AI meets Formal: verification and optimization of systems modelled using machine learning

Konstantin Korovin

joint work with Franz Brauße and Zurab Khasidashvili

[1] The University of Manchester, UK

[2] Intel Israel Development Center

AI meets Formal: verification and optimization of systems modelled using machine learning

Konstantin Korovin

joint work with Franz Brauße and Zurab Khasidashvili

[1] The University of Manchester, UK

[2] Intel Israel Development Center

CAV'24: SMLP: Symbolic Machine Learning Prover

IJCAI'22: Combining Constraint Solving and Bayesian Techniques for System Optimization

FMCAD'20 Selecting Stable Safe Configurations for Systems Modelled by Neural Networks with ReLU Activation

What is Artificial Intelligence (AI) ?

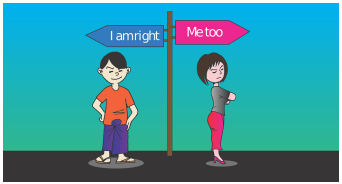
What is Artificial Intelligence (AI) ?

Data driven AI Symbolic AI

What is Artificial Intelligence (AI) ?

Data driven AI Symbolic AI

Learning Thinking



Data driven AI vs symbolic (AI)

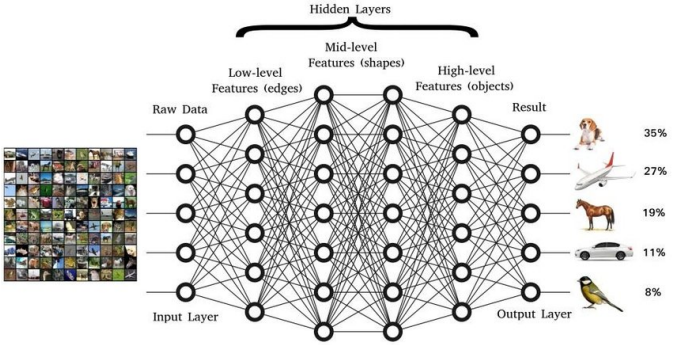
Data driven AI aka Machine Learning

- we don't know problem structure
- we have data
- goal: learn the model that fits the data
- method: machine learning
- no explanations;
  works but don't know why
- reasoning is very limited
- for many applications prediction accuracy is not sufficient: e.g., plane safety

Symbolic AI aka Formal Methods

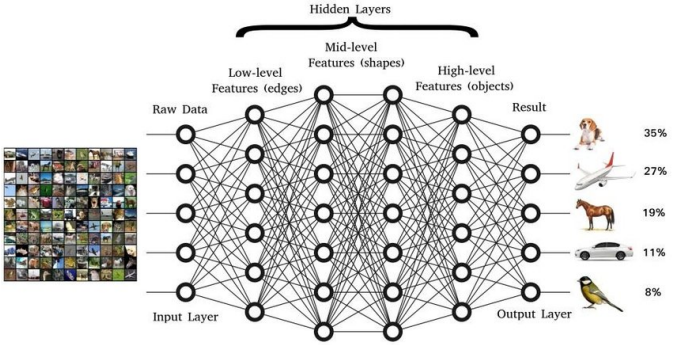
- problem has a mathematical model
- we want to infer properties of the problem
- method: automated reasoning, verification
- exact and general results
- detailed explanations – proofs
- mathematics is based on reasoning

Data driven AI: machine learning



Deep Neural Network [1]

Data driven AI: machine learning



Deep Neural Network [1]

- Image/speech recognition/self-driving cars
- Finance/targeted advertising/fraud prediction/social engineering
- Generative AI, LLMs. ChatGPT and all that

Why ML is so powerful ?

Why ML is so powerful ?

Why ML is so powerful ?

Why ML is so powerful ? Nobody knows....

Why ML is so powerful ?

Why ML is so powerful ? Nobody knows.... but...

From mathematical point of view ML model is just a function:

$$f : \mathbb{R}^n \mapsto \mathbb{R}^m$$

Why ML is so powerful ?

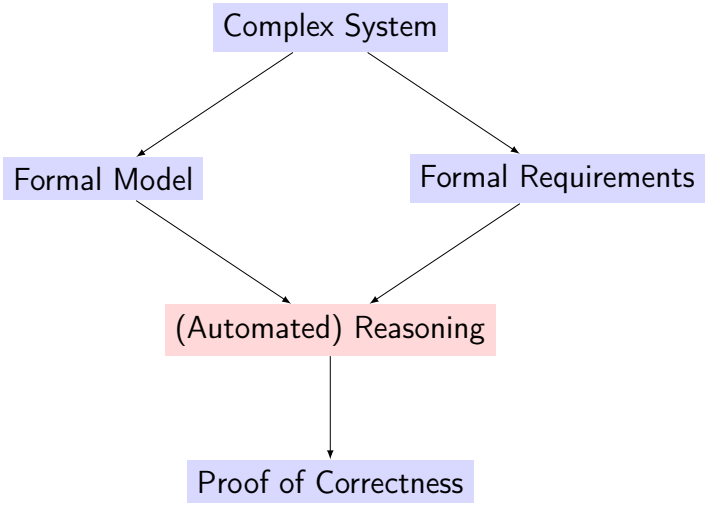
Why ML is so powerful ? Nobody knows.... but...

From mathematical point of view ML model is just a function:

$$f : \mathbb{R}^n \mapsto \mathbb{R}^m$$

Universal approximation theorem: neural networks can approximate any continuous function with sufficient accuracy.

Formal Methods



Why Formal Methods are so powerful ?

Why formal methods are so powerful ?

Why Formal Methods are so powerful ?

Why formal methods are so powerful ? Nobody knows....

Why Formal Methods are so powerful ?

Why formal methods are so powerful ? Nobody knows.... but...

"Universal representation" any mathematical problem can be represented in formal logic and in many cases complete reasoning methods exist.

Why Formal Methods are so powerful ?

Why formal methods are so powerful ? Nobody knows.... but...

"Universal representation" any mathematical problem can be represented in formal logic and in many cases complete reasoning methods exist.

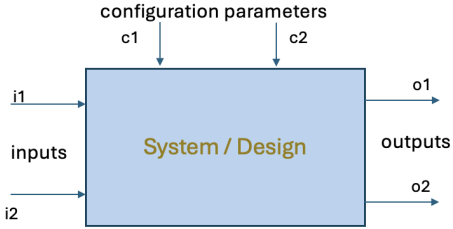
SAT/SMT solvers scale to formulas with millions of variables.

- Mathematical reasoning:
  Erdős discrepancy problem solved by a SAT solver (12Gb proof)
  Pythagorean triples problem solved by a SAT solver (200Tb proof)

- FM are used in large scale verification of hardware and software:
  Intel, Microsoft, NASA, Facebook, AWS

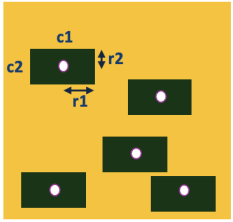
Motivating problem form hardware design

Safety and Optimality with Stability

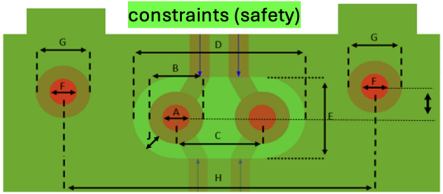


Some optimization applications require the selected configurations to (1) satisfy constraints (2) be robust (stable) against perturbations, and (3) be optimal:

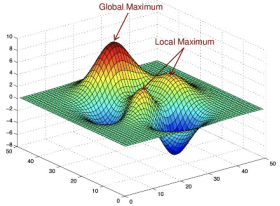
Robustness (stability)



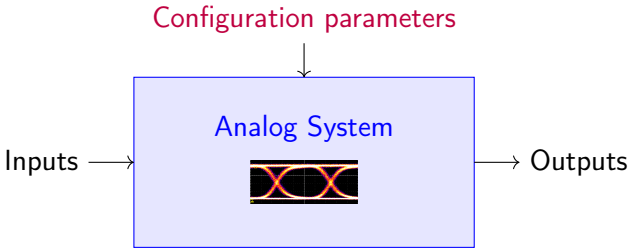
constraints (safety)



(pareto) near-optimality



Research Objective

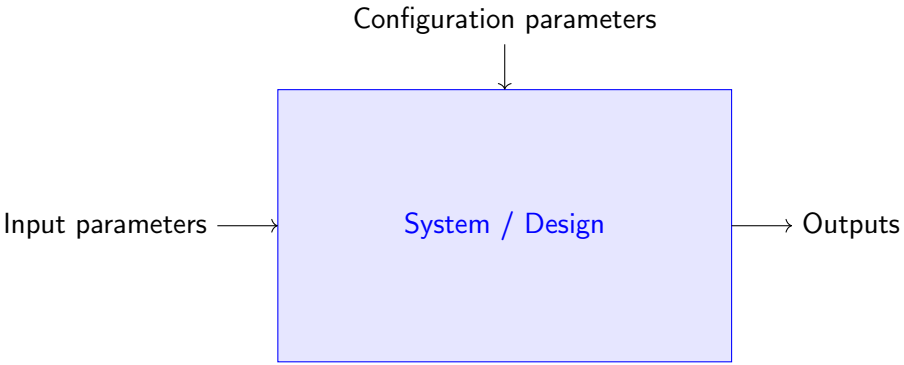


Objective: Find a close to optimal configurations such that the output satisfies a spec for all inputs.

