Formal Synthesis with Neural Templates - *Logic meets Learning*

A. Abate, with

D. Ahmed, A. Edwards, M. Giacobbe, A. Peruffo, H. Punchihewa, D. Roy

Department of Computer Science, University of Oxford

www.oxcav.org

April 2023

Outline

Outline

Motivations

- stability, reachability (vs. safety/invariance) of autonomous models

$$\dot{x} = f(x)$$

- termination, assertion violation of SW programs

$$\texttt{while g(x), } x^+ \texttt{ := f(x)}$$

- non-linear (e.g., polynomial) dynamics/updates
  stochastic, controlled models

Motivations

- stability, reachability (vs. safety/invariance) of autonomous models

$$\dot{x} = f(x)$$

- termination, assertion violation of SW programs

$$\texttt{while g(x), } x^{+} \texttt{ := f(x)}$$

- non-linear (e.g., polynomial) dynamics/updates
  stochastic, controlled models

- use of sufficient approaches: synthesis of certificates, e.g.
  Lyapunov functs $V(x)$, barrier certificates $B(x)$, ranking functs $\eta(\mathtt{x})$
- model abstractions $\hat{f}(x)$
- sound and automated synthesis (vs. numerical, manual)

Goals

- new approach to synthesise certificates for stability/safety/termination/...
- new approach is

Goals

- new approach to synthesise certificates for stability/safety/termination/...
- new approach is
  - automated: minimal user input
  - sound: SMT-verified

Goals

- new approach to synthesise certificates for stability/safety/termination/...
- new approach is
  - automated: minimal user input
  - sound: SMT-verified
  - in general, lacks completeness
- FOSSIL, a software tool

- variations, extensions, applications ...

Outline

Outline

Decision problems: SAT and SMT

- SAT is a decision problem (can be formulated as a yes/no question)
- finding satisfying assignment of Boolean functions
- e.g., assume x_i Boolean,

$$\exists x_1, x_2, x_3 : (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

Decision problems: SAT and SMT

- SAT is a decision problem (can be formulated as a yes/no question)
- finding satisfying assignment of Boolean functions
- e.g., assume x_i Boolean,

$$\exists x_1, x_2, x_3 : (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$$

- SMT: decision problem for logical formulae within one (or combination of) theories
- e.g., assume x_i integers,

$$\exists x_1, x_2 : x_1 \geq 0 \Rightarrow 3x_1 + 2x_2 + 1 > 0$$

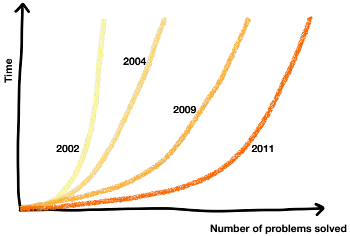
- instance: theory of arithmetics over real closed fields (e.g., with polynomial functions)

Decision problems: SAT and SMT

- SAT is hard! Let alone SMT!

Decision problems: SAT and SMT

- SAT is hard! Let alone SMT!
- yes but ...ever more powerful and numerous SW tools, of industrial relevance



[credit: E. Polgreen]

- breakthroughs: algorithmic improvements over DPLL/CDCL + applications

From decision to synthesis problems

- consider problem:
  $x_i \in \mathbb{Z}$ integers, $F : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$;

$$\exists F, \forall x_1, x_2,$$

$$F(x_1, x_2) \geq x_1 \wedge F(x_1, x_2) \geq x_2 \wedge (F(x_1, x_2) = x_1 \vee F(x_1, x_2) = x_2)$$

From decision to synthesis problems

- consider problem:
  $x_i \in \mathbb{Z}$ integers, $F : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$;

$$\exists F, \forall x_1, x_2,$$

$$F(x_1, x_2) \geq x_1 \wedge F(x_1, x_2) \geq x_2 \wedge (F(x_1, x_2) = x_1 \vee F(x_1, x_2) = x_2)$$

  $(F = \max\{x_1, x_2\})$

- synthesis: constructive, optimisation-based, *inductive*

Outline

Outline

Conditions for Lyapunov function $V(x)$

- assume $x_e \in \mathbb{R}^n$ is an equilibrium, $f(x_e) = 0$
- ensure asymptotic stability of x_e in $\mathcal{D} \subseteq \mathbb{R}^n$

1. lower bound:

$$V(x_e) = 0 \tag{1}$$

2. positive definiteness:

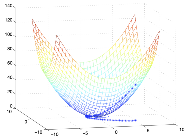
$$V(x) > 0, \ \forall x \in \mathcal{D} \setminus \{x_e\} \tag{2}$$

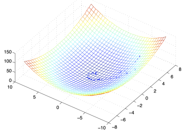
3. negative Lie derivative:

$$\dot{V}(x) = \nabla V(x) \cdot f(x) < 0, \ \forall x \in \mathcal{D} \setminus \{x_e\} \tag{3}$$

- no general, effective method: potential search within function space

Conditions for Lyapunov function $V(x)$





1. lower bound:

$$V(x_e) = 0 \tag{1}$$

2. positive definiteness:

$$V(x) > 0, \ \forall x \in \mathcal{D} \setminus \{x_e\} \tag{2}$$

3. negative Lie derivative:

$$\dot{V}(x) = \nabla V(x) \cdot f(x) < 0, \ \forall x \in \mathcal{D} \setminus \{x_e\} \tag{3}$$

- no general, effective method: potential search within function space

Lyapunov synthesis problem

- solve following synthesis problem:

$$\exists V : \mathcal{D} \to \mathbb{R} \;\; \forall x \in \mathcal{D} \;\; s.t. \text{ conditions } (1) \wedge (2) \wedge (3) \text{ are SAT}$$

Lyapunov synthesis problem

- solve following synthesis problem:

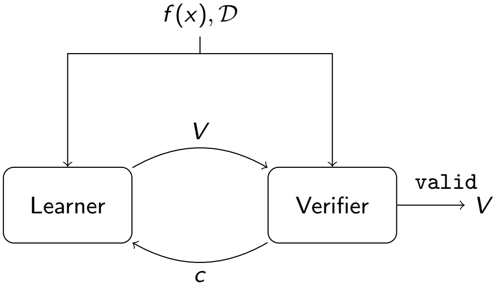
$$\exists V : \mathcal{D} \to \mathbb{R} \;\; \forall x \in \mathcal{D} \;\; s.t. \text{ conditions } (1) \wedge (2) \wedge (3) \text{ are SAT}$$

- inductive synthesis $\rightarrow$ "guess and check"

  1. sample (finite) set $S \subset \mathcal{D}$
  2. "guess" $V(s)$ that satisfies (1) $\wedge$ (2) $\wedge$ (3) on points $s \in S$
  3. "check" either $V(x)$ valid over dense $\mathcal{D}$ or counterexample c : query SMT solver whether $\exists c \in \mathcal{D} : \neg(1) \vee \neg(2) \vee \neg(3)$
  4. $S \leftarrow S \cup c$, loop back to 2

Counterexample-guided inductive synthesis (CEGIS)

1. Learner: generates candidates V over finite set "guess"
2. Verifier: certifies validity on $\mathcal{D}$, or provides c-example(s) c "check"



- sound but not complete: search space for V infinite (as domain $\mathcal{D}$)

Synthesis of Lyapunov functions from templates[1]

- work with fixed, given polynomial templates

$$V(x) = \sum_{l=1}^{c} x_l^T \; P_l \; x_l$$

c - degree that can be tuned

- "guess" - candidates generated by SMT solver call enforcing Lyapunov conditions in (1), (2), (3) on S

[1] D. Ahmed, A. Peruffo and A. Abate, "Automated and Sound Synthesis of Lyapunov Functions with SMT Solvers," TACAS20, LNCS 12078, pp. 97-114, 2020.

Synthesis of Lyapunov functions from templates[1]

- work with fixed, given polynomial templates

$$V(x) = \sum_{l=1}^{c} x_l^T \; P_l \; x_l$$

c - degree that can be tuned

- "guess" - candidates generated by SMT solver call
  enforcing Lyapunov conditions in (1), (2), (3) on S

- "check" via SMT solver on non-linear formulae over reals
- Z3, dReal: outcomes provably correct, sound and (for Z3) complete

[1] D. Ahmed, A. Peruffo and A. Abate, "Automated and Sound Synthesis of Lyapunov Functions with SMT Solvers," TACAS20, LNCS 12078, pp. 97-114, 2020.

