Emerging Algorithms for Verifying Deep Neural Networks

Changliu Liu[1], Tomer Arnon[2], Chris Lazarus[2], Clark Barrett[2], and Mykel Kochenderfer[2]

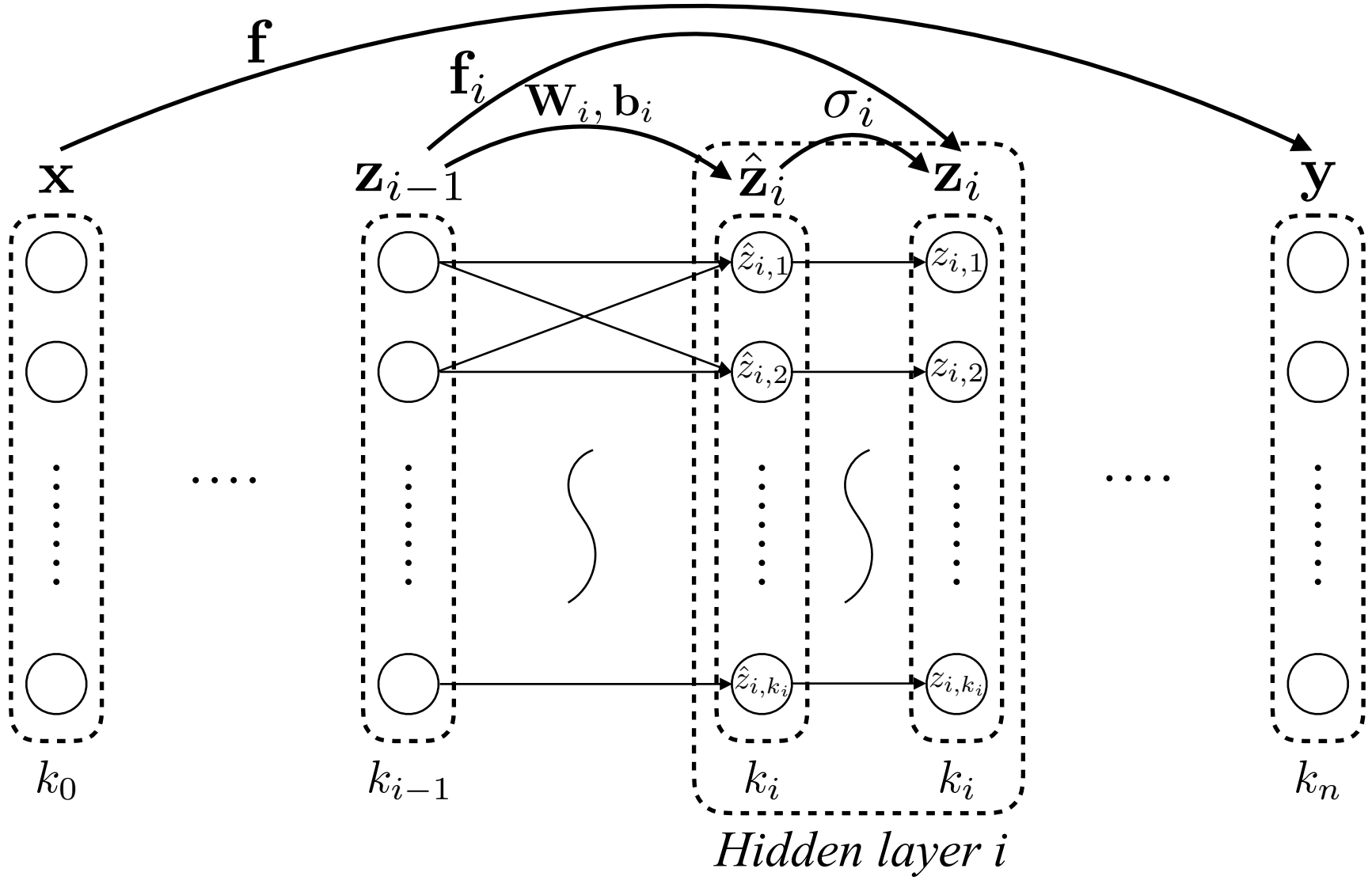
[1] Carnegie Mellon University
[2] Stanford University

Overview

- Deep neural networks are widely used for **nonlinear function approximation** with applications spanning from computer vision to control.
- Although these networks involve the composition of simple arithmetic operations, it can be very challenging to verify whether a particular network satisfies certain **input-output properties**.
- We survey methods that have emerged recently for soundly verifying such properties. In this talk, we will
    1. Discuss fundamental differences and connections between existing algorithms;
    2. Introduce NeuralVerification.jl toolbox which contains pedagogical implementations of existing methods.