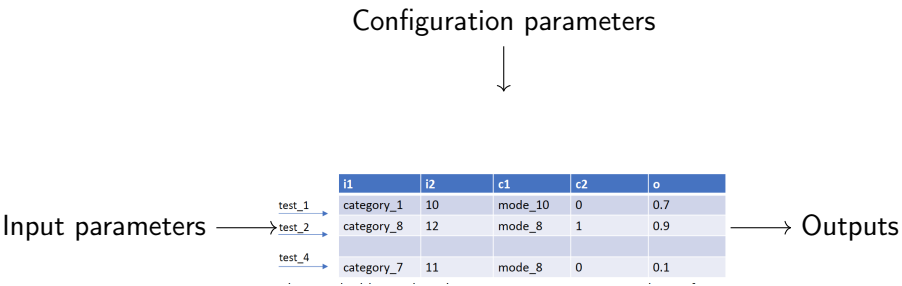
Applications:

- Finding synthesis parameters for area/power/performance tradeoff
- Circuit layout
- Signal integrity
- Design/Arch exploration for correct and close to optimal design
- BIOS configuration optimisation

Our approach: High level overview

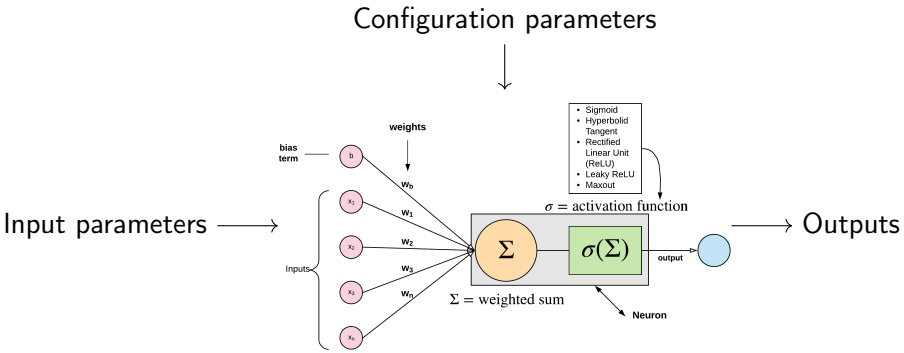


Our approach: High level overview



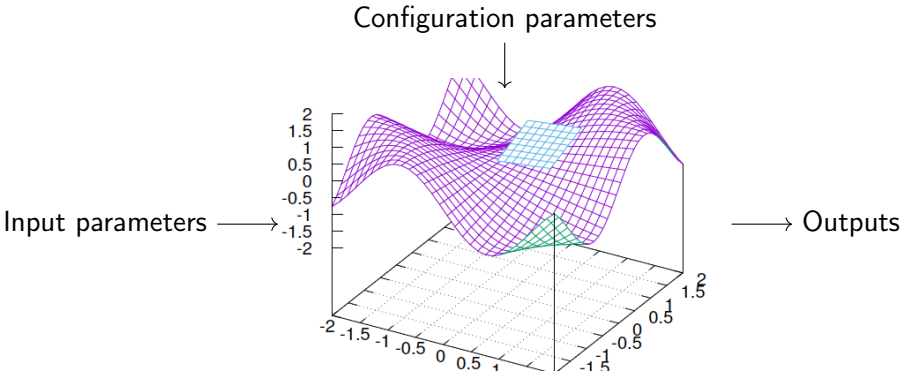
1. Generate training data representing input/output system behavior.

Our approach: High level overview



1. Generate training data representing input/output system behavior.
2. Build mathematical model approximating the system using machine learning.

Our approach: High level overview



1. Generate training data representing input/output system behavior.
2. Build mathematical model approximating the system using machine learning.
3. Use symbolic methods: constraint solving and theorem proving to analyze the model.

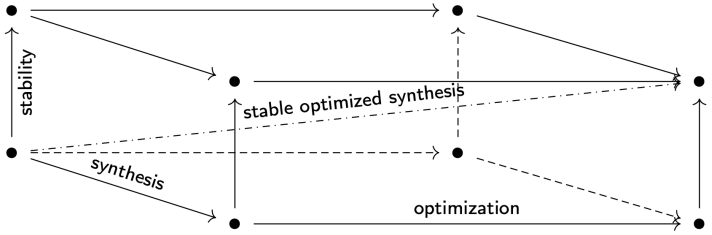
System Exploration Cube

Parameter synthesis Finding model parameter values s.t. system constraints are valid.

Parameter optimization Optimization of one (or more) objective functions under constraints.

Stability of solutions Guaranteed within entire stability region.

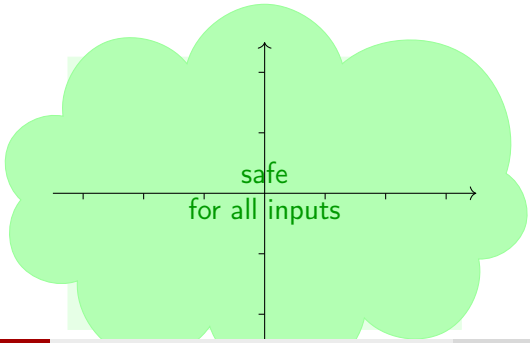
- Expressed as a relation θ on configuration parameters.



System exploration modes are combinations of these directions.

Configuration problems

No config. parameters: verification – for all inputs x system satisfies spec.: constraint solvers/SMT solvers $\forall x.\, \varphi_{spec}(x)$

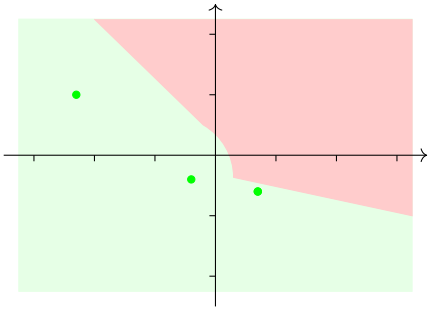


Configuration problems

No config. parameters: verification – for all inputs x system satisfies spec.: constraint solvers/SMT solvers $\forall x.\, \varphi_{spec}(x)$

SMLP:

Find configurations safe for all inputs: $\exists p\, \forall x.\, \varphi_{spec}(p, x)$



Configuration problems

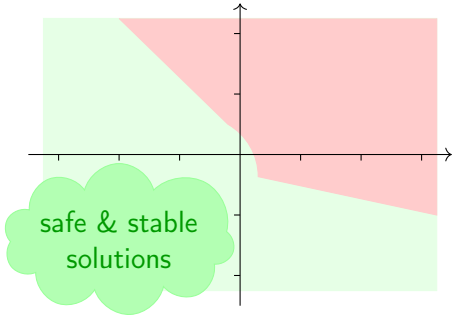
No config. parameters: verification – for all inputs x system satisfies spec.: constraint solvers/SMT solvers $\forall x.\, \varphi_{spec}(x)$

SMLP:

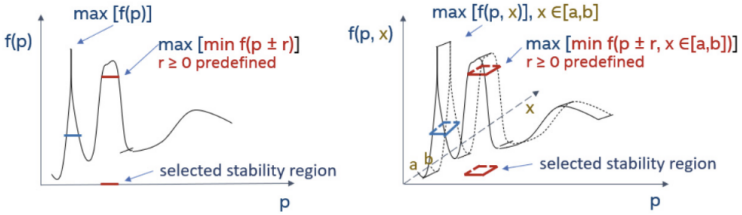
Find configurations safe for all inputs: $\exists p \forall x.\, \varphi_{spec}(p, x)$

Find configurations safe and stable for all inputs: $\exists p \forall q \forall x.\, (\|p - q\| \leq r \rightarrow \varphi_{spec}(q, x))$

$\varphi_{spec}(q, x)$ includes spec of the ML model and optimality constraints.



Safe and stable solutions



Finding safe and stable solutions as max-min optimization:

$$[\![\varphi_M]\!]_{o,\theta} = \max_{p} \min_{x,p'} \{z \mid \eta(p) \wedge \forall y \ [\theta(p,p') \rightarrow (\varphi_M(p',x,y) \rightarrow \varphi^{\leq}_{cond}(p',x,y,z))]\}$$

SMLP

Optimization procedure for systems represented using neural networks, polynomial models, trees, random forests.

Finds safe & stable regions with formally guaranteed accuracy.

SMLP

Optimization procedure for systems represented using neural networks, polynomial models, trees, random forests.

Finds safe & stable regions with formally guaranteed accuracy.

safety solutions satisfy user-defined constraints specified in the language

SMLP

Optimization procedure for systems represented using neural networks, polynomial models, trees, random forests.

Finds safe & stable regions with formally guaranteed accuracy.

safety	solutions satisfy user-defined constraints specified in the language
language	linear arithmetic over $\mathbb{R}$, $\mathbb{Z}$ (`QF_LIRA`), non-linear, equality on UF

SMLP

Optimization procedure for systems represented using neural networks, polynomial models, trees, random forests.

Finds safe & stable regions with formally guaranteed accuracy.

safety	solutions satisfy user-defined constraints specified in the language
language	linear arithmetic over $\mathbb{R}$, $\mathbb{Z}$ (`QF_LIRA`), non-linear, equality on UF
stable regions	typically defined by a radius around a point (aka robustness) *all points* in region satisfy constraints

SMLP

Optimization procedure for systems represented using neural networks, polynomial models, trees, random forests.

Finds safe & stable regions with formally guaranteed accuracy.

safety	solutions satisfy user-defined constraints specified in the language
language	linear arithmetic over $\mathbb{R}$, $\mathbb{Z}$ (`QF_LIRA`), non-linear, equality on UF
stable regions	typically defined by a radius around a point (aka robustness) *all points* in region satisfy constraints
accuracy	given $\varepsilon > 0$, worst-case point in region is at most ε below the optimum

SMLP

Optimization procedure for systems represented using neural networks, polynomial models, trees, random forests.

Finds safe & stable regions with formally guaranteed accuracy.

safety	solutions satisfy user-defined constraints specified in the language
language	linear arithmetic over $\mathbb{R}$, $\mathbb{Z}$ (`QF_LIRA`), non-linear, equality on UF
stable regions	typically defined by a radius around a point (aka robustness) *all points* in region satisfy constraints
accuracy	given $\varepsilon > 0$, worst-case point in region is at most ε below the optimum

- based on GearSAT procedure solving $\exists^*\forall^*$ formulas over the language [FMCAD'20]

Setting

Consider theory $\mathcal{T}_{\|\cdot\|}$ with quantifier alternations:

- sorts for reals, integers,
- linear arithmetic with real coefficients and variables of mixed real and integer sorts,
- usual arithmetic comparison operators,
- norm $\|\cdot\|$,
- Boolean operators, and
- a collection of activation functions AF.

