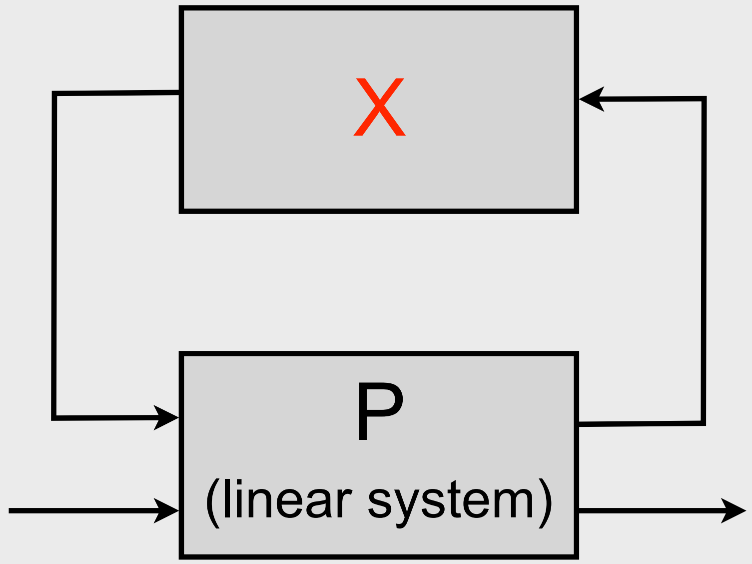
Affine Multiplexing Networks:

System Analysis, Learning, and Computation

Ivan Papusha (joint work with Ufuk Topcu)
Institute for Computational Engineering and Sciences
University of Texas at Austin

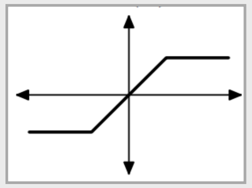
X-in-the-loop control

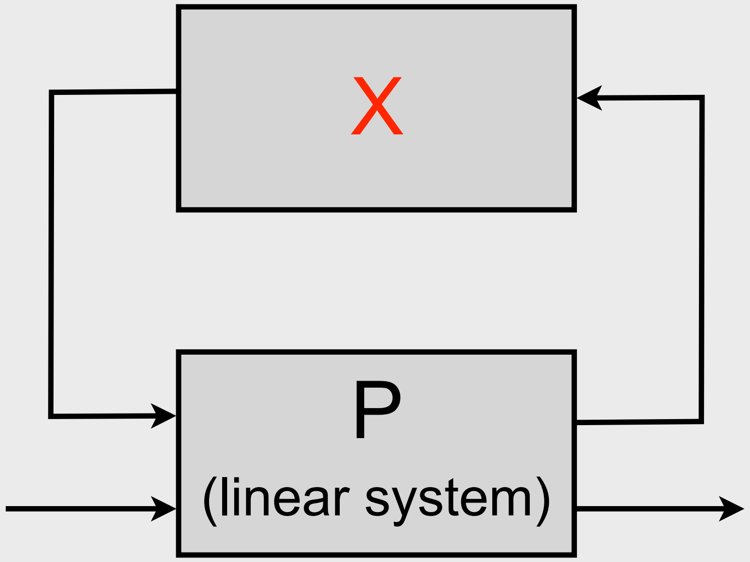
(and verification for it)



X-in-the-loop control

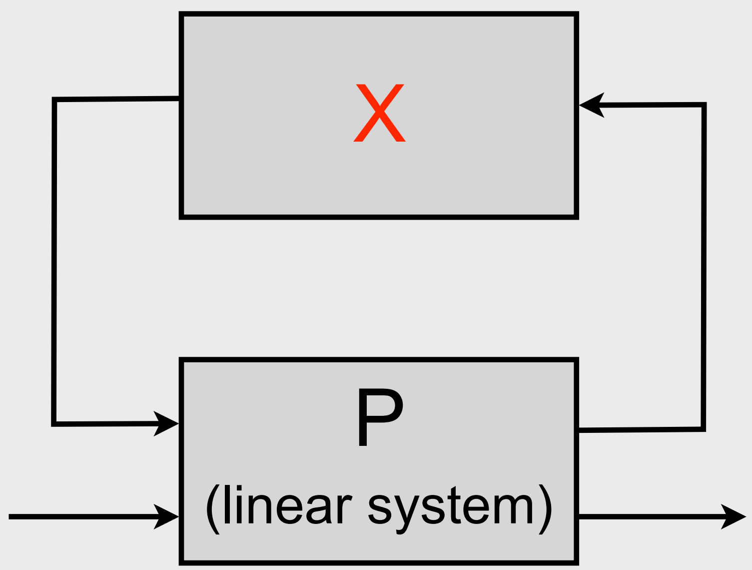
(and verification for it)

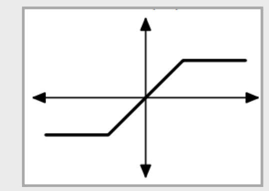
saturation X = 



X-in-the-loop control

(and verification for it)



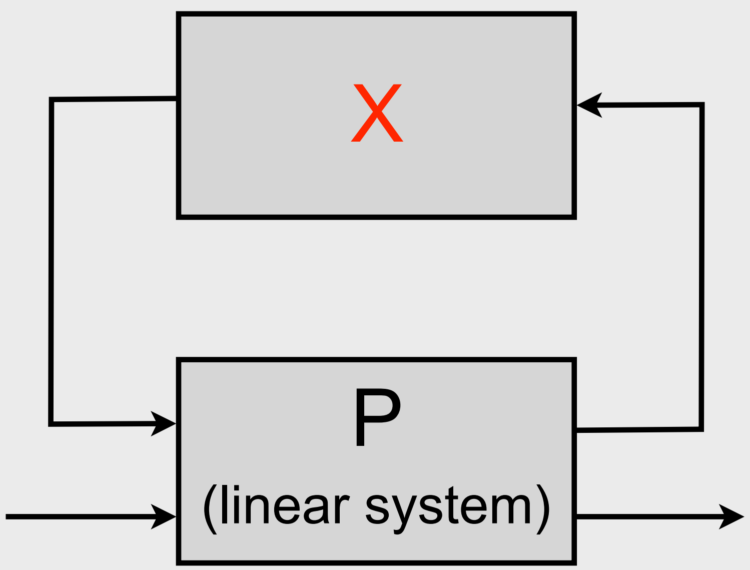
saturation X = 

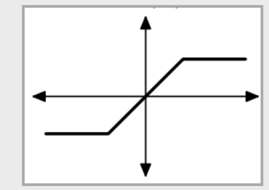
software X =

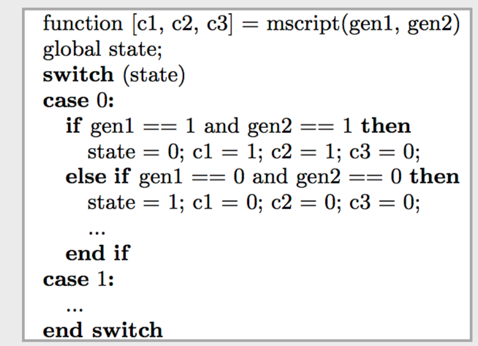
```
function [c1, c2, c3] = mscript(gen1, gen2)
global state;
switch (state)
case 0:
  if gen1 == 1 and gen2 == 1 then
    state = 0; c1 = 1; c2 = 1; c3 = 0;
  else if gen1 == 0 and gen2 == 0 then
    state = 1; c1 = 0; c2 = 0; c3 = 0;
    ...
  end if
case 1:
  ...
end switch
```

X-in-the-loop control

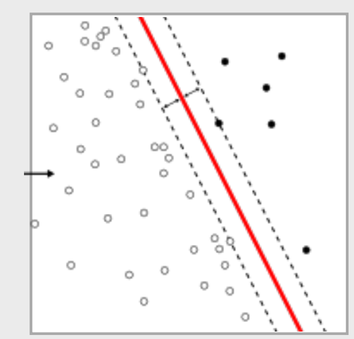
(and verification for it)



saturation X = 

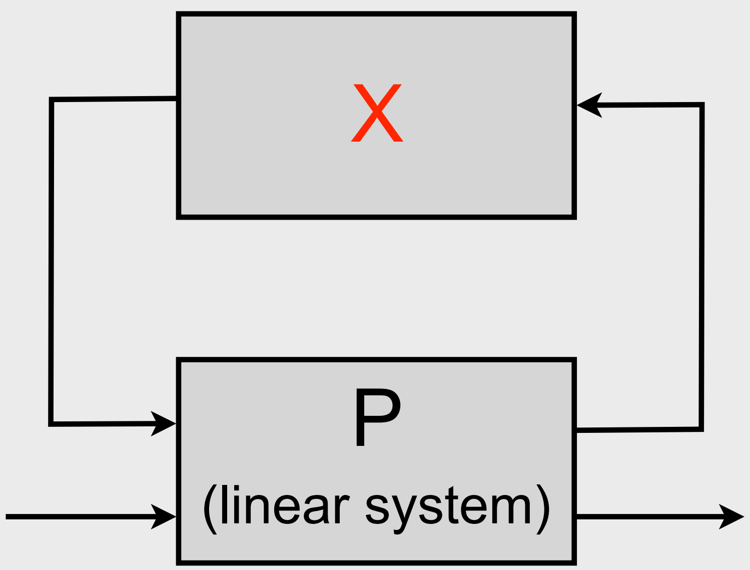
software X = 

```
function [c1, c2, c3] = mscript(gen1, gen2)
global state;
switch (state)
case 0:
  if gen1 == 1 and gen2 == 1 then
    state = 0; c1 = 1; c2 = 1; c3 = 0;
  else if gen1 == 0 and gen2 == 0 then
    state = 1; c1 = 0; c2 = 0; c3 = 0;
    ...
  end if
case 1:
  ...
end switch
```

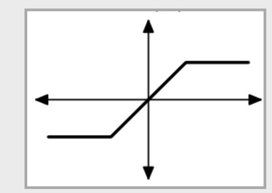
classifier X = 

X-in-the-loop control

(and verification for it)



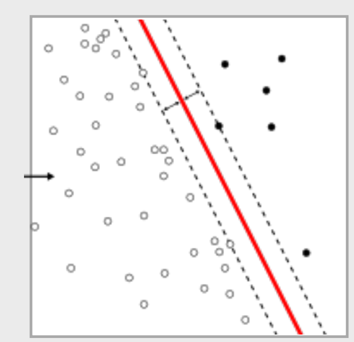
saturation X =



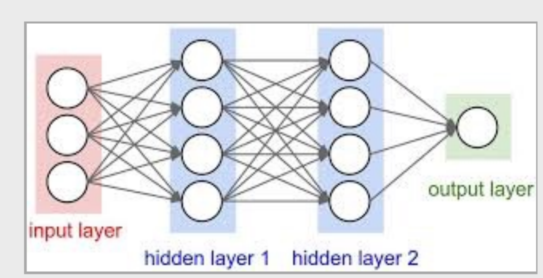
software X =

```
function [c1, c2, c3] = mscript(gen1, gen2)
global state;
switch (state)
case 0:
  if gen1 == 1 and gen2 == 1 then
    state = 0; c1 = 1; c2 = 1; c3 = 0;
  else if gen1 == 0 and gen2 == 0 then
    state = 1; c1 = 0; c2 = 0; c3 = 0;
    ...
  end if
case 1:
  ...
end switch
```

classifier X =



neural network X =



Affine Multiplexing Networks (AMNs)

A directed interconnection of…

1) affine transformations

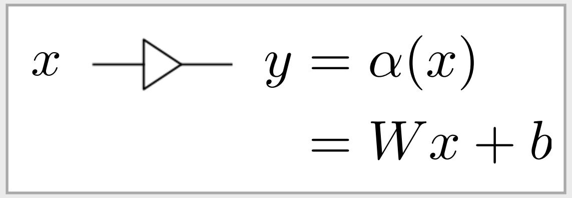
$$x \longrightarrow \triangleright \longrightarrow y = \alpha(x) = Wx + b$$

2) multiplexing functions

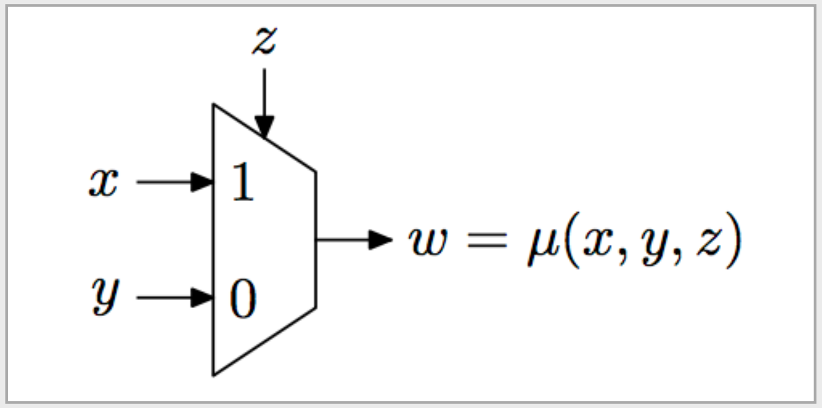
Affine Multiplexing Networks (AMNs)

A directed interconnection of…

1) affine transformations



2) multiplexing functions

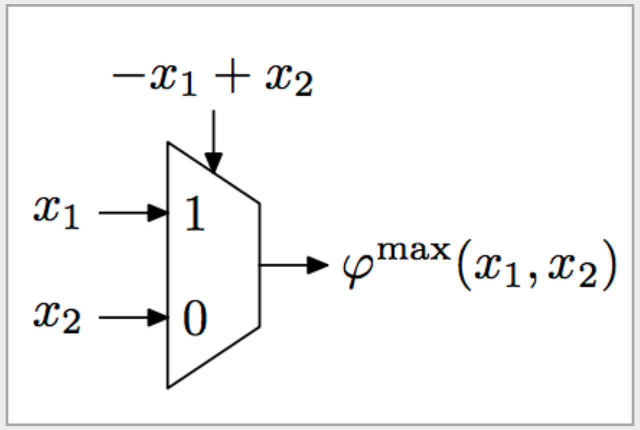


$$\mu : \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$$

$$\mu(x, y, z) = \begin{cases} x, & \text{if } z \leq 0 \\ y, & \text{otherwise} \end{cases}$$

Examples of AMNs

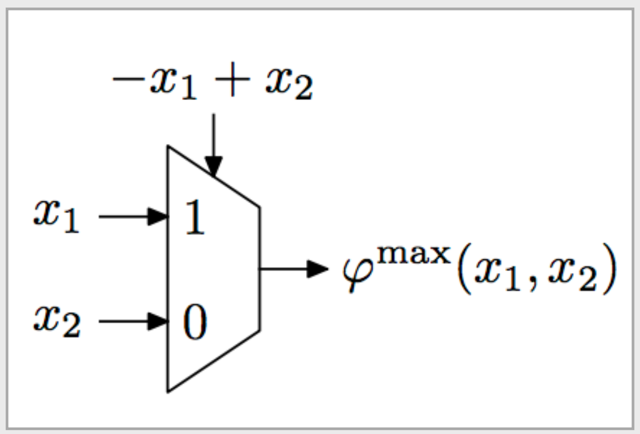
maximum



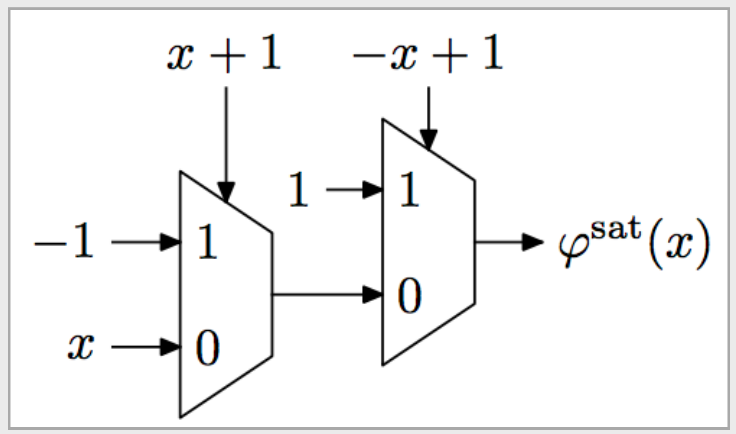
$$\mu(x_1, x_2, -x_1 + x_2)$$

Examples of AMNs

maximum



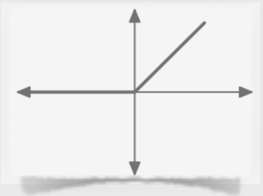
saturation



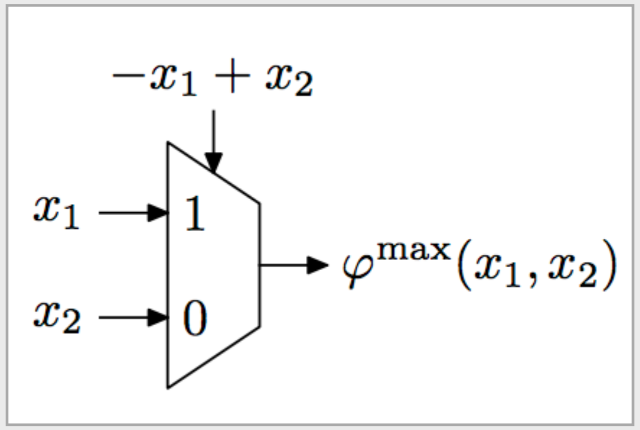
$$\mu(x_1, x_2, -x_1 + x_2)$$

$$\mu(1, \mu(-1, x, x+1), -x+1)$$

Examples of AMNs

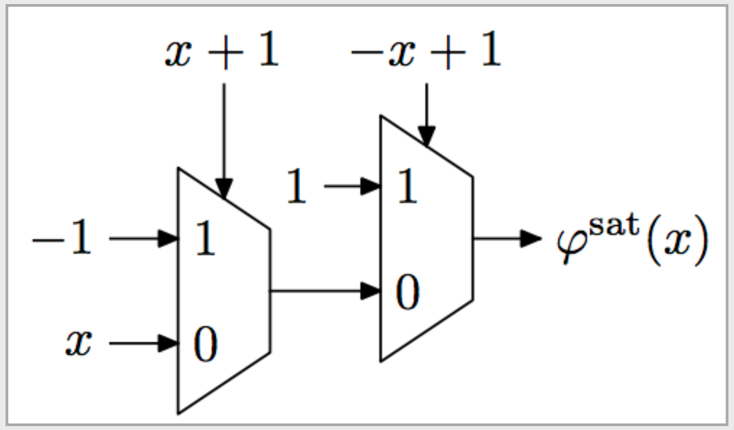


maximum



$$\mu(x_1, x_2, -x_1 + x_2)$$

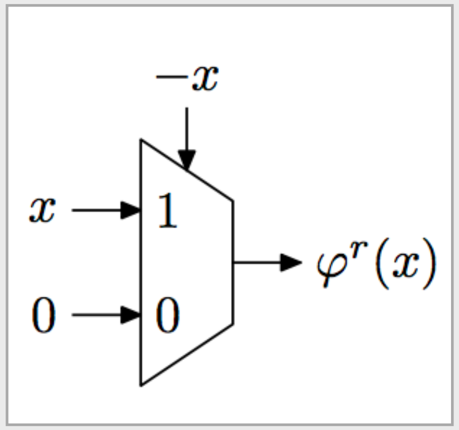
saturation



$$\mu(1, \mu(-1, x, x + 1), -x + 1)$$

ReLU

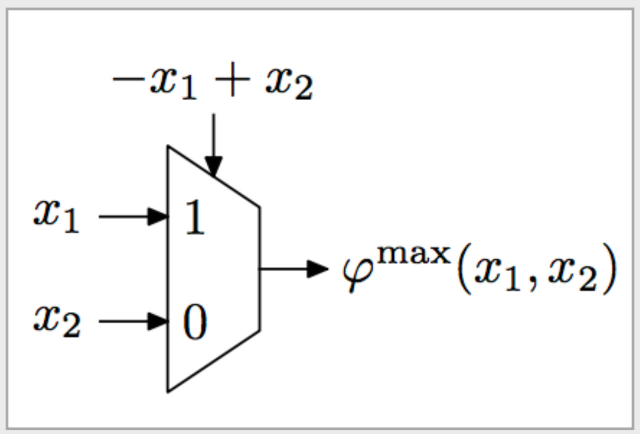
(rectified linear unit)



