Neural State Classification for Hybrid Systems

Nicola Paoletti
Royal Holloway, University of London, UK

JWW: D Phan, T Zhang, SA Smolka, SD Stoller (Stony Brook University) and R Grosu (TU Wien)

Appeared in ATVA 2018, 16th International Symposium on Automated Technology for Verification and Analysis

Stony Brook University – 12 Oct 2018

Agenda

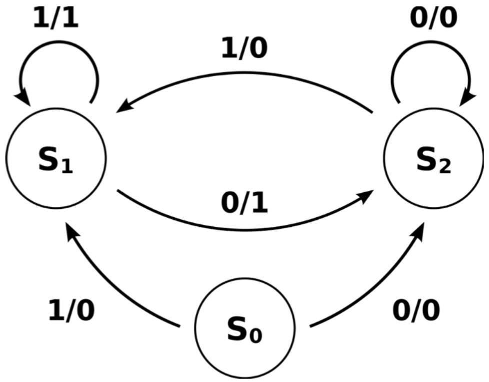
- Background: hybrid systems verification
  - What are HS? Real-world examples
  - Why verify? Safety-critical applications
  - How verify? Formal models, reachability checking, online verification.
- Contribution: Neural State Classification
  - NN-based method to approximate verification results for online analysis
  - Sampling methods
  - Statistical guarantees
  - Reducing errors via falsification
- Experimental results

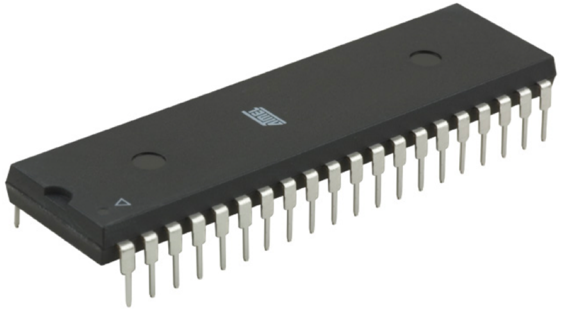
Hybrid systems, informally

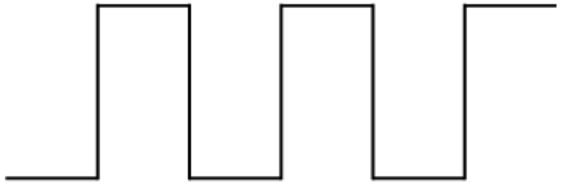
continuous / physical / analog + discrete / digital components

$$\frac{dn}{dt} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n$$

$$\frac{dm}{dt} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m$$

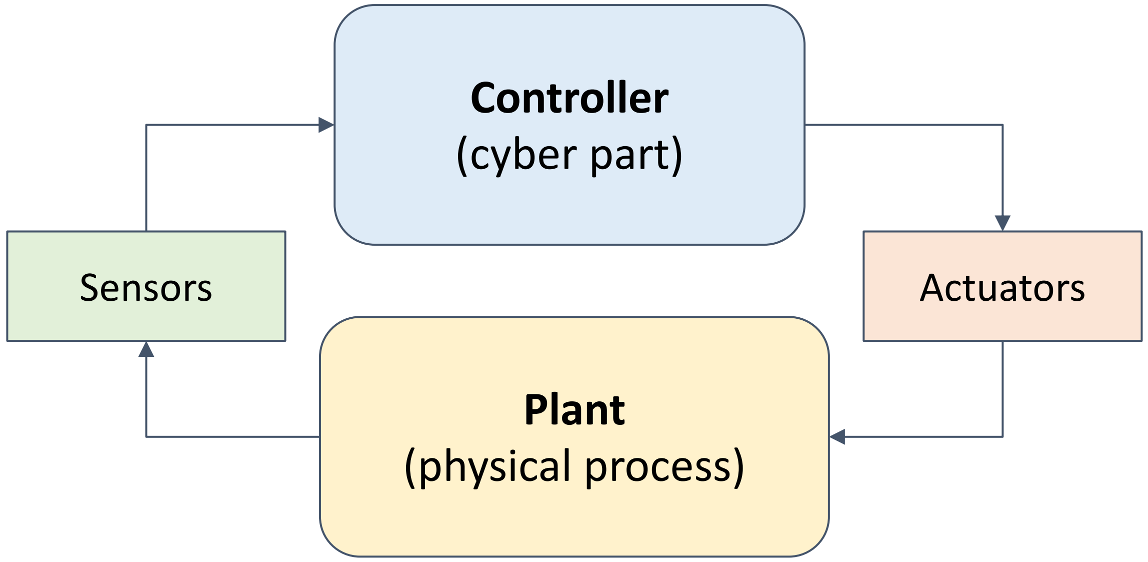






Hybrid systems, examples

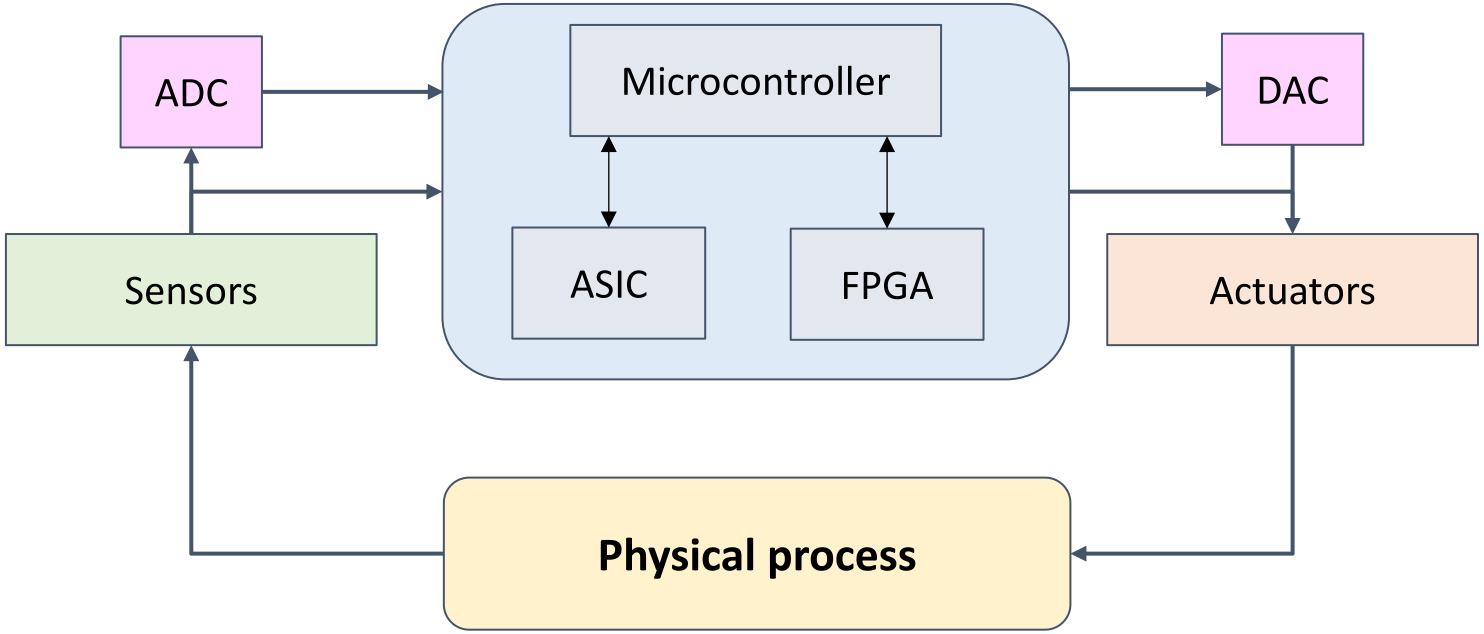
Cyber-physical systems
(aka control systems)

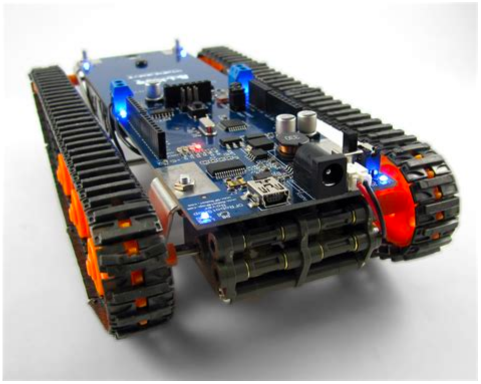


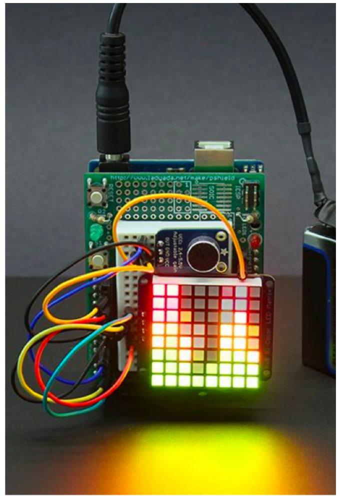
Hybrid systems, examples

Embedded systems

(building blocks of the Internet of Things)

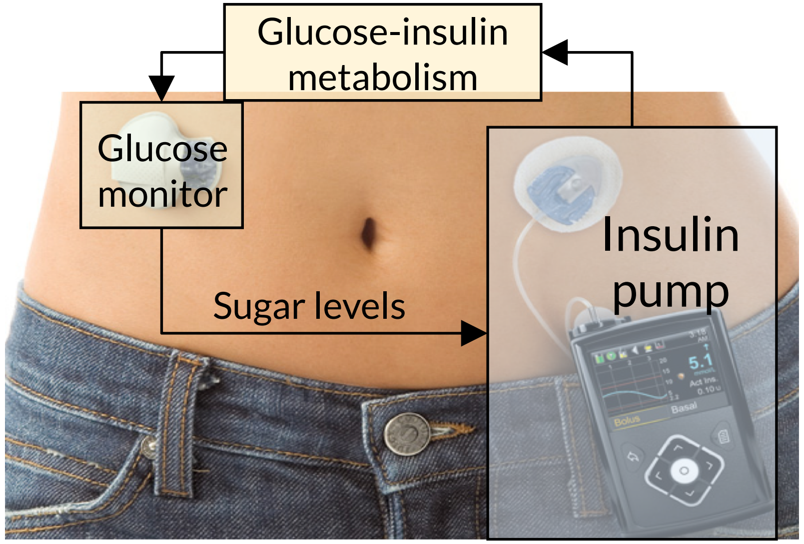




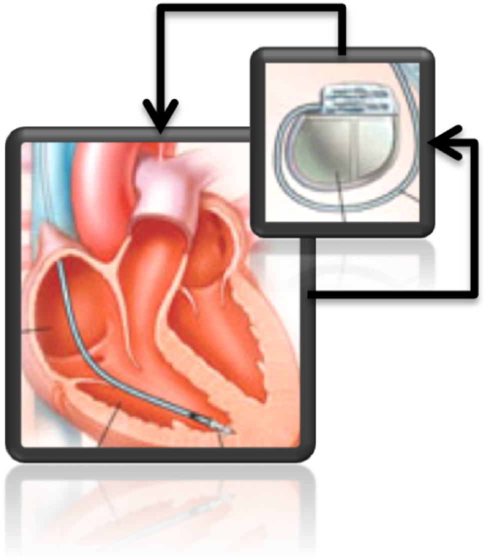


Hybrid systems, examples

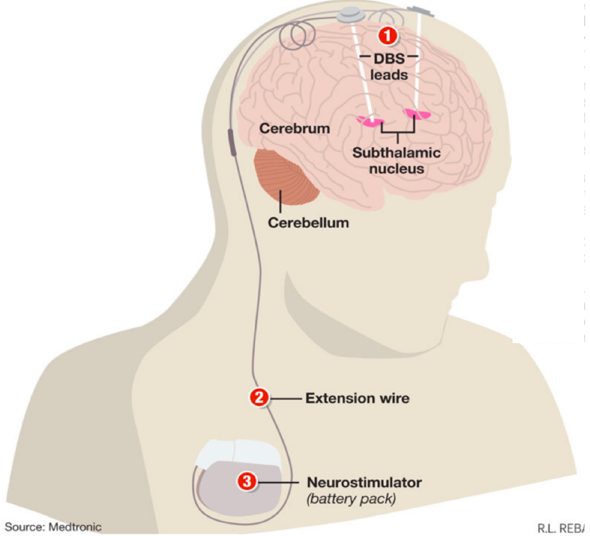
Artificial pancreas



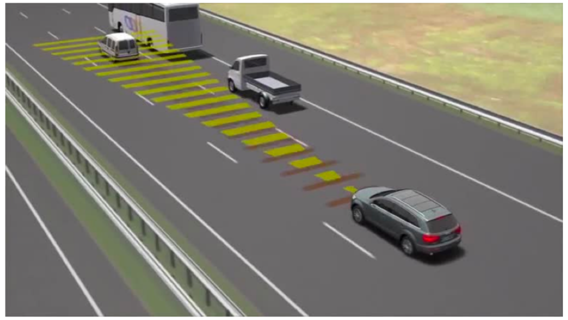
Cardiac devices



Closed-loop deep brain stimulation



Hybrid systems, examples



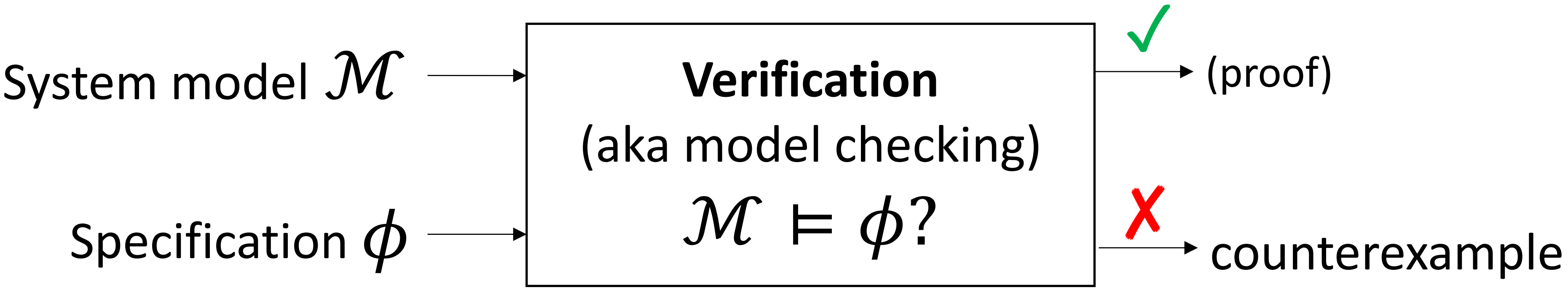
Safety assurance, how?