Assume $\texttt{ReLU} \in AF$. Other functions possible.

Assume there is a decision procedure S for the quantifier-free fragment of $\mathcal{T}_{\|\cdot\|}$.

GEAR fragment of reflexively guarded $\exists^*\forall^*$ formulas

Definition (GEAR fragment)

Closed first-order formulas of the form

$$\exists p.\, \eta(p) \wedge \forall q.\, (\theta(p,q) \rightarrow \forall x.\, \psi(q,x))$$

where

- η, θ, ψ are quantifier-free over $\mathcal{T}_{\|\cdot\|}$, and
- the θ defines a reflexive relation.

η	config spec.
θ	stability guard
ψ	system spec.

Z3, CVC5 fail to solve a single example in our domain in a direct encoding.

An incomplete procedure

decision problem $\exists p.\, \forall x.\, \varphi(p, x)$

Loop:

1. Find assignment α of p, x satisfying
$$\varphi(p, x)$$
or return `unsat`.
2. If $\neg\varphi([\![p]\!]^\alpha, x)$ is `unsat`
return `sat` with solution $[\![p]\!]^\alpha$.
3. Add lemma $\varphi(p, x) \leftarrow \varphi(p, x) \wedge (p \neq [\![p]\!]^\alpha)$.

- sound
- termination on finite domain
- learns weak lemmas excluding the whole assignment

GearSat

η config spec.
θ stability guard
ψ system spec.

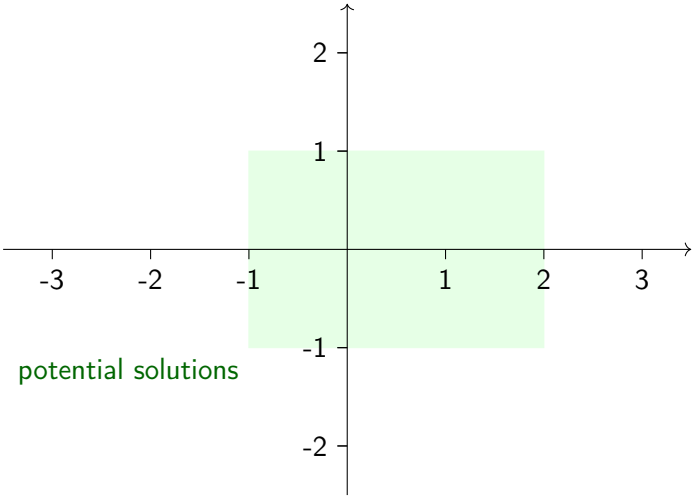
decision problem $\quad \exists p.\, \eta(p) \wedge \forall q\, \forall x.\, (\theta(p,q) \rightarrow \psi(q,x))$

Initial list of lemmas $E(p) \leftarrow \eta(p)$.
Until S returns `unsat`:

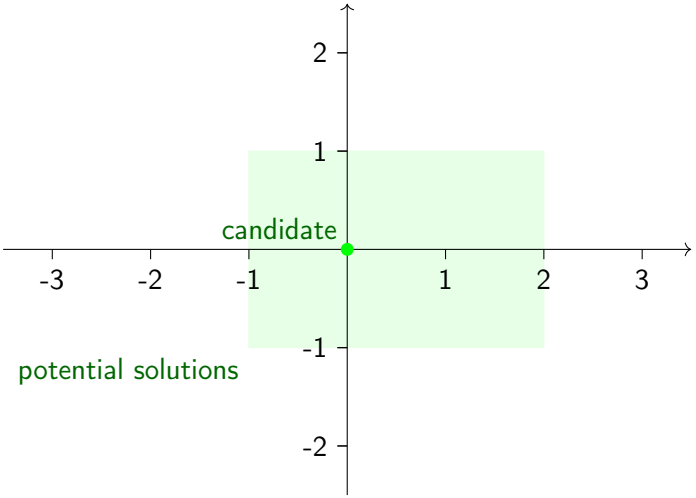
1. Use S to find assignment α of p, x satisfying
$$\psi(p,x) \wedge E(p)$$
or return `unsat`.
2. Use S to find assignment β of q, x satisfying
$$\neg(\theta([\![p]\!]^\alpha, q) \rightarrow \psi(q,x))$$
or return `sat` with solution $[\![p]\!]^\alpha$.
3. Add lemma $E(p) \leftarrow E(p) \wedge \neg\theta(p, [\![q]\!]^\beta)$.

- uses special structure of formula
- learn stronger lemmas that exclude regions based on the stability guard
  - counter-examples to stability
  - represent large regions
- guard connects $\exists$ and $\forall$ quantified variables

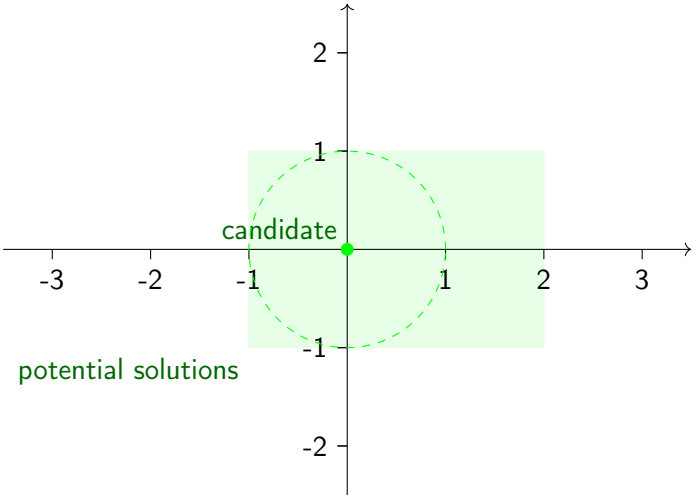
Example



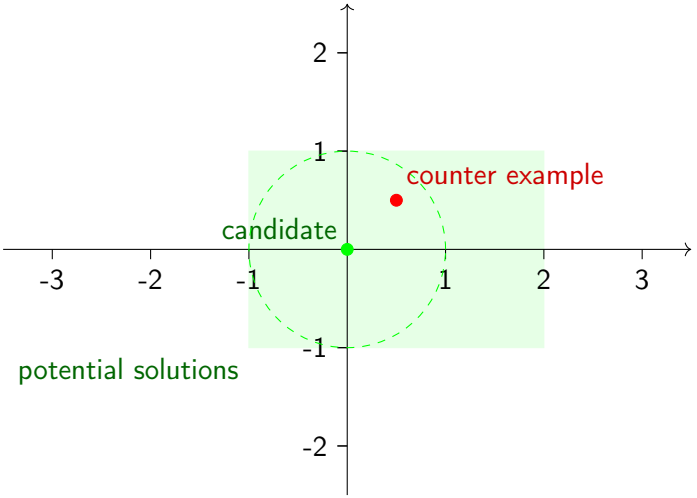
Example



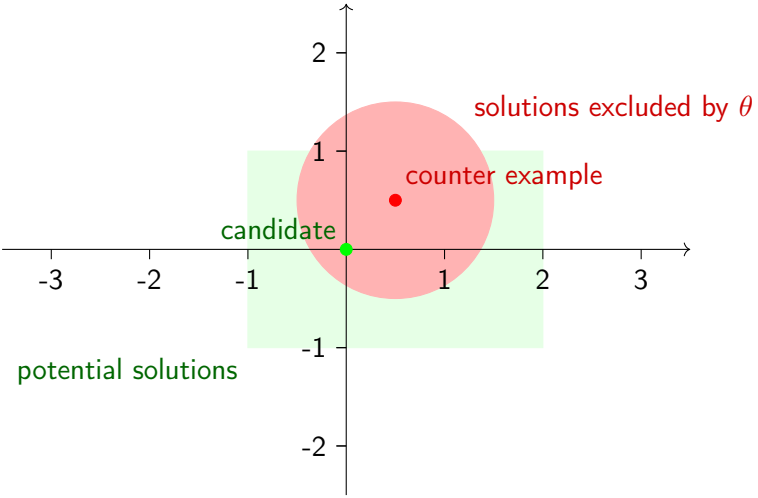
Example



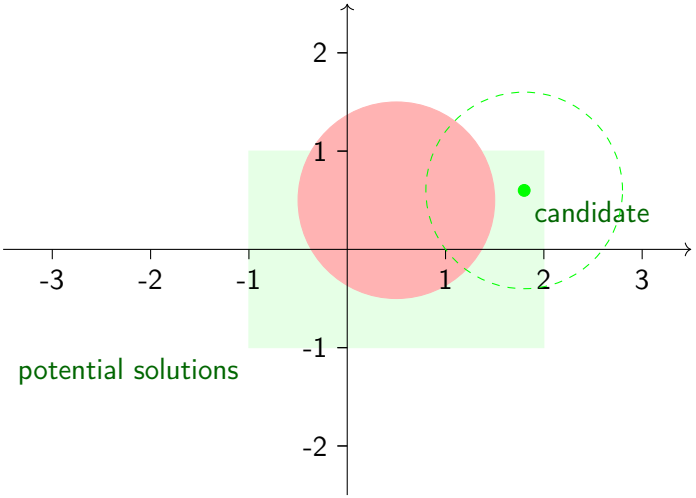
Example


2
1
-1
-2
-3
-2
-1
1
2
3
counter example
candidate
potential solutions

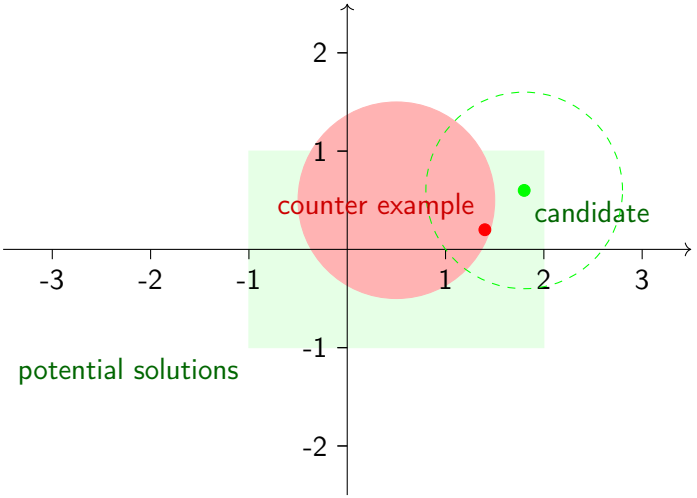
Example


2
1
-1
-2
-3
-2
-1
1
2
3
solutions excluded by θ
counter example
candidate
potential solutions