$$\mu(x, 0, -x)$$

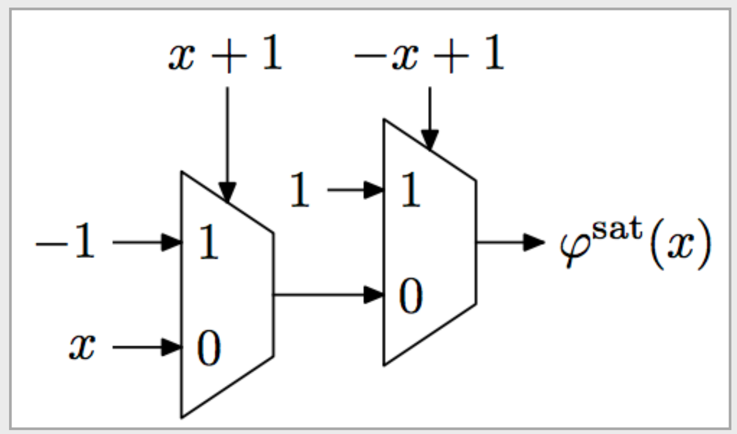
Examples of AMNs

maximum



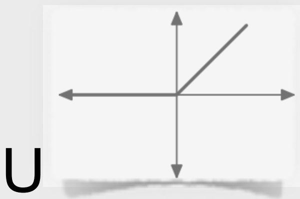
$\mu(x_1, x_2, -x_1 + x_2)$

saturation

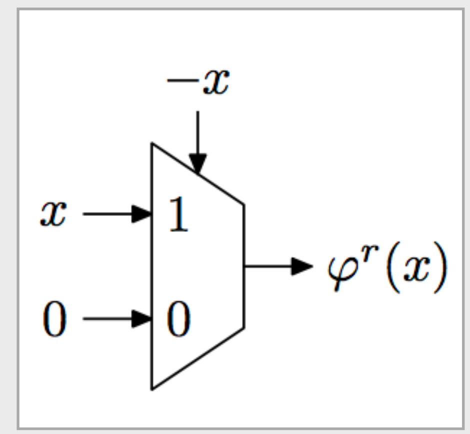


$\mu(1, \mu(-1, x, x + 1), -x + 1)$

ReLU

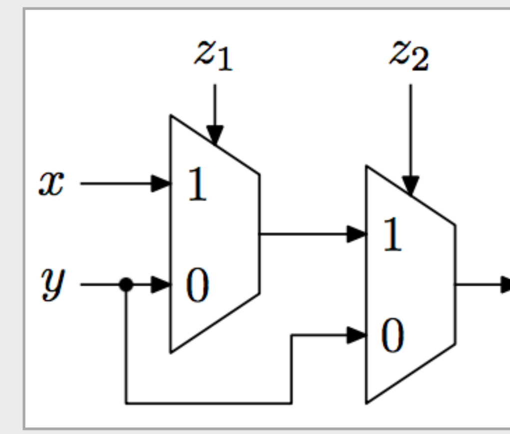


(rectified linear unit)



$\mu(x, 0, -x)$

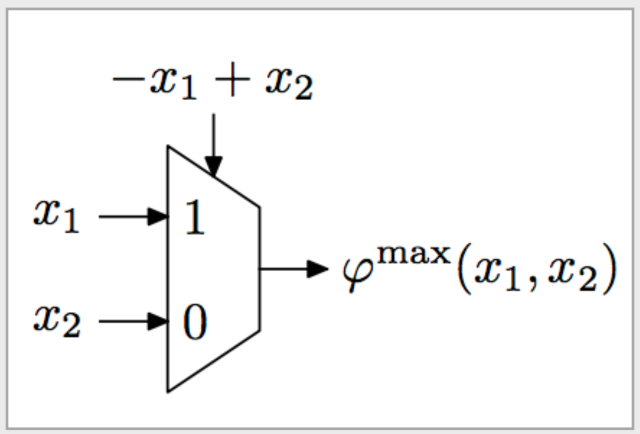
AND gate



$\mu(\mu(x, y, z_1), y, z_2)$

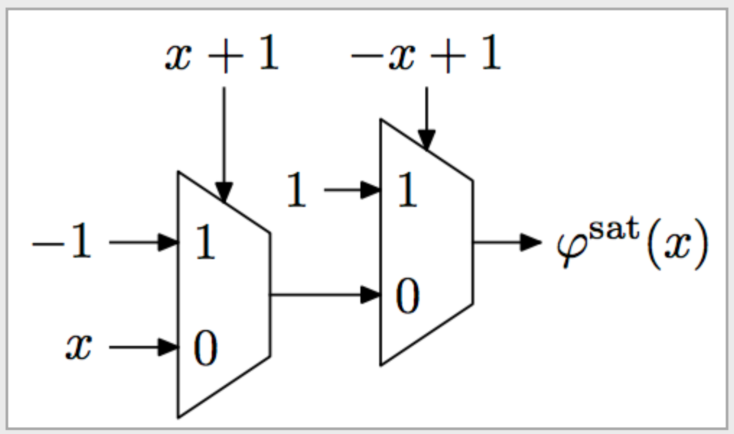
Examples of AMNs

maximum



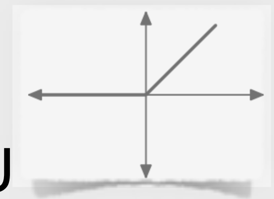
$\mu(x_1, x_2, -x_1 + x_2)$

saturation

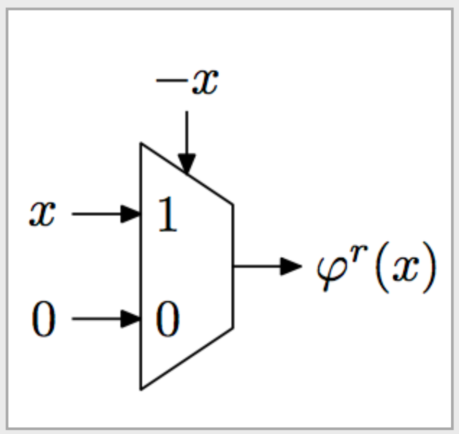


$\mu(1, \mu(-1, x, x + 1), -x + 1)$

ReLU

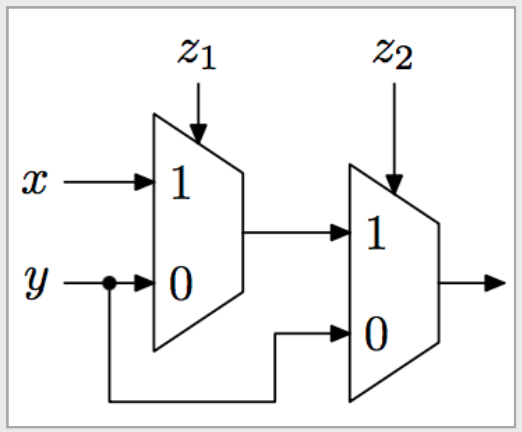


(rectified linear unit)



$\mu(x, 0, -x)$

AND gate

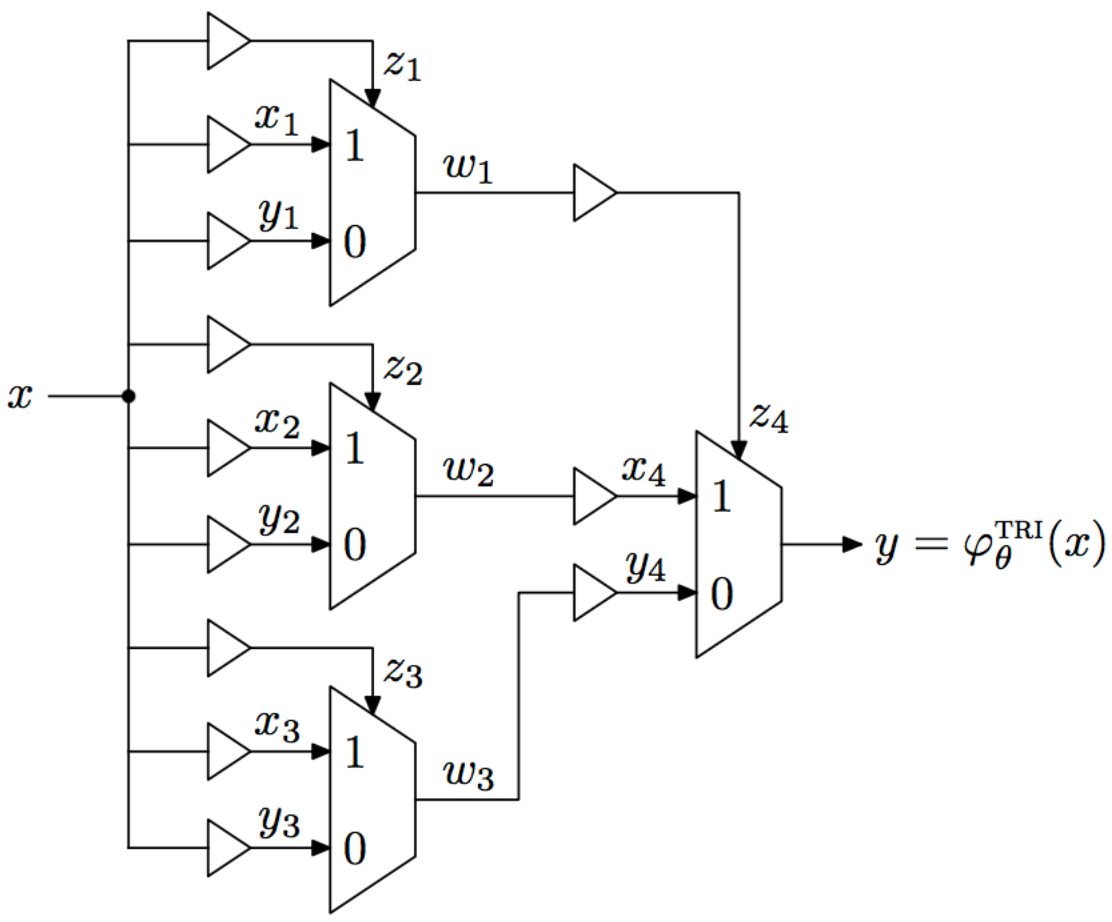


$\mu(\mu(x, y, z_1), y, z_2)$

$\mathbf{card}(x)$	$\sum_{i=1}^{n} \mu(\mu(1, 0, x_i), 0, -x_i)$
$\|x\|_\infty$	$\mu(x_1, \mu(x_2 \ldots), -x_1 + \mu(x_2 \ldots))$
XOR	$\mu(\mu(y, x, z_1), \mu(x, y, z_1), z_2)$
LE	$\mu(x, y, z)$

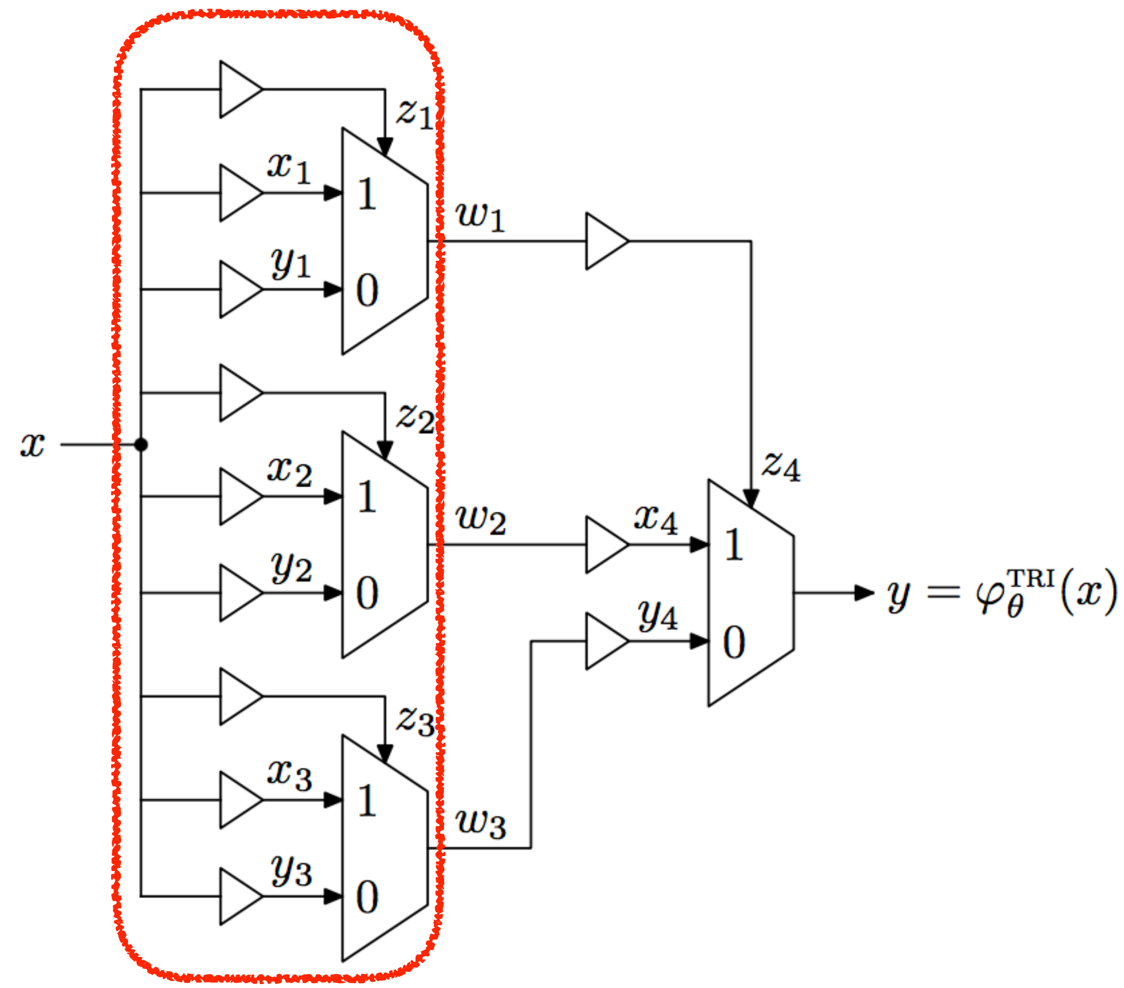
AMNs compose

A 2-layer,
4-mux
AMN



AMNs compose

A 2-layer,
4-mux
AMN



layer 1

weights

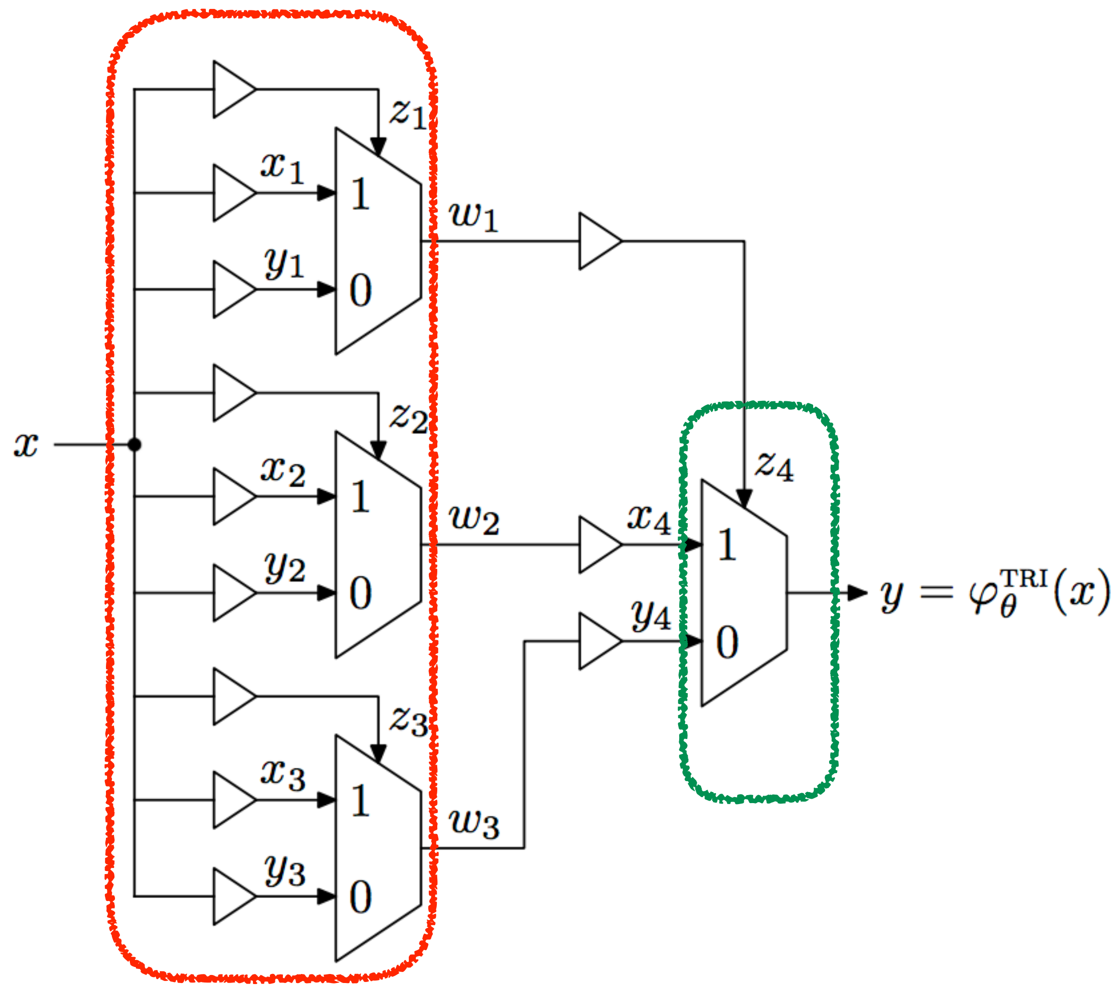
$$x_1 := a_1 x + b_1 \quad \cdots \quad x_3 := a_3 x + b_3$$
$$y_1 := c_1 x + d_1 \quad \cdots \quad y_3 := c_3 x + d_3$$
$$z_1 := e_1 x + f_1 \quad \cdots \quad z_3 := e_3 x + f_3$$