Hybrid systems are ubiquitous and found in many safety-critical applications

How do we ensure that they work as intended?

e.g., pacemaker always keeps its pacing rate within healthy bounds,
cruise control always maintains safety distance, collision freedom, etc

The verification problem



- Verification is automated and exhaustive (considers all possible system's behaviors)
- $\mathcal{M}$ is a formal, executable model
- ϕ is a correctness property over time
  - Liveness: "at any time, something good must eventually happen"
  - Safety: "something bad will never happen"
  - …

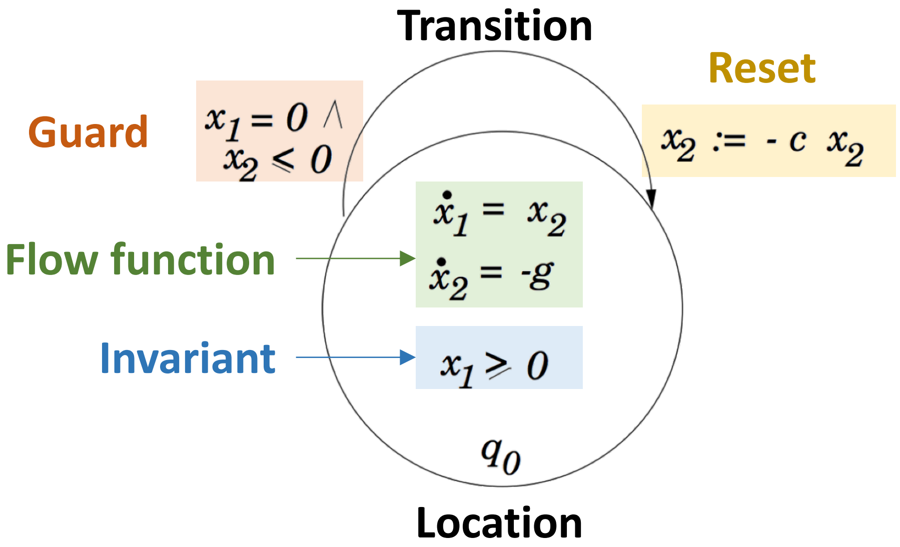
Hybrid systems, formally

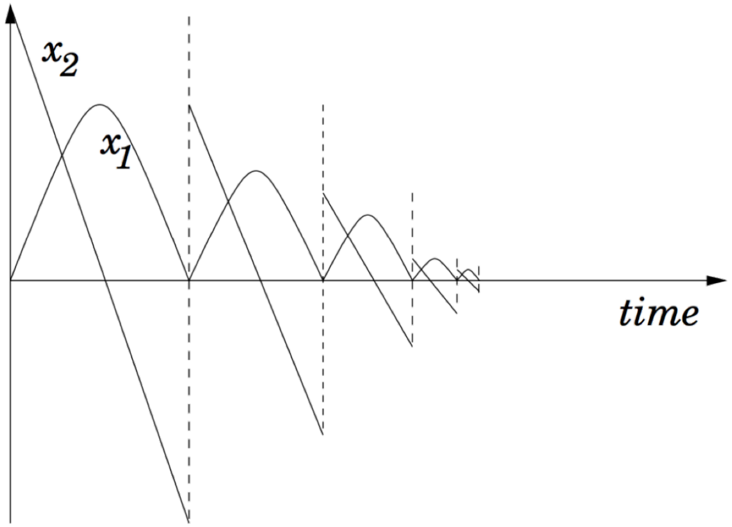
Hybrid automata [Henzinger, LICS 1996]

- Set of discrete locations: Loc
- Set of continuous variables: Var, over $\mathrm{X} \subseteq \mathbb{R}$
- Initial set of states: $Init \subseteq Loc \times X$
- Invariant: $Inv\colon\ Loc \to 2^X$
- Flow function (continuous evolution, ODEs): $Flow\colon Loc \to (X \to X)$
- Transition relation (discrete jumps):
  - Jumps from source location to target location if guard condition holds
  - Updates variables before reaching target

Hybrid automata - Examples

Bouncing ball

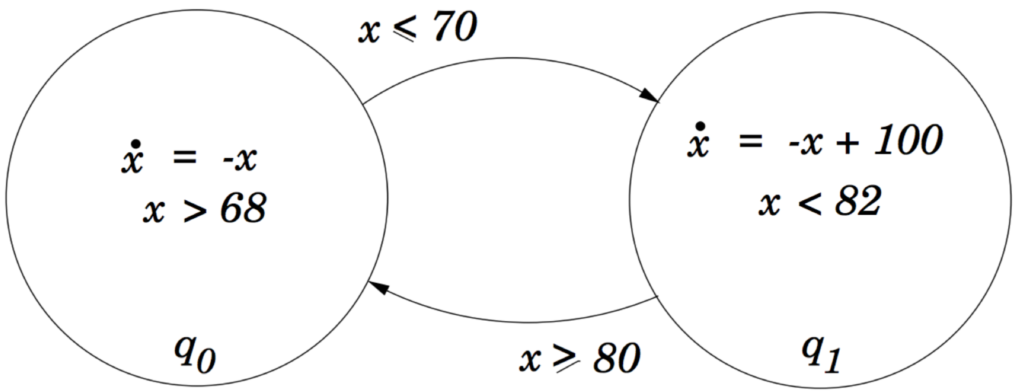


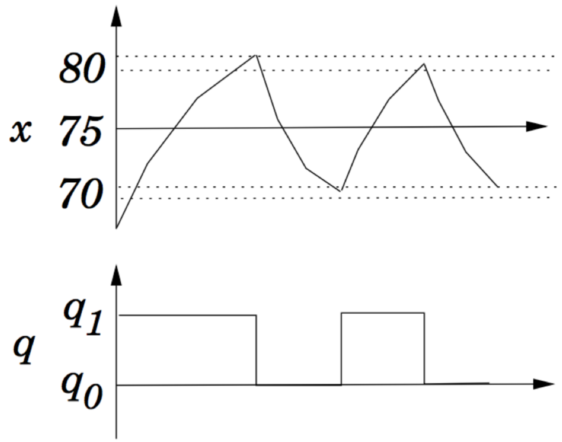


Claire J. Tomlin AA278A lecture notes

Hybrid automata - Examples

Thermostat

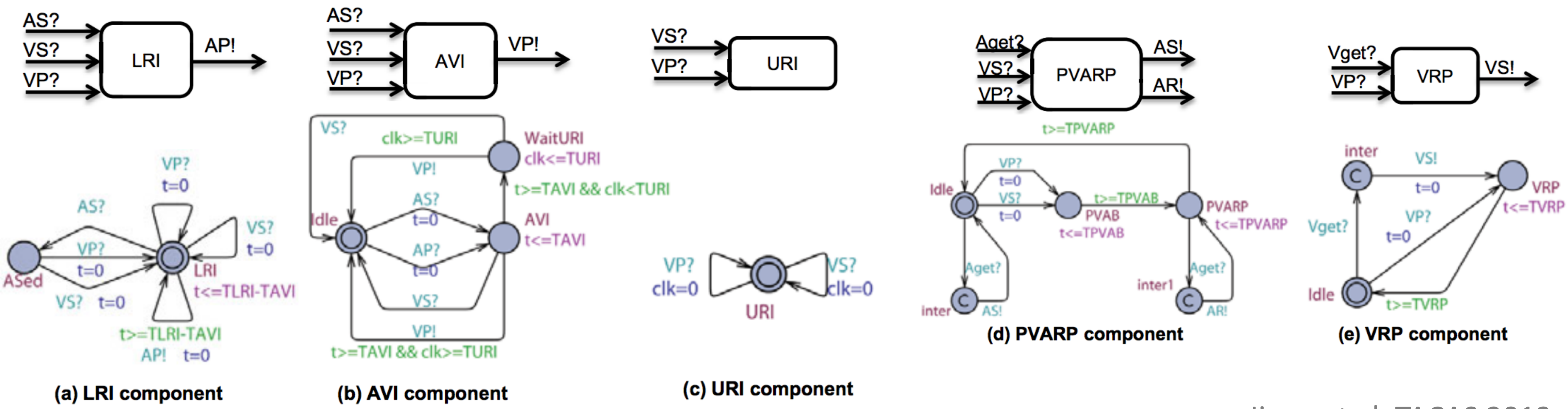




Claire J. Tomlin AA278A lecture notes

Hybrid automata in action

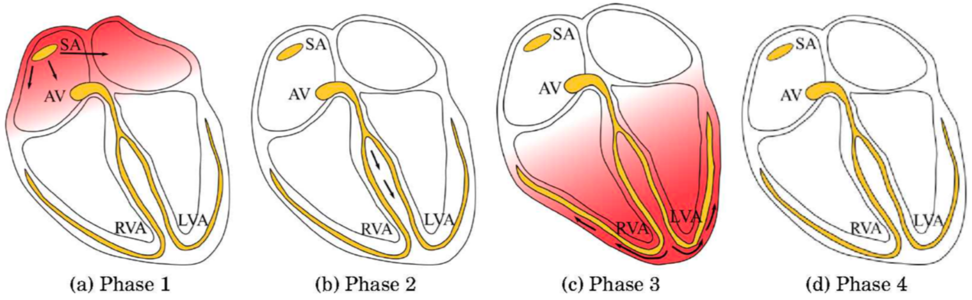
Timed automata network of Boston Scientific dual chamber pacemaker