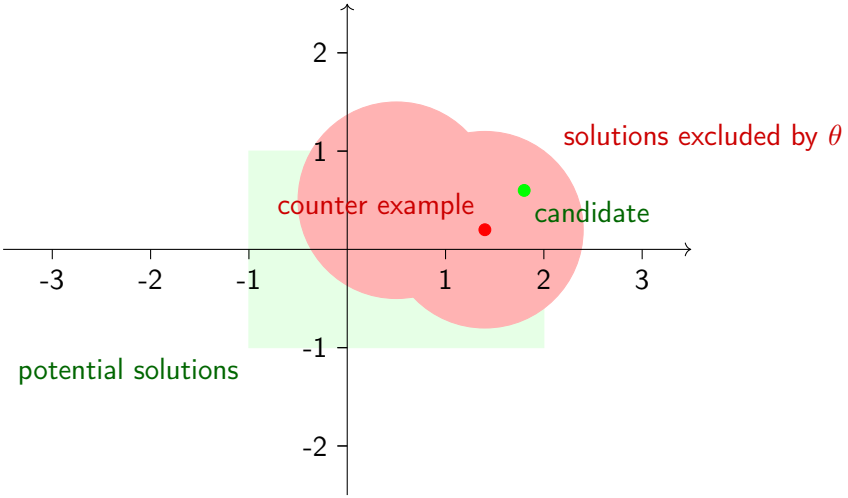
Example



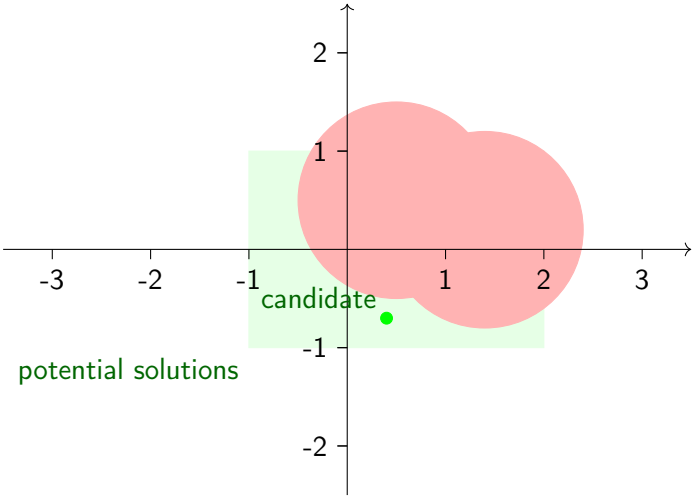
Example


2
1
-1
-2
-3
-2
-1
1
2
3
counter example
candidate
potential solutions

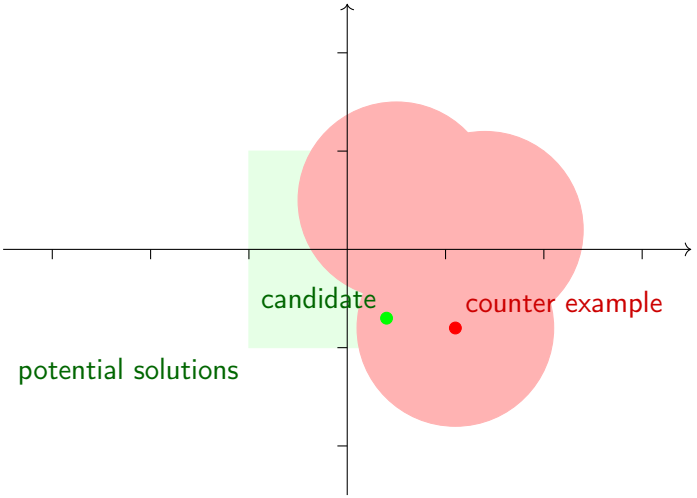
Example


2
1
-1
-2
-3
-2
-1
1
2
3
solutions excluded by θ
counter example
candidate
potential solutions

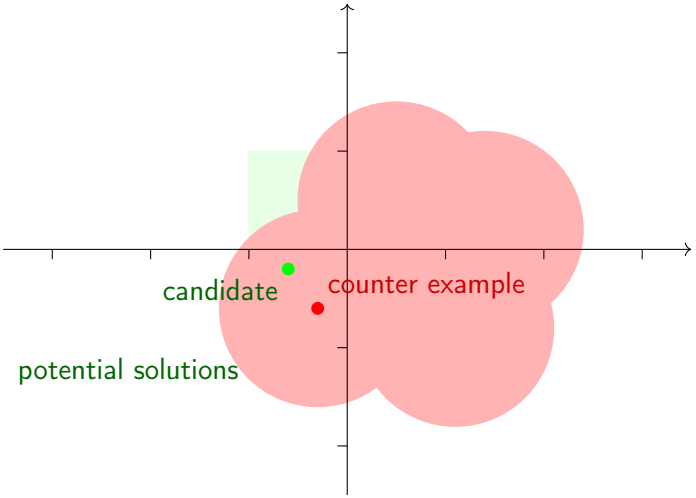
Example



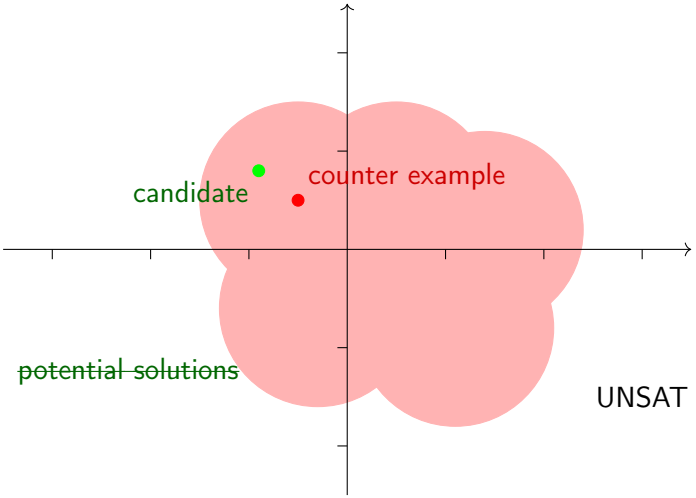
Example



Example



Example



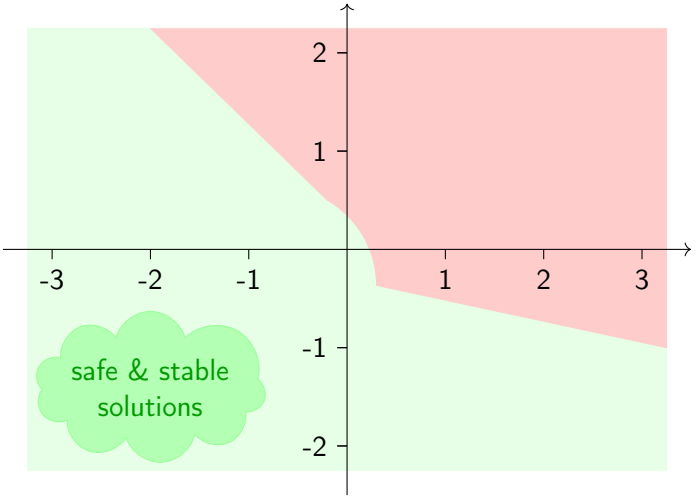
Soundness

Theorem

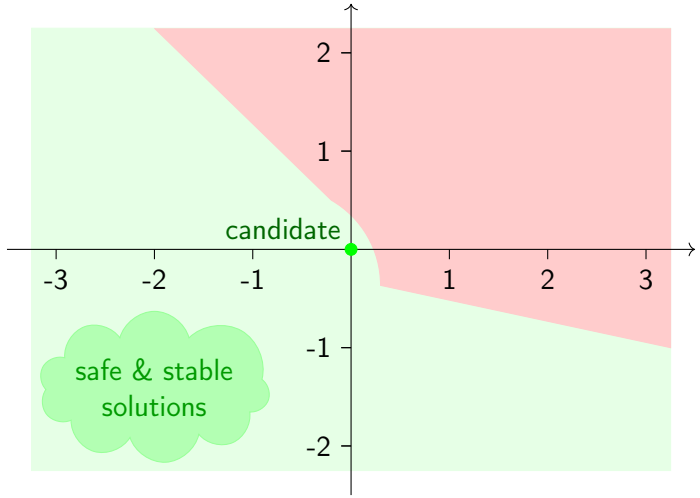
GearSat is sound for the GEAR fragment.