* C. Liu, T. Arnon, C. Lazarus, C. Barrett, and M. Kochenderfer, "Algorithms for Verifying Deep Neural Networks," arXiv:1903.06758.

Deep Feed Forward Neural Networks



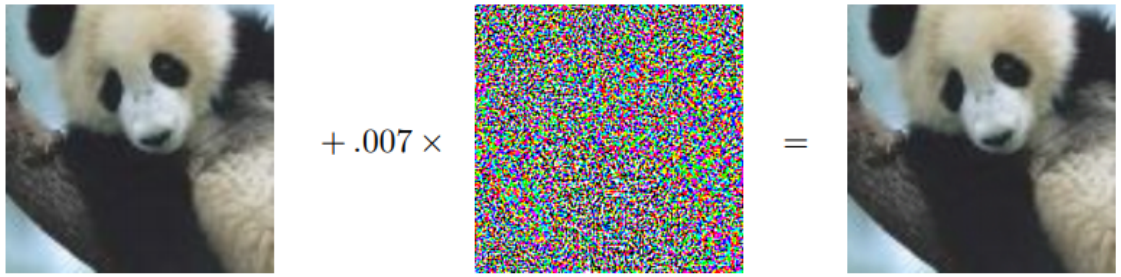
What to Verify?

Input-Output Relationship

$$\forall \mathbf{x} \in \mathcal{X}, \mathbf{y} = \mathbf{f}(\mathbf{x}) \in \mathcal{Y}.$$

```
struct Problem{P, Q}
    network::Network
    input::P
    output::Q
end
```