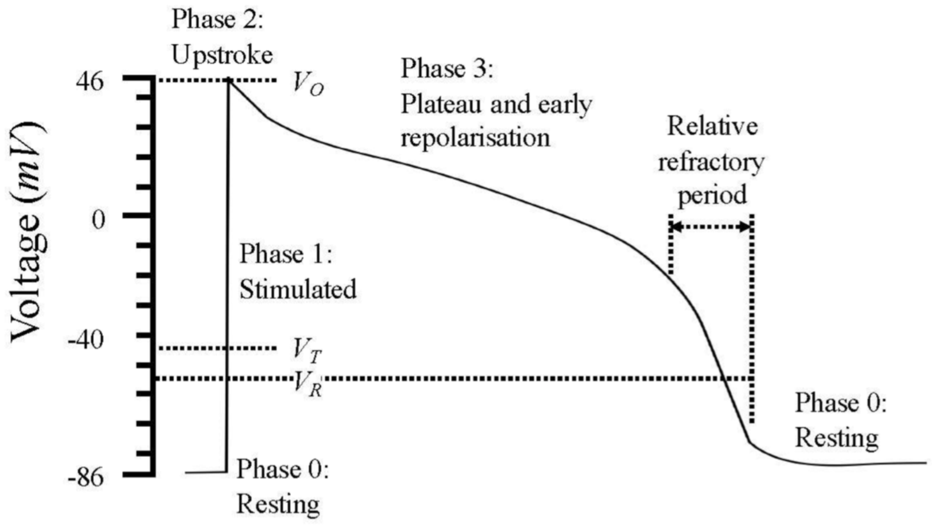


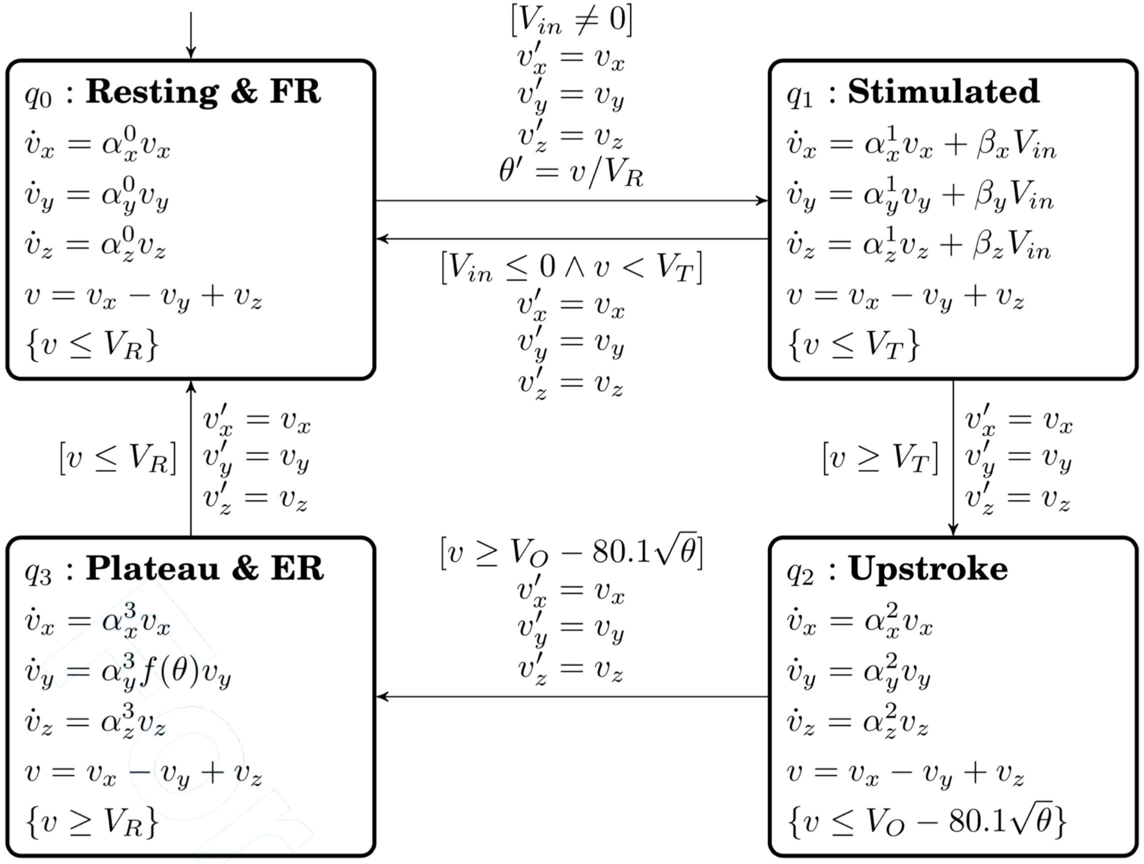
Jiang et al, TACAS 2012

Hybrid automata in action

HA model of cardiac cell action potential (Smolka et al)







Hybrid automata in action

HA model of prostate cancer treatment

Mode 1 (on-treatment)

$\text{flow}_1:$

$$\frac{dx}{dt} = \left(\alpha_x\left(k_1 + \frac{(1-k_1)z}{z+k_2}\right) - \beta_x\left(k_3 + \frac{(1-k_3)z}{z+k_4}\right) - m_1\left(1-\frac{z}{z_0}\right)\right)x$$

$$\frac{dy}{dt} = m_1\left(1-\frac{z}{z_0}\right)x + \left(\alpha_y\left(1-d\,\frac{z}{z_0}\right) - \beta_y\right)y$$

$$\frac{dz}{dt} = \frac{-z}{\tau}$$

$$\frac{dv}{dt} = \left(\alpha_x\left(k_1 + \frac{(1-k_1)z}{z+k_2}\right) - \beta_x\left(k_3 + \frac{(1-k_3)z}{z+k_4}\right) - m_1\left(1-\frac{z}{z_0}\right)\right)x + m_1\left(1-\frac{z}{z_0}\right)x + \left(\alpha_y\left(1-d\,\frac{z}{z_0}\right) - \beta_y\right)y$$

$\text{jump}_{1\to 2}:$

$$x + y \le r_0 \wedge \frac{dx}{dt} + \frac{dy}{dt} < 0$$

$\text{jump}_{2\to 1}:$

$$x + y \ge r_1 \wedge \frac{dx}{dt} + \frac{dy}{dt} > 0$$

Mode 2 (off-treatment)

$\text{flow}_2:$

$$\frac{dx}{dt} = \left(\alpha_x\left(k_1 + \frac{(1-k_1)z}{z+k_2}\right) - \beta_x\left(k_3 + \frac{(1-k_3)z}{z+k_4}\right) - m_1\left(1-\frac{z}{z_0}\right)\right)x$$

$$\frac{dy}{dt} = m_1\left(1-\frac{z}{z_0}\right)x + \left(\alpha_y\left(1-d\,\frac{z}{z_0}\right) - \beta_y\right)y$$

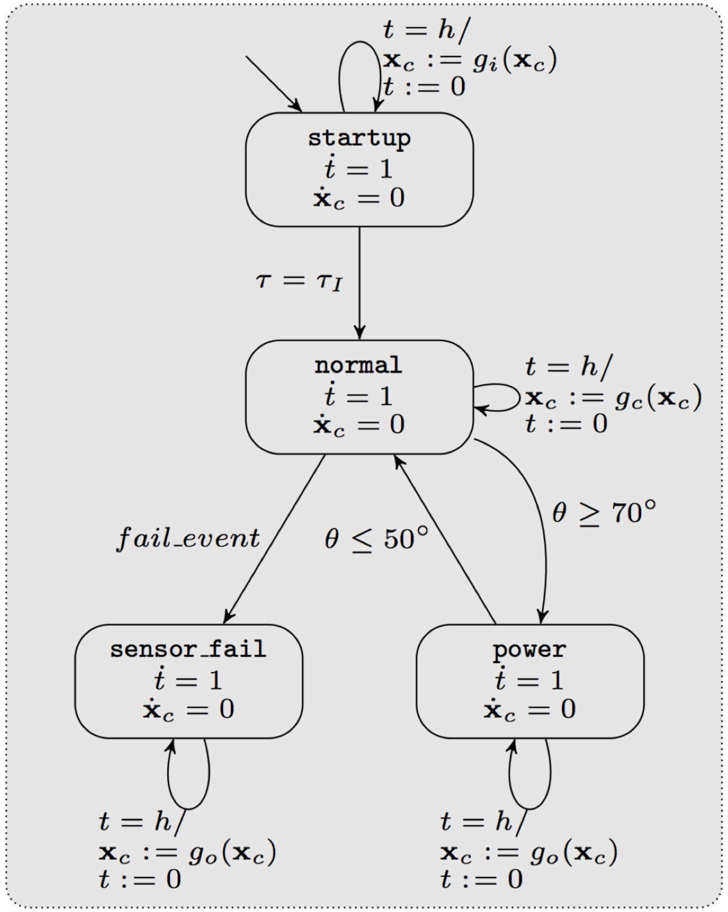
$$\frac{dz}{dt} = \frac{z_0 - z}{\tau}$$

$$\frac{dv}{dt} = \left(\alpha_x\left(k_1 + \frac{(1-k_1)z}{z+k_2}\right) - \beta_x\left(k_3 + \frac{(1-k_3)z}{z+k_4}\right) - m_1\left(1-\frac{z}{z_0}\right)\right)x + m_1\left(1-\frac{z}{z_0}\right)x + \left(\alpha_y\left(1-d\,\frac{z}{z_0}\right) - \beta_y\right)y$$

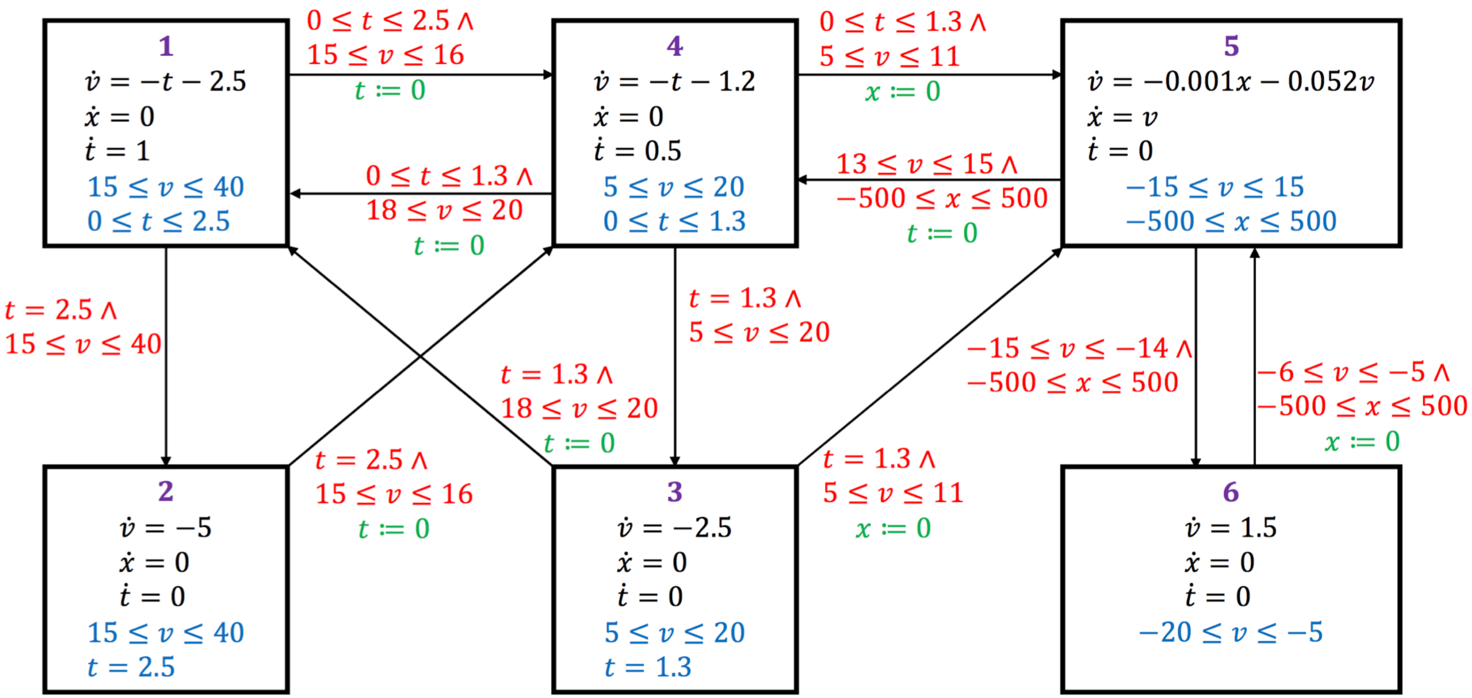
Ideta et al, J. Nonlinear Sci. 18 (2008)

Hybrid automata in action

Powertain system by Toyota



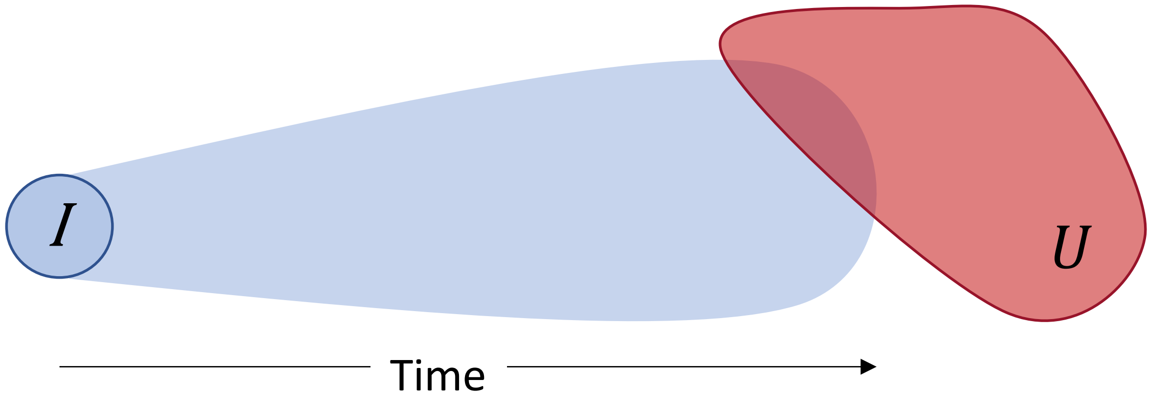
Cruise control HA model



Hybrid automata verification

HA verification problem usually formulated as reachability

(Time-bounded) reachability:
can an HA $\mathcal{M}$, starting in an initial region I, reach a state $u \in U$ (within time T)?