- stronger lemmas from stability guard are not *too* strong
- proof uses reflexivity of θ, and
- soundness of decision procedure S for quantifier-free $\mathcal{T}_{\|\cdot\|}$

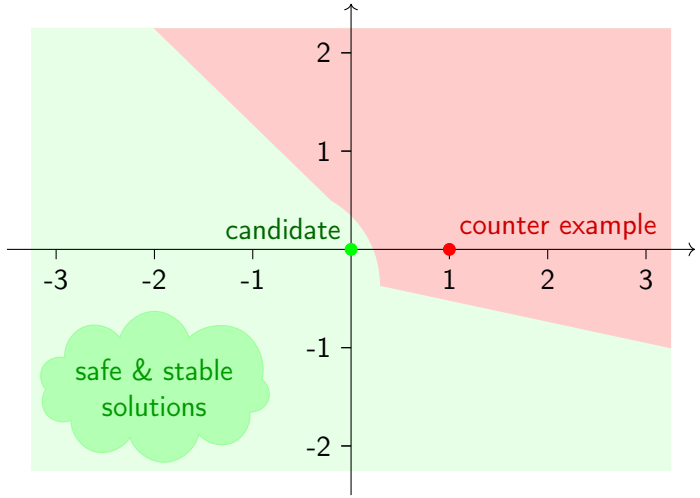
Candidates can be arbitrarily close



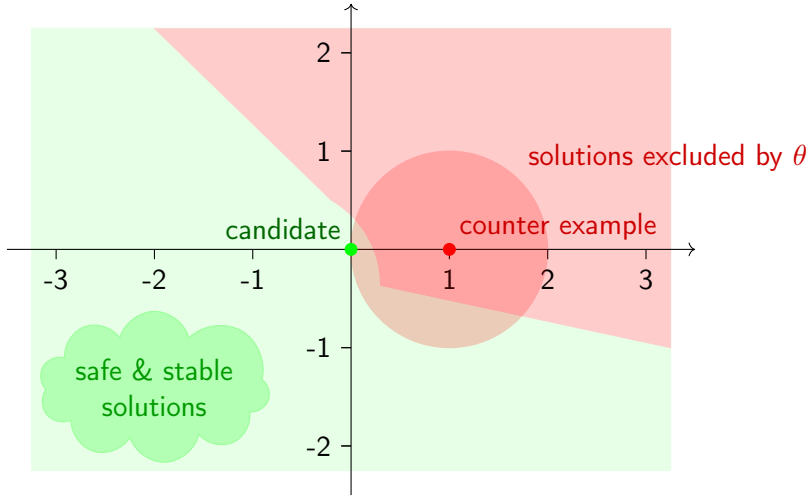
Candidates can be arbitrarily close



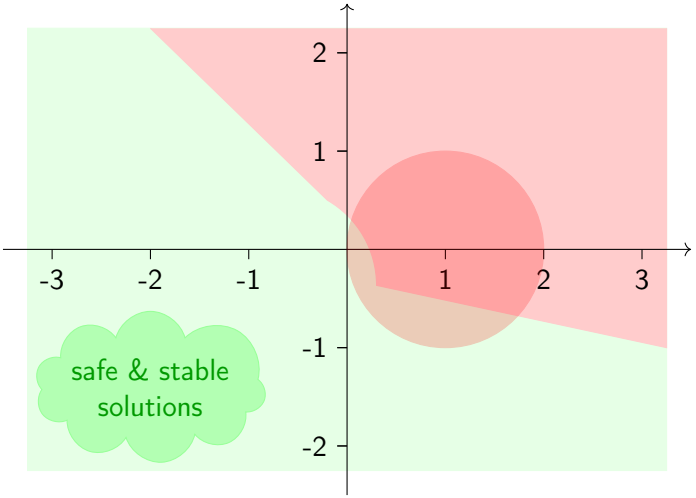
Candidates can be arbitrarily close



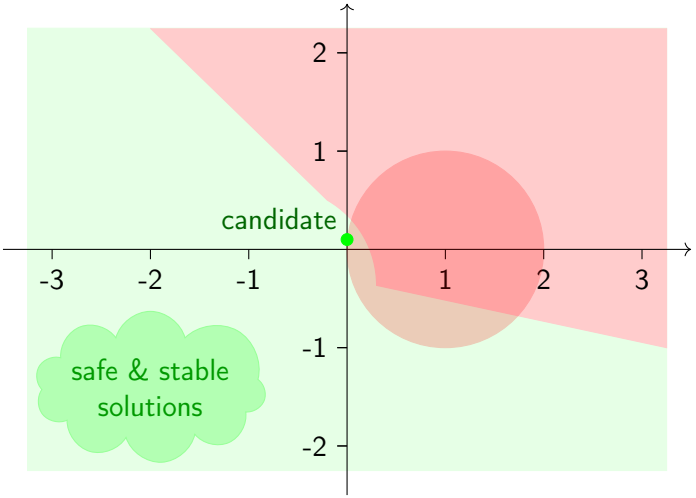
Candidates can be arbitrarily close



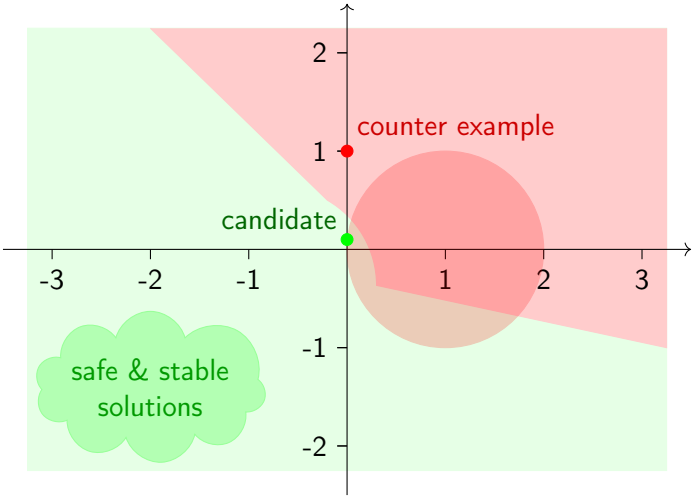
Candidates can be arbitrarily close



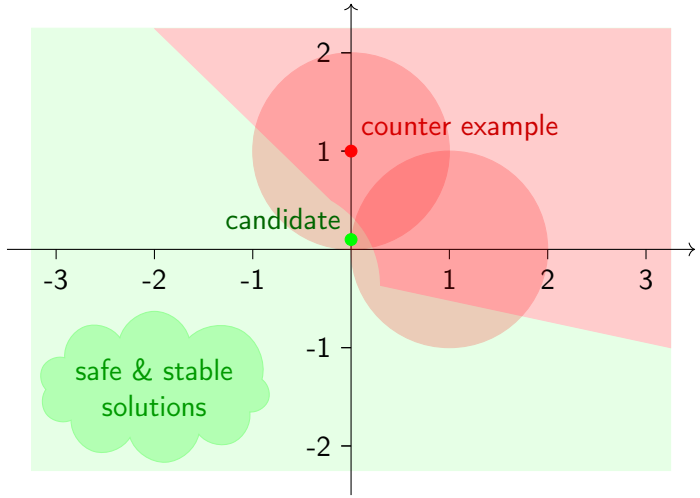
Candidates can be arbitrarily close



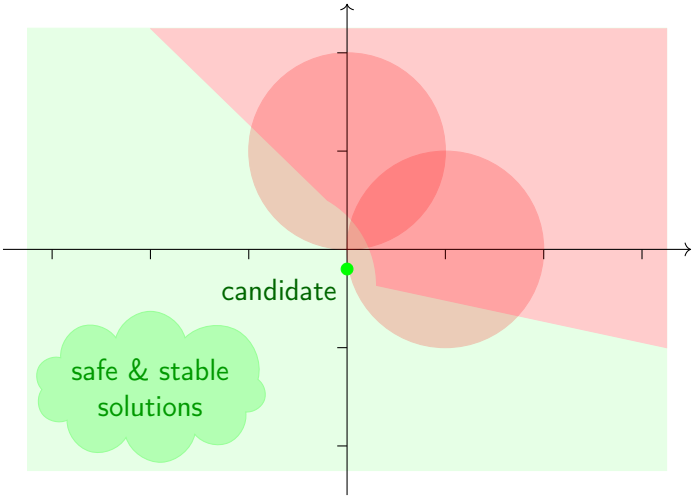
Candidates can be arbitrarily close



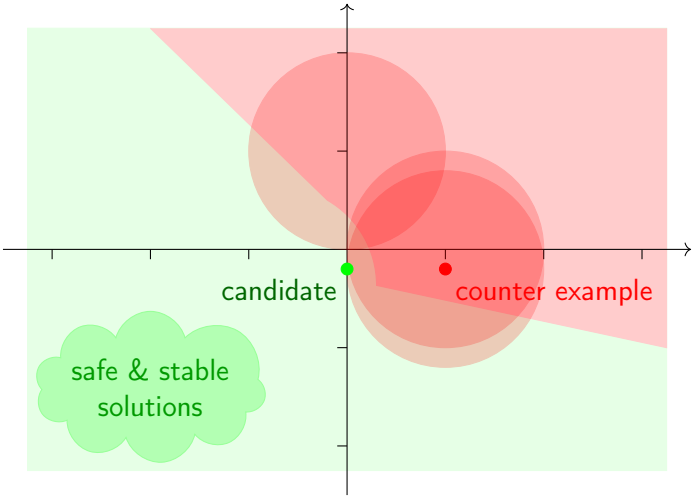
Candidates can be arbitrarily close



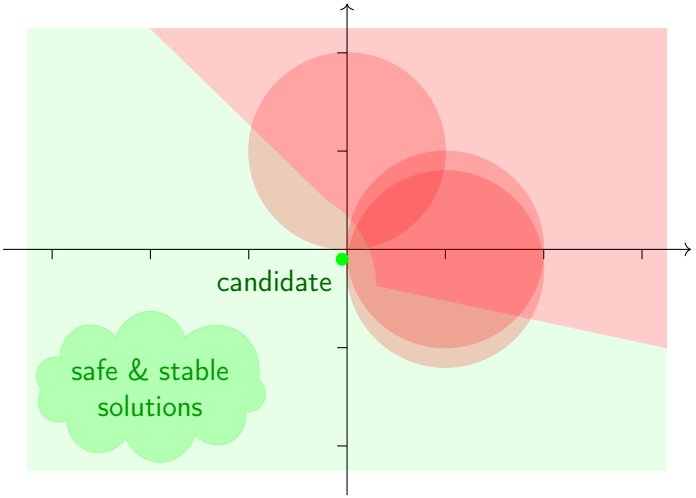
Candidates can be arbitrarily close



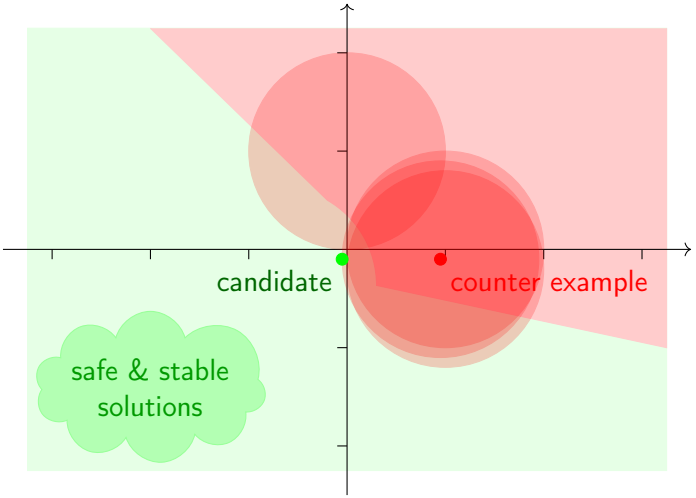
Candidates and counter examples can be arbitrarily close