Robustness Problem

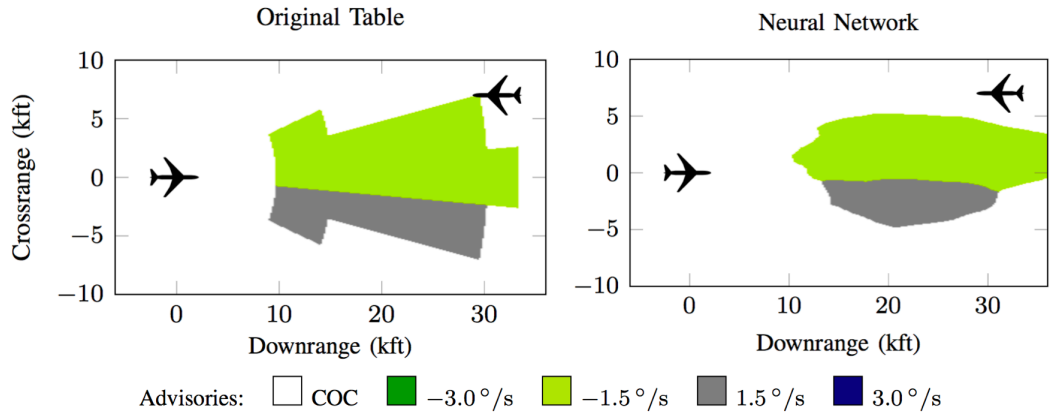


"panda"
57.7% confidence

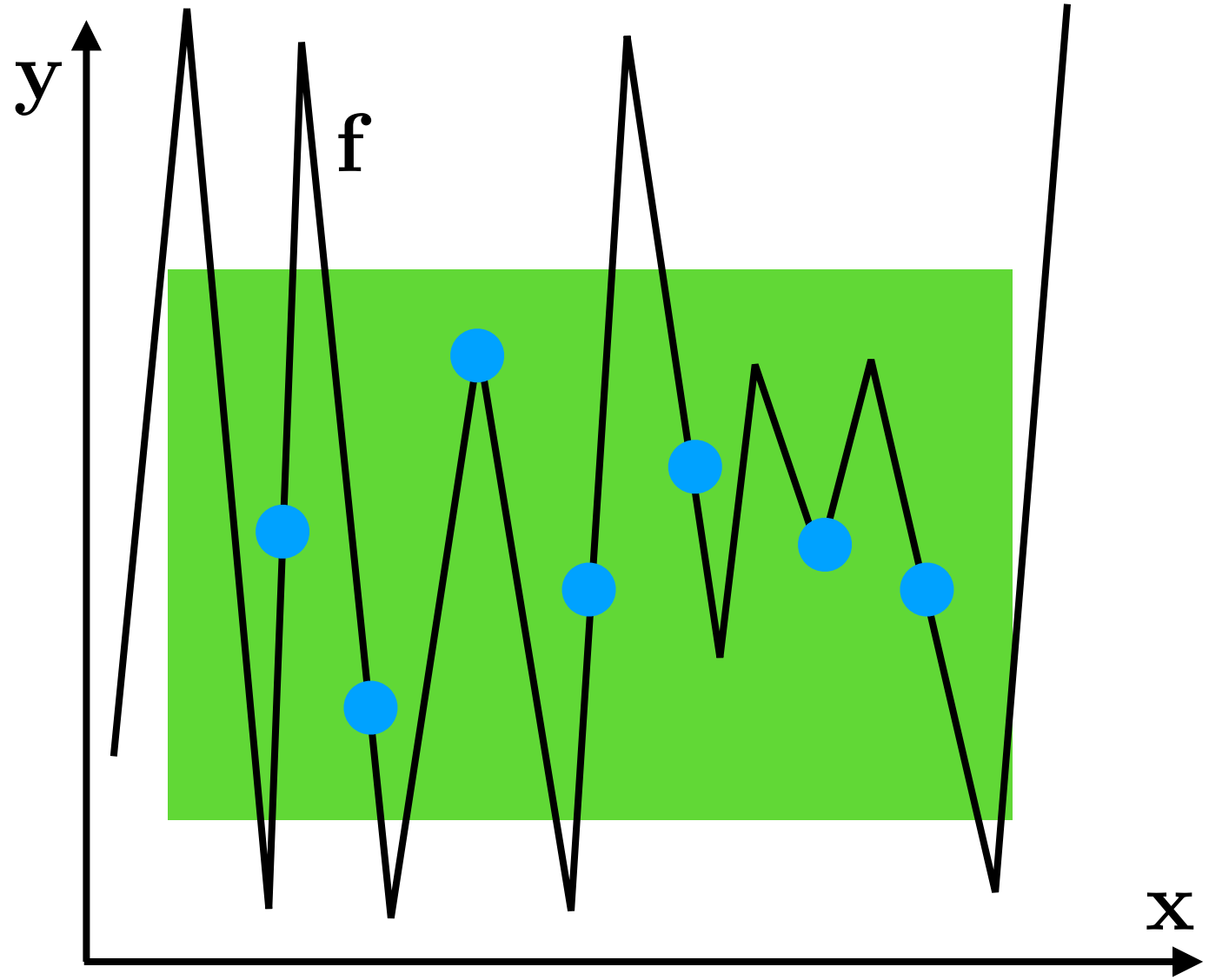
"nematode"
8.2% confidence

"gibbon"
99.3 % confidence

Safety Problem



Point-wise Verification can Fail



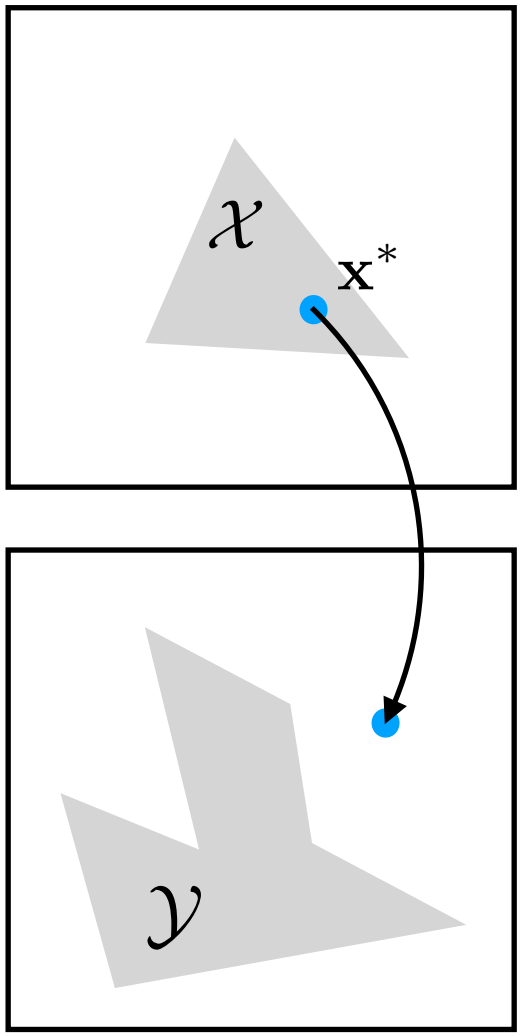
Result and Failure Modes

```
abstract type Result end

struct BasicResult <: Result
    status::Symbol
end

struct CounterExampleResult <: Result
    status::Symbol
    counter_example::Vector{Float64}
end

struct AdversarialResult <: Result
    status::Symbol
    max_disturbance::Float64