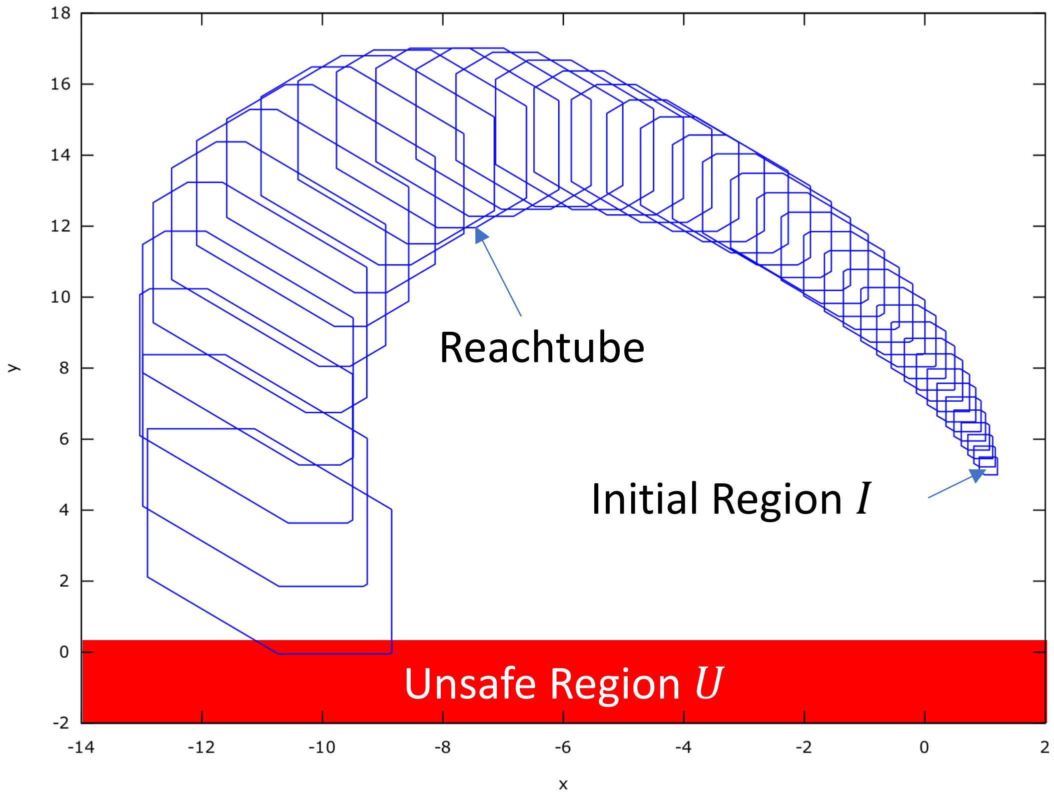


Both bounded and unbounded versions are undecidable

[Henzinger et al, *JCSS 57 1* (1998); Brihaye et al, *ICALP* (2011)]

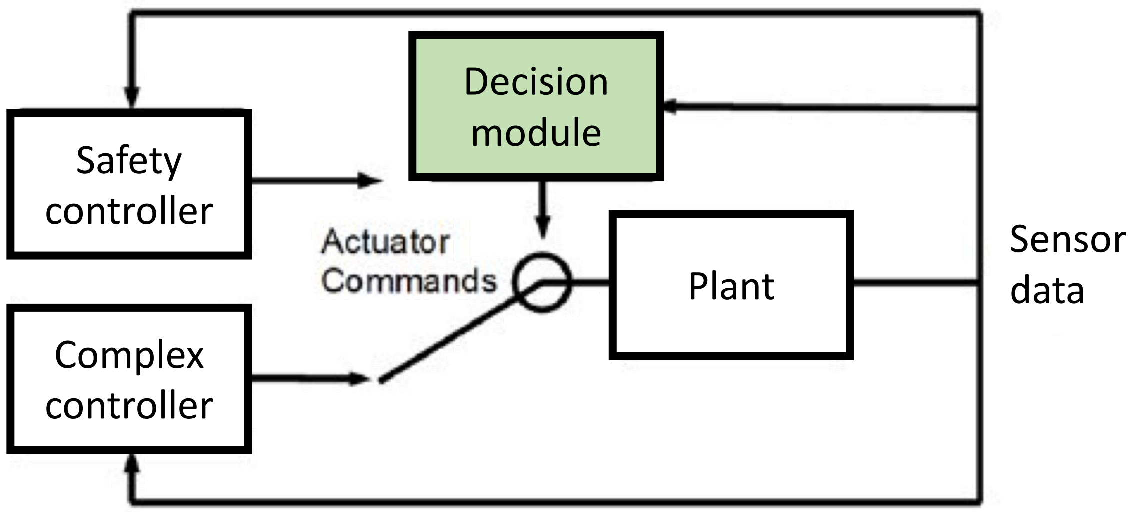
Reachability checkers for HAs

- Over-approximate the set of states reachable from the initial region
  - Given initial region I of an HA $\mathcal{M}$ and a time bound T, compute $ReachTube(\mathcal{M}, I, T)$
  - Check if $ReachTube(\mathcal{M}, I, T)$ intersects the unsafe region U
    - No: 100% safe
    - Yes: *maybe* unsafe, s.t. false positives
- Tools: HyCreate, Flow*, SpaceEx, iSAT, dReal, etc.
- HA reachability is computationally expensive



Motivation - Online model checking (OMC)

- OMC – *predicting at runtime future violations from current state* – is as important as offline model verification for HSs and CPSs
  - switch to fail-safe operation mode when failure is imminent (e.g. Simplex architecture of [Sha, *IEEE Software* (2001)])



Motivation - Online model checking (OMC)

Offline

- Reachability from a (large) region
- One-off analysis, potentially long time horizons (blow-up of over-approximation)
- No hard time constraints
- Controlled settings
  - Model is ground truth

Online

- Reachability from a single state
- Analysis run periodically → short time horizons
- Strict time constraints
- Less predictable settings
  - Real system might differ from model
  - Noisy observations

Motivation - Online model checking (OMC)

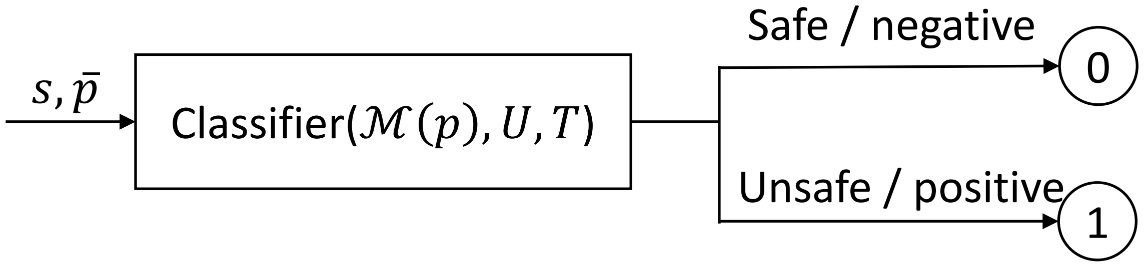
- OMC focus is on reachability from a single state, and not from a (large) region
- OMC runs the the analysis periodically → short time horizons
- Runtime settings are less predictable

Does OMC need fully-fledged reachability checking?

- We rather need methods that can work under real-time constraints
  - Reachability checking is too expensive for online analysis

State Classification Problem (SCP)

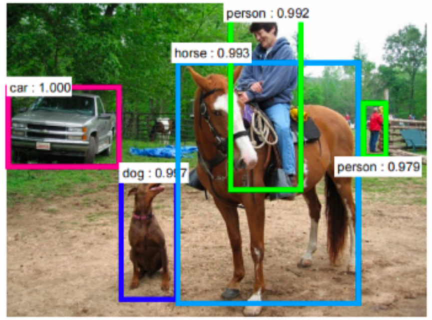
- We want a function that, given HA $\mathcal{M}$ with state space S, set of unsafe states U, and time bound T, classifies every state $s \in S$ as either *positive* or *negative*
  - s is *positive* if $\mathcal{M}$, starting in s, can reach a state in U within time T;
  - *negative* o/w



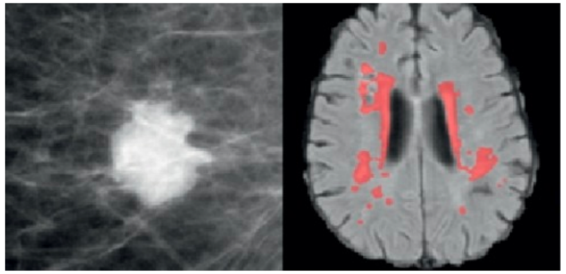
- We call such a function a *state classifier,* a solution to the SCP
- $\mathcal{M}$ can be parameterized by a set of parameters p

Neural networks (NNs) as state classifiers

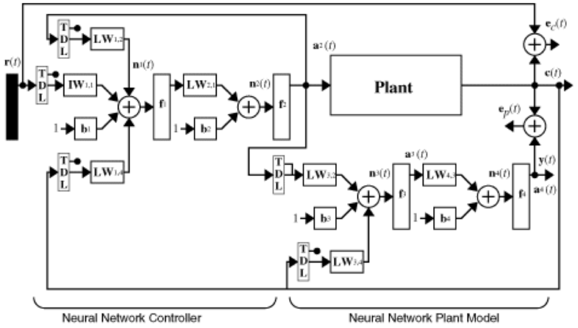
(Deep) NNs are extremely successful at complex classification and regression tasks



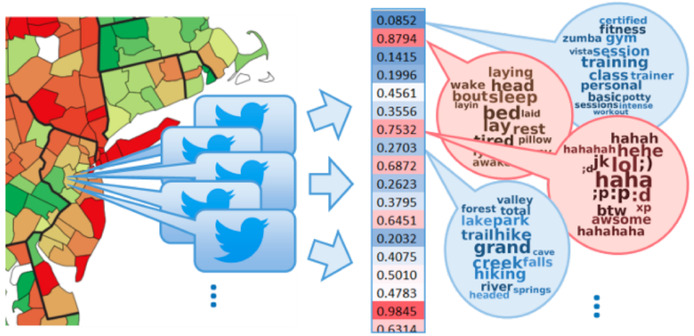
Object detection



Classification of tumor and diseases from medical images



System identification and control

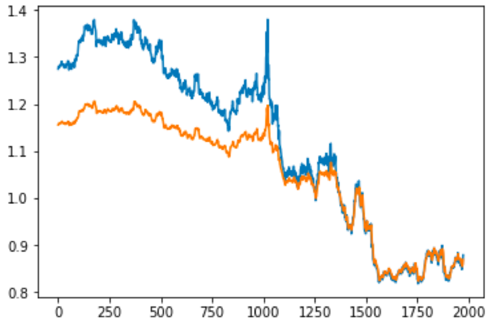


Natural language processing, sentiment analysis

Image credits: H. Andrew Schwartz

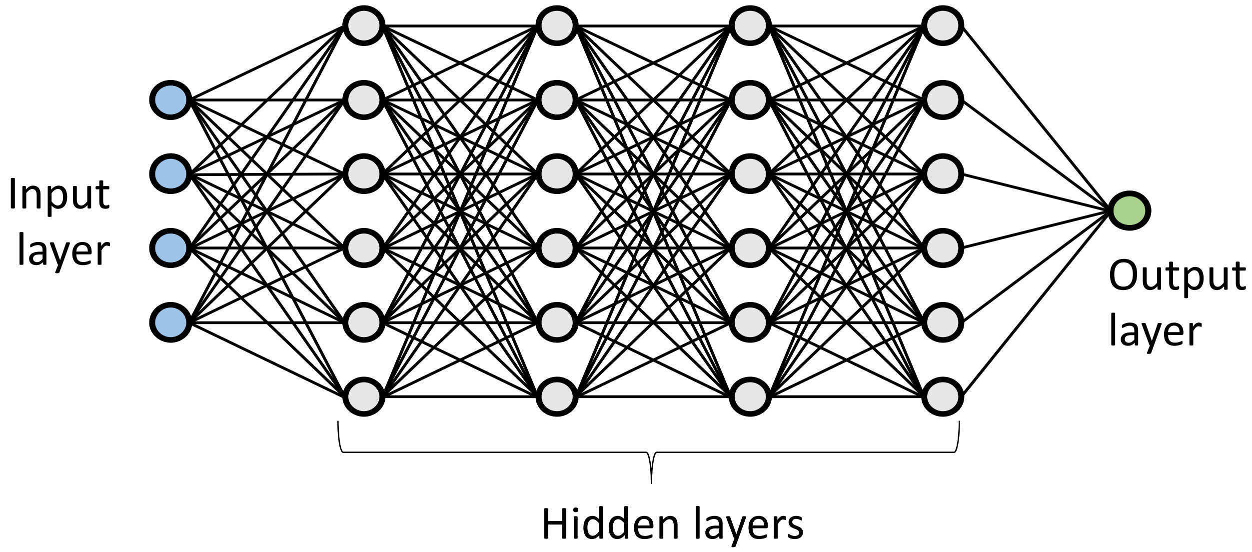
?