nonlinearities

$$w_1 := \mu(x_1, y_1, z_1)$$
$$w_2 := \mu(x_2, y_2, z_2)$$
$$w_3 := \mu(x_3, y_3, z_3)$$

AMNs compose

A 2-layer,
4-mux
AMN



layer 1

weights

$$x_1 := a_1 x + b_1 \qquad \cdots \qquad x_3 := a_3 x + b_3$$
$$y_1 := c_1 x + d_1 \qquad \qquad y_3 := c_3 x + d_3$$
$$z_1 := e_1 x + f_1 \qquad \qquad z_3 := e_3 x + f_3$$

nonlinearities

$$w_1 := \mu(x_1, y_1, z_1)$$
$$w_2 := \mu(x_2, y_2, z_2)$$
$$w_3 := \mu(x_3, y_3, z_3)$$

layer 2

weights

$$x_4 := a_4 w_2 + b_4$$
$$y_4 := c_4 w_3 + d_4$$
$$z_4 := e_4 w_1 + f_4$$

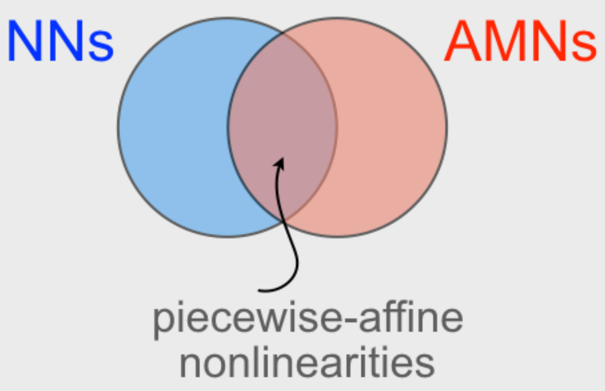
nonlinearity

$$y := \mu(x_4, y_4, z_4)$$

Why care about AMNs?

AMNs are expressive.

Deep multilayer feedforward networks with **piecewise-affine nonlinearities** are a special case.

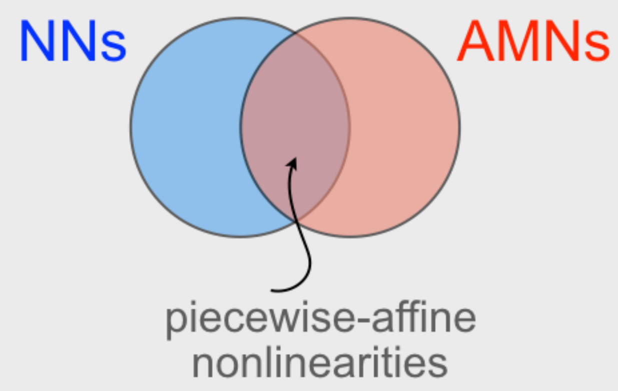


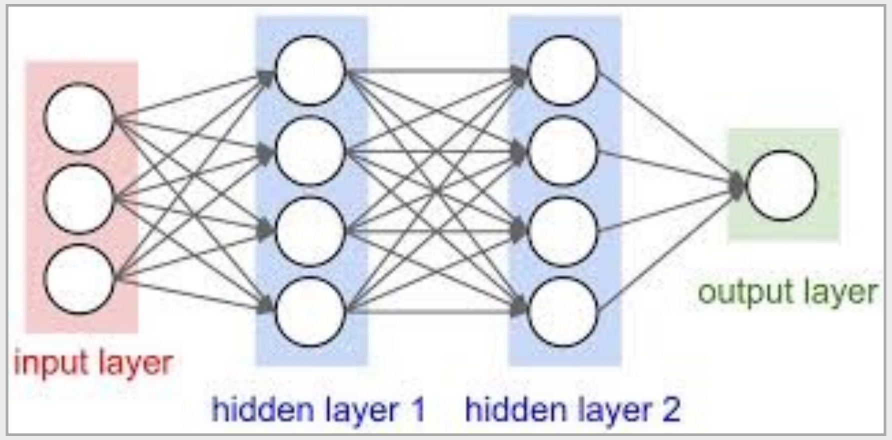
Why care about AMNs?

AMNs are expressive.

Deep multilayer feedforward networks with **piecewise-affine nonlinearities** are a special case.

Can automatically translate them into AMNs.



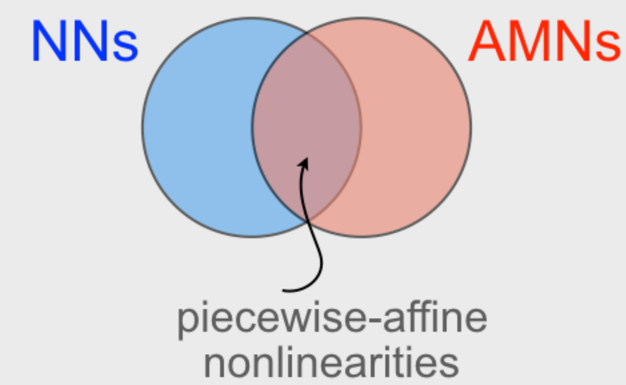


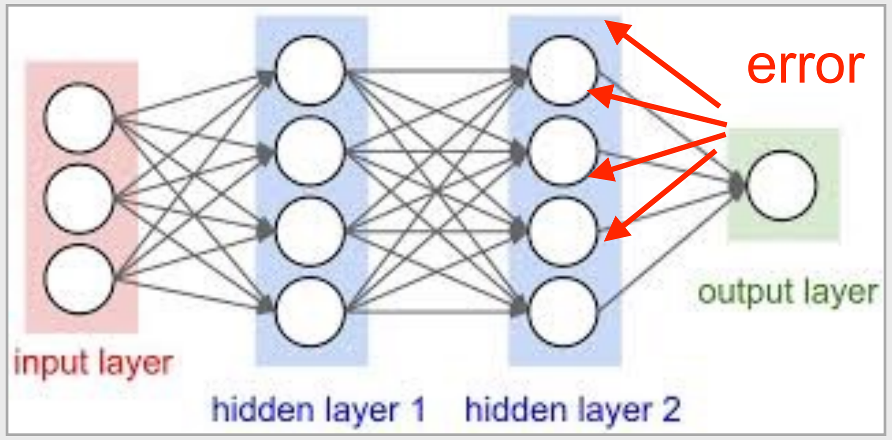
Why care about AMNs?

AMNs are expressive.

Deep multilayer feedforward networks with **piecewise-affine nonlinearities** are a special case.

Can automatically translate them into AMNs.





AMNs can be *trained* through a modification of the back-propagation algorithm
(i.e. weak derivatives of μ can be efficiently computed).

Sounds like a rehash of neural networks?

They are similar but…

AMNs can express discontinuous functions
whereas classical (continuous) neural networks cannot.

Sounds like a rehash of neural networks?

They are similar but…

AMNs can express discontinuous functions
whereas classical (continuous) neural networks cannot.

a switched system	its AMN representation
$x(t+1) = \begin{cases} A^{-}x(t), & \text{if } x_1(t) \leq 0, \\ A^{+}x(t), & \text{otherwise,} \end{cases}$	$\varphi^{\text{sw}}(x) = \mu(A^{-}x, A^{+}x, e_1^T x)$

Sounds like a rehash of neural networks?

They are similar but…

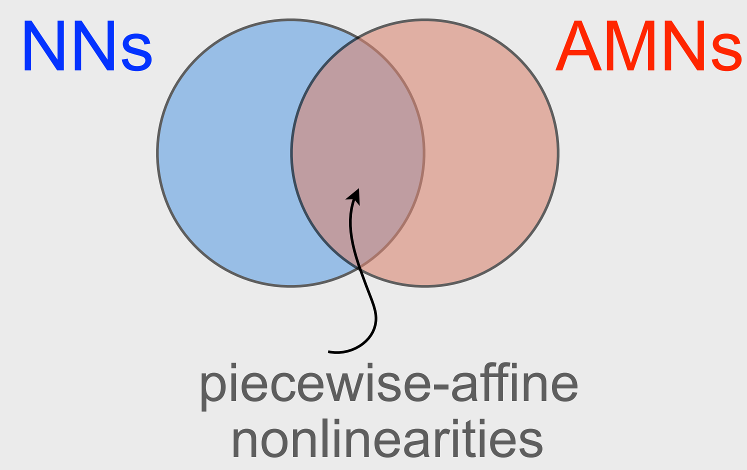
AMNs can express discontinuous functions
whereas classical (continuous) neural networks cannot.

a switched system	its AMN representation
$x(t+1) = \begin{cases} A^- x(t), & \text{if } x_1(t) \leq 0, \\ A^+ x(t), & \text{otherwise,} \end{cases}$	$\varphi^{\mathrm{sw}}(x) = \mu(A^- x, A^+ x, e_1^T x)$

AMNs can be encoded as satisfiability modulo theory (SMT) instances.

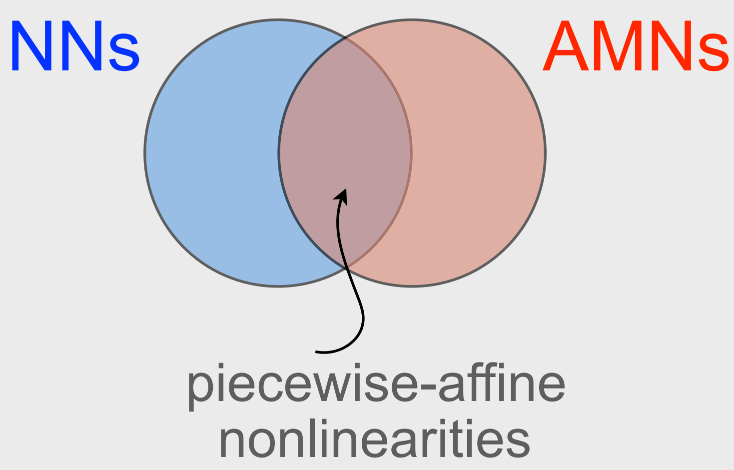
Fine, but does the difference matter?

The answer depends on what you want to do.



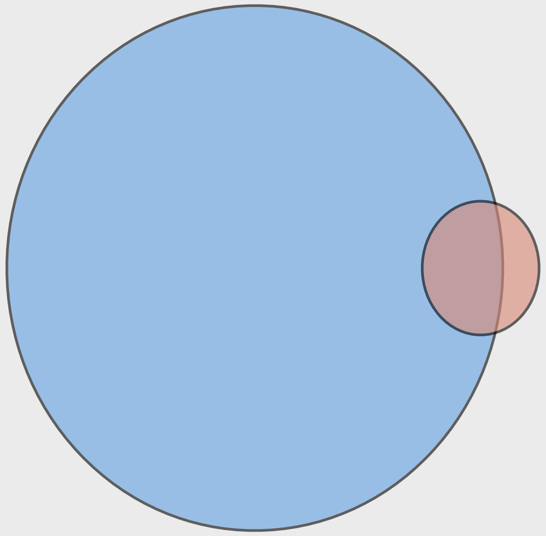
Fine, but does the difference matter?

The answer depends on what you want to do.



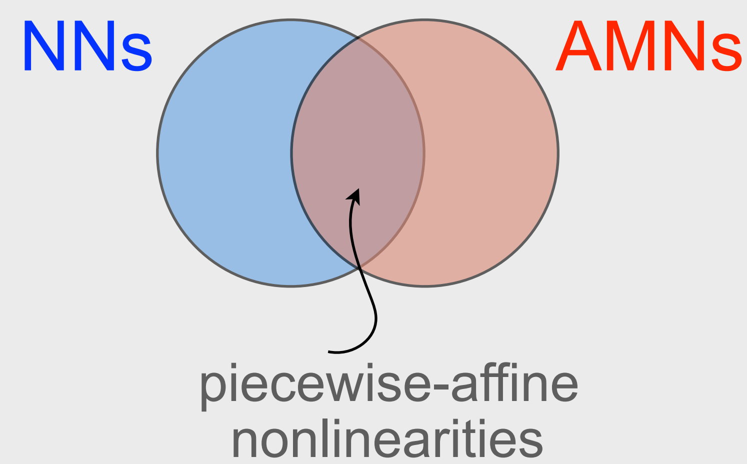
Learning view:

"Can train so much more"

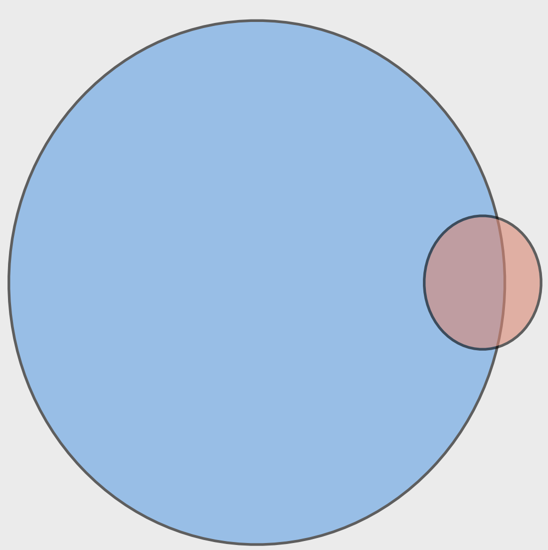


Fine, but does the difference matter?

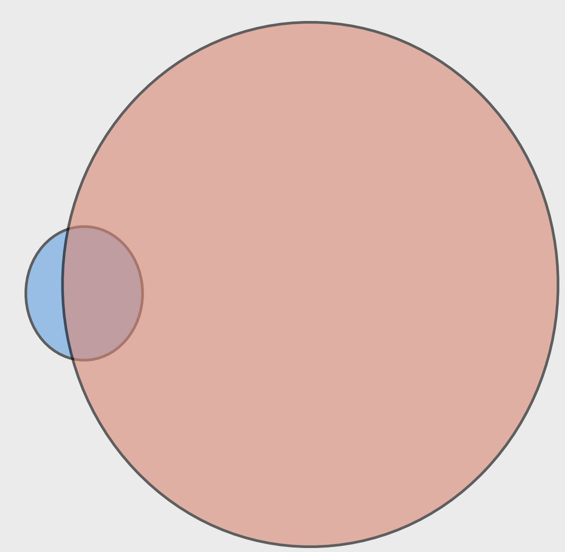
The answer depends on what you want to do.