end

struct ReachabilityResult <: Result
    status::Symbol
    reachable::Vector{<:AbstractPolytope}
end
```

Result and Failure Modes

Counter example

$$\mathbf{x}^* \in \mathcal{X}, \mathbf{f}(\mathbf{x}^*) \notin \mathcal{Y},$$



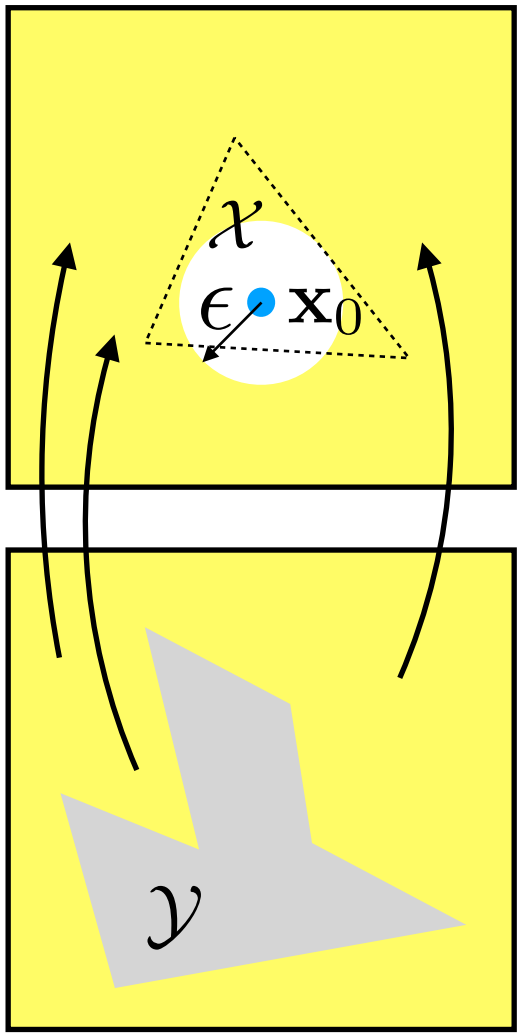
Result and Failure Modes

Counter example

$$\mathbf{x}^* \in \mathcal{X}, \mathbf{f}(\mathbf{x}^*) \notin \mathcal{Y},$$

Adversarial input

$$\epsilon := \min_{\mathbf{f}(\mathbf{x}) \notin \mathcal{Y}} \|\mathbf{x} - \mathbf{x}_0\|_p < \text{radius}(\mathcal{X}),$$