Verification



Time-series analysis and prediction

Feedforward neural networks



$$F = \mathrm{f}_l \circ \mathrm{f}_{l-1} \circ \ldots \circ \mathrm{f}_1 \circ \mathrm{f}_0$$

$$\underbrace{\mathrm{f}_i}_{\text{Output of layer } i}\left(\mathrm{p}_{i-1}\right) = \underbrace{\mathrm{g}_i}_{\text{Activation function (sigmoid, ReLU, …)}}\left(\underbrace{\mathrm{W}_{i,i-1}}_{\text{weights}} \cdot \underbrace{\mathrm{p}_{i-1}}_{\text{Output of layer } i\text{-}1} + \underbrace{\mathrm{b}_i}_{\text{biases}}\right), \quad i = 1, \ldots, l$$

Supervised learning of NN =
finding weights and biases that maximize the fit between predictions and training data

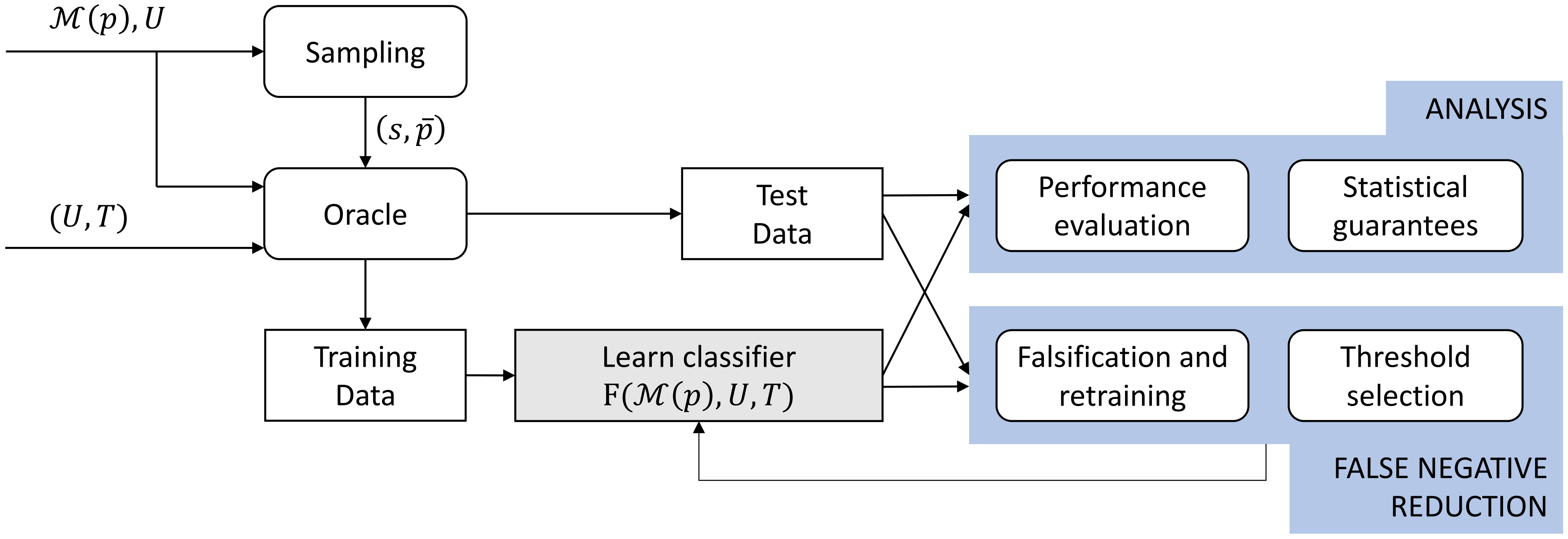
Neural networks (NNs) as state classifiers

- *Can we train a NN to learn a HA reachability function, i.e., solve the SCP?*
- In principle, **YES:** NNs are universal approximators [Hornik et al, *Neural networks 2(5)* (1989)]
- In practice, good accuracy but prediction errors can't be avoided
- Trained NN state classifier runs in **constant time** -> suitable for online model checking

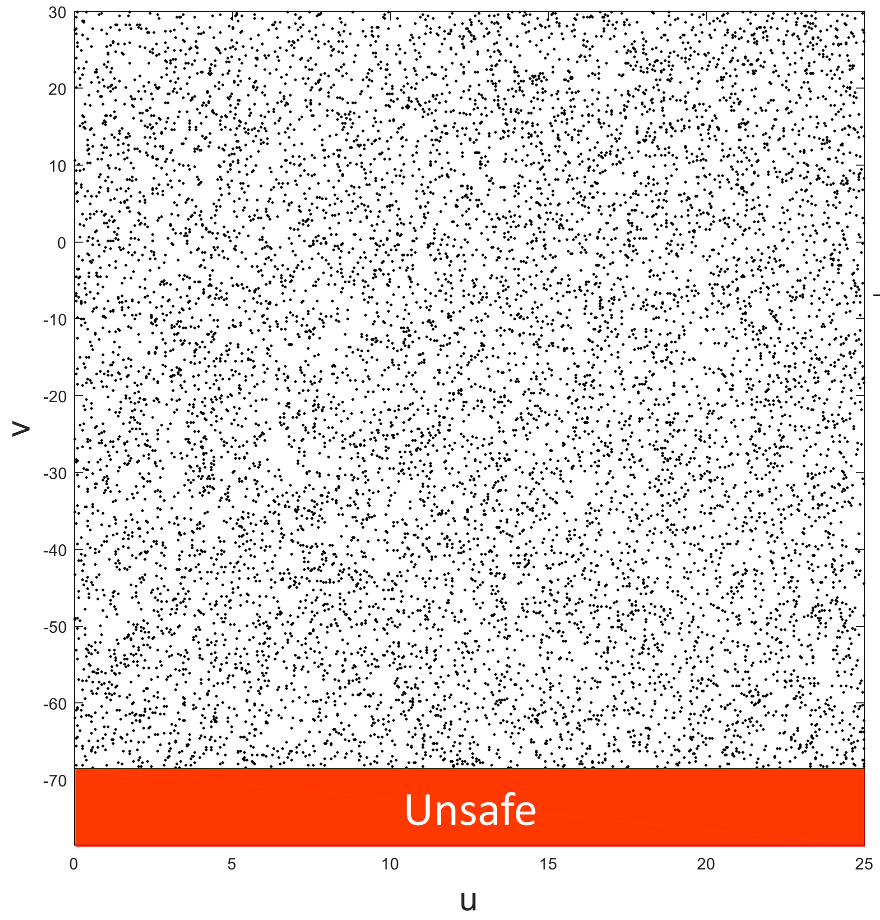
Two kinds of errors in **neural state classification**:

- **False positives:** a negative state is predicted to be positive (conservative decision)
- **False negatives:** a positive state is predicted to be negative (can compromise system's safety!)

Neural State Classification (NSC)

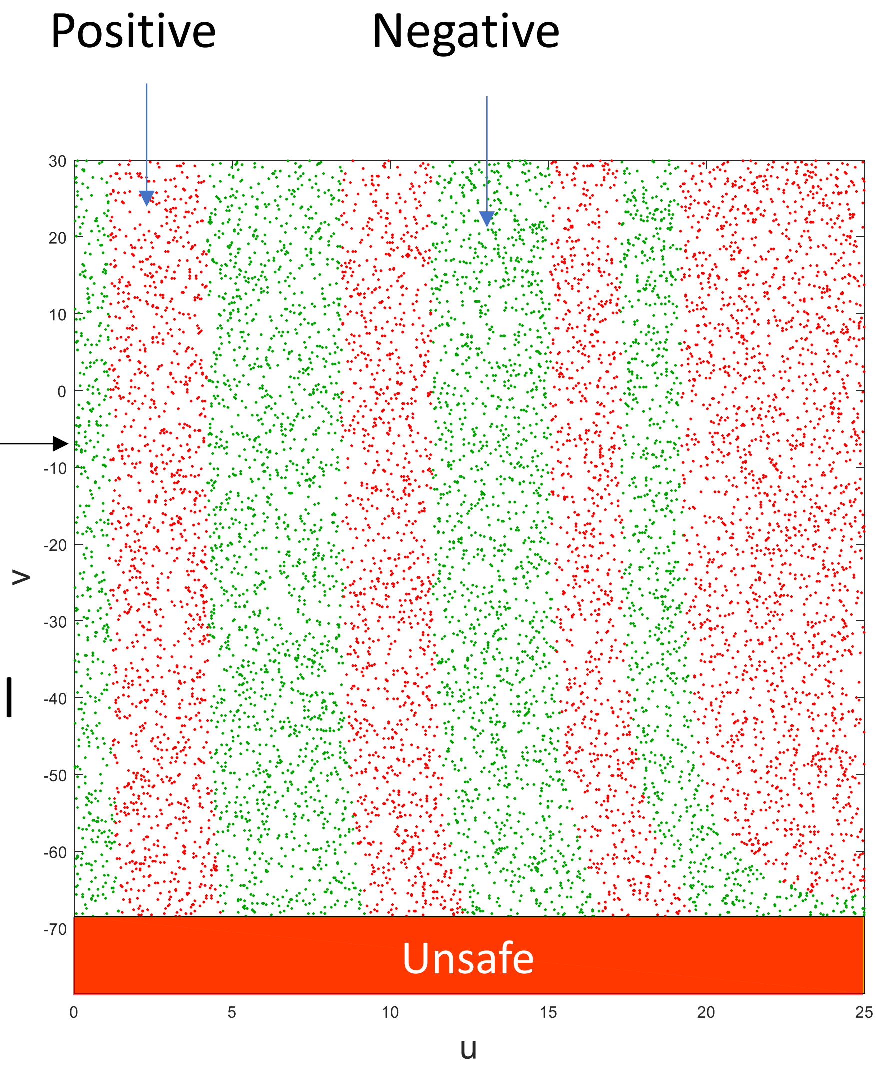


Oracles

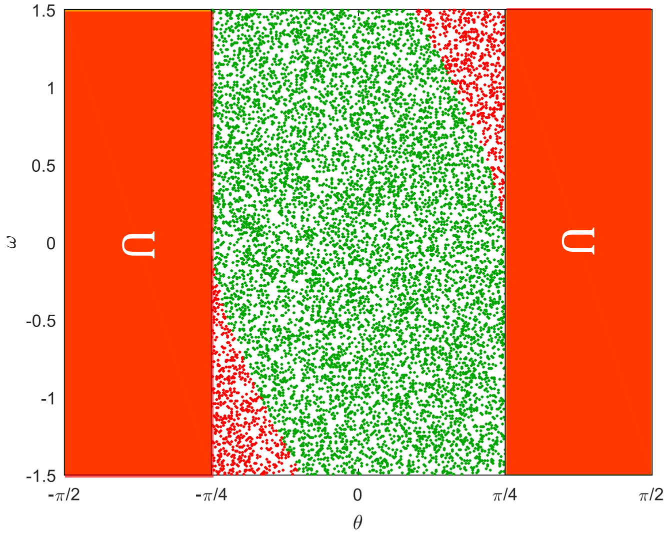


Oracle

- Simulator (deterministic)
- Reachability checker (dReal [Gao et al, *CADE* (2013)])
- Backwards simulator

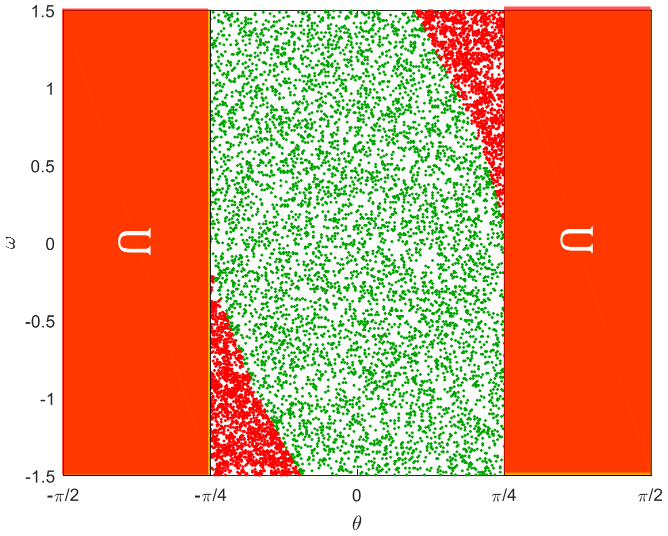


Sampling methods



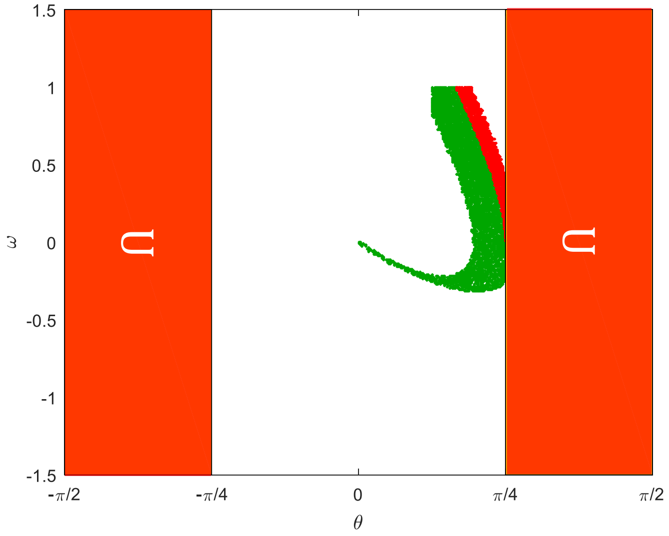
Uniform Sampling

- all states equally important



Balanced Sampling

- balanced number of pos. and neg. samples
- suitable when unsafe set U is small
- based on **backwards HA simulation**