Synthesis of Lyapunov functions from templates[1]

- work with fixed, given polynomial templates

$$V(x) = \sum_{l=1}^{c} x_l^T \; P_l \; x_l$$

c - degree that can be tuned

- "guess" - candidates generated by SMT solver call
  enforcing Lyapunov conditions in (1), (2), (3) on S
- alternatively, solution of optimisation problem (via Gurobi)

- "check" via SMT solver on non-linear formulae over reals
- Z3, dReal: outcomes provably correct, sound and (for Z3) complete

[1] D. Ahmed, A. Peruffo and A. Abate, "Automated and Sound Synthesis of Lyapunov Functions with SMT Solvers," TACAS20, LNCS 12078, pp. 97-114, 2020.

Experimental results

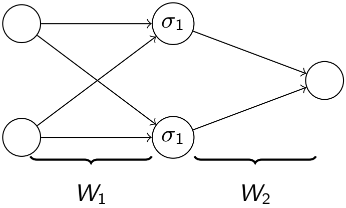
n	Gurobi-CEGIS			Z3-CEGIS		
	Iterations	Time [sec]	OOT	Iterations	Time [sec]	OOT
3	3 [3, 3]	0.48 [0.33, 0.77]	–	3.03 [3, 4]	0.49 [0.4, 0.70]	–
4	3.10 [3, 4]	0.53 [0.36, 1.20]	–	5.93 [4, 7]	0.68 [0.54,1.07]	–
5	4.15 [4, 5]	1.33 [1.08, 1.97]	–	7.38 [5, 12]	1.67 [1.10, 3.03]	–
6	6.99 [4, 10]	3.88 [2.41, 4.97]	–	9.10 [6, 10]	7.48 [2.40, 54.44]	–
7	8.56 [4, 12]	12.64 [2.9, 62.3]	–	12.88 [5, 17]	17.63 [5.41, 20.3]	1
8	9.14 [3, 13]	21.50 [3.9, 114.16]	1	16.2 [3, 25]	23.91 [4.05, 35.08]	1
9	15.72 [3, 32]	29.98 [3.87, 78.5]	2	22.47 [4, 35]	34.41 [5.67, 48.96]	5
10	18.45 [3,41]	40.63 [6.17, 46.65]	5	27.25 [5, 47]	44.63 [6.32, 101.2]	7

- Gurobi-CEGIS vs. Z3-CEGIS
- $N = 100$ randomly generated n-dimensional linear models
- OOT = number of runs (out of N) not finishing within 180 [sec]
- time: average [min, max] performance [sec]

Lyapunov functions as neural networks[2]

- learner trains shallow neural network

$$V(x) = W_2 \cdot \sigma_1(W_1 x + b_1)$$



- generality and flexibility (univ. function approximator)
- alternative activation fcns (σ_1)
- loss function enforces Lyapunov conditions in (1), (2), (3):

$$L(S) = \sum_{s \in S} \max\{0, -V(s)\} + \sum_{s \in S} \max\{0, \dot{V}(s)\}$$

- loss function L is *"pretty good"* proxy of synthesis formula

[2] A. Abate, D. Ahmed, M. Giacobbe and A. Peruffo "Automated Formal Synthesis of Lyapunov Neural Networks," IEEE Control Systems Letters, 5 (3), 773-778, 2020.

From learner to verifier, and back

- communication btw *Learner* $\leftrightarrow$ *Verifier* is crucial

$\rightarrow$ translate Neural Network into formula

$$V(x) = W_2 \cdot \sigma_1(W_1 x + b_1)$$

$$\dot{V}(x) = \nabla V(x) \cdot f(x) = W_2 \cdot \sigma_1'(W_1 x + b_1) \cdot W_1 \cdot f(x)$$

- SMT engine accepts negation of synthesis specification, namely

$$\exists x \ s.t. \ x \in \mathcal{D} \setminus \{x_e\} \wedge V(x) \leq 0 \vee \dot{V}(x) \geq 0$$

- outcomes:
  1. sat: counter-example c and loop back to Learner, now with $S \cup c$
  2. unsat: V is valid (proper Lyapunov fcn)

From learner to verifier, and back

- communication btw *Learner* $\leftrightarrow$ *Verifier* is crucial

$\rightarrow$ translate Neural Network into formula

$$V(x) = W_2 \cdot \sigma_1(W_1 x + b_1)$$

$$\dot{V}(x) = \nabla V(x) \cdot f(x) = W_2 \cdot \sigma_1'(W_1 x + b_1) \cdot W_1 \cdot f(x)$$

- SMT engine accepts negation of synthesis specification, namely

$$\exists x \ s.t. \ x \in \mathcal{D} \setminus \{x_e\} \wedge V(x) \leq 0 \vee \dot{V}(x) \geq 0$$

- outcomes:
  1. sat: counter-example c and loop back to Learner, now with $S \cup c$
  2. unsat: V is valid (proper Lyapunov fcn)

$\leftarrow$ extract more details around c-ex c, provide more info to Learner
  1. generate random samples around c
  2. unfold model trajectories from c

Experimental results

- non-linear models that do not admit global polynomial $V(x)$
- vs. related work, provided larger domains in comparable runtimes

LNN Time	LNN r	NLC[3] Time	NLC Domain	CBS[4] Time	CBS r	SOS[5] Time	SOS r
12.01	500	6.28	1	0.22	1	6.67	800
0.29	100	5.45	1	0.30	1	7.76	25
0.32	1000	54.12	1	2.22	1	11.80	OOT
33.27	1000	37.80	1	0.42	1	9.65	OOT

Table: Comparison between proposed approach (LNN), NLC and CBS approaches, and SOSTOOLS: total Time (guess and check) [sec], Domain $\mathcal{D}$ (radius r). Timeouts = OOT.

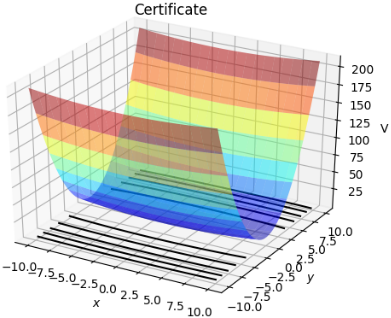
[3] Chang, Y.C., Roohi, N. and Gao, S., "Neural Lyapunov Control", NeurIPS, pp. 3245-3254, 2019.

[4] D. Ahmed, A. Peruffo and A. Abate, "Automated and Sound Synthesis of Lyapunov Functions with SMT Solvers," TACAS20, LNCS 12078, pp. 97-114, 2020.

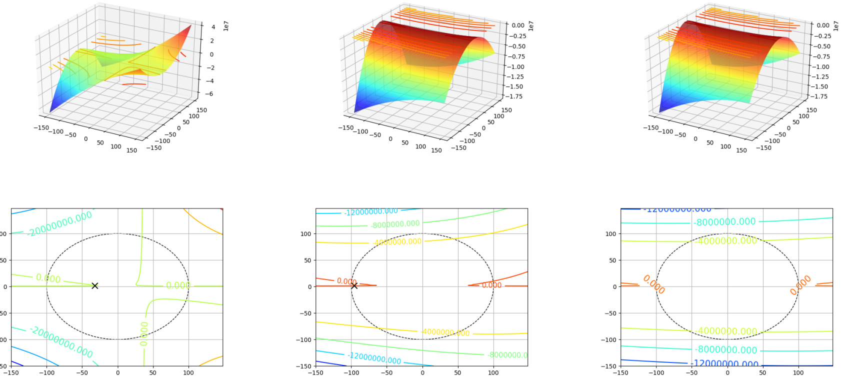
[5] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. A. Parrilo. SOSTOOLS: Sum of squares optimization toolbox for MATLAB, 2013.