Result and Failure Modes

Counter example

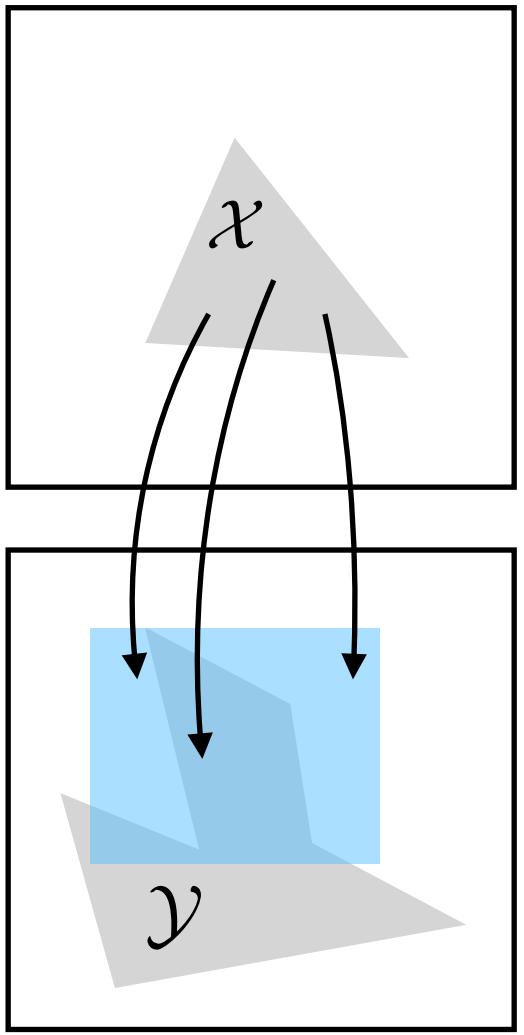
$$\mathbf{x}^* \in \mathcal{X}, \mathbf{f}(\mathbf{x}^*) \notin \mathcal{Y},$$

Adversarial input

$$\epsilon := \min_{\mathbf{f}(\mathbf{x}) \notin \mathcal{Y}} \|\mathbf{x} - \mathbf{x}_0\|_p < \text{radius}(\mathcal{X}),$$

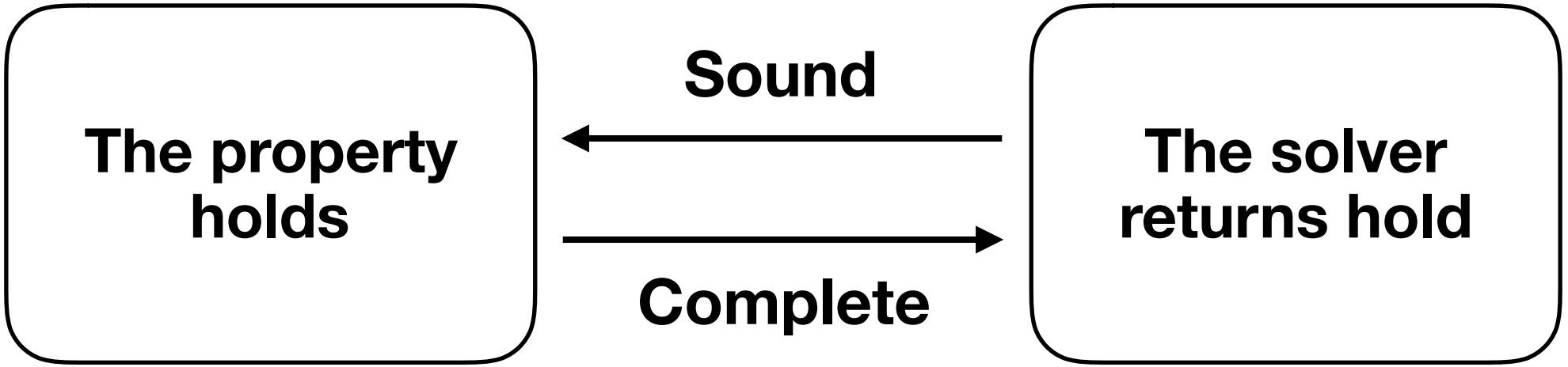
Output Reachable set

$$\mathcal{R}(\mathcal{X}, \mathbf{f}) := \{\mathbf{y} : \mathbf{y} = \mathbf{f}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}\} \not\subseteq \mathcal{Y}.$$