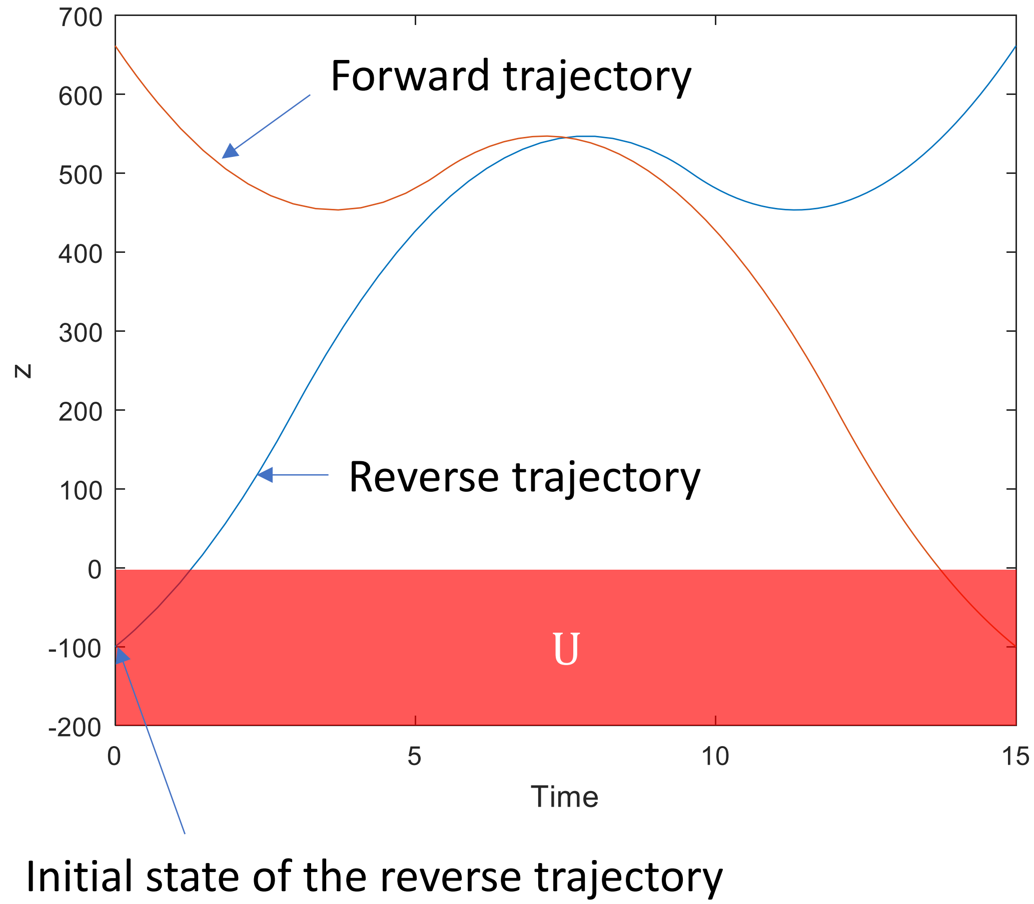


Dynamics-Aware Sampling

- reflects the likelihood of visiting a state from the initial region
- based on estimating state distribution from random HA runs

Backwards simulator

- For generating arbitrarily many positive samples for a balanced dataset
- Given an unsafe state $u \in U$, simulate $\overleftarrow{\mathcal{M}}$, the *reverse HA* of $\mathcal{M}$, for up to time T
- Every state in the reverse trajectory is positive
- We provide a constructive definition of reverse HA and prove its correctness (more general than [Henzinger et al, *STOC* (1995)] for rectangular automata)



Statistical guarantees via hypothesis testing

- We don’t just want empirical performance, but also to establish guaranteed performance requirements
  - Accuracy (probability of correct prediction): $P_A \geq \theta_A$
  - FN rate (probability that prediction is an FN): $P_{FN} \leq \theta_{FN}$
- Deriving absolute guarantees is infeasible
- statistical guarantees (precise up to a small error probability) via the *sequential probability ratio test* (SPRT) [Wald and Wolfowitz (1948)]

Sequential probability ratio test

- *Sequential* means that we only need the number of test samples necessary to decide the threshold
- Precise up to arbitrary error bounds α (prob of type-I errors) and β (prob of type-II errors)
- To ensure both bounds, the test $P \geq \Theta$ vs $P < \Theta$ is relaxed to
  - $H_0 \colon P \geq p_0$ vs $H_1 \colon P \leq \mathrm{p}_1$ where $p_1 < \Theta < p_0$ (but both close to Θ)
- H_0 accepted if $\frac{p_{1m}}{p_{0m}} \leq \frac{1-\beta}{\alpha}$; H_1 accepted if $\frac{p_{1m}}{p_{0m}} \geq \frac{\beta}{1-\alpha}$
- $\frac{p_{1m}}{p_{0m}} = \frac{p_1^{t_m}(1-p_1)^{f_m}}{p_0^{t_m}(1-p_0)^{f_m}}$, t_m: # pos. samples; f_m: # neg. samples

Reducing FN rate via falsification

- Make the classifier more conservative (reduce FN) through re-training with new FN samples
  - **Dual of CEGAR** [Clarke et al, CAV (2000)]**:** CEGAR refines an overapproximation using counterexamples (FPs)
- FNs found via a falsifier / adversarial sampling, an algorithm that finds states maximizing the discrepancy between predictions and true labels

$$\max_{s \in S} |b(s) - F(s)|$$

True label of *s* Network prediction for *s*

Input: classifier (NN) F,
training samples D
Output: ”conservative” classifier F
do
- $\widehat{FN} \leftarrow$ subset of the true FN set of F
  /*found via falsifier (genetic alg)*/
- $D \leftarrow D \cup \widehat{FN}$
- $F \leftarrow$ **train**(D)

while $\widehat{FN} \neq \emptyset$ **or** *max_iter*

Iterative falsification / re-training algorithm

Reducing FN rate via falsification

- The *algorithm converges to an empty set of FNs with high probability* (proof based on bounds on generalization error of ML models [Vapnik, *The nature of statistical learning theory* (2013)])

$$\boxed{\textit{for all } \eta \in (0,1), \Pr(\lim_{k\to\infty} FN_k = \emptyset) \geq 1-\eta}$$

under assumptions that:

- Falsifier always finds a FN if it exists
- Classifier doesn't make mistakes on positive training samples
- FP rate for test data is not below that for training data

Input: classifier (NN) F, training samples D
Output: "conservative" classifier F
do
- $\widehat{FN} \leftarrow$ subset of the true FN set of F /*found via falsifier (genetic alg)*/
- $D \leftarrow D \cup \widehat{FN}$
- $F \leftarrow$ **train**(D)

while $\widehat{FN} \neq \emptyset$ **or** *max_iter*

Iterative falsification / re-training algorithm

Experimental design

Hybrid system benchmark:

- Spiking neuron
- Inverted pendulum
- Quadcopter dynamics
- Cruise control
- Powertrain
- Helicopter

State classifier models:

- Feed-forward deep NNs (3 hidden layers, 10 neurons each, sigmoid and ReLU)
- Feed-forward shallow NNs (1 hidden layer, 20 neurons, sigmoid)
- Support Vector Machines (SVMs)
- Binary Decision Trees (BDTs)
- Nearest neighbor (returns label of closest training sample)

Accuracy and FNs

	Neuron		Pendulum		Quadcopter		Cruise		Powertrain		Helicopter		
	Acc	FN	Acc	FN	Acc	FN	Acc	FN	Acc	FN	Acc	FN	
DNN-S	**99.81**	**0.1**	**99.98**	**0**	99.83	0.1	99.95	0.01	**96.68**	1.28	**98.49**	**0.84**	**Uniform**
DNN-R	99.52	0.29	99.93	0.04	**99.89**	**0.06**	**99.98**	**0**	96.21	**1.08**	98	0.96	
SNN	99.17	0.43	99.81	**0**	99.85	0.08	99.84	0.15	96.02	1.37	97.69	1.25	
SVM	98.73	0.75	99.84	**0**	97.33	0.69	99.88	0.1	92.26	3.48	95.58	2.42	
BDT	99.3	0.37	99.6	0.17	99.52	0.2	99.84	0.08	95.59	2.19	80.07	9.8	
NBOR	97.03	1.22	99.69	0.14	99.53	0.25	99.49	0.33	71.44	14.51	67.39	16.98	
	Acc	FN	Acc	FN	Acc	FN	Acc	FN	Acc	FN	Acc	FN	
DNN-S	**99.83**	**0.12**	**99.89**	**0**	**99.82**	0.04	99.94	**0**	**97.2**	**0.86**	**98.24**	**0.79**	**Balanced**
DNN-R	99.48	0.24	99.63	0.01	99.67	0.09	**99.95**	**0**	96.07	1.24	97.91	1.2	
SNN	98.89	0.69	99.2	**0**	99.49	**0.01**	99.6	**0**	95.21	1.79	97.58	1.16	
SVM	98.63	0.78	99.37	**0**	96.93	0.2	99.61	**0**	91.84	3.3	95.36	1.85	
BDT	99.07	0.45	99.46	0.05	99.36	0.22	99.9	0.03	95.86	2.4	79.03	10.26	
NBOR	96.95	1.62	99.51	0.04	99.11	0.56	99.47	0.11	71.33	13.99	65.18	17.48	

20K training samples,
10K test samples

DNN-S: Sigmoid DNN

SVM: Support Vector Machine

SNN: Shallow NN

DNN-R: ReLU DNN

BDT: Binary Decision Tree

SNN: Shallow NN

Accuracy and FNs

	Neuron		Pendulum		Quadcopter		Cruise		**Powertrain**		**Helicopter**		
	Acc	FN	Acc	FN	Acc	FN	Acc	FN	Acc	FN	Acc	FN	
If we increase training samples from 20K to 1M:									**99.25**	**0.33**	**99.92**	**0.04**	**Uniform**
DNN-R	99.52	0.29	99.93	0.04	**99.89**	**0.06**	**99.98**	**0**	96.21	**1.08**	98	0.96	
SNN	99.17	0.43	99.81	0	99.85	0.08	99.84	0.15	96.02	1.37	97.69	1.25	
SVM	98.73	0.75	99.84	0	97.33	0.69	99.88	0.1	92.26	3.48	95.58	2.42	
BDT	99.3	0.37	99.6	0.17	99.52	0.2	99.84	0.08	95.59	2.19	80.07	9.8	
NBOR	97.03	1.22	99.69	0.14	99.53	0.25	99.49	0.33	71.44	14.51	67.39	16.98	
	Acc	FN	Acc	FN	Acc	FN	Acc	FN	Acc	FN	Acc	FN	**Balanced**
DNN-S	**99.83**	**0.12**	**99.89**	**0**	**99.82**	0.04	99.94	**0**	**97.2**	**0.86**	**98.24**	**0.79**	
DNN-R	99.48	0.24	99.63	0.01	99.67	0.09	**99.95**	**0**	96.07	1.24	97.91	1.2	
SNN	98.89	0.69	99.2	**0**	99.49	**0.01**	99.6	**0**	95.21	1.79	97.58	1.16	
SVM	98.63	0.78	99.37	**0**	96.93	0.2	99.61	**0**	91.84	3.3	95.36	1.85	
BDT	99.07	0.45	99.46	0.05	99.36	0.22	99.9	0.03	95.86	2.4	79.03	10.26	
NBOR	96.95	1.62	99.51	0.04	99.11	0.56	99.47	0.11	71.33	13.99	65.18	17.48	