Learning view:
"Can train so much more"



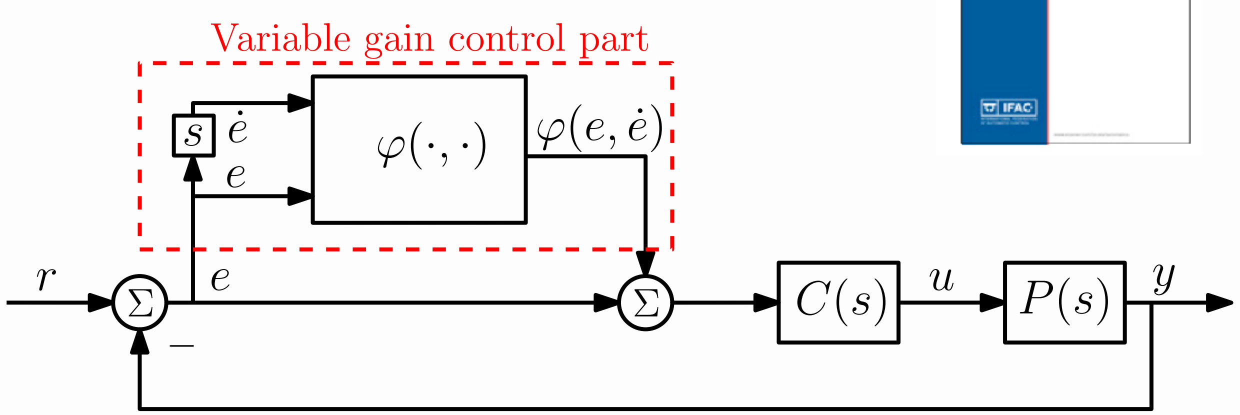
Verification view:
"Can do computational analysis"



Beating fundamental limitations of linear feedback

Overcoming a fundamental time-domain performance limitation by nonlinear control

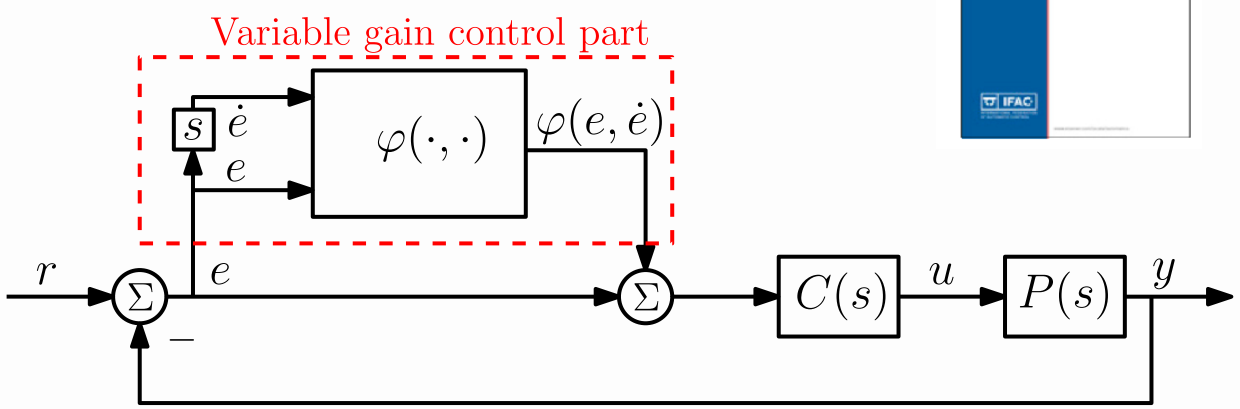
Hunnekens, Wouw, Nesic (2016)



Beating fundamental limitations of linear feedback

Overcoming a fundamental time-
performance limitation by nonlin

Hunnekens, Wouw, Nesic (2016)



Seron, et al. (1997)

$$y_{os} \geq \frac{(pt_r - 1)e^{pt_r} + 1}{pt_r} \geq \frac{pt_r}{2}\,.$$

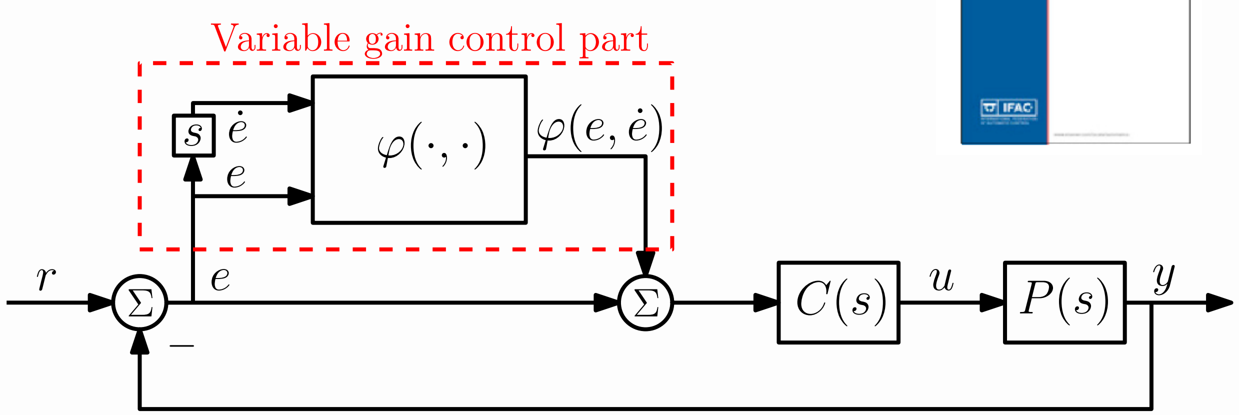
Beating fundamental limitations of linear feedback

Seron, et al. (1997)

$$y_{os} \geq \frac{(pt_r - 1)e^{pt_r} + 1}{pt_r} \geq \frac{pt_r}{2}\ .$$

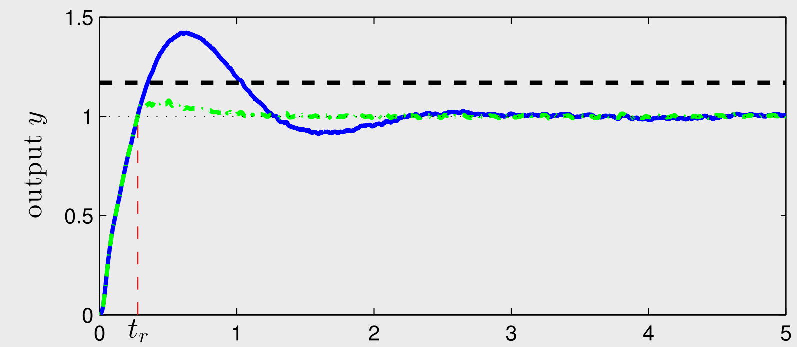
Overcoming a fundamental time-domain performance limitation by nonlin

Hunnekens, Wouw, Nesic (2016)



Phase-based,
variable-gain nonlinearity

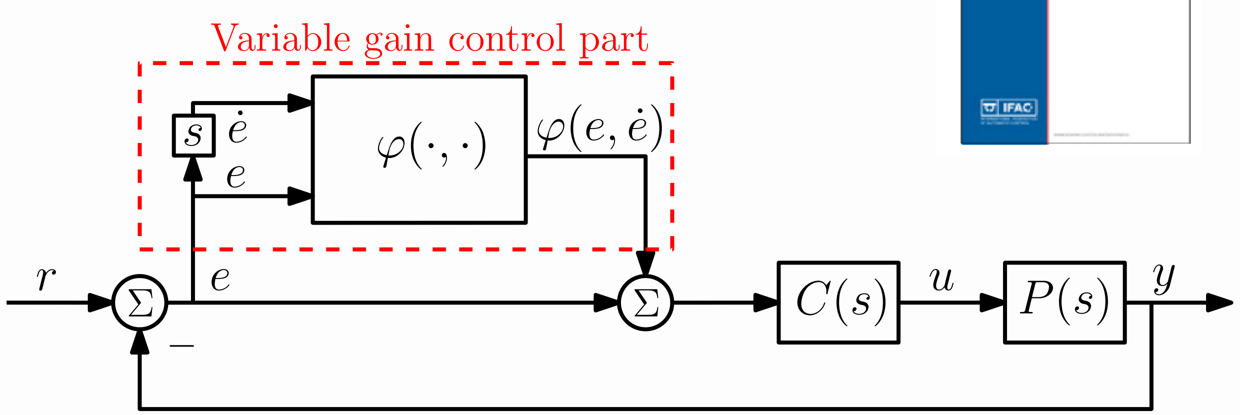
$$\varphi^{\text{vgc}}(e,\dot{e}) = \begin{cases} \alpha e, & \text{if } e\dot{e} > 0 \\ 0, & \text{otherwise} \end{cases}$$



Beating fundamental limitations of linear feedback

Overcoming a fundamental time-
performance limitation by nonlin

Hunnekens, Wouw, Nesic (2016)

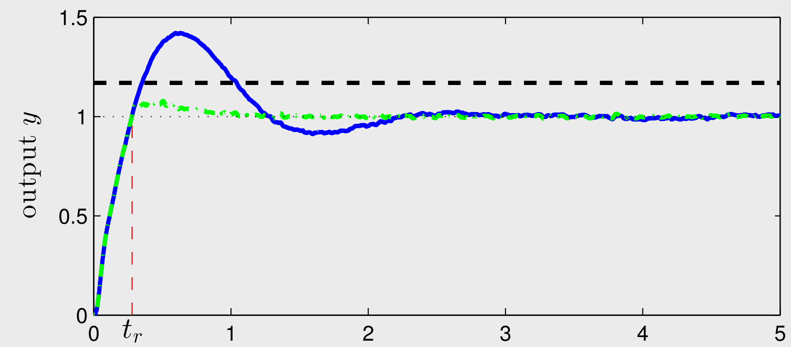


Seron, et al. (1997)

$$y_{os} \geq \frac{(pt_r - 1)e^{pt_r} + 1}{pt_r} \geq \frac{pt_r}{2} \, .$$

Phase-based,
variable-gain nonlinearity

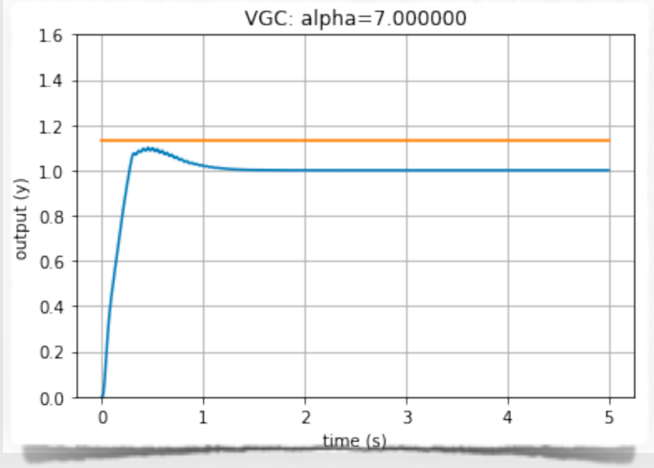
$$\varphi^{\text{vgc}}(e,\dot{e}) = \begin{cases} \alpha e, & \text{if } e\dot{e} > 0 \\ 0, & \text{otherwise} \end{cases}$$



The nonlinearity as an AMN (with 4 multiplexing nonlinearities)

$$\varphi^{\text{vgc}}(e,\dot{e}) = \varphi^{\vee}\left(\varphi^{\vee}(0, \alpha e, -e, -\dot{e}), \alpha e, e, \dot{e}\right)$$

$$\varphi^{\vee}(x, y, z_1, z_2) = \mu(x, \mu(x, y, z_1), z_2)$$



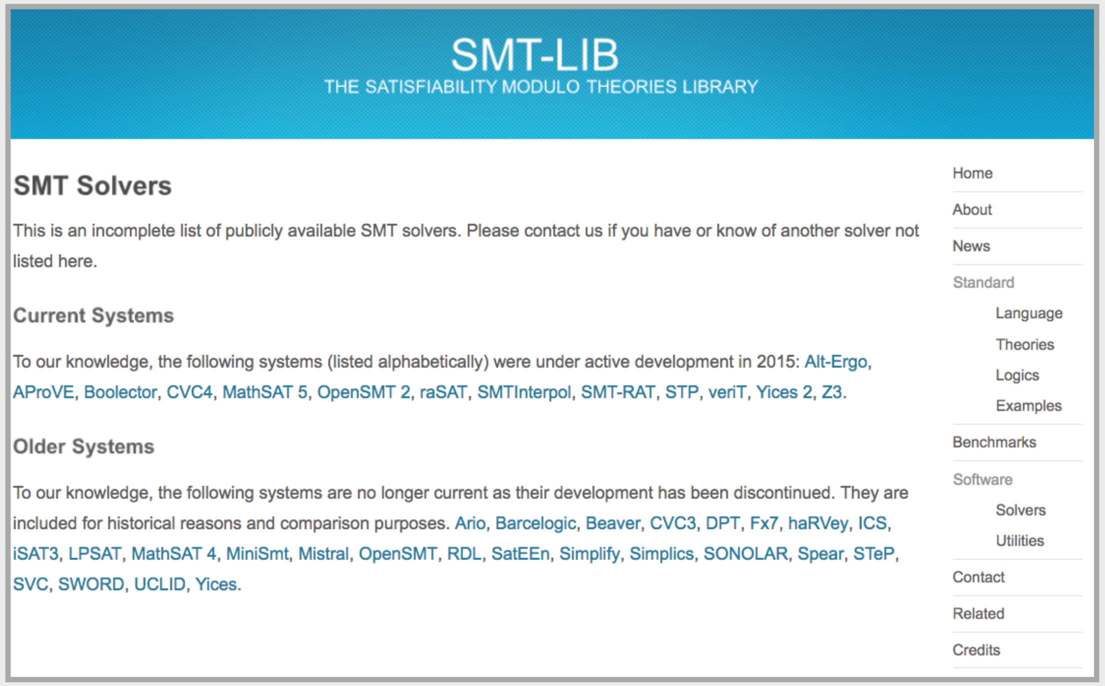
What is SMT (satisfiability modulo theories)?

- Essentially, **constraint solving**
- A bad name for a useful and **extremely practical** idea.
- Burgeoning and vibrant area of CS research

What is SMT (satisfiability modulo theories)?

- Essentially, **constraint solving**
- A bad name for a useful and **extremely practical** idea.
- Burgeoning and vibrant area of CS research

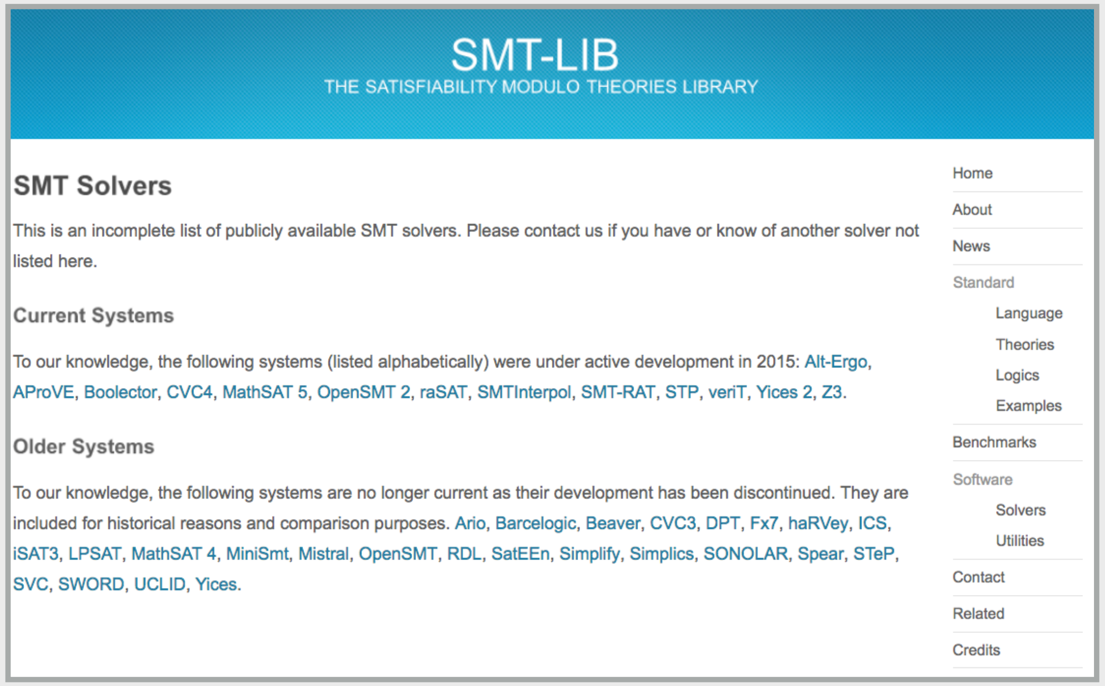
smt-lib.org



What is SMT (satisfiability modulo theories)?

- Essentially, **constraint solving**
- A bad name for a useful and **extremely practical** idea.
- Burgeoning and vibrant area of CS research

smt-lib.org


SMT-LIB
THE SATISFIABILITY MODULO THEORIES LIBRARY

SMT Solvers

This is an incomplete list of publicly available SMT solvers. Please contact us if you have or know of another solver not listed here.

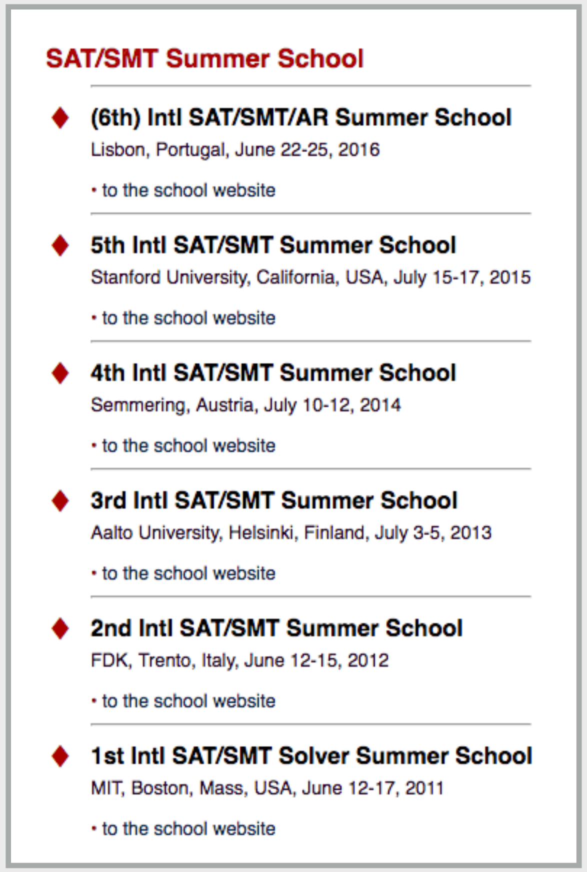
Current Systems

To our knowledge, the following systems (listed alphabetically) were under active development in 2015: Alt-Ergo, AProVE, Boolector, CVC4, MathSAT 5, OpenSMT 2, raSAT, SMTInterpol, SMT-RAT, STP, veriT, Yices 2, Z3.