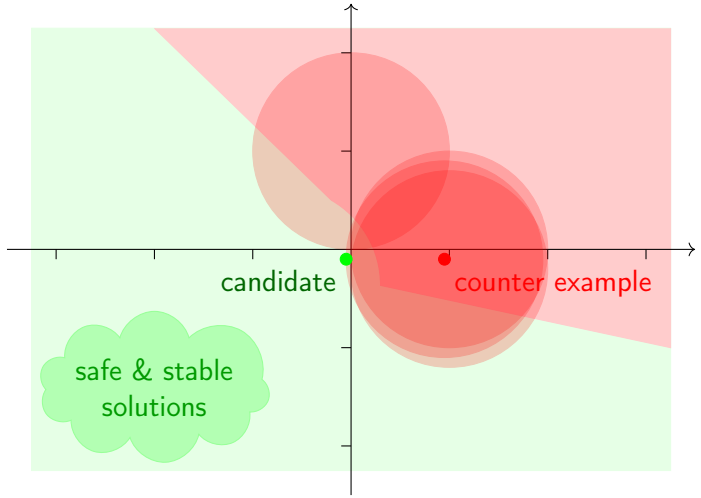
Candidates and counter examples can be arbitrarily close



Candidates and counter examples can be arbitrarily close



Candidates and counter examples can be arbitrarily close



SAT, but may not terminate

Normed GEAR formulas

Definition

Let $r \geq 0$ and φ_r be the GEAR formula

$$\exists p.\, \eta(p) \wedge \forall q\, \forall x.\, (\|p - q\| \leq r \rightarrow \psi(q, x))$$

where η defines a bounded set. Then φ_r is called a normed GEAR formula.
The δ-perturbation $\varphi_{r+\delta}$ of φ_r is

$$\exists p.\, \eta(p) \wedge \forall q\, \forall x.\, (\|p - q\| \leq r + \delta \rightarrow \psi(q, x))$$

for $\delta > 0$.

δ-unsatisifiability

Motiviation: twofold:

1. termination for instances over transcendental functions, e.g., δ-`ksmt`, and
2. reject solutions if counter-examples are within some tolerance of the safe region.

Definition (δ-decision problem for normed GEAR formulas)

Let $\delta > 0$. Show $\begin{cases} \varphi_r & \text{is satisfiable, or} \\ \varphi_{r+\delta} & \text{is unsatisfiable.} \end{cases}$

Related to δ-SMT problem [Gao, Avigad, Clarke]: Show $\begin{cases} \xi_{r+\delta} & \text{is satisfiable, or} \\ \xi_r & \text{is unsatisfiable.} \end{cases}$

GearSat

η config spec.
θ stability guard
ψ system spec.

decision problem $\quad \exists p.\, \eta(p) \wedge \forall q \forall x.\, \|p - q\| \leq r \rightarrow \psi(q, x)$

Initial list of lemmas $E(p) \leftarrow \eta(p)$.
Until S returns `unsat`:

1. Use S to find assignment α of p, x satisfying
$$\psi(p, x) \wedge E(p)$$
or return `unsat`.
2. Use S to find assignment β of q, x satisfying
$$\neg(\|[\![p]\!]^{\alpha} - q\| \leq r \rightarrow \psi(q, x))$$
or return `sat` with solution $[\![p]\!]^{\alpha}$.
3. Add lemma $E(p) \leftarrow E(p) \wedge \|p - [\![q]\!]^{\beta}\| > r$.

GearSat$_\delta$

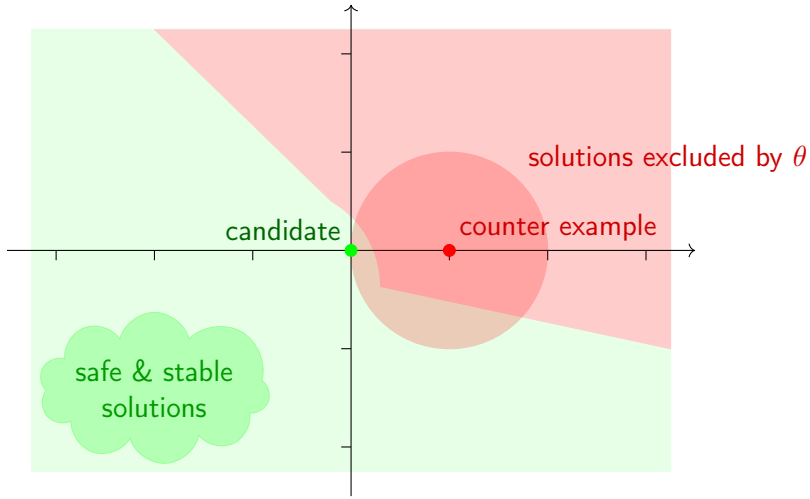
η config spec.
θ stability guard
ψ system spec.

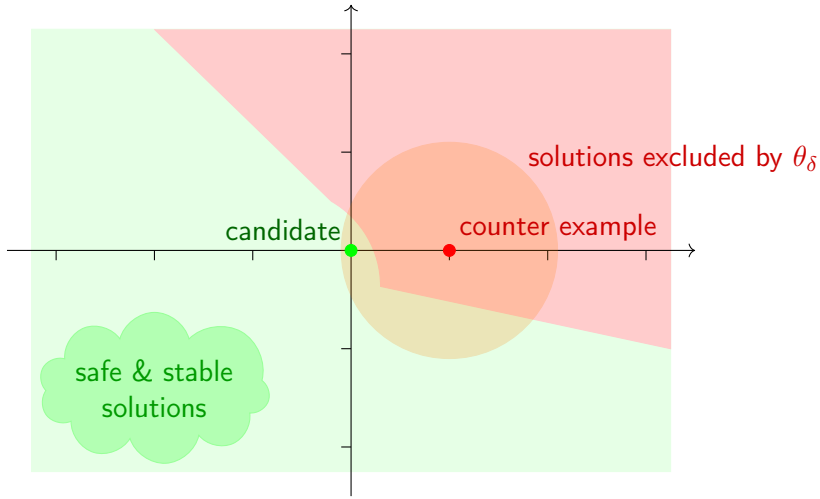
δ-decision problem $\quad \exists p.\, \eta(p) \wedge \forall q \forall x.\, \|p - q\| \leq r \rightarrow \psi(q, x)$

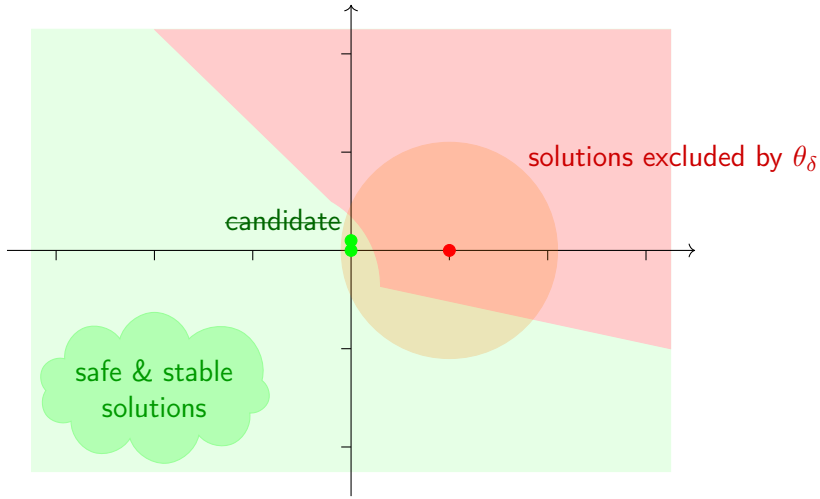
Initial list of lemmas $E(p) \leftarrow \eta(p)$.
Until S returns `unsat`:

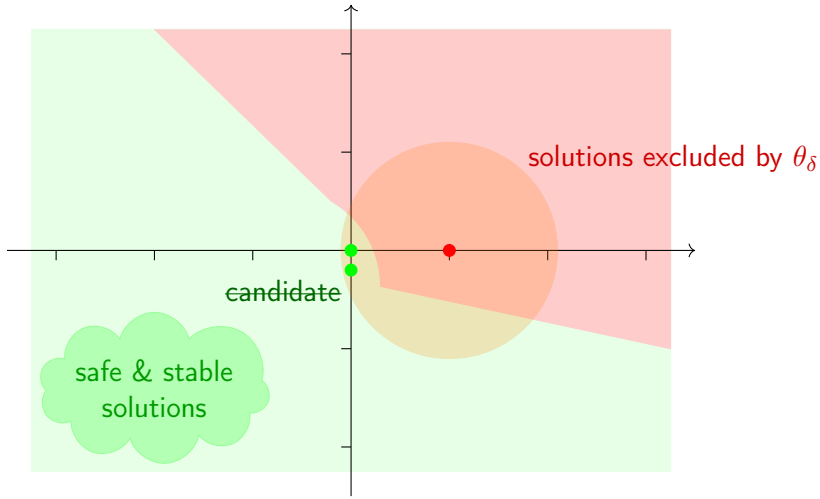
1. Use S to find assignment α of p, x satisfying
$$\psi(p, x) \wedge E(p)$$
or return `unsat`.
2. Use S to find assignment β of q, x satisfying
$$\neg(\|[\![p]\!]^\alpha - q\| \leq r \rightarrow \psi(q, x))$$
or return `sat` with solution $[\![p]\!]^\alpha$.
3. Add lemma $E(p) \leftarrow E(p) \wedge \|p - [\![q]\!]^\beta\| > r + \delta$.