Recent Approaches

- **Handles piece-wise linear activation function**
- **A majority of the methods only consider ReLU activation**
- **Soundness, Completeness, Terminating**



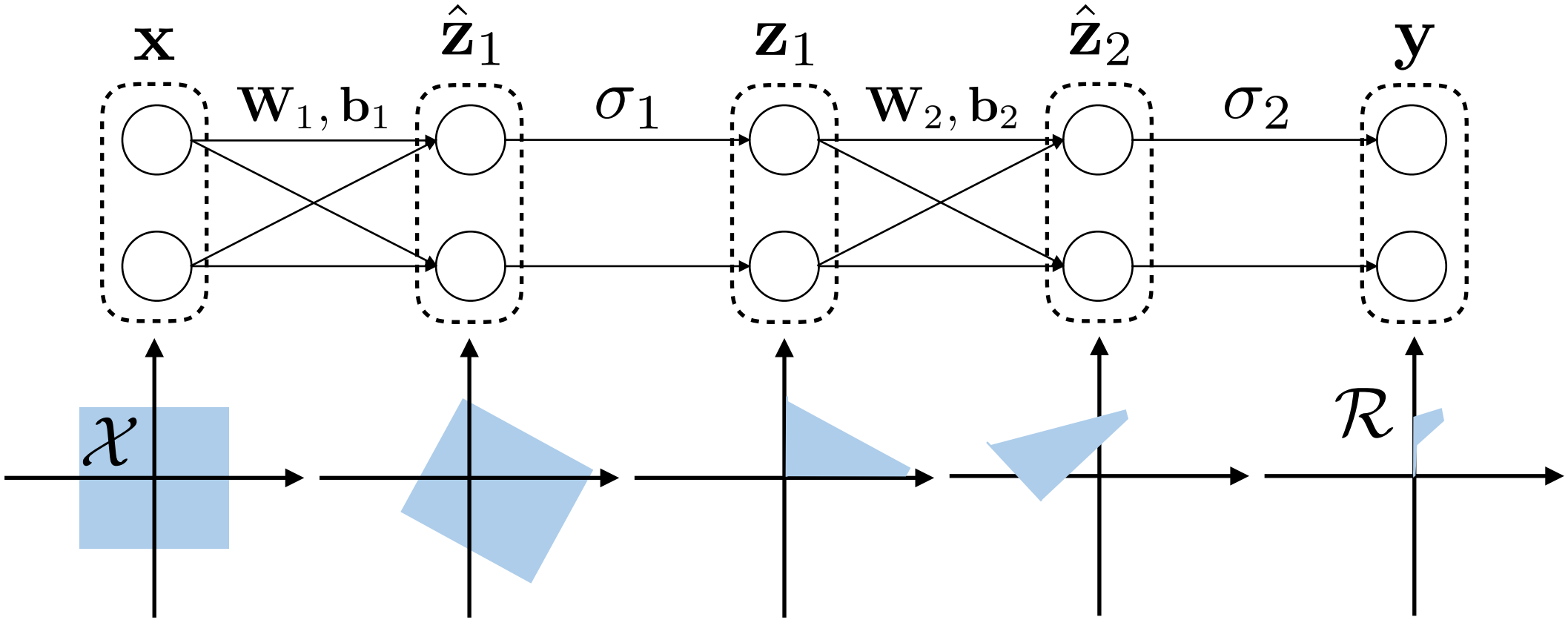
Three basic methods

Reachability

Optimization

Search

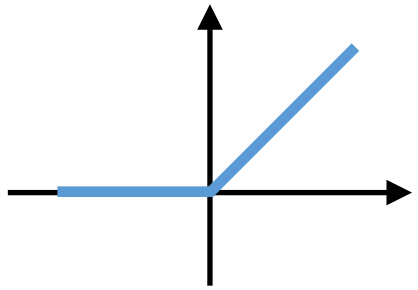
Reachability



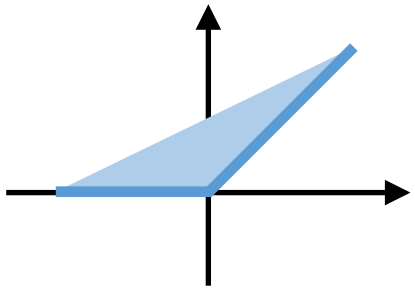
Optimization

$$\min_{\mathbf{x},\mathbf{y}} J(\mathbf{x}, \mathbf{y}, \mathcal{X}, \mathcal{Y}),$$