Older Systems

To our knowledge, the following systems are no longer current as their development has been discontinued. They are included for historical reasons and comparison purposes. Ario, Barcelogic, Beaver, CVC3, DPT, Fx7, haRVey, ICS, iSAT3, LPSAT, MathSAT 4, MiniSmt, Mistral, OpenSMT, RDL, SatEEn, Simplify, Simplics, SONOLAR, Spear, STeP, SVC, SWORD, UCLID, Yices.

Home
About
News
Standard
Language
Theories
Logics
Examples
Benchmarks
Software
Solvers
Utilities
Contact
Related
Credits


SAT/SMT Summer School

- **(6th) Intl SAT/SMT/AR Summer School**
  Lisbon, Portugal, June 22-25, 2016
  • to the school website
- **5th Intl SAT/SMT Summer School**
  Stanford University, California, USA, July 15-17, 2015
  • to the school website
- **4th Intl SAT/SMT Summer School**
  Semmering, Austria, July 10-12, 2014
  • to the school website
- **3rd Intl SAT/SMT Summer School**
  Aalto University, Helsinki, Finland, July 3-5, 2013
  • to the school website
- **2nd Intl SAT/SMT Summer School**
  FDK, Trento, Italy, June 12-15, 2012
  • to the school website
- **1st Intl SAT/SMT Solver Summer School**
  MIT, Boston, Mass, USA, June 12-17, 2011
  • to the school website

What is SMT (satisfiability modulo theories)?

- Essentially, **constraint solving**
- A bad name for a useful and **extremely practical** idea.
- Burgeoning and vibrant area of CS research

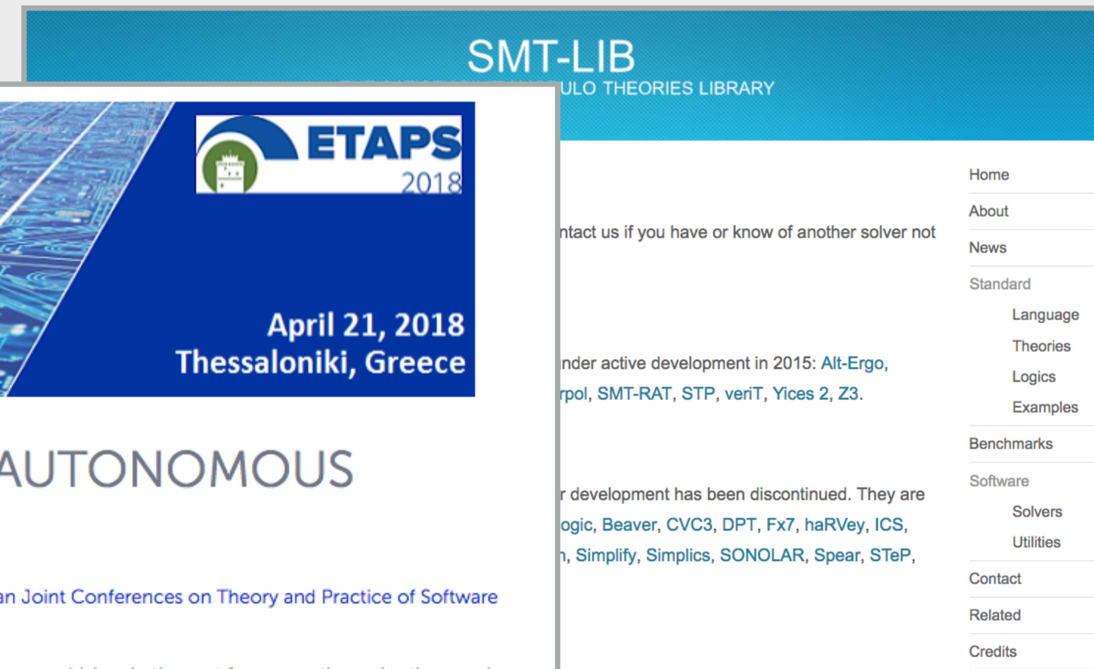
smt-lib.org

fomlas2018.fortiss.org

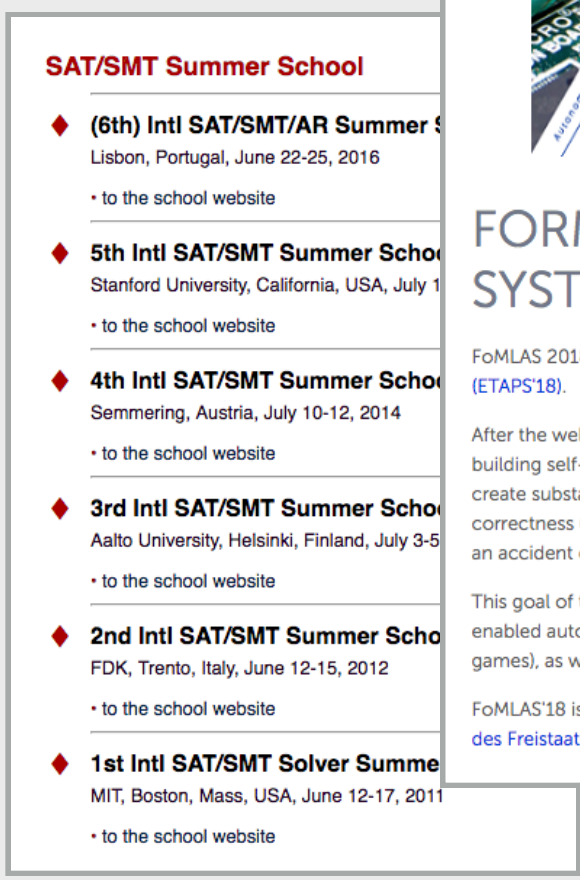
What is SMT (satisfiability modulo theories)?

- Essentially, **constraint solving**
- A bad name for a useful and **extremely practical** idea.
- Burgeoning and vibrant area of CS research

smt-lib.org



fomlas2018.fortiss.org



Chih-Hong Cheng
Fortiss (Germany)

Indranil Saha
IIT Kanpur (India)

TÉCNICO LISBOA
SAT Association
Microsoft Research

What is SMT (satisfiability modulo theories)?

$$((x_1 + 2x_3 < 5) \vee \neg(x_3 \leq 1) \wedge (x_1 \geq 1))$$

satisfiability: Boolean combination

of

theory: linear arithmetic predicates

What is SMT (satisfiability modulo theories)?

$$((x_1 + 2x_3 < 5) \vee \neg(x_3 \leq 1) \wedge (x_1 \geq 1))$$

satisfiability: Boolean combination

of

theory: linear arithmetic predicates

Boolean satisfiability (SAT) solver

Linear programming solver

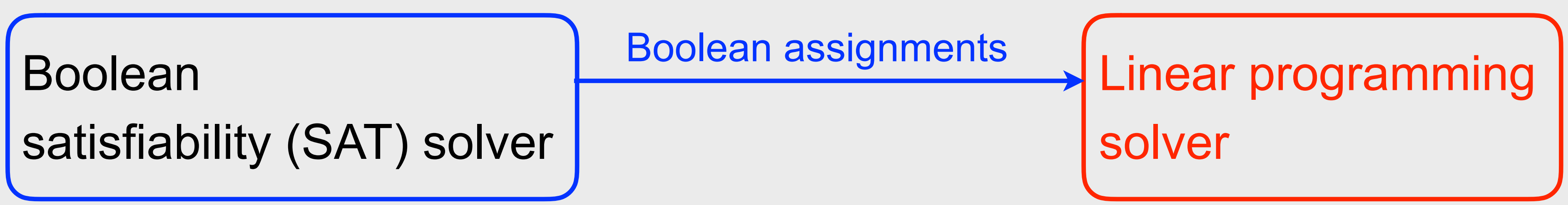
What is SMT (satisfiability modulo theories)?

$$((x_1 + 2x_3 < 5) \lor \neg(x_3 \leq 1) \land (x_1 \geq 1))$$

satisfiability: Boolean combination

of

theory: linear arithmetic predicates



What is SMT (satisfiability modulo theories)?

satisfiability: Boolean combination

of

theory: linear arithmetic predicates

$$\boxed{((x_1 + 2x_3 < 5) \vee \neg(x_3 \leq 1) \wedge (x_1 \geq 1))}$$

Boolean satisfiability (SAT) solver $\xrightarrow{\text{Boolean assignments}}$ Linear programming solver

Linear programming solver $\xrightarrow{\text{Variable assignments or counterexamples}}$ Boolean satisfiability (SAT) solver

What is SMT (satisfiability modulo theories)?

$$((x_1 + 2x_3 < 5) \vee \neg(x_3 \leq 1) \wedge (x_1 \geq 1))$$

satisfiability: Boolean combination

of

theory: linear arithmetic predicates

Boolean satisfiability (SAT) solver

Boolean assignments →

← Variable assignments or counterexamples

Linear programming solver

State-of-the-art SAT solvers: Brute force yet can handle gigantic problem instances

Nature

MATHEMATICS

Maths proof smashes size record

Supercomputer produces a 200-terabyte proof — but is it really mathematics?

Wikipedia

Nevertheless, as of 2016, heuristical SAT-algorithms are able to solve problem instances involving tens of thousands of variables and formulas consisting of millions of symbols,[1]

Future and why (I think) formal methods (will) matter

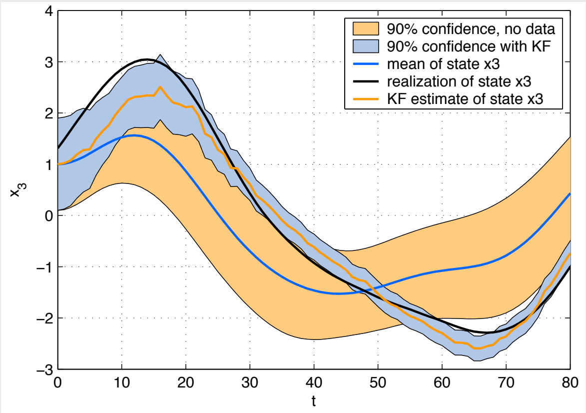
Future and why (I think) formal methods (will) matter

Feedback formalized
Bode, Nyquist, Black

1930s-40s

en.wikipedia.org/wiki/PID_controller

Future and why (I think) formal methods (will) matter



State space, filtering
(Wiener, Bellman, Kalman)

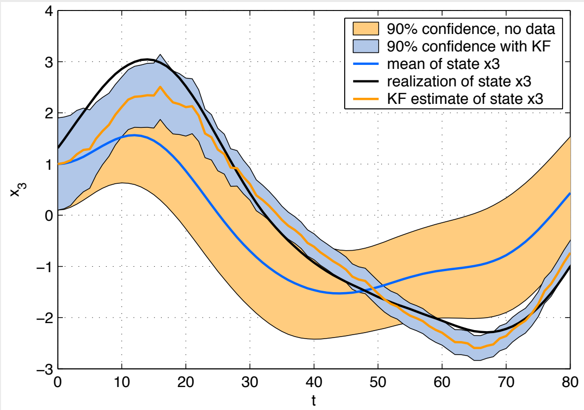
$$\hat{x}(t \mid s) = \mathbf{E}\left(x(t) \mid y(0), \ldots, y(s)\right),$$
$$\hat{\Sigma}_{t|s} = \mathbf{cov}\left(x(t) \mid y(0), \ldots, y(s)\right).$$

Feedback formalized
Bode, Nyquist, Black

1930s-40s 1950s-60s

en.wikipedia.org/wiki/PID_controller

Future and why (I think) formal methods (will) matter



State space, filtering
(Wiener, Bellman, Kalman)

$$\hat{x}(t \mid s) = \mathbf{E}\left(x(t) \mid y(0), \ldots, y(s)\right),$$
$$\hat{\Sigma}_{t|s} = \mathbf{cov}\left(x(t) \mid y(0), \ldots, y(s)\right).$$

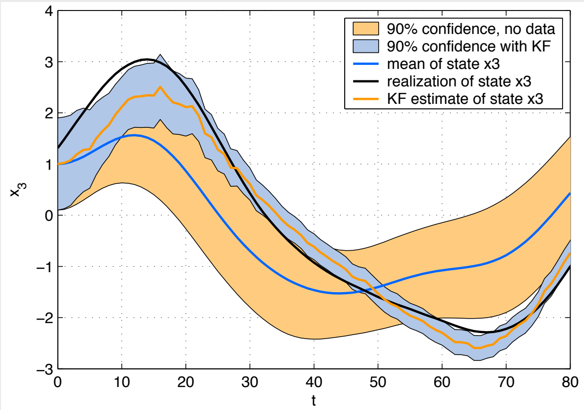
Golden age
(robust control, adaptive control
KYP, LMIs, convex optimization)

Feedback formalized
Bode, Nyquist, Black

1930s-40s 1950s-60s 1970s-2000s

en.wikipedia.org/wiki/PID_controller

Future and why (I think) formal methods (will) matter



State space, filtering
(Wiener, Bellman, Kalman)

$$\hat{x}(t \mid s) = \mathbf{E}\left(x(t) \mid y(0), \ldots, y(s)\right),$$
$$\hat{\Sigma}_{t|s} = \mathbf{cov}\left(x(t) \mid y(0), \ldots, y(s)\right).$$

Feedback formalized
Bode, Nyquist, Black

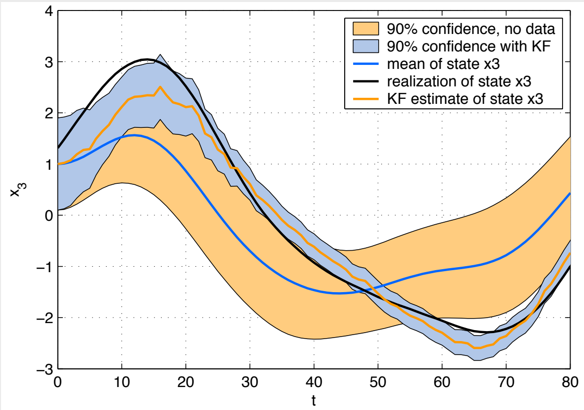
Golden age
(robust control, adaptive control
KYP, LMIs, convex optimization)

1930s-40s   1950s-60s   1970s-2000s   2004

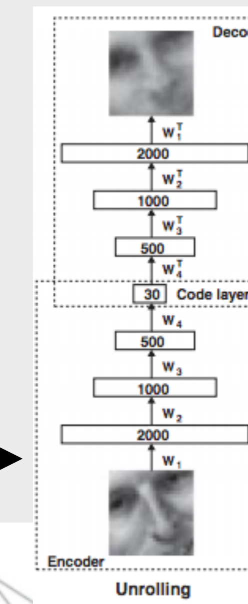
en.wikipedia.org/wiki/PID_controller

Stanley the driverless car.

Future and why (I think) formal methods (will) matter



State space, filtering
(Wiener, Bellman, Kalman)



Feedback formalized
Bode, Nyquist, Black

$$\hat{x}(t \mid s) = \mathbf{E}\left(x(t) \mid y(0), \ldots, y(s)\right),$$
$$\hat{\Sigma}_{t|s} = \mathbf{cov}\left(x(t) \mid y(0), \ldots, y(s)\right).$$

Golden age
(robust control, adaptive control
KYP, LMIs, convex optimization)

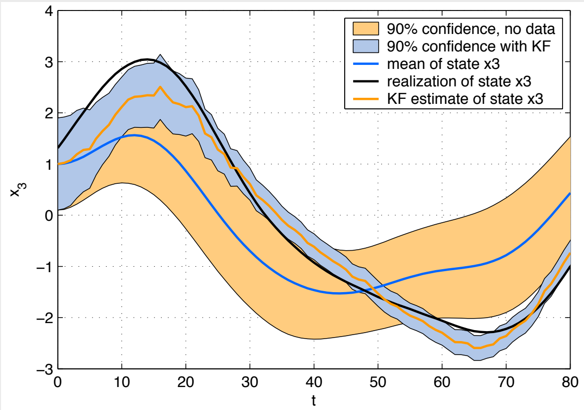
1930s-40s 1950s-60s 1970s-2000s 2004 2006

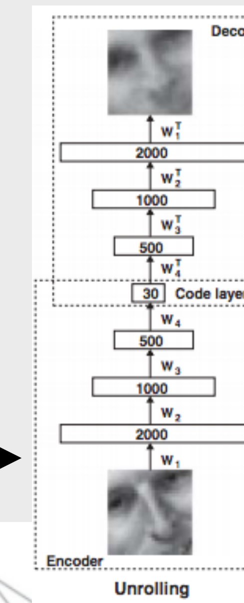
Autonomous vehicles, ML deep networks

en.wikipedia.org/wiki/PID_controller

Stanley the driverless car.

