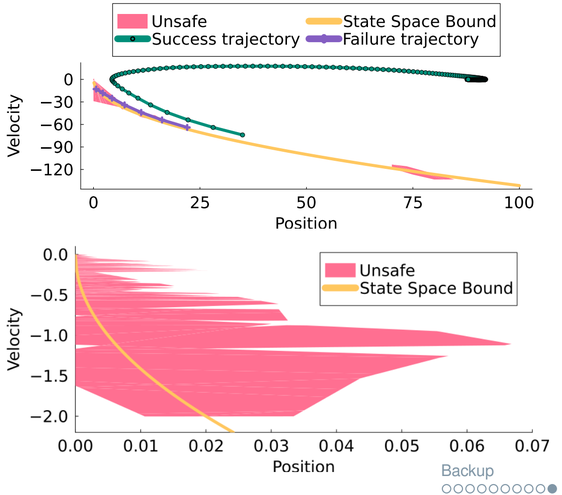


Evaluation: Adaptive Cruise Control

Verification Property

No crash ($p_{\text{rel}} > 0$)

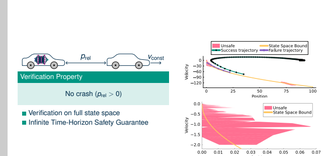
- Verification on full state space
- Infinite Time-Horizon Safety Guarantee



References

KIT

Backup

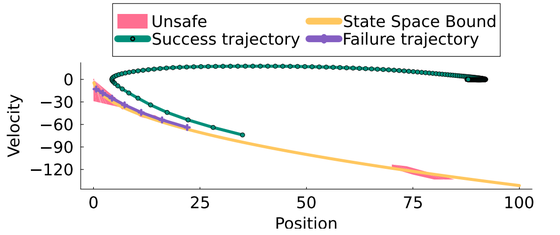


Evaluation: Adaptive Cruise Control

Verification Property

No crash ($p_{\text{rel}} > 0$)

- Verification on full state space
- Infinite Time-Horizon Safety Guarantee
- Some experiments on retraining with counterexample regions



References

KIT