Experimental results

- non-linear models that do not admit global polynomial $V(x)$
- vs. related work, provided larger domains in comparable runtimes



Example: Synthesis of Lyapunov function in 3 CEGIS loops



Outline

Outline

Conditions for barrier certificate $B(x)$

- from notion of stability (equilibria) to safety (set invariance)
- consider sets X_0 (initial) and X_u (unsafe)
- ensure there exists no trajectory starting in X_0 entering X_u, over $\mathcal{D}$

1. negativity in initial set X_0:

$$B(x) \leq 0 \ \forall x \in X_0 \tag{4}$$

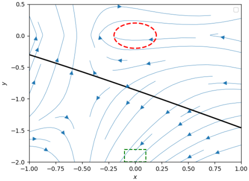
2. positivity in unsafe set X_U:

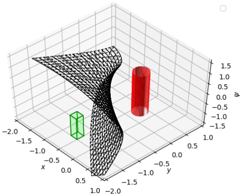
$$B(x) > 0 \ \forall x \in X_U \tag{5}$$

3. set invariance property via Lie derivative:

$$\dot{B}(x) < 0 \ \forall x \text{ s.t.} B(x) = 0 \tag{6}$$

Conditions for barrier certificate $B(x)$





1. negativity in initial set X_0:

$$B(x) \leq 0 \;\forall x \in X_0 \tag{4}$$

2. positivity in unsafe set X_U:

$$B(x) > 0 \;\forall x \in X_U \tag{5}$$

3. set invariance property via Lie derivative:

$$\dot{B}(x) < 0 \;\forall x \text{ s.t.} B(x) = 0 \tag{6}$$

Barrier certificates as neural nets[3]

- barrier certificates often require a more arbitrary (non-linear) structure than Lyapunov functions (cf. shapes of X_0, X_U)
- neural networks provide powerful, rich template for diverse potential candidates:
  - canonical activations such as $\sigma(x) = x^n$
  - more neuro-typical activation functions such as $\sigma(x) = \tanh(x)$ and $\sigma(x) = \ln(1 + \exp(x))$
  - use of *leaky* and *saturated* ReLU's

[3] A. Abate, D. Ahmed and A. Peruffo, "Automated Formal Synthesis of Neural Barrier Certificates for Dynamical Models," TACAS21, LNCS 12651, pp. 370–388, 2021.

Experimental results

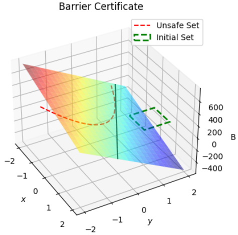
Benchmark	CEGIS (this work)				BC[4]			SOS[5]	
	Learn	Verify	Samples	Iters	Learn	Verify	Samples	Synth	Verify
Darboux	31.6	0.01	0.5 k	2	54.9	20.8	65 k	×	–
Exponential	15.9	0.07	1.5 k	2	234.0	11.3	65 k	×	–
Obstacle	55.5	1.83	2.0 k	9	3165.3	1003.3	2097 k	×	–
Polynomial	64.5	4.20	2.3 k	2	1731.0	635.3	65 k	8.10	×
Hybrid mod	0.58	2.01	0.5 k	1	–	–	–	12.30	0.11
4-d ODE	29.31	0.07	1 k	1	–	–	–	12.90	OOT
6-d ODE	89.52	1.61	1 k	3	–	–	–	16.60	OOT
8-d ODE	104.5	82.51	1 k	3	–	–	–	26.10	OOT

- time for Learning and Verification steps in [sec]
- 'Samples' = size of input data for Learner (in thousands)
- 'Iters' = number of iterations of CEGIS loop
- × = synthesis or verification failure, OOT = verification timeout

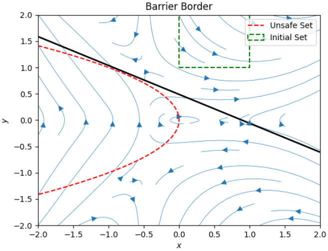
[4] H. Zhao, X. Zeng, T. Chen, and Z. Liu. Synthesizing Barrier Certificates Using Neural Networks. In Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control, HSCC, 2020.

[5] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. A. Parrilo. SOSTOOLS: Sum of squares optimization toolbox for MATLAB, 2013.

Synthesised barrier certificates - examples

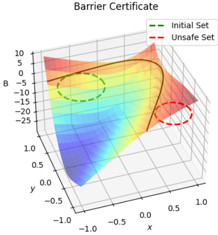


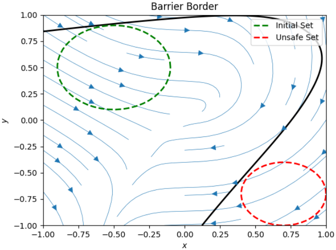
$$\dot{x} = y + 2xy,$$
$$\dot{y} = -x + 2x^2 - y^2$$



[10] · Linear

Synthesised barrier certificates - examples

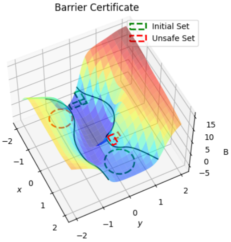


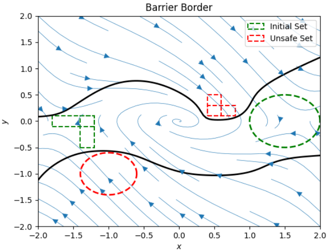


$$\dot{x} = \exp(-x) + y - 1,$$
$$\dot{y} = -\sin(x)^2$$

[20] · Softplus

Synthesised barrier certificates - examples





$$\dot{x} = y,$$
$$\dot{y} = -x - y + \frac{1}{3}x^3$$

$[20, 20] \cdot \text{Sigmoid, Sigmoid}$

Outline

Outline

Software Tool - FOSSIL[6]

- Formal Synthesis of Lyapunov Functions and Barrier Certificates using Neural Networks

  - CEGIS with NN templates
  - two SMT engines: Z3, dReal
  - improved Learner/Verifier communication

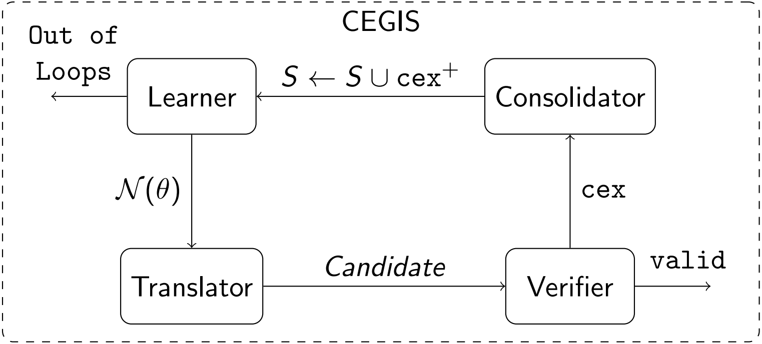
  - supports non-polynomial dynamics with multiple equilibria
  - handles continuous- and discrete-time models
  - scalable and robust

  - easy-to-use Jupyter-based user interface

`https://github.com/oxford-oxcav/fossil`

[6] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe and A. Peruffo, "FOSSIL: A Software Tool for the Formal Synthesis of Lyapunov Functions and Barrier Certificates using Neural Networks," HSCC, pp. 1-11, 2021.

FOSSIL - architecture

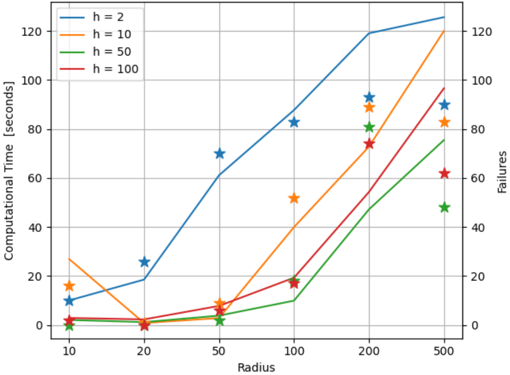


- **Translator**
  - translates numerical neural network into symbolic candidate
  - trade-off precision of symb variables vs tractability of verification
- **Consolidator**
  - samples around counterexample $\rightarrow$ more data to learner
  - computes trajectory of maximum constraint violation

FOSSIL - robustness

- random initialisation influences results
- variation of NN width (h) and domain radius
- average times and number of failures for Lyapunov synthesis with

$$\begin{cases} \dot{x} = -x + xy \\ \dot{y} = -y \end{cases}$$



FOSSIL - robustness

- model dynamics

$$\begin{cases} \dot{x} = y + 2xy, \\ \dot{y} = -x + 2x^2 - y^2 \end{cases}$$

- sets

$$X_0 = \{0 \le x \le 1, 1 \le y \le 2\}, \quad X_u = \{x + y^2 \le 0\}$$

Table: Minimum and average running times and number of failures (in parenthesis), out of 25 runs, for barrier synthesis task.