20K training samples,
10K test samples

DNN-S: Sigmoid DNN

SVM: Support Vector Machine

SNN: Shallow NN

DNN-R: ReLU DNN

BDT: Binary Decision Tree

SNN: Shallow NN

Statistical guarantees based on SPRT

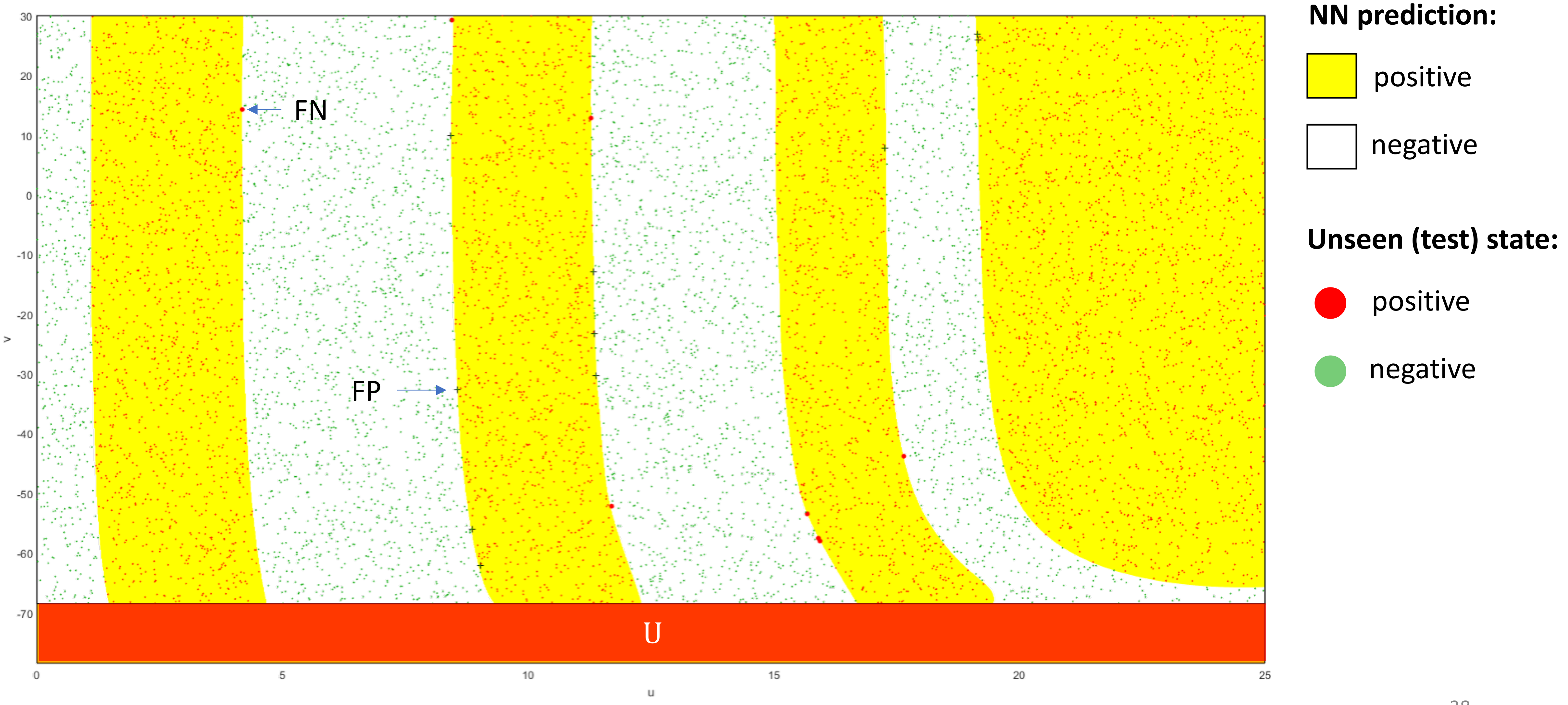
$\theta_A = 99.7\%, \theta_{FN} = 0.2\%$

In parenthesis: number of samples needed to reach the decision

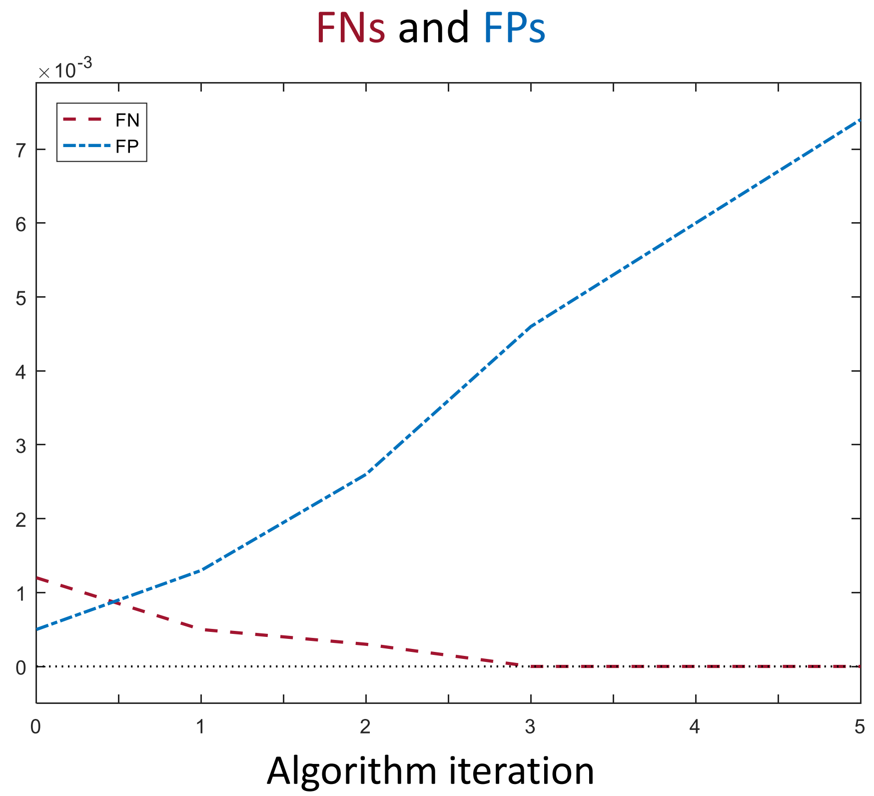
	Neuron		Pendulum		Quadcopter		Cruise	
	$P_A \geq \theta_A$	$P_{FN} \leq \theta_{FN}$	$P_A \geq \theta_A$	$P_{FN} \leq \theta_{FN}$	$P_A \geq \theta_A$	$P_{FN} \leq \theta_{FN}$	$P_A \geq \theta_A$	$P_{FN} \leq \theta_{FN}$
DNN-S	✓ (5800)	✓ (2900)	✓ (2300)	✓ (2300)	✓ (4400)	✓ (2300)	✓ (3000)	✓ (2300)
DNN-R	✗ (3600)	✗ (8600)	✓ (15500)	✓ (4000)	✗ (1400)	✓ (7300)	✓ (3000)	✓ (2300)
SNN	✗ (700)	✗ (1000)	✗ (2900)	✓ (2300)	✗ (1500)	✓ (3400)	✗ (3600)	✓ (2300)
SVM	✗ (400)	✗ (600)	✗ (6600)	✓ (2300)	✗ (200)	✗ (5300)	✗ (3400)	✓ (2300)
BDT	✗ (1700)	✗ (3300)	✗ (6300)	✓ (15000)	✗ (800)	✗ (1100)	✓ (2700)	✓ (2900)
NBOR	✗ (300)	✗ (300)	✗ (28500)	✓ (2900)	✗ (1000)	✗ (1300)	✗ (3400)	✗ (2300)

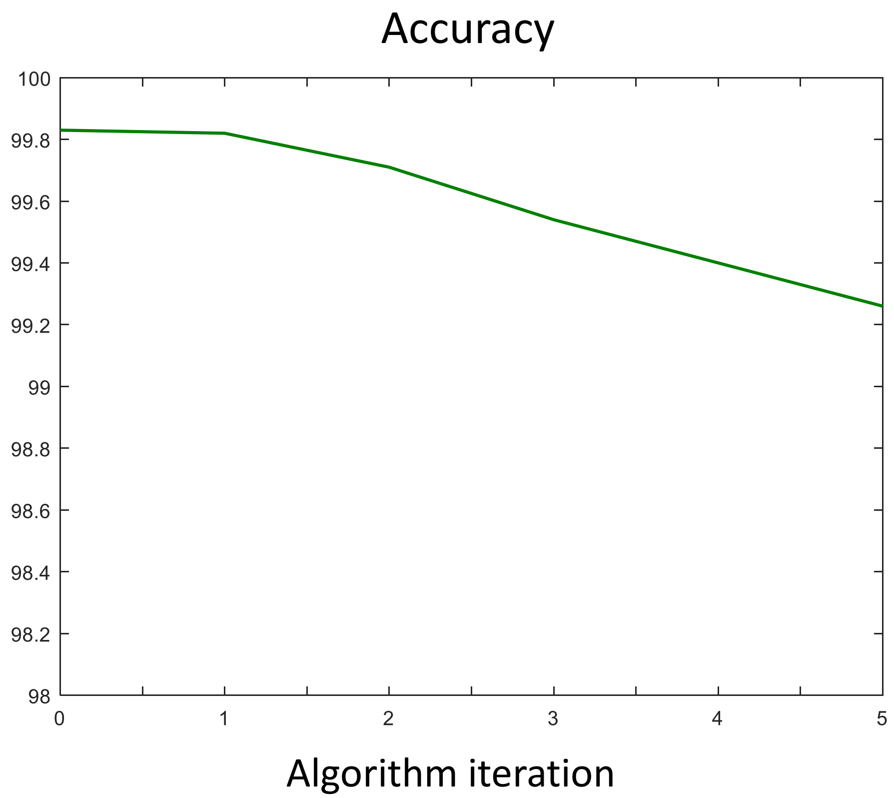
Strength of test: $\alpha = \beta = 0.01$.