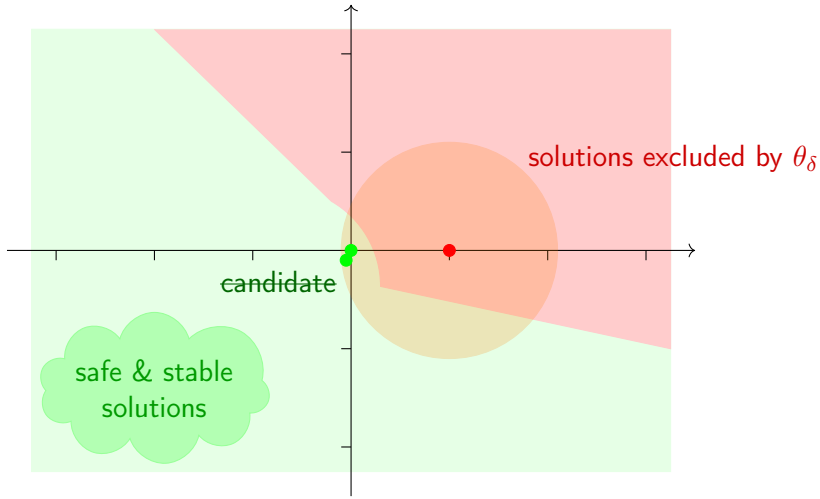
- reject solutions if counter-examples to stability are within tolerance $\delta > 0$.
- $\begin{cases} \varphi_r & \text{is satisfiable, or} \\ \varphi_{r+\delta} & \text{is unsatisfiable} \end{cases}$

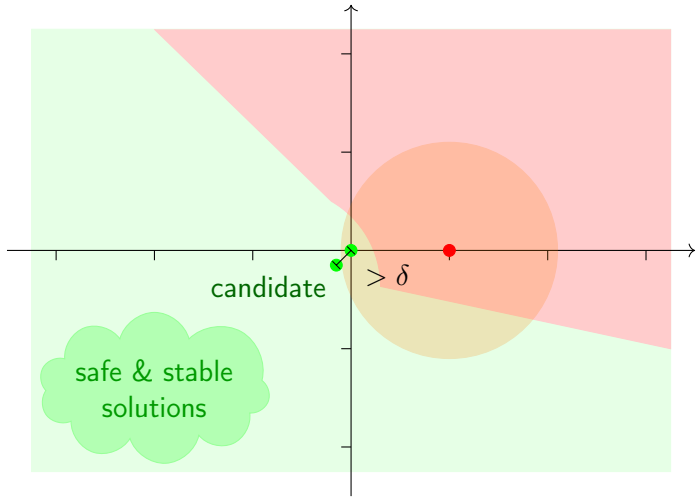

solutions excluded by θ
candidate
counter example
safe & stable
solutions

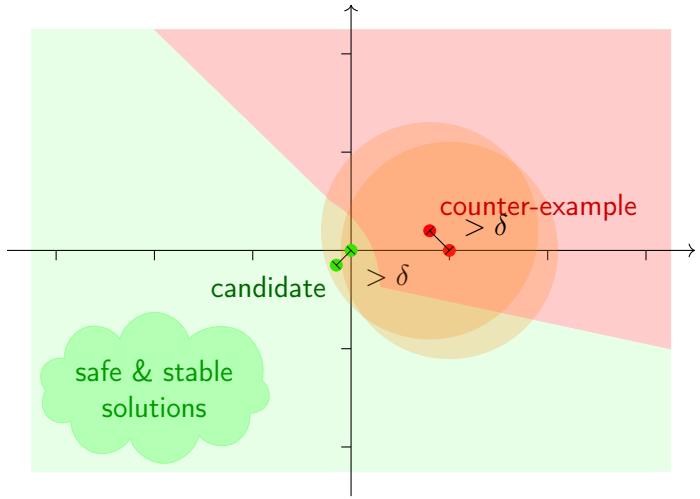

solutions excluded by θ_δ
candidate
counter example
safe & stable
solutions

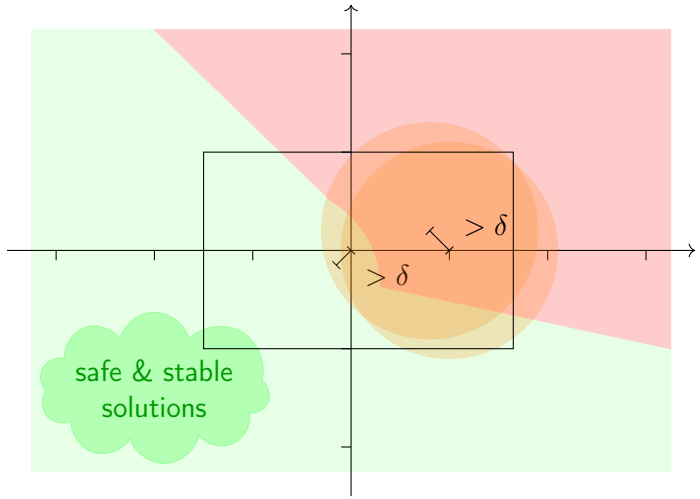

solutions excluded by θ_δ
candidate
safe & stable
solutions


solutions excluded by θ_δ
~~candidate~~
safe & stable
solutions


solutions excluded by θ_δ
candidate
safe & stable
solutions


candidate
$> \delta$
safe & stable
solutions


counter-example
$> \delta$
candidate
$> \delta$
safe & stable
solutions



terminates on
bounded instances

Completeness

Theorem

Let $\delta > 0$ be rational. GearSat$_\delta$ terminates for normed GEAR formulas.

Proof by a finite cover argument due to properties of $\|\cdot\|$.

Theorem

GearSat$_\delta$ is a δ-decision procedure for normed GEAR formulas.

Scalability: Combination of GearSat with Bayesian optimization

Scalability to large problems is a major issue.

GearSat with SMT/constraint solvers:

- + exact solutions with formal guarantees
- - scalability to large problems can be an issue

Scalability: Combination of GearSat with Bayesian optimization

Scalability to large problems is a major issue.

GearSat with SMT/constraint solvers:

- + exact solutions with formal guarantees
- - scalability to large problems can be an issue

Bayesian optimization

- + efficient method for optimisation
- - no guarantees that solutions satisfy spec or optimality;
- - point solutions in our approach we can find large regions of near optimal stable solutions

GearSat + Baesyan Optimization

GearSat + Baesyan Optimization general idea:

- Using BO for finding candidate solutions and counter-examples and use SMT to verify the solutions.

Finding counter-examples

Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k \in [[\boldsymbol{a}_i, \boldsymbol{b}_i]]$ ▷ BO
if $f(\boldsymbol{e}_j) < T$ **for some** $j \in \{1, \ldots, k\}$ **then**
  $\boldsymbol{d}_i \leftarrow \boldsymbol{e}_j$
else
  $B_i \leftarrow B^{\min}$.Init$(\boldsymbol{a}_i, \boldsymbol{b}_i, (\boldsymbol{e}_j, f(\boldsymbol{e}_j))_j)$
  for $j = 1, \ldots, MaxIter$ **do**
    $\boldsymbol{d}_i \leftarrow B_i$.Suggest
    $B_i \leftarrow B_i$.Observe$(\boldsymbol{d}_i, f(\boldsymbol{d}_i))$
    if $f(\boldsymbol{d}_i) < T$ **then return fi**
  end
  $\boldsymbol{d}_i \leftarrow$ sol. of $D_i(\boldsymbol{x}', y')$ restricted to $\boldsymbol{x}'$ or unsat ▷ SMT
fi

Finding candidates using BO

if $i > 1$ **then** ▷ record previous counter-example
  $A_i \leftarrow A_{i-1}$.Observe$(\boldsymbol{c}_{i-1}, f(\boldsymbol{d}_{i-1}))$
fi
for $j = 1, \ldots, MaxIter$ **do** ▷ BO
  $\boldsymbol{c}_i \leftarrow A_i$.Suggest
  if $\theta(\boldsymbol{c}_i, \boldsymbol{d}_m)$ **for any** $m \in \{1, \ldots, i-1\}$ **then**
    $z \leftarrow f(\boldsymbol{d}_m)$ ▷ $\boldsymbol{c}_i$ is excluded by lemmas
  else
    $z \leftarrow f(\boldsymbol{c}_i)$ ▷ $\boldsymbol{c}_i$ is an eligible candidate
  fi
  if $z \geq T$ **then return fi** ▷ $f(\boldsymbol{c}_i) \geq T$
  $A_i \leftarrow A_i$.Observe$(\boldsymbol{c}_i, z)$ ▷ $z < T$
end
$\boldsymbol{c}_i \leftarrow$ solution of $C_i(\boldsymbol{x}, y)$ restricted to $\boldsymbol{x}$ or unsat ▷ SMT

[Combining Constraint Solving and Bayesian Techniques for System Optimization.
F. Brausse, Z. Khasidashvili, K. Korovin, IJCAI 2022]

GearOpt$_\delta$-BO

Optimization algorithm GearOpt$_\delta$-BO which combines:

- SMT constraint solving for formal guarantees on
  - safety for all inputs wrt spec constraints
  - optimality of objective function
  - stability – all points in a specified parameter region are near-optimal
- Bayesian optimization for efficient search of both candidates and counter-examples.