Hidden Neurons	2	10	50	100
Min Time	18.11	21.72	61.06	173.01
Avg Time (Fail)	35.46 (0)	33.27 (0)	73.77 (0)	217.68 (0)

FOSSIL - tests on benchmarks from literature

Benchmark	Dim	Iters	Time	N-Net	Verifier
$\texttt{nonpoly}_0$	2	1	0.22	$2{\cdot}[\phi_2]$	dReal
$\texttt{nonpoly}_1$	2	1	1.54	$20{\cdot}[\phi_1, \phi_2]$	Z3
$\texttt{nonpoly}_2$	3	1	1.63	$10{\cdot}[\phi_1, \phi_2]$	Z3
$\texttt{nonpoly}_3$	3	1	12.06	$3{\cdot}[\phi_2]$	Z3
$\texttt{poly}_1$	3	1	0.81	$5{\cdot}[\phi_2]$	dReal
$\texttt{poly}_2$	2	3	19.13	$10{\cdot}[\phi_2]$	Z3
$\texttt{poly}_3$	2	3	17.25	$5{\cdot}[\phi_2]$	dReal
$\texttt{poly}_4$	2	2	5.88	$5{\cdot}[\phi_2]$	dReal
$\texttt{barr}_1$	2	1	26.85	$10{\cdot}[\phi_1, \phi_{1,2}, \phi_1]$	dReal
$\texttt{barr}_2$	2	1	0.92	$10{\cdot}$[tanh]	dReal
$\texttt{barr}_3$	2	2	10.03	$20{\cdot}$[tanh]	dReal
hy-lyap	2	1	2.24	$10{\cdot}[\phi_2]$	Z3
hy-barr	2	1	8.46	$3{\cdot}[\phi_{1,2}]$	Z3
$\texttt{hi-ord}_4$	4	3	98.94	$20{\cdot}[\phi_1]$	dReal
$\texttt{hi-ord}_6$	6	1	29.39	$10{\cdot}[\phi_1]$	dReal
$\texttt{hi-ord}_8$	8	1	41.72	$10{\cdot}[\phi_1]$	dReal

FOSSIL - ongoing extensions

- class of properties:

$$\forall \xi(t_0) \in \mathcal{X}_I, \exists T \in \mathbb{R}^+, \forall t \in [t_0, T], \forall \tau \geq T : \\ \xi(t) \notin \mathcal{X}_U \wedge \xi(T) \in \mathcal{X}_G \wedge \xi(\tau) \in \mathcal{X}_F$$

- encompasses stab/reach/safety/invariance/reach-avoid/RSWA/. . .

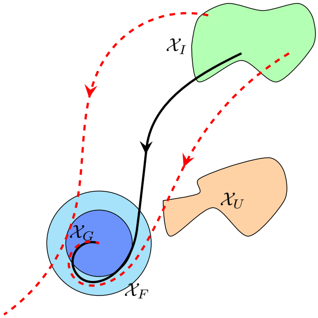


Figure: Black trajectory satisfies, red ones violate property.

FOSSIL - ongoing extensions

- dynamical models with inputs (a.k.a., external non-determinism)

$\rightarrow$ synthesis of "control certificates" (e.g., control LFs, BCs, ...- well known in literature)

- approach: control policies are NN-templated,
  synthesis follows from above (albeit being more complex)

- (SW to be released hopefully soon)

Outline

Outline

Extension: discrete-time, probabilistic models

- discrete-time models (e.g. SW programs), $x^+ = f(x)$
- same Lyapunov conditions, except concerning "next step":

$$V(f(x)) < V(x), \quad \forall x \in \mathcal{D} \setminus \{x_e\}$$

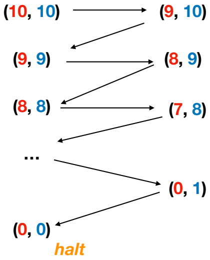
- stochastic models: same story, except for "next step"-condition is in expectation:

$$\mathbb{E}[V(f(x)) \mid x] < V(x), \quad \forall x \in \mathcal{D} \setminus \{x_e\}$$

Program termination[7]

- discrete-time programs

```
while (red > 0 or blue > 0)
  if ((red + blue) % 2 == 0)
   red = red - 1
  else
   blue = blue - 1
```

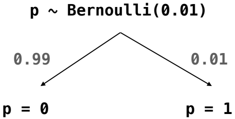


[7] D. Roy, M. Giacobbe, and A. Abate, "Learning Probabilistic Termination Proofs," CAV21, LNCS 12760, pp. 3–26, 2021.

Program termination[7]

- discrete-time programs

```
while (red > 0 or blue > 0)
  p ~ Bernoulli(0.01)
  if (p == 1)
   red = red - 1
  else
   blue = blue - 1
```



[7] D. Roy, M. Giacobbe, and A. Abate, "Learning Probabilistic Termination Proofs," CAV21, LNCS 12760, pp. 3–26, 2021.

Program termination[7]

- discrete-time programs

```
while (red > 0 or blue > 0)
  p ~ Bernoulli(0.01)
  if (p == 1)
   red = red - 1
  else
   blue = blue - 1
```

$$X_0 = (2, 2)$$

$$(2, 2) \stackrel{p=1}{\rightarrow} (1, 2) \stackrel{p=0}{\rightarrow} (1, 1) \stackrel{p=0}{\rightarrow} (1, 0) \stackrel{p=1}{\rightarrow} (0, 0) \text{ (halt)}$$

$$(2, 2) \stackrel{p=0}{\rightarrow} (2, 1) \stackrel{p=0}{\rightarrow} (2, 0) \stackrel{p=0}{\rightarrow} (2, -1) \stackrel{p=0}{\rightarrow} \ldots$$

[7] D. Roy, M. Giacobbe, and A. Abate, "Learning Probabilistic Termination Proofs," CAV21, LNCS 12760, pp. 3–26, 2021.

Program termination[7]

- discrete-time programs

```
while (red > 0 or blue > 0)
  p ~ Bernoulli(0.01)
  if (p == 1)
   red = red - 1
  else
   blue = blue - 1
```

$$X_0 = (2, 2)$$

$$(2,2) \overset{p=1}{\rightarrow} (1,2) \overset{p=0}{\rightarrow} (1,1) \overset{p=0}{\rightarrow} (1,0) \overset{p=1}{\rightarrow} (0,0) \ \text{(halt)}$$

$$(2,2) \overset{p=0}{\rightarrow} (2,1) \overset{p=0}{\rightarrow} (2,0) \overset{p=0}{\rightarrow} (2,-1) \overset{p=0}{\rightarrow} \dots$$

- what is the probability that allowable traces terminate?
- PAST - positive, almost-sure termination:

$$\mathbb{E}\,[\text{execution steps}] < \infty$$

- PAST implies a.s.-termination (w.p. 1)

[7] D. Roy, M. Giacobbe, and A. Abate, “Learning Probabilistic Termination Proofs,” CAV21, LNCS 12760, pp. 3–26, 2021.

Ranking functions for program termination

- probabilistic programs in discrete time
- certificates for PAST: ranking super-martingales (RSMs) η

Ranking functions for program termination

- probabilistic programs in discrete time
- certificates for PAST: ranking super-martingales (RSMs) η

- seek $\eta : x \to \mathbb{R}$, where x are program configs, such that
    1. $\eta(x) \geq K$
    2. $\mathbb{E}\left[\eta(X_{t+1}) | X_t = x\right] \leq \eta(x) - \epsilon, \quad \epsilon > 0,$

- conditions above entail
    1. bound (K) from below
    2. decrease in expectation at each iteration by at least an ϵ

CEGIS-based synthesis of RSMs

- learner trains ReLU-activated NN

CEGIS-based synthesis of RSMs

- learner trains ReLU-activated NN

- lower bound on NN (K) satisfied by construction
- NN ought to "decrease in expectation" from program state x

$\rightarrow$ loss at state x is $\max\left\{\mathbb{E}\left[\eta(X_{t+1})|X_t = x\right] - \eta(x) + \epsilon, 0\right\}$

CEGIS-based synthesis of RSMs

- learner trains ReLU-activated NN

- lower bound on NN (K) satisfied by construction
- NN ought to "decrease in expectation" from program state x

$\rightarrow$ loss at state x is $\max\left\{\mathbb{E}\left[\eta(X_{t+1})|X_t = x\right] - \eta(x) + \epsilon, 0\right\}$

- approximate expectation by Monte Carlo estimate from sampled executions x' from program state x:

$\rightarrow$ loss at state x is $\max\left\{\frac{1}{|P'|}\sum_{x' \in P'} \eta(x') - \eta(x) + \epsilon, 0\right\}$

- smoothen loss (easier derivative)

$\rightarrow$ loss at state x is $\text{softplus}\left\{\frac{1}{|P'|}\sum_{x' \in P'} \eta(x') - \eta(x) + \epsilon, 0\right\}$