Reducing FNs…

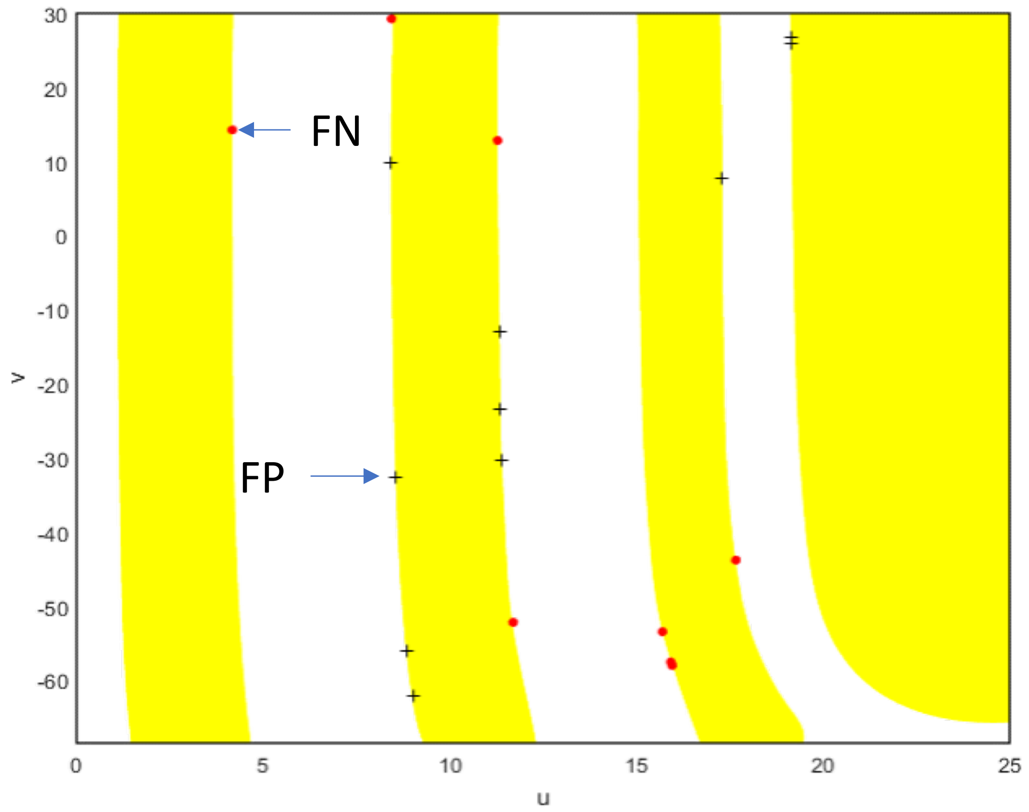


…with falsification and re-training



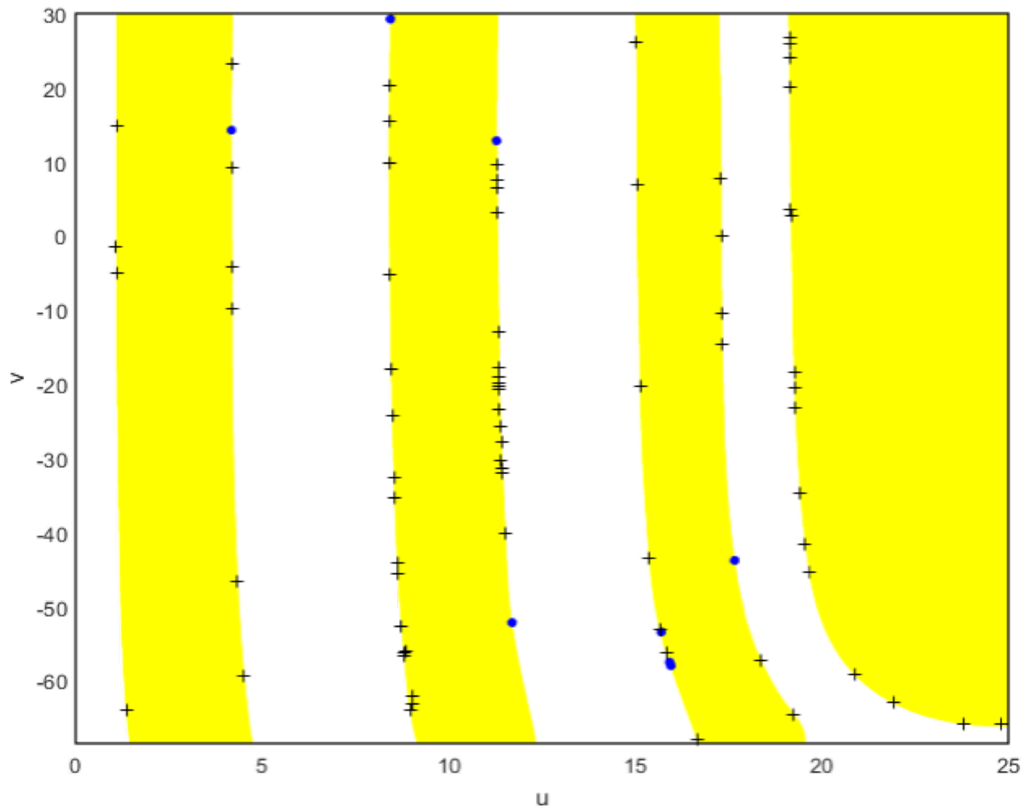


Reducing FNs



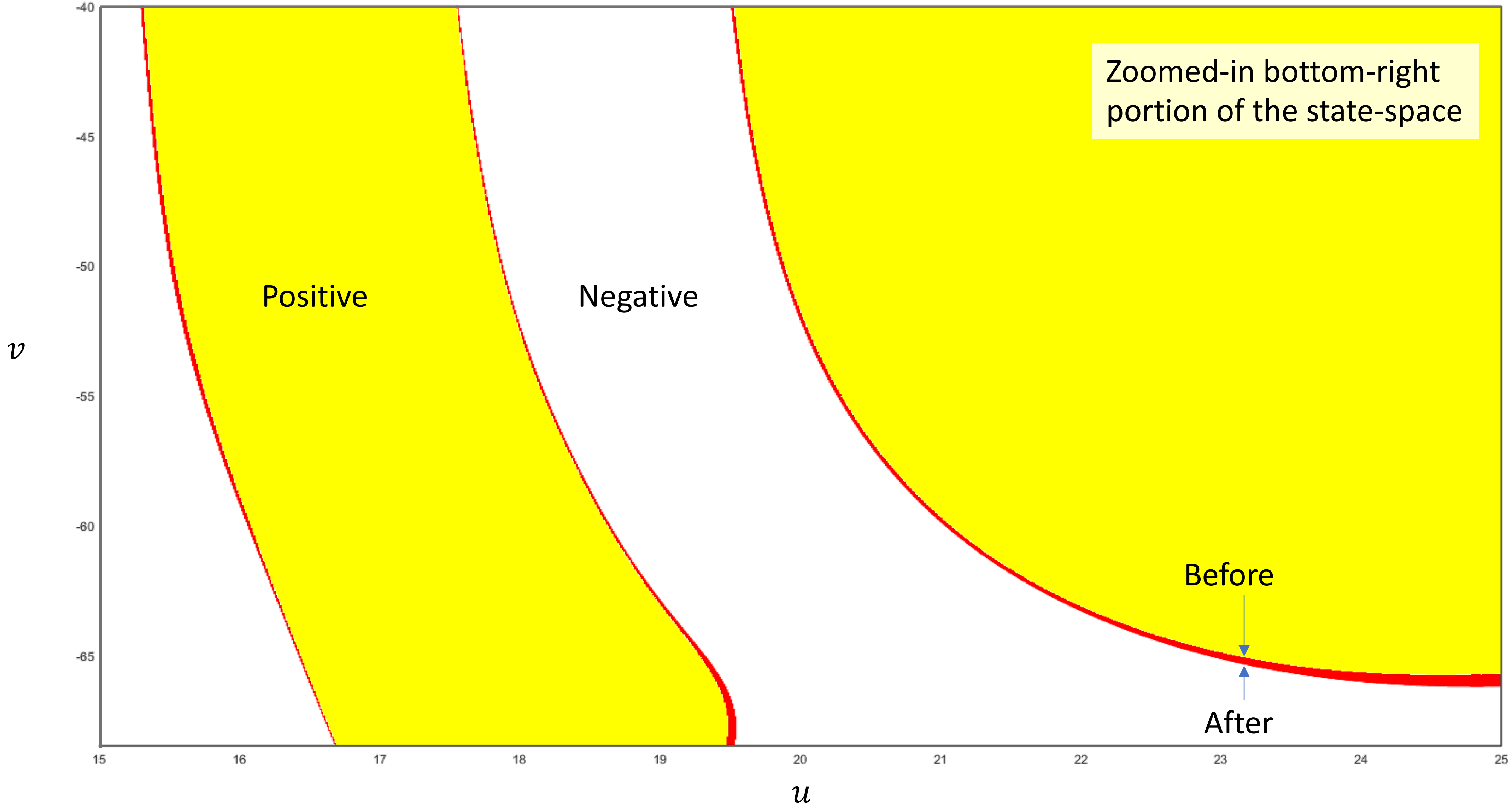
Before

Test FNs are eliminated and
the state classifier becomes more conservative



After

Pushing the DNN decision boundary



Related work

Machine-learning-aided verification

- Gaussian processes to approximate the satisfaction function of continuous-time Markov chains
  [Bortolussi et al, *Information and Computation 247* (2016)]
- NeuroSAT, learning to solve SAT problems from examples
  [Selsam et al, *arXiv:1802.03685* (2018)]
- Reinforcement learning of DNN policies for heuristics in QBF solvers [Lederman et al, *arXiv:1807.08058* (2018)]
- NN-based program synthesis from I/O examples
  [Parisotto et al, *arXiv:1611.01855* (2016)]

Verification of NNs

- Robustness (absence of adversarial inputs)
  [Huang et al, *CAV* (2017); Gopinath et al, *ATVA* (2018)]
- Convex specifications
  [Katz et al, *CAV* (2017); Ehlers, *ATVA* (2017)]
- Analysis of NN components in-the-loop with CPS models
  [Dreossi et al, *NFM* (2017)]
- Range estimation for NNs (compute ”reach set” of NN function)
  [Dutta et al, *NFM* (2018); Xiang et al, *IEEE Trans on Neural Networks and Learning Systems* (2018)]

Conclusion

- State classification problem for hybrid systems
- NSC, a solution based on neural networks, efficient and with high accuracy
- Reverse HA construction for balanced sampling
- Statistical guarantees on classifier accuracy and FN rate
- Falsification-based techniques to reduce FNs and make classifier more conservative

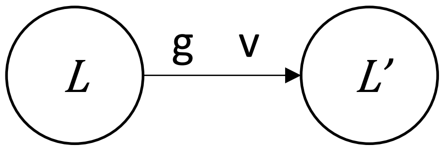
Future work:

- More expressive properties, quantitative semantics, confidence intervals of point predictions

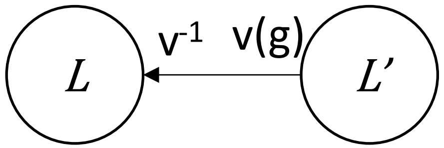
Backup slides

Reverse HA automaton

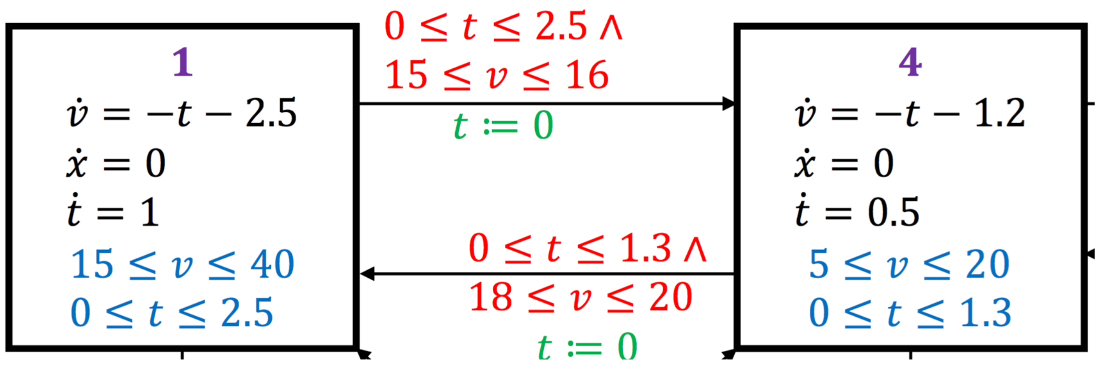
- Locations and invariants stay the same
- Flows are reversed (sign changes)



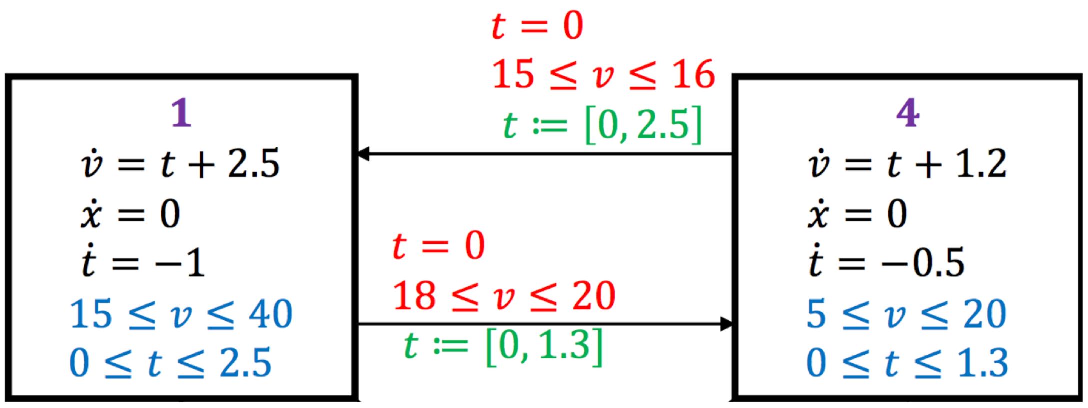
becomes



Forward

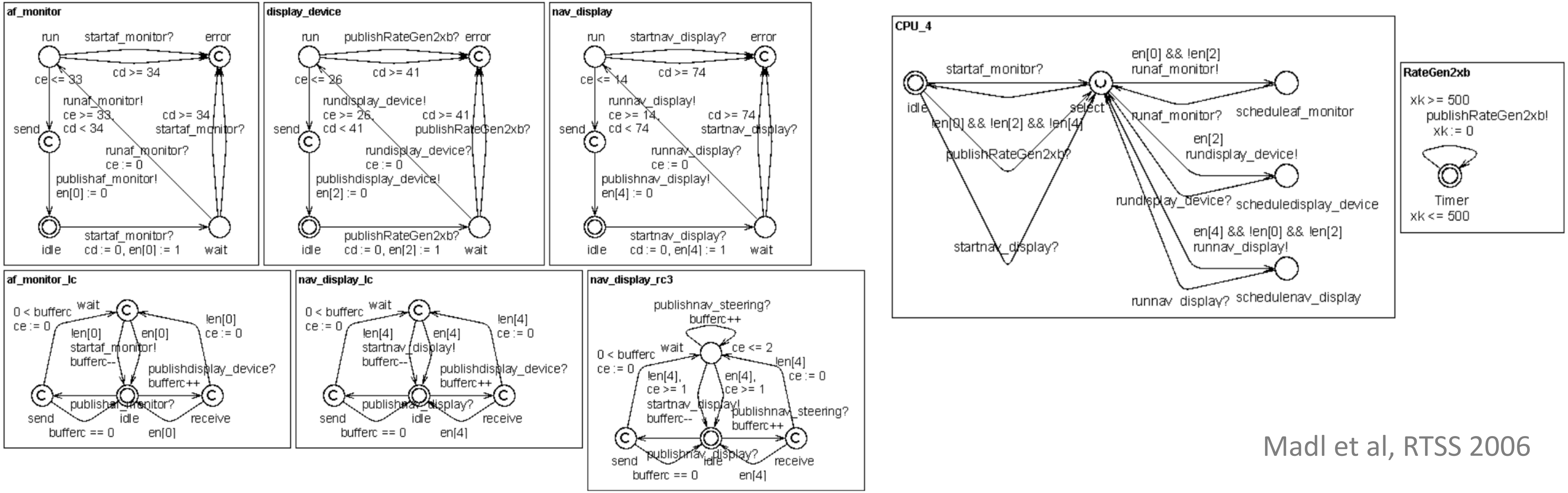


Reverse



Hybrid automata in action

Timed automata network of task scheduling in Boeing Bold Stroke platform



Madl et al, RTSS 2006