Future and why (I think) formal methods (will) matter





Feedback formalized
Bode, Nyquist, Black

State space, filtering
(Wiener, Bellman, Kalman)

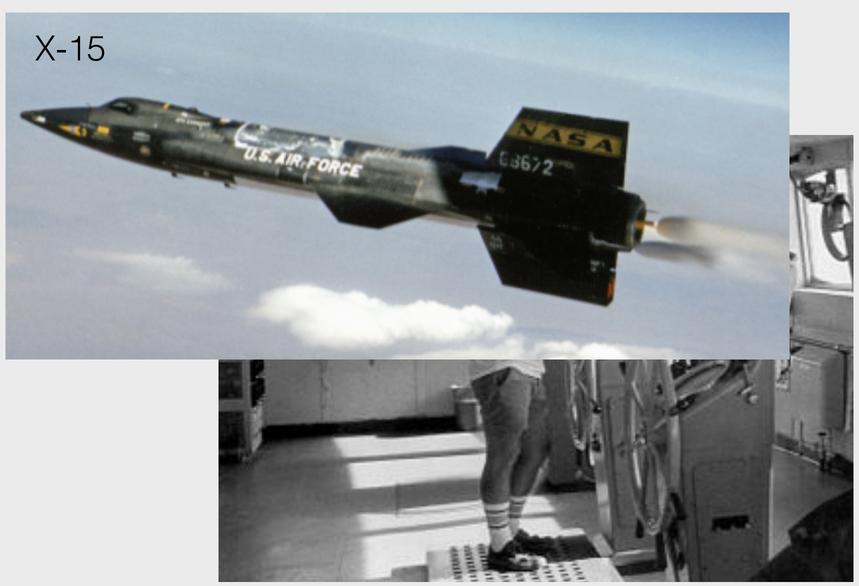
$$\hat{x}(t \mid s) = \mathbf{E}\,(x(t) \mid y(0), \dots, y(s))\,,$$
$$\hat{\Sigma} \quad (t) \mid y(0), \dots, y(s))\,.$$

1967

Golden age
(robust control, adaptive control
KYP, LMIs, convex optimization)

1930s-40s | 1950s-60s | 1970s-2000s | 2004 | 2006

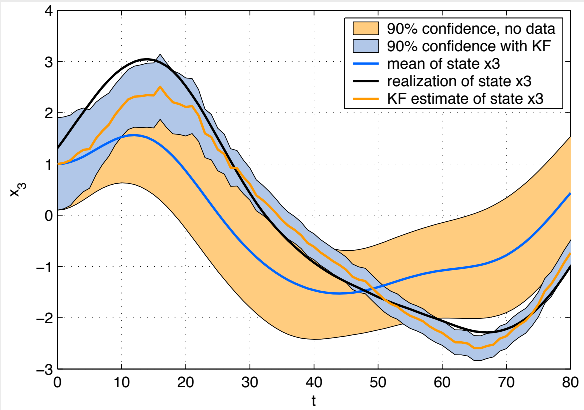
Autonomous vehicles, ML deep networks

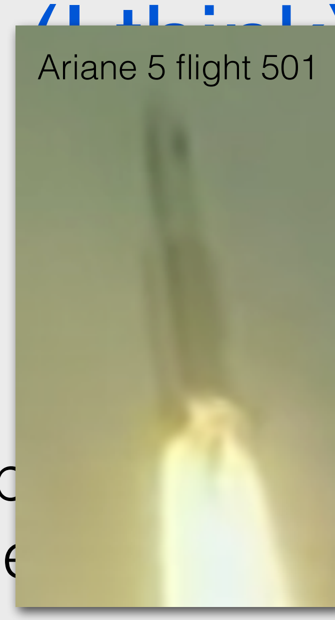


en.wikipedia.org/wiki/PID_controller

Stanley the driverless car.

Future and why (I think) formal methods (will) matter





State sp… g
(Wiener, Be… man)

$\hat{x}(t \mid s) = \mathbf{E}\,(x(t) \mid y(0) \ldots y(s))\,,$

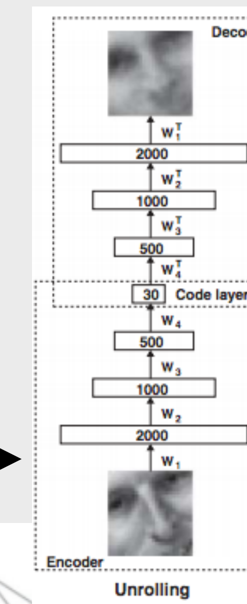
$\hat{\Sigma} \ldots (t) \ldots (s))\,.$

1967

1996

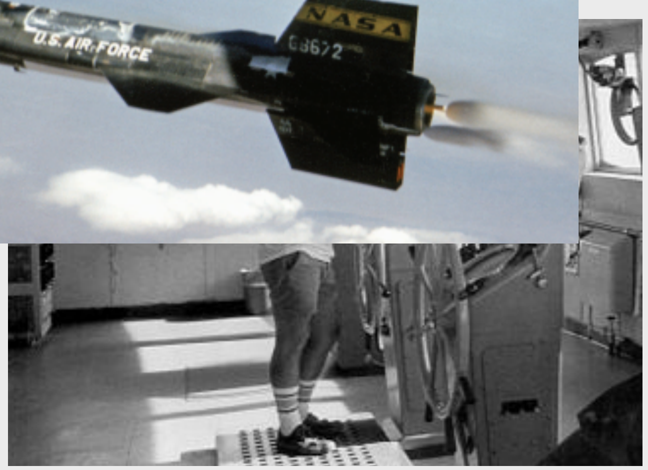
Feedback formalized
Bode, Nyquist, Black

Golden age
(robust control, adaptive control
KYP, LMIs, convex optimization)



1930s-40s 1950s-60s 1970s-2000s 2004 2006

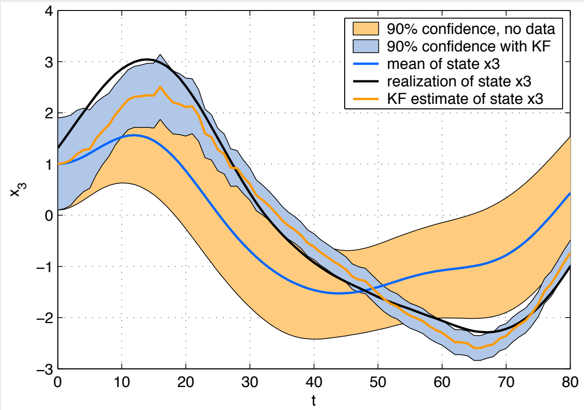
Autonomous vehicles, ML deep networks

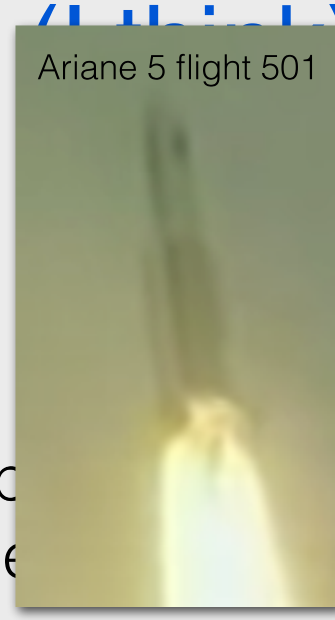


en.wikipedia.org/wiki/PID_controller

Stanley the driverless car.

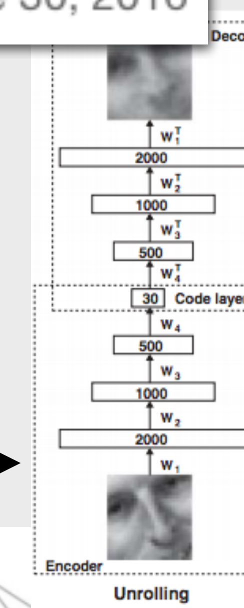
Future and why formal methods (will) matter





A Tragic Loss

The Tesla Team • June 30, 2016



State sp g

(Wiener, Be man)

$\hat{x}(t \mid s) = \mathbf{E}\,(x(t) \mid y(0) \quad y(s))\,,$

$\hat{\Sigma} \quad (t) \quad (s))\,.$

Feedback formalized
Bode, Nyquist, Black

Golden age
(robust control, a e control
KYP, LMIs, conve mization)

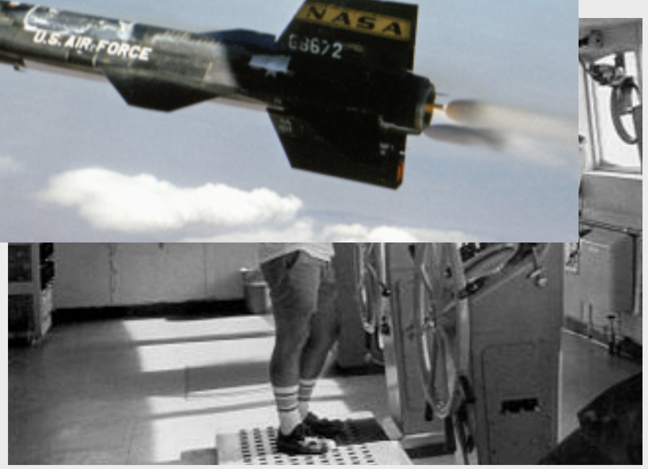
1967

1996

2016

1930s-40s 1950s-60s 1970s-2000s 2004 2006

Autonomous vehicles, ML deep networks

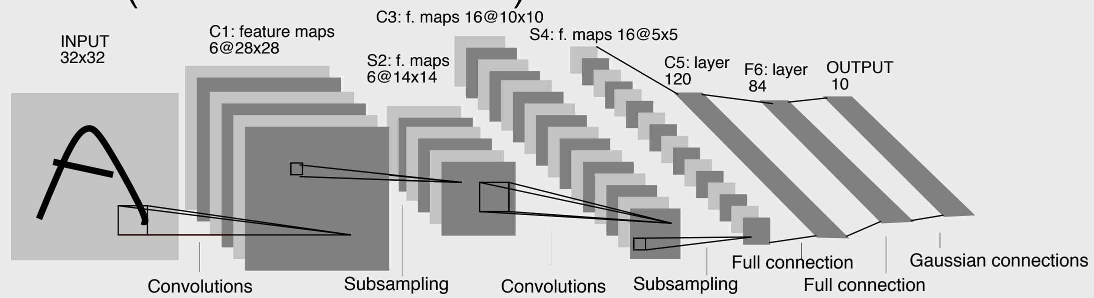


en.wikipedia.org/wiki/PID_controller

Stanley the driverless car.

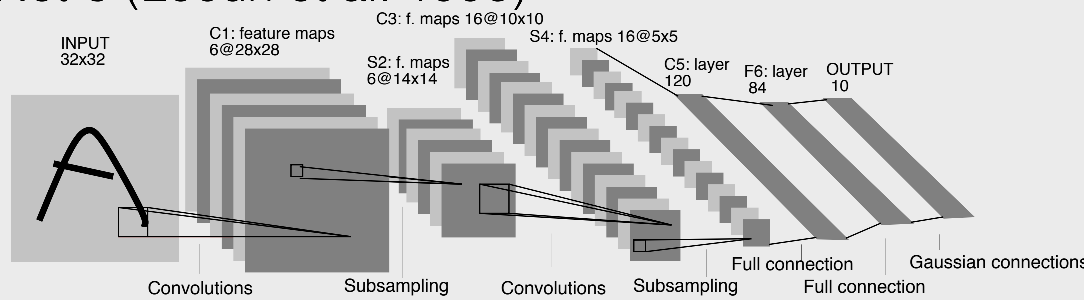
Verifying vision algorithms

LeNet-5 (Lecun et al. 1998)



Verifying vision algorithms

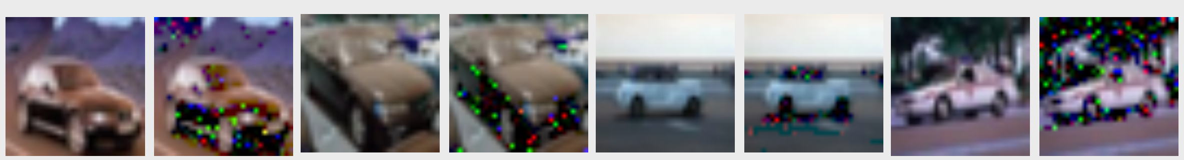
LeNet-5 (Lecun et al. 1998)



Safety Verification of Deep Neural Networks*

Xiaowei Huang, Marta Kwiatkowska, Sen Wang and Min Wu

Department of Computer Science, University of Oxford



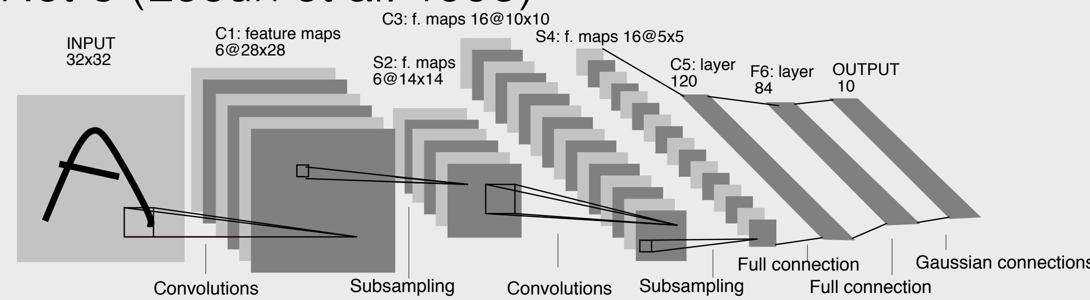
automobile to bird   automobile to frog   automobile to airplane   automobile to horse

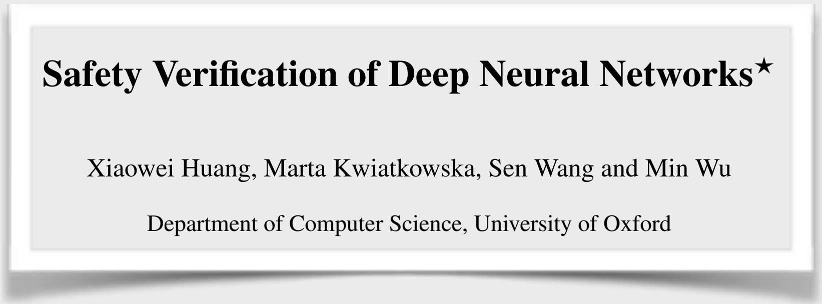
Fig. 1. Automobile images (classified correctly) and their perturbed images (classified wrongly)

Fig. 10. Street sign images. Found an adversarial example for the left image (class changed into bird house), but cannot find an adversarial example for the right image for 20,000 dimensions.

Verifying vision algorithms

LeNet-5 (Lecun et al. 1998)

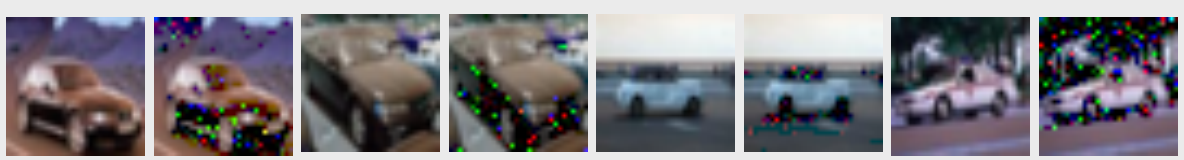




Safety Verification of Deep Neural Networks★

Xiaowei Huang, Marta Kwiatkowska, Sen Wang and Min Wu

Department of Computer Science, University of Oxford



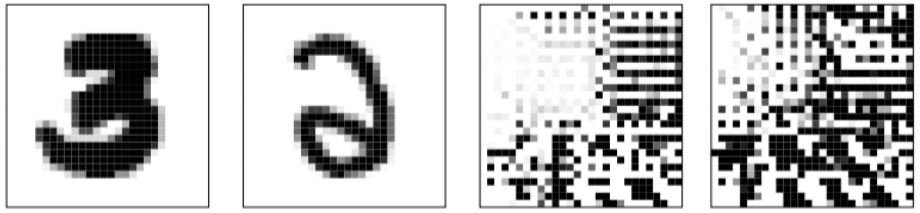
automobile to bird | automobile to frog | automobile to airplane | automobile to horse

Fig. 1. Automobile images (classified correctly) and their perturbed images (classified wrongly)

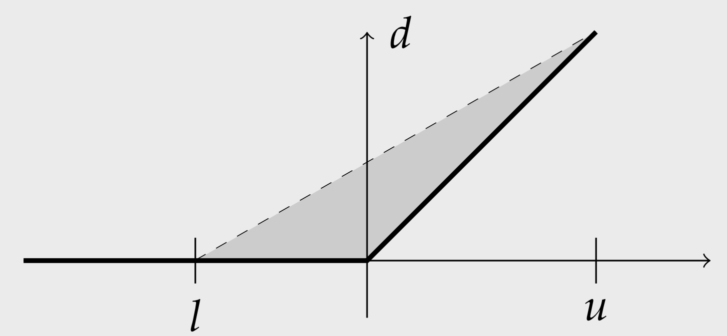
Fig. 10. Street sign images. Found an adversarial example for the left image (class changed into bird house), but cannot find an adversarial example for the right image for 20,000 dimensions.

Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks

Rüdiger Ehlers



(a) '3' digit from the MNIST dataset (b) '2' digit from the MNIST dataset (c) Image classified as digit 2 with $\delta = 20$ (d) Image classified as digit 2 with $\delta = 30$



linear over-approximation of ReLU activation function

SMT encoding of an AMN

Variable (identity map)

$$\boxed{\mathrm{SMT}_x[x, y] \equiv \{y = x\}}$$

SMT encoding of an AMN

Variable
(identity map)

$$\mathrm{SMT}_x[x, y] \equiv \{y = x\}$$

Affine
transformation

$$\alpha(\xi) = W\xi + b,$$

$$\mathrm{SMT}_{\alpha(\varphi_1)}[x, y] \equiv \left\{ \begin{array}{r} \exists v.\,(y = Wv + b) \\ \land\ \mathrm{SMT}_{\varphi_1}[x, v] \end{array} \right\}$$

SMT encoding of an AMN

Variable (identity map)

$$\mathrm{SMT}_x[x, y] \equiv \{y = x\}$$

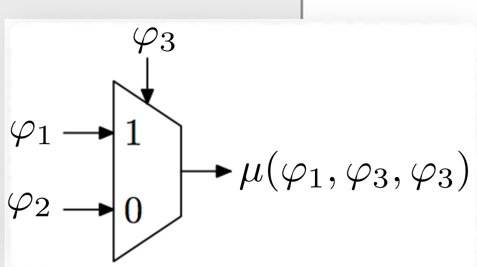
Affine transformation

$$\alpha(\xi) = W\xi + b,$$

$$\mathrm{SMT}_{\alpha(\varphi_1)}[x, y] \equiv \left\{ \begin{array}{r} \exists v.\,(y = Wv + b) \\ \wedge\ \mathrm{SMT}_{\varphi_1}[x, v] \end{array} \right\}$$

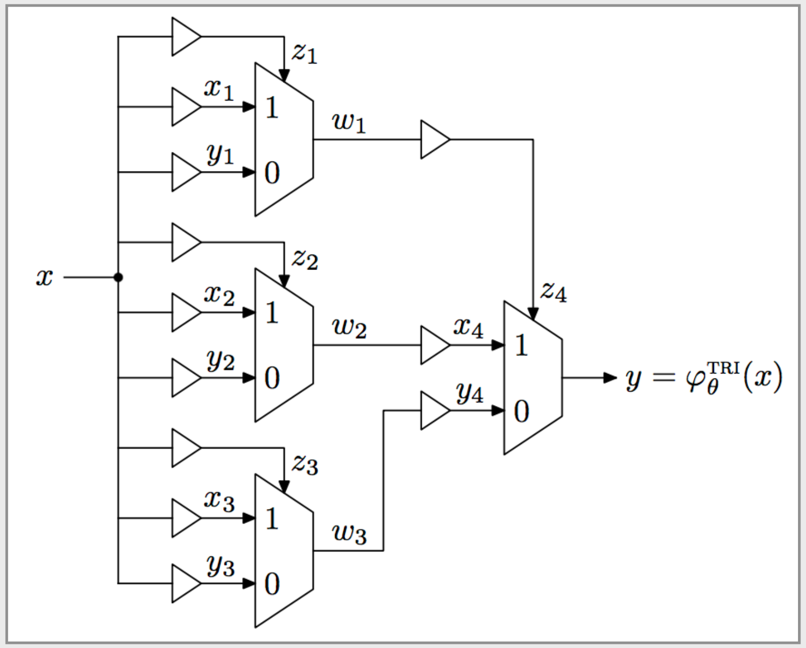
Multiplexing function

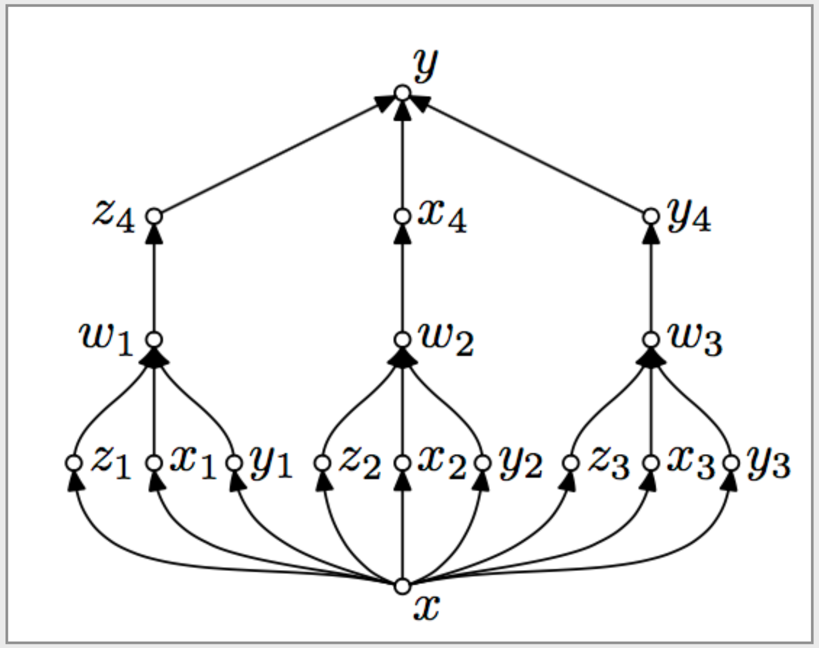
$$\mathrm{SMT}_{\mu(\varphi_1,\varphi_2,\varphi_3)}[x, y] \equiv \left\{ \begin{array}{l} \exists u, v, w.\,((w \leq 0) \to (y = u)) \\ \qquad\qquad \wedge\ (\neg(w \leq 0) \to (y = v)) \\ \qquad\qquad \wedge\ \mathrm{SMT}_{\varphi_1}[x, u] \\ \qquad\qquad \wedge\ \mathrm{SMT}_{\varphi_2}[x, v] \\ \qquad\qquad \wedge\ \mathrm{SMT}_{\varphi_3}[x, w] \end{array} \right\}$$



SMT encoding of a “triplexer”

Triplexer

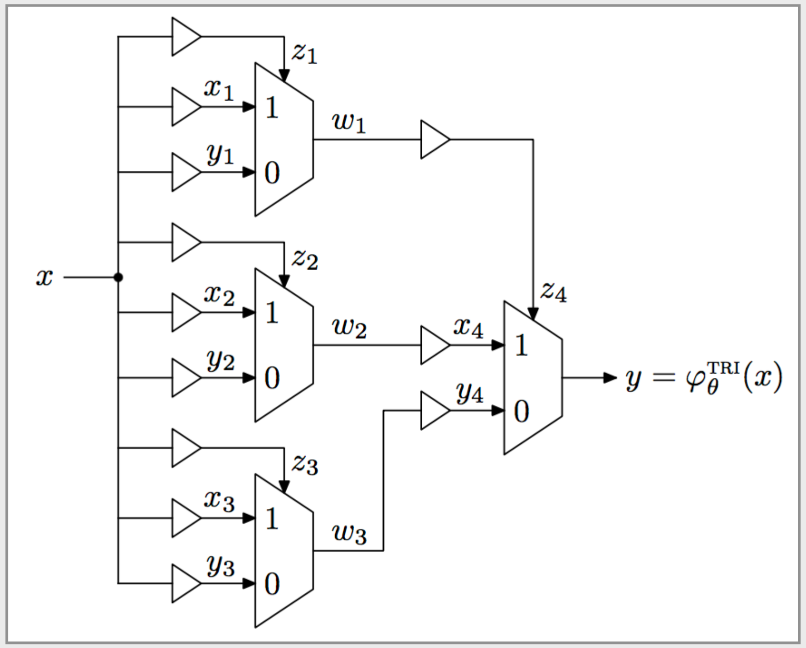


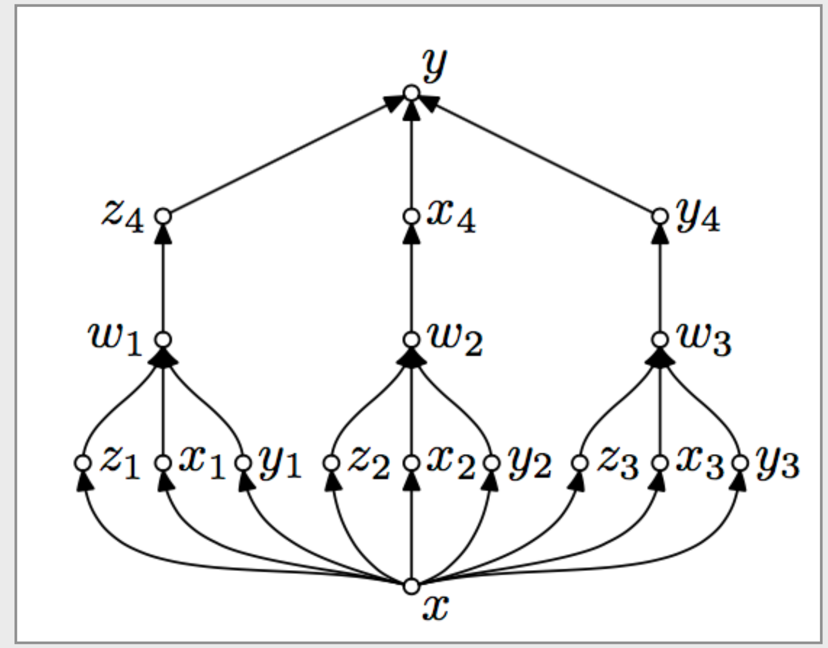


Computation graph

SMT encoding of a “triplexer”

Triplexer





Computation graph

SMT encoding

$$\mathrm{SMT}_{\varphi_\theta^{\mathrm{TRI}}}[x,y] \equiv \left\{ \begin{array}{l} \exists (x_1, y_1, z_1, \ldots, x_4, y_4, z_4, w_1, w_2, w_3) \in \mathbf{R}^{15}. \\ \quad \displaystyle\bigwedge_{i=1}^{3} (x_i = a_i x + b_i \wedge y_i = c_i x + d_i \wedge z_i = e_i x + f_i) \\ \quad \wedge \displaystyle\bigwedge_{j=1}^{3} ((z_j \leq 0) \rightarrow (w_j = x_j)) \wedge (\neg(z_j \leq 0) \rightarrow (w_j = y_j)) \\ \quad \wedge (x_4 = a_4 w_2 + b_4 \wedge y_4 = c_4 w_3 + d_4 \wedge z_4 = e_4 w_1 + f_4) \\ \quad \wedge ((z_4 \leq 0) \rightarrow (y = x_4)) \wedge (\neg(z_4 \leq 0) \rightarrow (y = y_4)) \end{array} \right\}$$

System analysis with AMNs in the loop

discrete-**time** system
with
an AMN φ

$$x^+ = x(t+1) = \varphi(x(t)), \quad x(0) = x_0, \quad t = 0, 1, 2, \ldots,$$

System analysis with AMNs in the loop

discrete-**time** system
with
an AMN φ

$$x^+ = x(t+1) = \varphi(x(t)), \quad x(0) = x_0, \quad t = 0, 1, 2, \ldots,$$

Global asymptotic stability: If there exists V: $R^n \rightarrow R$ such that

$$(V(0) = 0) \ \wedge \ (x \neq 0 \rightarrow V(x) > 0) \ \wedge \ (V(x^+) - V(x) < 0), \ldots$$

System analysis with AMNs in the loop

discrete-**time** system with an AMN φ

$$x^+ = x(t+1) = \varphi(x(t)), \quad x(0) = x_0, \quad t = 0, 1, 2, \ldots,$$

Global asymptotic stability: If there exists *V*: $R^n \rightarrow R$ such that

$$(V(0) = 0) \ \wedge \ (x \neq 0 \rightarrow V(x) > 0) \ \wedge \ (V(x^+) - V(x) < 0), \ldots$$

General idea: Pick a class of V's such that the sufficient conditions can be expressed as an SMT (e.g., with linear arithmetic as the theory).

An example:

$$V(x) = \max_i \ \{a_i^T x + b_i\}$$

Yet, any AMN-representable *V* will do, too.

Search for Lyapunov certificates

Want to solve: $$\exists V \in \mathcal{V}.\, \forall x \in \mathcal{X}.\ \textsc{lyap}(V, x)$$

there exists *V*
satisfying the sufficient conditions
for all x

But, cannot (directly) handle “exists for all” quantifiers.

Search for Lyapunov certificates

Want to solve: $$\boxed{\exists V \in \mathcal{V}.\, \forall x \in \mathcal{X}.\ \textsc{lyap}(V, x)}$$

there exists *V*
satisfying the sufficient conditions
for all x

But, cannot (directly) handle "exists for all" quantifiers.

Counterexample-guided search:

algorithm: Counterexample guided Lyapunov search
initialize: $k := 0$, $x_0 \in \mathcal{X}$, $\mathcal{X}_0 := \{x_0\}$
repeat:
1. *Search for a candidate Lyapunov function.*
   if $\text{E-}\textsc{solve}(\mathcal{X}_k)$ **then:**
   $V_k :=$ solution to $\text{E-}\textsc{solve}(\mathcal{X}_k)$
   else return FALSE/UNKNOWN
2. *Generate counterexample.*
   if $\text{F-}\textsc{solve}(V_k)$ **then:**
   $x_{k+1} :=$ solution to $\text{F-}\textsc{solve}(V_k)$
   else return TRUE
3. *Update counterexample set.*
   $\mathcal{X}_{k+1} := \mathcal{X}_k \cup \{x_{k+1}\}$
   $k := k + 1$

until: stopping criterion

Search for Lyapunov certificates

Want to solve: $\boxed{\exists V \in \mathcal{V}.\, \forall x \in \mathcal{X}.\ \textsc{lyap}(V, x)}$

there exists *V*
satisfying the sufficient conditions
for all x

But, cannot (directly) handle “exists for all” quantifiers.

Counterexample-guided search:

algorithm: Counterexample guided Lyapunov search
initialize: $k := 0$, $x_0 \in \mathcal{X}$, $\mathcal{X}_0 := \{x_0\}$
repeat:
1. *Search for a candidate Lyapunov function.*
   if E-SOLVE($\mathcal{X}_k$) **then**:
     $V_k :=$ solution to E-SOLVE($\mathcal{X}_k$)
   else return FALSE/UNKNOWN
2. *Generate counterexample.*
   if F-SOLVE(V_k) **then**:
     $x_{k+1} :=$ solution to F-SOLVE(V_k)
   else return TRUE
3. *Update counterexample set.*
   $\mathcal{X}_{k+1} := \mathcal{X}_k \cup \{x_{k+1}\}$
   $k := k + 1$

until: stopping criterion

Generate candidate *V*
(LYAP imposed over a subset of the states)

$$\boxed{\text{E-SOLVE}(\mathcal{X}_c) \equiv \exists V \in \mathcal{V}.\ \bigwedge_{x_c \in \mathcal{X}_c} \textsc{lyap}(V, x_c)}$$

Search for Lyapunov certificates

Want to solve: $\boxed{\exists V \in \mathcal{V}.\, \forall x \in \mathcal{X}.\ \textsc{lyap}(V, x)}$

there exists V
satisfying the sufficient conditions
for all x

But, cannot (directly) handle “exists for all” quantifiers.

Counterexample-guided search:

algorithm: Counterexample guided Lyapunov search
initialize: $k := 0$, $x_0 \in \mathcal{X}$, $\mathcal{X}_0 := \{x_0\}$
repeat:
1. *Search for a candidate Lyapunov function.*
   if E-SOLVE($\mathcal{X}_k$) **then:**
   $V_k :=$ solution to E-SOLVE($\mathcal{X}_k$)
   else return FALSE/UNKNOWN
2. *Generate counterexample.*
   if F-SOLVE(V_k) **then:**
   $x_{k+1} :=$ solution to F-SOLVE(V_k)
   else return TRUE
3. *Update counterexample set.*
   $\mathcal{X}_{k+1} := \mathcal{X}_k \cup \{x_{k+1}\}$
   $k := k + 1$

until: stopping criterion

Generate candidate *V*
(LYAP imposed over a subset of the states)

$$\text{E-SOLVE}(\mathcal{X}_c) \equiv \exists V \in \mathcal{V}.\ \bigwedge_{x_c \in \mathcal{X}_c} \textsc{lyap}(V, x_c)$$

$$\text{F-SOLVE}(V) \equiv \exists x \in \mathcal{X}.\, \neg\textsc{lyap}(V, x)$$

Find states at which LYAP is violated.
Revise the conditions for step (1)

Sanity check

A randomly generated linear system
with spectral radius of 0.75.

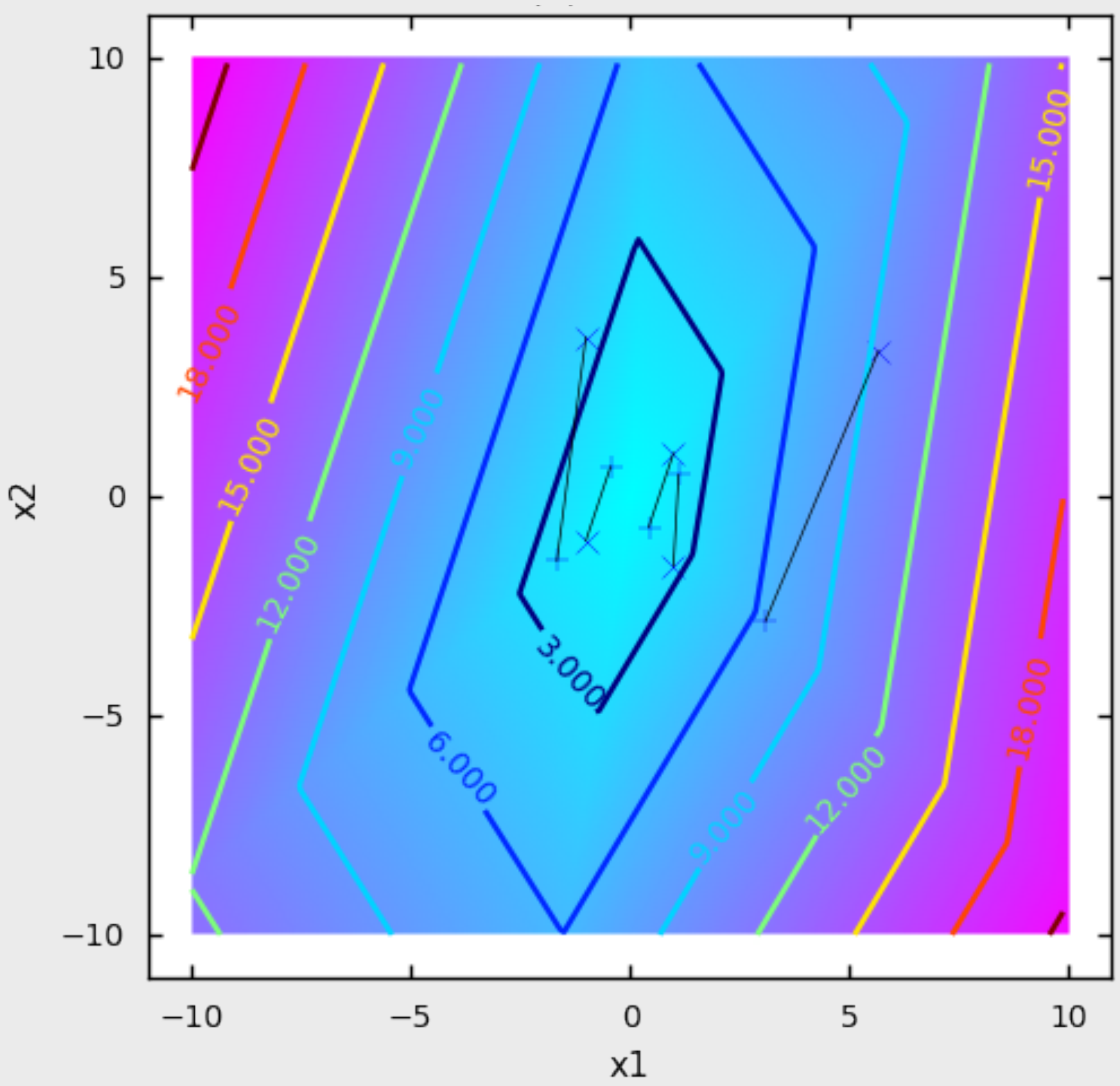
Sanity check

A randomly generated linear system with spectral radius of 0.75.