CEGIS-based synthesis of RSMs - example

```
while (red > 0 or blue > 0)
  p ~ Bernoulli(0.01)
  if (p == 1)
   red = red - 1
  else
   blue = blue - 1
```

$$X_0 = (2, 2)$$

$$(2,2) \overset{p=1}{\to} (1,2) \overset{p=0}{\to} (1,1) \overset{p=0}{\to} (1,0) \overset{p=1}{\to} (0,0) \;\; \text{(halt)}$$

$$(2,2) \overset{p=0}{\to} (2,1) \overset{p=0}{\to} (2,0) \overset{p=0}{\to} (2,-1) \overset{p=0}{\to} \dots$$

- synthesised RSM: $\eta(\texttt{red}, \texttt{blue}) = \max\{\texttt{red},\ 0\} + \max\{\texttt{blue},\ 0\}$

CEGIS-based synthesis of RSMs - benchmarks[8]

Program	AMBER [39]	Farkas' lemma [2]	AB-SYNTH [41]	Succ. rate	Inter.	Train.	Verif.	#iter	NRSM
`Hare & Tortoise` (d)	0.04	≈ 0	0.09	10/10	0.61	3.86	0.70	0	SOR
`exmini/terminate` (d)	—	0.02	oot	10/10	1.75	29.35	7.67	2	SOR
`aaron2` (d)	0.03	0.02	0.02	10/10	0.04	2.27	0.01	0	SOR
`catmouse` (c)	0.03	0.02	—	9/10	0.39	12.41	3.68	1	SOS
`counterex1c` (d)	—	0.02	0.22	8/10	1.00	6.71	0.02	0	SOR
`easy1` (d)	0.12	0.01	0.05	10/10	1.12	5.55	1.27	0	SOR
`easy2` (c)	0.04	0.02	—	10/10	1.55	6.79	0.18	0	SOS
`ndecr` (d)	0.04	0.02	0.03	10/10	1.18	5.63	0.02	0	SOR
`random1d` (c)	0.05	0.02	—	10/10	1.14	4.86	0.79	0	SOS
`rsd` (d)	error	0.01	oot	10/10	1.14	6.18	2.04	0	SOR
`speedFails1` (d)	0.07	0.01	0.04	10/10	0.45	4.09	0.67	0	SOR
`speedPldi2` (d)	—	0.02	0.40	9/10	1.36	7.85	0.02	0	SOR
`speedPldi3` (d)	—	0.02	0.36	8/10	2.58	30.70	2.12	1	SOR
`speedPldi4` (d)	—	0.02	0.17	10/10	0.68	5.07	0.04	0	SOR
`speedSingleSingle` (c)	0.03	0.02	—	10/10	0.39	2.85	0.51	0	SOS
`speedSingleSingle2` (d)	—	0.02	0.15	10/10	0.83	7.30	0.04	0	SOR
`wcet0` (d)	—	0.02	0.10	10/10	1.45	5.64	0.09	0	SOR
`wcet1` (d)	—	0.02	0.10	10/10	0.85	4.31	0.09	0	SOR

- admittedly slower than existing tools
- robust (cf. success rate)
- generalises well (d = discrete, c = continuous distributions; SOR = sum of ReLU, SOS = sum of squares)

[8] D. Roy, M. Giacobbe, and A. Abate, "Learning Probabilistic Termination Proofs," CAV21, LNCS 12760, pp. 3–26, 2021.

CEGIS-based synthesis of RSMs - benchmarks[8]

Program	AMBER [39]	Farkas' lemma [2]	AB-SYNTH [41]	Succ. rate	Inter.	Train.	Verif.	#iter	NRSM
`probfact` (d)	—	—	n/a	10/10	0.49	6.12	0.16	0	SOR
`probfact2` (d)	—	—	n/a	10/10	0.45	5.89	0.23	0	SOR
`marbles` (d)	—	—	n/a	10/10	0.84	10.83	0.91	0	SOR
`marbles3` (d)	—	—	n/a	10/10	0.40	70.14	7.87	2	SOR
`crwalk` (c)	—	—	—	10/10	0.53	3.06	1.56	1	SOS
`crwalk2` (c)	—	—	—	10/10	1.32	3.11	0.75	1	SOS
`expdistrw` (c)	n/a	—	—	10/10	0.05	1.53	0.01	0	SOS
`expdistrw2` (c)	n/a	—	—	10/10	4.92	3.15	1.03	1	SOS
`gaussrw` (c)	—	—	—	10/10	10.30	3.45	0.75	0	SOS
`gaussrw2` (c)	—	—	—	9/10	15.46	4.91	5.33	0	SOS
`slicedcauchy` (c)	—	—	—	10/10	0.02	3.31	0.01	0	SOR
`slicedcauchy2` (c)	—	—	—	10/10	0.01	2.16	0.03	0	SOR

- generalises well to non-linear programs
- ongoing work:
  1. more general program control-flow, e.g. nested loops
  2. use of better sampling techniques, deeper networks, symbolic post expectations
  3. problems beyond PAST: quantitative verification

[8] D. Roy, M. Giacobbe, and A. Abate, "Learning Probabilistic Termination Proofs," CAV21, LNCS 12760, pp. 3–26, 2021.

Quantitative probabilistic verification[9]

- probabilistic programs on real-valued variables
- quantitative verification questions, i.e. probabilistic reachability analysis
  - termination and assertion violation
  - safety and invariance verification

- certificates: neural indicating super-martingales (ISMs)

$\rightarrow$ upper bound on probabilistic reachability

(quite a few alternatives in literature)

- post expectation: program-aware vs -agnostic
- tight bounds, beyond linear certificates

[9] A. Abate, A. Edwards, M. Giacobbe, H. Punchihewa, and D. Roy, "Quantitative Verification With Neural Networks For Probabilistic Programs and Stochastic Systems," arXiv:2301.06136, 2023.

Neural ISMs - benchmarks

Benchmark	Farkas' Lemma	Quantitative Neural Certificates				Network Arch.
		Program-Agnostic		Program-Aware		
	p	p	Success Ratio	p	Success Ratio	
persist_2d	-	≤ 0.1026	0.9	≤ 0.1175	0.9	(3, 1)
faulty_marbles	-	≤ 0.0739	0.9	≤ 0.0649	0.8	3
faulty_unreliable	-	≤ 0.0553	0.9	≤ 0.0536	1.0	3
faulty_regions	-	≤ 0.0473	0.9	≤ 0.0411	0.9	(3, 1)
cliff_crossing	≤ 0.4546	≤ 0.0553	0.9	≤ 0.0591	0.8	4
repulse	≤ 0.0991	≤ 0.0288	1.0	≤ 0.0268	1.0	3
repulse_uniform	≤ 0.0991	≤ 0.0344	1.0	-	-	2
repulse_2d	≤ 0.0991	≤ 0.0568	1.0	≤ 0.0541	1.0	3
faulty_varying	≤ 0.1819	≤ 0.0864	1.0	≤ 0.0865	1.0	2
faulty_concave	≤ 0.1819	≤ 0.1399	1.0	≤ 0.1356	0.9	(3, 1)
fixed_loop	≤ 0.0091	≤ 0.0095	1.0	≤ 0.0094	1.0	1
faulty_loop	≤ 0.0181	≤ 0.0195	1.0	≤ 0.0184	1.0	1
faulty_uniform	≤ 0.0181	≤ 0.0233	1.0	≤ 0.0221	1.0	1
faulty_rare	≤ 0.0019	≤ 0.0022	1.0	≤ 0.0022	1.0	1
faulty_easy1	≤ 0.0801	≤ 0.1007	1.0	≤ 0.0865	1.0	1
faulty_ndecr	≤ 0.0561	≤ 0.0723	1.0	≤ 0.0630	1.0	1
faulty_walk	≤ 0.0121	≤ 0.0173	1.0	≤ 0.0166	1.0	1

Table: Neural ISMs vs Farkas Lemma; p - average prob bound from certificate; Success Ratio - terminating runs out of 10 repeats in CEGIS; NN architecture - (h_1, h_2) are 2 hidden layers.