$\rightsquigarrow$ Stable regions with guarantees on distance to the optimum.

GearOpt$_\delta$-BO

Optimization algorithm GearOpt$_\delta$-BO which combines:

- SMT constraint solving for formal guarantees on
  - safety for all inputs wrt spec constraints
  - optimality of objective function
  - stability – all points in a specified parameter region are near-optimal
- Bayesian optimization for efficient search of both candidates and counter-examples.

$\rightsquigarrow$ Stable regions with guarantees on distance to the optimum.

Theorem (IJCAI'22)

For any accuracy $\varepsilon > 0$, any stability guard θ and $\delta > 0$, GearOpt$_\delta$-BO is a sound, δ-complete and terminating procedure for the problem of finding safe, optimal and θ-stable ε-solutions.

Implementation – SMLP.

SMLP (CAV'24)

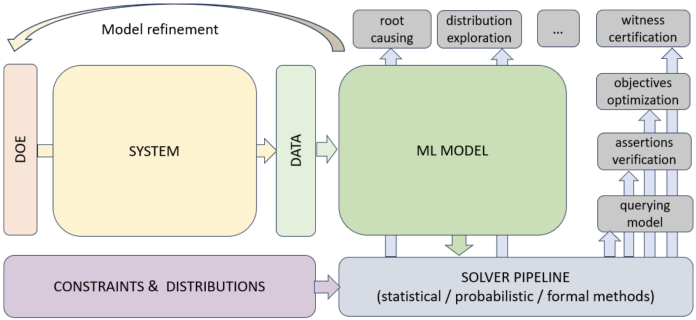
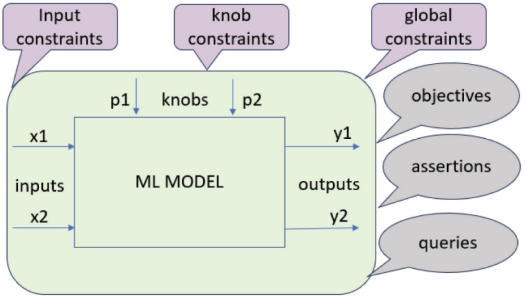


Fig. 2. SMLP Tool Architecture

Applied at Intel for: circuit layout optimization and signal integrity

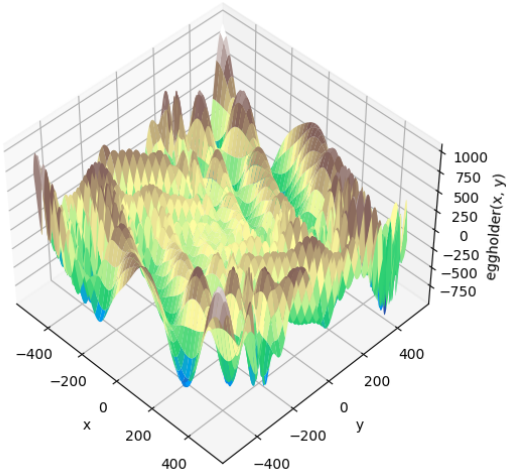
Problem specification file

```
{
  "version": "1.2",
  "variables": [
    {"label":"y1", "interface":"output", "type":"real"},
    {"label":"y2", "interface":"output", "type":"real"},
    {"label":"x1", "interface":"input", "type":"real", "range":[0,10]},
    {"label":"x2", "interface":"input", "type":"int", "range":[-1,1]},
    {"label":"p1", "interface":"knob", "type":"real", "range":[0,10], "rad-rel":0.1, "grid":[2,4,7]},
    {"label":"p2", "interface":"knob", "type":"int", "range":[3,7], "rad-abs":0.2}
  ],
  "alpha": "p2<5 and x1==10 and x2<12",
  "beta": "y1>=4 and y2==8",
  "eta": "p1==4 or (p1==8 and p2 > 3)",
  "assertions": {
    "assert1": "(y2**3+p2)/2>6",
    "assert2": "y1>=0",
    "assert3": "y2>0"
  },
  "objectives": {
    "objective1": "(y1+y2)/2",
    "objective2": "y1"
  }
}
```



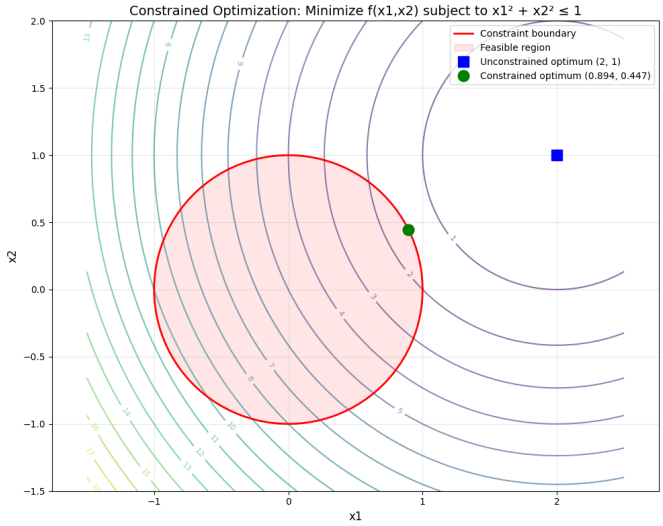
SMLP tutorial (Dmitry Messerman)

1. Black-box function optimization:



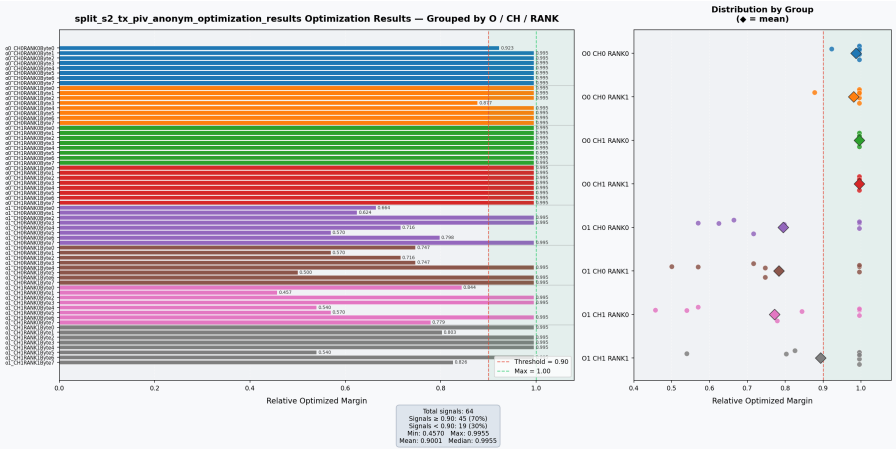
SMLP tutorial (Dmitry Messerman)

2. Constrained DORA (Distance to Optimal with Radial Adjustment)



SMLP tutorial (Dmitry Messerman)

3. Intel Signal Integrity domain



SMLP implementation

Implementation of GearSat algorithms in SMLP system: Python, C, C++

- combination of Z3/δ-ksmt SMT for quantifier-free NRA reasoning
-
  - speed-up by combining Bayesian Optimization solvers (producing point solutions) with SMT based constraint solving for finding stable regions
  - neural networks, polynomial models, tree models, random forest
  - different exploration modes
  - Pareto

GitHub: `https://github.com/SMLP-Systems/smlp`

Experimental Results on Intel problems

RX/TX system with 2 channels each with 8 bytes approximated by NNs:

- input dimension 7-8, integer and real, 1 real output
- 2 internal layers of widths 14-16, 7-8, ReLU activation
- used Tensorflow for training on normalized data up to MSE 0.005
- $\sim$100k training samples of signal integrity measurements of the system

Experimental results: obtain up to 100 different safe and stable configurations.

	RX				TX			
C:B	th	safe	ce	time	th	safe	ce	time
0:0	0.9	100	54	319 s	0.9	100	156	131 s
0:1	0.85	100	69	700 s	0.9	51	1006	372 s
0:2	0.9	29	2867	3034 s	0.9	100	69	114 s
0:3	0.85	100	251	512 s	0.9	100	120	78 s
0:4	0.9	100	128	830 s	0.9	100	315	211 s
0:5	0.85	100	82	627 s	0.9	100	221	135 s
0:6	0.85	100	121	680 s	0.9	20	110	41 s
0:7	0.85	41	2620	3409 s	0.9	100	176	129 s
1:0	0.8	100	762	606 s	0.9	100	84	89 s
1:1	0.8	1	188	264 s	0.85	100	467	226 s
1:2	0.9	100	369	700 s	0.9	100	360	128 s
1:3	0.8	100	3449	1328 s	0.9	100	169	82 s
1:4	0.85	100	16	381 s	0.9	100	304	79 s
1:5	0.85	35	287	769 s	0.85	100	357	205 s
1:6	0.8	100	1088	980 s	0.9	100	259	68 s
1:7	0.9	100	84	405 s	0.9	100	60	60 s

Between 232 and 7 098 Z3 calls.

Conclusions

- Combination of ML and Formal Methods
- SMLP: analysis and optimization of systems modelled using ML
- δ-decision procedure GearSat$_\delta$ for the normed GEAR fragment producing near optimal safety & stability regions in machine learning models.
  - Combination of GearSat with Bayesian optimization: GearSat$_\delta$-BO
  - SMT support for neural networks encodings

Future:

- Optimize encoding of NNs + use more specialized solvers
- Increase scalability to larger models
- Other applications for the configuration problem: detection of adversarial attacks on ML, Kubernetes load optimization, cyber-physical systems...