5 iterations of the algorithm constructs a Lyapunov function.

Each iteration adds a “face” to *V*.

$$V(x) = \max_i \; \{a_i^T x + b_i\}$$



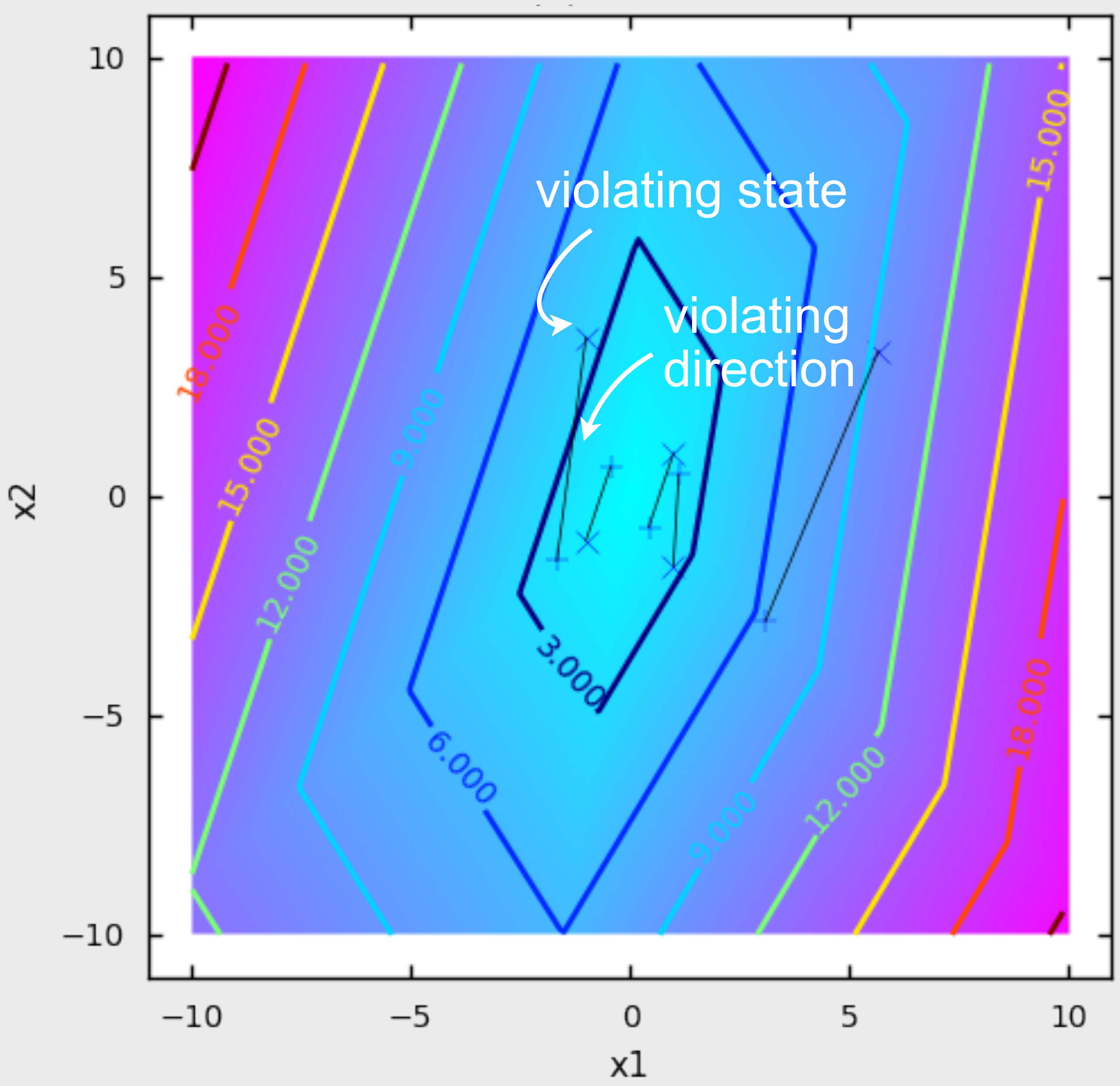
Sanity check

A randomly generated linear system with spectral radius of 0.75.

5 iterations of the algorithm constructs a Lyapunov function.

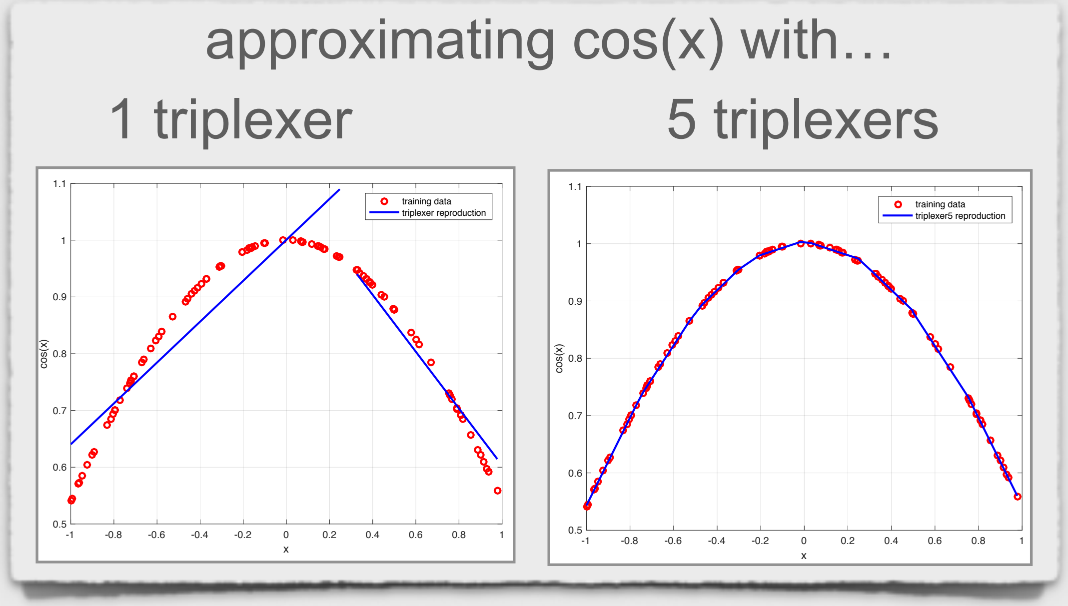
Each iteration adds a “face” to *V*.

$$V(x) = \max_i \ \{a_i^T x + b_i\}$$



Completeness of AMNs in function approximation

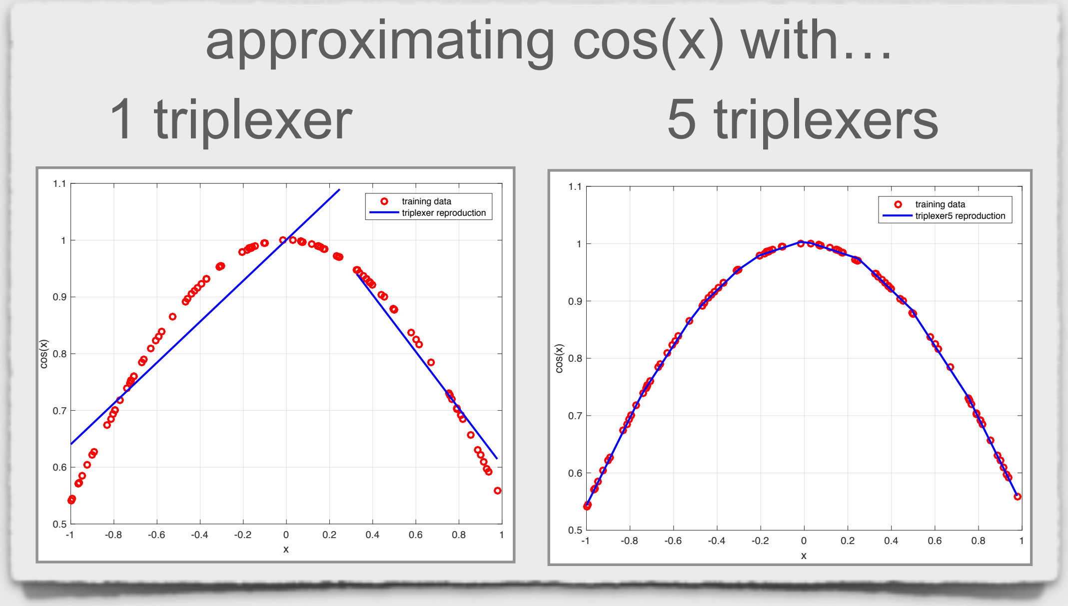
Completeness over continuous functions is inherited from similar completeness of neural networks.



Given a compact subset $X \subset \mathbf{R}^n$ and a continuous function $f : X \to \mathbf{R}$, there exists an affine multiplexing network $\varphi : X \to \mathbf{R}$ that approximates f arbitrarily well on X in L_2-norm.

Completeness of AMNs in function approximation

Completeness over continuous functions is inherited from similar completeness of neural networks.



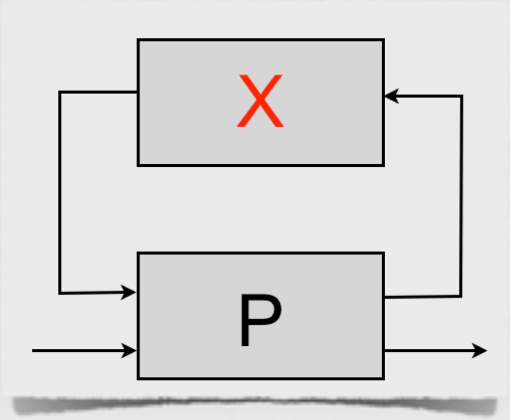
Given a compact subset $X \subset \mathbf{R}^n$ and a continuous function $f : X \to \mathbf{R}$, there exists an affine multiplexing network $\varphi : X \to \mathbf{R}$ that approximates f arbitrarily well on X in L_2-norm.

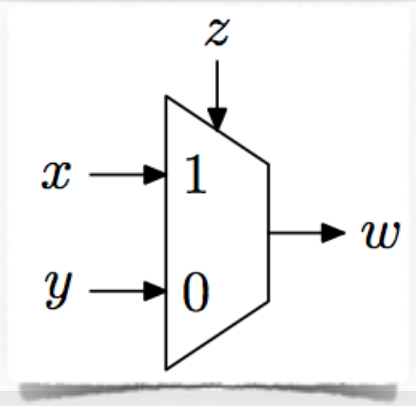
Open question: Can the added expressivity of AMNs be utilized for completeness over a broader family of functions?

Summary

Summary

- X-in-the-loop control
- Affine multiplexing networks
- Connections to system analysis, learning and decision procedures

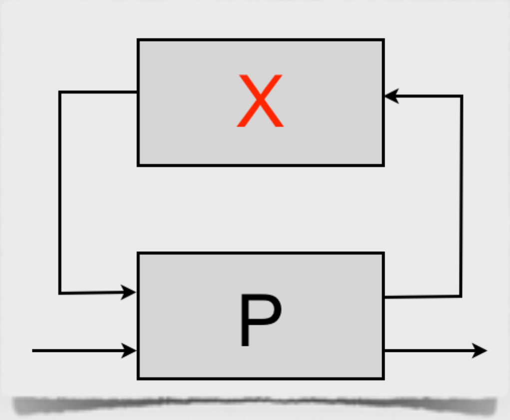


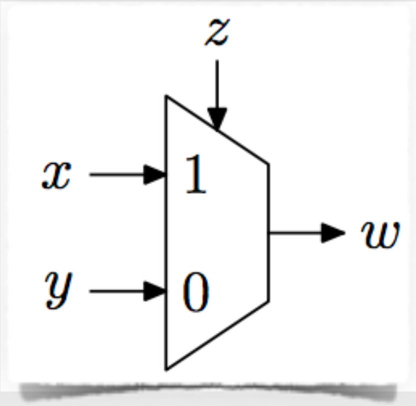


Summary

Summary

- X-in-the-loop control
- Affine multiplexing networks
- Connections to system analysis, learning and decision procedures





A few (potential) **future steps**

- Alternative means for nonlinear control synthesis and beating fundamental limitations
- Training Lyapunov-type certificates in AMN form directly from data (i.e., simulations)
- Continuous-time dynamics (modulo technicalities)
- Control-oriented theories in SMT
- Domain-specific languages

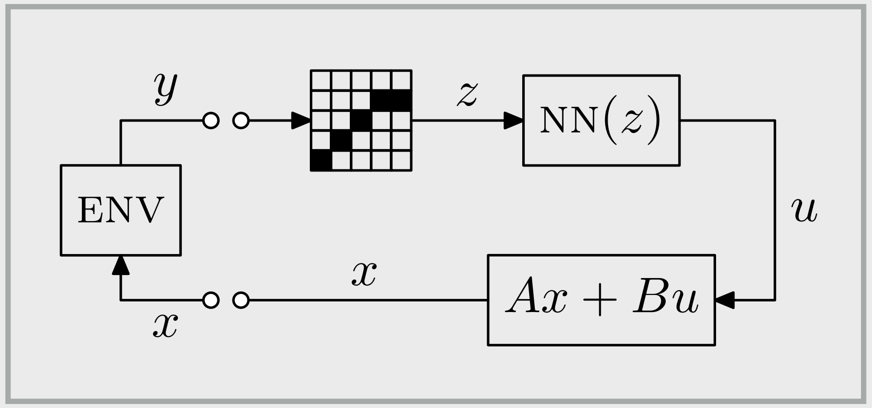
$$\mu_{\mathcal{K}}(x,y,z) = \begin{cases} x, & \text{if } z \in \mathcal{K}, \\ y, & \text{otherwise.} \end{cases}$$

arbitrary cone
(AMN: $z \leq 0$ for $z \in \mathbb{R}$)

Difficulties

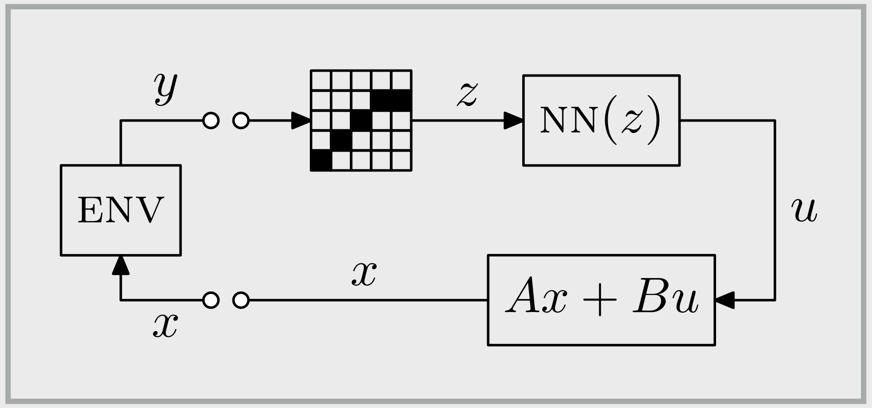
- At its heart, involves at least (worst-case) an **NP-complete** computation
- Model-based (IQCs, robust control)
- Some good tools exist (Z3, yices), but these require expert knowledge to use
- Writing a new Python toolbox (AMNET) as a modeling and query layer

Difficulties



- At its heart, involves at least (worst-case) an **NP-complete** computation
- Model-based (IQCs, robust control)
- Some good tools exist (Z3, yices), but these require expert knowledge to use
- Writing a new Python toolbox (AMNET) as a modeling and query layer

Difficulties

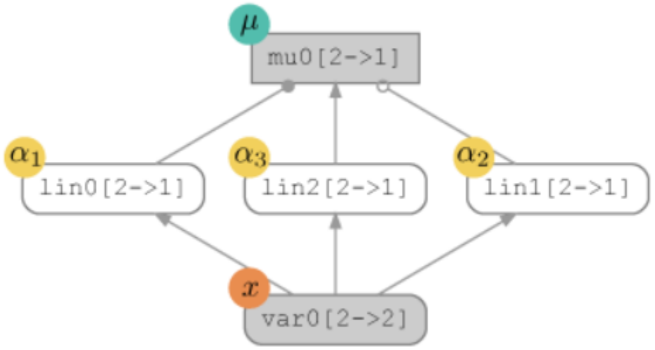


- At its heart, involves at least (worst-case) an **NP-complete** computation
- Model-based (IQCs, robust control)
- Some good tools exist (Z3, yices), but these require expert knowledge to use
- Writing a new Python toolbox (AMNET) as a modeling and query layer

https://github.com/ipapusha/amnet

AMNET: Affine Multiplexing Network Toolbox

AMNET is a Python toolbox that assists in building certain kinds of neural networks, and formally verifying their behavior in-the-loop (under development).



1. Multiplexing nonlinearities

$$\mathtt{max2}(x) = \mu(\alpha_1(x), \alpha_2(x), \alpha_3(x))$$
$$= \mathtt{if}\ (-x_1 + x_2 \le 0)\ \mathtt{then}\ x_1\ \mathtt{else}\ x_2$$

2. Affine transformations

$$\alpha_1(x) := e_1^T x \quad \alpha_2(x) := e_2^T x \quad \alpha_3(x) := -x_1 + x_2$$

3. Input variables

$$x \in \mathbf{R}^2$$

```
import numpy as np
from amnet import Variable, Linear, Mu

# a two-dimensional input variable
x = Variable(2, name='x')

# choose components
a1 = Linear(np.array([[1, 0]]), x)
a2 = Linear(np.array([[0, 1]]), x)

# find difference
a3 = Linear(np.array([[-1, 1]]), x)

# if a3 <= 0, returns a1; otherwise a2
phimax = Mu(a1, a2, a3)

# equivalently, we can also write
# phimax = amnet.atoms.max_all(x)

print phimax
print phimax.eval([1, -2]) # returns: 1
```