Neural ISMs - benchmarks

Benchmark	Farkas' Lemma	Quantitative Neural Certificates			
		Program-Agnostic		Program-Aware	
	Time	Learn Time	Verify Time	Learn Time	Verify Time
persist_2d	-	169.14	85.31	44.96	74.90
faulty_marbles	-	114.24	29.23	15.86	28.68
faulty_unreliable	-	123.85	45.48	18.34	33.97
faulty_regions	-	17.92	35.85	17.55	32.38
cliff_crossing	0.11	134.61	19.02	21.27	29.07
repulse	0.19	16.65	5.00	6.49	3.74
repulse_uniform	0.19	21.28	14.18	-	-
repulse_2d	0.12	122.92	64.54	15.75	47.70
faulty_varying	0.36	21.74	5.06	4.71	3.28
faulty_concave	0.39	49.12	13.37	13.49	7.82
fixed_loop	0.15	14.16	3.14	3.34	2.43
faulty_loop	0.16	25.52	3.81	3.73	2.66
faulty_uniform	0.34	20.20	1.91	6.75	1.33
faulty_rare	0.27	25.52	4.27	3.71	2.96
faulty_easy1	0.31	104.20	12.78	4.95	7.51
faulty_ndecr	0.33	104.89	9.06	5.37	4.66
faulty_walk	0.32	15.08	4.00	6.97	3.33

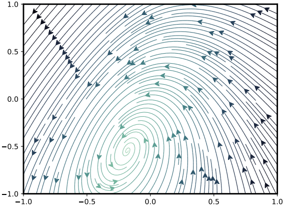
Table: Neural ISMs vs Farkas Lemma - Runtimes. Obviously slower! Learning takes time.

Outline

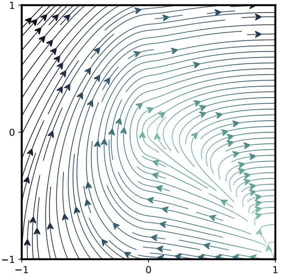
Outline

Model hybridisations

- motivation: safety verification of non-linear dynamical systems $\dot{x} = f(x)$ over $x \in \mathcal{X} \subset \mathbb{R}^n$, is in general hard, not automated



$$\begin{cases} \dot{x} & = -y - 1.5x^2 - 0.5x^3 - 0.5 \\ \dot{y} & = 3x - y \\ \mathcal{X} & = [-1, 1]^2 \end{cases}$$

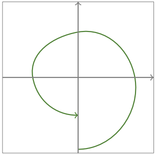


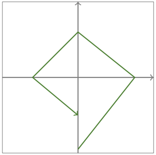
$$\begin{cases} \dot{x} & = x^2 + y \\ \dot{y} & = \sqrt[3]{x^2} - x \\ \mathcal{X} & = [-1, 1]^2 \end{cases}$$

Model hybridisations

- motivation: safety verification of non-linear dynamical systems $\dot{x} = f(x)$ over $x \in \mathcal{X} \subset \mathbb{R}^n$, is in general hard, not automated

- leverage formal abstractions for verification
- namely, simpler models encapsulating original dynamics (sim, bisim)

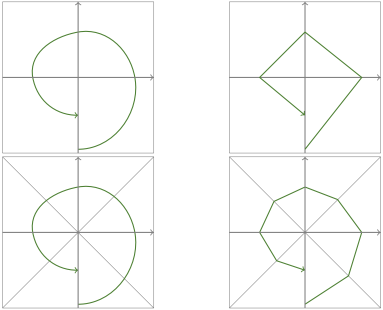




Model hybridisations

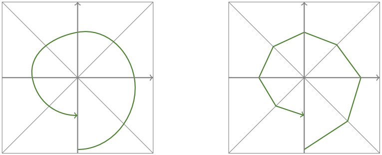
- motivation: safety verification of non-linear dynamical systems $\dot{x} = f(x)$ over $x \in \mathcal{X} \subset \mathbb{R}^n$, is in general hard, not automated

- leverage formal abstractions for verification
- namely, simpler models encapsulating original dynamics (sim, bisim)



Model hybridisations

- motivation: safety verification of non-linear dynamical systems $\dot{x} = f(x)$ over $x \in \mathcal{X} \subset \mathbb{R}^n$, is in general hard, not automated
- leverage formal abstractions for verification
- namely, simpler model encapsulating original dynamics



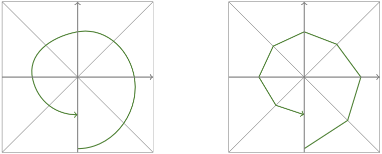
- hybridisation: within $\mathcal{X}$, convert $f(x)$ to hybrid system $\mathcal{H}$ with multiple partitions, where

  each partition has own flow $\tilde{f}(x)$ & transitions to other parts
- upper-bound ϵ to error, so that

$$\dot{x} = \tilde{f}(x) + d, \quad \|d\| \leq \epsilon, \quad x \in \mathcal{X}$$

Model hybridisations

- motivation: safety verification of non-linear dynamical systems $\dot{x} = f(x)$ over $x \in \mathcal{X} \subset \mathbb{R}^n$, is in general hard, not automated
- leverage formal abstractions for verification
- namely, simpler model encapsulating original dynamics

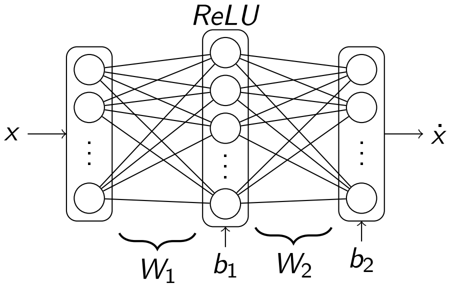


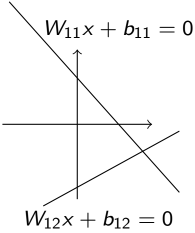
- hybridisation: within $\mathcal{X}$, convert $f(x)$ to hybrid system $\mathcal{H}$ with multiple partitions, where

  each partition has own flow $\tilde{f}(x)$ & transitions to other parts
- more partitions $\rightarrow$ more complex (larger) abstraction

! mesh size & shape important in achieving low error bound ϵ

Model hybridisations as "neural abstractions"[10]

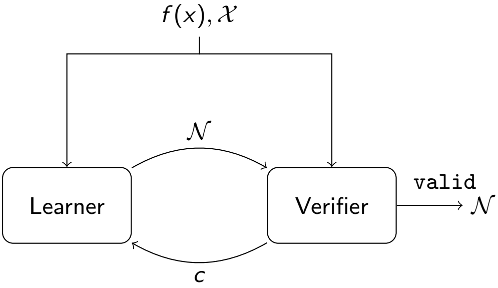




- neural network $\mathcal{N}$ as abstraction $\tilde{f}$ of nonlinear vector field f
- $\mathcal{N}(x) : \mathbb{R}^n \to \mathbb{R}^n$ approximates $f(x)$
- H neurons $\to 2^H$ total modes (at most! but not, really)

[10] A. Abate, A. Edwards, and M. Giacobbe, "Neural Abstractions," NeurIPS, Advances in Neural Information Processing Systems 35, 26432-26447, 2022.

Model hybridisations as neural abstractions - Synthesis

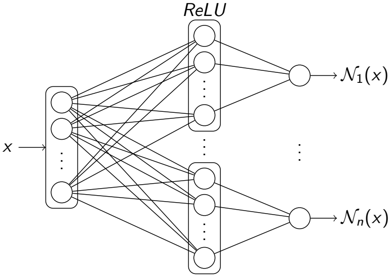


- synthesis of neural abstractions via CEGIS
  1. learn parameters of NN $\mathcal{N}$ w/ MSE loss $\mathcal{L} = \|f(S) - \mathcal{N}(S)\|$, S finite
  2. SMT solver provides formal upper bound ϵ on approximation error:

$$\exists c \in \mathcal{X} \quad s.t. \ \|f(c) - \mathcal{N}(c)\| > \epsilon$$

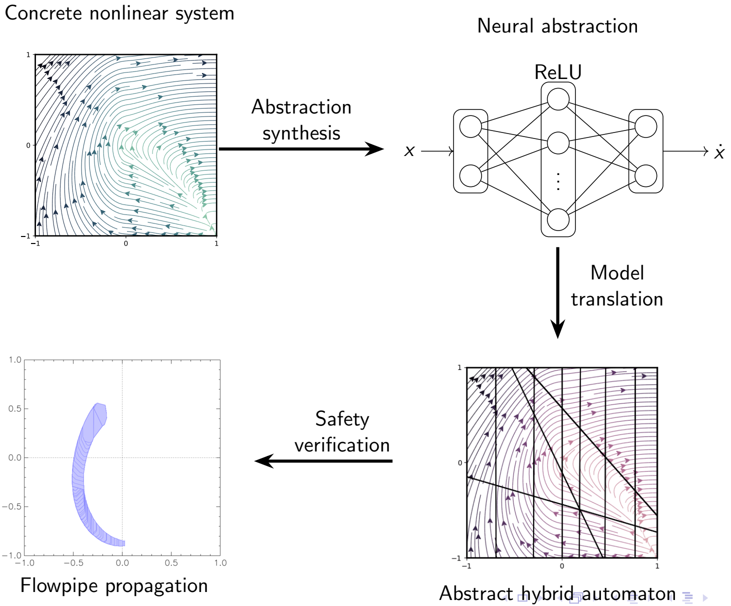
Model hybridisations as neural abstractions - synthesis

- Learner
  - multi-output regression when outputs of different scales is difficult
  - instead of learning $\mathcal{N}(x) : \mathbb{R}^n \to \mathbb{R}^n$, learn n nets $\mathcal{N}_i(x) : \mathbb{R}^n \to \mathbb{R}$



- Verifier: dReal

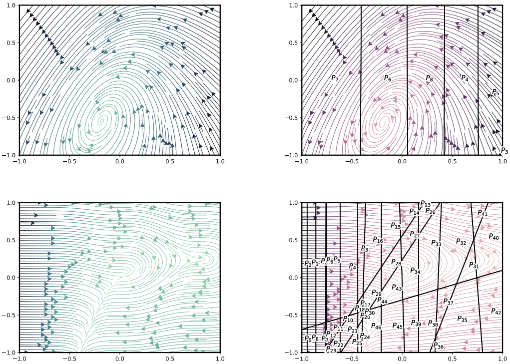
Model hybridisations as neural abstractions - verification



Model hybridisations as neural abstractions - examples

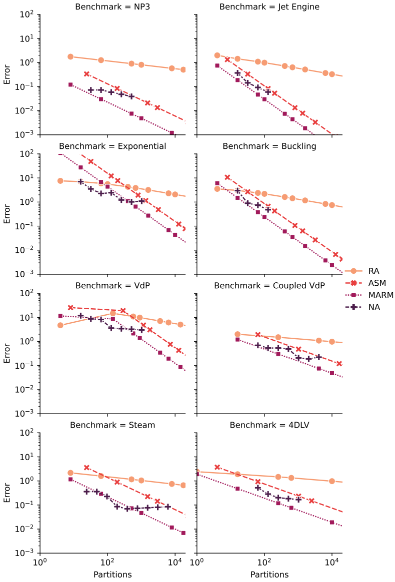
$$\begin{cases} \dot{x} &= -y - 1.5x^2 \\ & \quad -0.5x^3 - 0.5 \\ \dot{y} &= 3x - y \end{cases}$$

$$\begin{cases} \dot{x} &= \exp(-x) + y - 1 \\ \dot{y} &= -\sin(x)^2 \end{cases}$$



concrete model neural abstraction

Model hybridisations as neural abstractions - benchmarks



- error denotes accuracy
  - NA = neural abstractions
- mesh-based hybridisations:
  - RA = rectangular abst. on rectangular mesh
  - ASM = affine abst. on simplicial mesh
  - MARM = multi-affine abst. on rectangular mesh

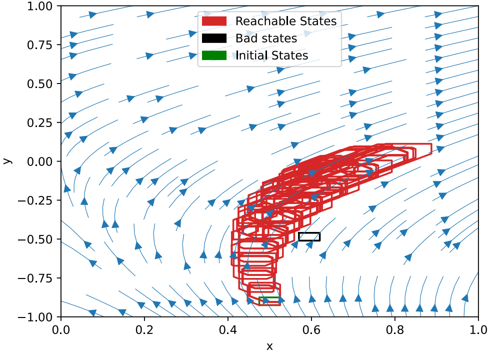
Model hybridisations as neural abstractions - benchmarks

Width Benchmark	2	3	4	5	6	7	8	9	10
4DLV	0.60	0.667	0.567	0.633	0.367	0.0	0.0	0.0	0.0
Buckling	1.0	0.50	0.167	0.333	0.0	0.0	0.0	0.0	0.0
Coupled-VdP	1.0	1.0	1.0	0.933	0.067	0.40	0.20	0.0	0.0
Exponential	1.0	1.0	1.0	1.0	0.933	0.733	0.433	0.0	0.0
Jet Engine	0.533	0.267	0.333	0.067	0.0	0.0	0.0	0.0	0.0
Log	0.867	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
NP3	0.967	1.0	0.967	0.433	0.033	0.0	0.0	0.0	0.0
Steam	0.967	1.0	1.0	1.0	1.0	0.967	0.80	0.967	0.633
VdP	0.967	0.967	1.0	1.0	1.0	0.667	0.567	0.0	0.0

- NA vs ASM: proportion of experiments outperforming ASM hybridisation for equivalent numbers of partitions.
- *"Width"* refers to width of NN

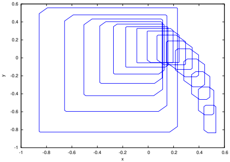
Model hybridisations as neural abstractions - verification

- initial states in $\mathcal{X}_0$, bad states $\mathcal{X}_U$
- ensure no trajectory of f starting in $\mathcal{X}_0$ enters $\mathcal{X}_B$ over time horizon T
- safety verification via over-approximations of reachable set from X_0

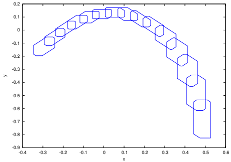


Model hybridisations as neural abstractions - verification

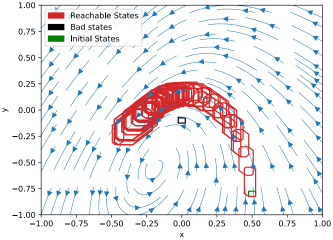
- initial states in $\mathcal{X}_0$, bad states $\mathcal{X}_U$
- ensure no trajectory of f starting in $\mathcal{X}_0$ enters $\mathcal{X}_B$ over time horizon T
- safety verification via over-approximations of reachable set from X_0



(a) Flow* reachable states, $T = 1.5s$, $\tau = 0.1s$ and 2^{nd} order Taylor model



(b) Flow* reachable states, $T = 1.5s$, $\tau = 0.1s$ and 3^{rd} order Taylor model



(c) Neural Abstraction reachable states, $T = 1.5s$, $\tau = 0.1s$

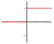
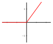
Model hybridisations as neural abstractions - verification

- initial states in $\mathcal{X}_0$, bad states $\mathcal{X}_U$
- ensure no trajectory of f starting in $\mathcal{X}_0$ enters $\mathcal{X}_B$ over time horizon T
- safety verification via over-approximations of reachable set from X_0

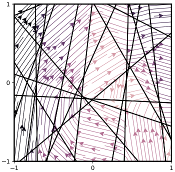
Model	T	**Flow***				**Neural Abstractions**			
		TM	δ	Safety Ver.	t	W	M	Safety Ver.	t
Jet Engine	1.5	10	0.1	Yes	**1.3**	[10, 16]	8	Yes	215
Steam Governor	2.0	10	0.1	Yes	**62**	[12]	29	Yes	219
Exponential	1.0	30	0.05	Blw	1034	[14, 14]	12	Yes	**308**
Water Tank	2.0	-	-	No	-	[12]	6	Yes	**49**
Non-Lipschitz 1	1.4	-	-	No	-	[10]	12	Yes	**19**
Non-Lipschitz 2	1.5	-	-	No	-	[12, 10]	32	Yes	**59**

- ground truth is Safe - note that $M << 2^H$ - SoA is Flow*

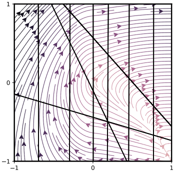
Neural abstractions: alternative templates

	Piecewise constant	**Piecewise affine**	**Nonlinear**
Activation function	 H	 ReLU	 Sigmoid
Equivalent abstract model	LHA I with disturbance	LHA II with disturbance	Nonlinear ODE with disturbance
Safety verification technology	Symbolic model checking	STC algorithm	Taylor models
Safety verification tool	PHAVer	SpaceEx	Flow*

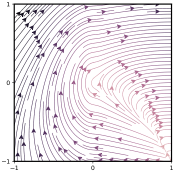
Neural abstractions: alternative templates



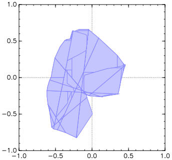
(a) Neural PWC abstraction



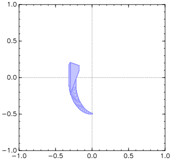
(b) Neural PWA abstraction



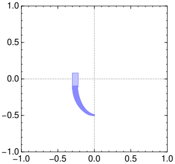
(c) Sigmoidal abstraction.



(a) Flowpipes for neural PWC model. 11.6s



(b) Flowpipe for neural PWA model. 76.5s



(c) Flowpipe for sigmoidal model. 1084.3s

Outline

Outline

Safe autonomous driving - setup[11]

- given: non-linear (3-dim) car dynamics, (2-dim) feedback controller (v, ω) for reference tracking

$$\begin{cases} \frac{d}{dt}x(t) = v\cos\theta(t) \\ \frac{d}{dt}y(t) = v\sin\theta(t) \\ \frac{d}{dt}\theta(t) = \omega \end{cases}$$

[11] A. Abate, S. Bogomolov, A. Edwards, K. Potomkin, S. Soudjani, P. Zuliani, "Safe Reach Set Computation via Neural Barrier Certificates," Under Review, 2023.

Safe autonomous driving - setup[11]

- given: non-linear (3-dim) car dynamics, (2-dim) feedback controller (v, ω) for reference tracking

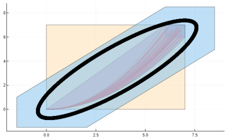
$$\begin{cases} \frac{d}{dt}x(t) = v\cos\theta(t) \\ \frac{d}{dt}y(t) = v\sin\theta(t) \\ \frac{d}{dt}\theta(t) = \omega \end{cases}$$

- goal: *safely* execute closed-loop dynamics, avoiding obstacles in environment
- approach: safe Reach-Set computation with NN-based barrier certificates

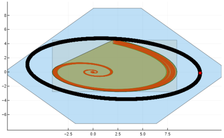
[11] A. Abate, S. Bogomolov, A. Edwards, K. Potomkin, S. Soudjani, P. Zuliani, "Safe Reach Set Computation via Neural Barrier Certificates," Under Review, 2023.

Safe autonomous driving - approach

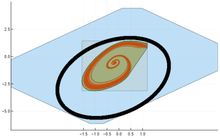
1. from set of initial states X_0, given pair (v, ω), generate over-approximation of reach set via model trajectories over T
2. consider dual of over-approximation as unsafe set X_U
3. generate *sound* barrier certificate with FOSSIL



(a) Linear system with with real eigenvalues.



(b) Spiralling stable dynamics.

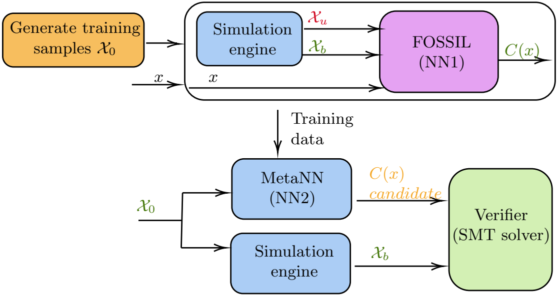


(c) Jet engine system.

Safe autonomous driving - approach

- generalise over sets of initial states X_0, time horizons T, and input pairs (v, ω)
- done via MetaNN (NN2), validated with SMT

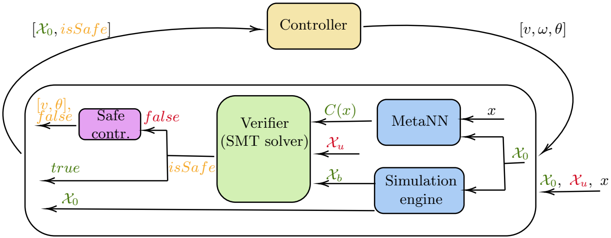
  highly accurate ($\sim 99\%$) outcomes



Safe autonomous driving - case study

- 3D state space, 2 inputs (v, ω) with tracking controller; dynamics:

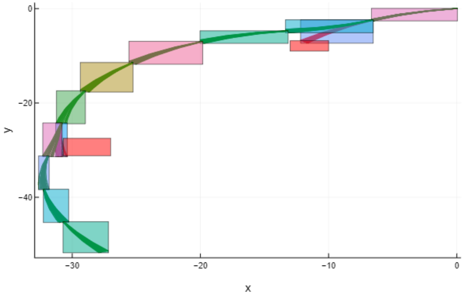
$$\begin{cases} \frac{d}{dt}x(t) = v\cos\theta(t) \\ \frac{d}{dt}y(t) = v\sin\theta(t) \\ \frac{d}{dt}\theta(t) = \omega \end{cases}$$

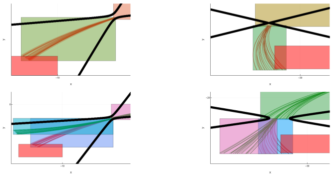


Safe autonomous driving - case study

- 3D state space, 2 inputs (v, ω) with tracking controller; dynamics:

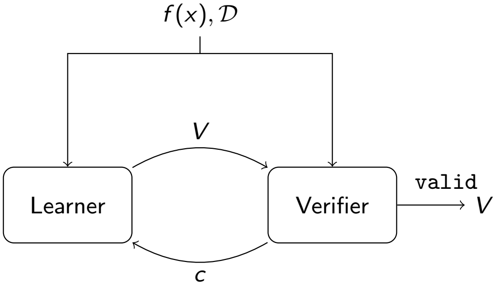
$$\begin{cases} \frac{d}{dt}x(t) = v\cos\theta(t) \\ \frac{d}{dt}y(t) = v\sin\theta(t) \\ \frac{d}{dt}\theta(t) = \omega \end{cases}$$





Ongoing work[12]

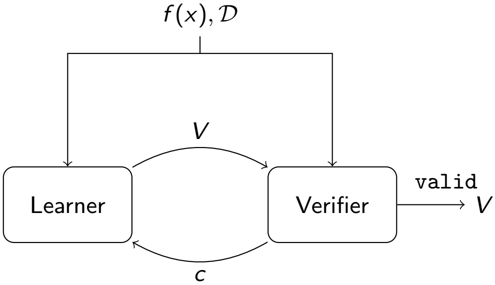
- synthesis of provably-correct controllers (not discussed)



[12] A. Abate, I. Bessa, D. Cattaruzza, L. Cordeiro, C. David, P. Kesseli, D. Kroening and E. Polgreen, "Automated Formal Synthesis of Provably Safe Digital Controllers for Continuous Plants," Acta Informatica, 57(3), 2020.

Ongoing work[12]

- synthesis of provably-correct controllers (not discussed)



- automated, sound, relatively scalable

→ applications in *provably-correct synthesis/safe learning*, for autonomy

- as we speak, we're looking at:
  - controlled, stochastic models
  - synthesis "modulo oracles"

[12] A. Abate, I. Bessa, D. Cattaruzza, L. Cordeiro, C. David, P. Kesseli, D. Kroening and E. Polgreen, "Automated Formal Synthesis of Provably Safe Digital Controllers for Continuous Plants," Acta Informatica, 57(3), 2020.

Thank you for your attention

www.oxcav.org

Selected references

A. Abate, A. Edwards, M. Giacobbe, H. Punchihewa, and D. Roy, "Quantitative Verification With Neural Networks For Probabilistic Programs and Stochastic Systems," arXiv:2301.06136, 2023.

D. Roy, M. Giacobbe, and A. Abate, "Learning Probabilistic Termination Proofs," CAV21, LNCS 12760, pp. 3–26, 2021.

A. Abate, D. Ahmed, A. Edwards, M. Giacobbe and A. Peruffo, "FOSSIL: A Software Tool for the Formal Synthesis of Lyapunov Functions and Barrier Certificates using Neural Networks," HSCC, pp. 1-11, 2021.

A. Abate, D. Ahmed and A. Peruffo, "Automated Formal Synthesis of Neural Barrier Certificates for Dynamical Models," TACAS21, LNCS 12651, pp. 370–388, 2021.

D. Ahmed, A. Peruffo and A. Abate, "Automated and Sound Synthesis of Lyapunov Functions with SMT Solvers," TACAS20, LNCS 12078, pp. 97-114, 2020.

A. Abate, D. Ahmed, M. Giacobbe and A. Peruffo "Automated Formal Synthesis of Lyapunov Neural Networks," IEEE Control Systems Letters, 5 (3), 773-778, 2020.

A. Abate, A. Edwards, and M. Giacobbe, "Neural Abstractions," Advances in Neural Information Processing Systems 35, 26432-26447, 2022.

A. Abate, S. Bogomolov, A. Edwards, K. Potomkin, S. Soudjani, P. Zuliani, "Safe Reach Set Computation via Neural Barrier Certificates," Under Review, 2023.

A. Abate, I. Bessa, D. Cattaruzza, L. Cordeiro, C. David, P. Kesseli, D. Kroening and E. Polgreen, "Automated Formal Synthesis of Provably Safe Digital Controllers for Continuous Plants," Acta Informatica, 57(3), 2020.