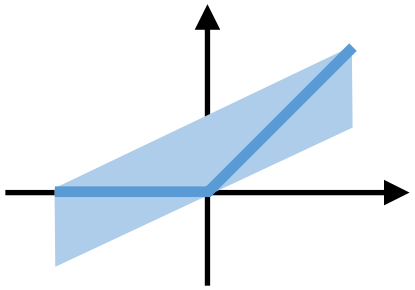
$$\text{s.t. } \mathbf{x} \in \mathcal{X}, \mathbf{y} \notin \mathcal{Y}, \mathbf{y} = \mathbf{f}(\mathbf{x}),$$



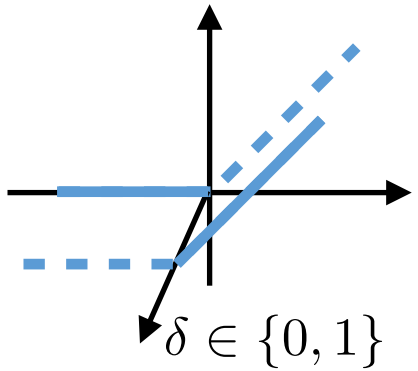
ReLU



Triangle Relaxation



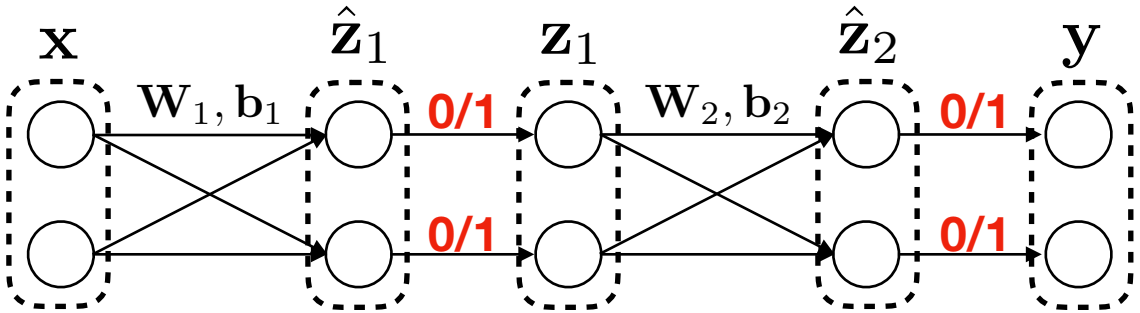
Parallel Relaxation



Mixed Integer Encoding

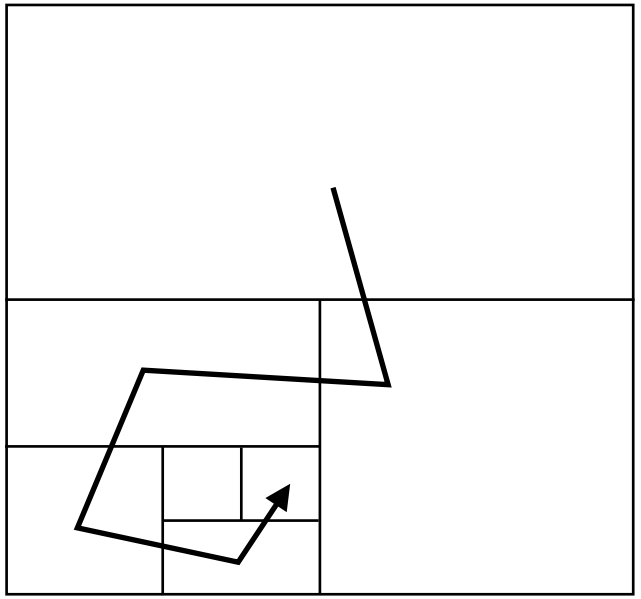
Search

1. Search in function space



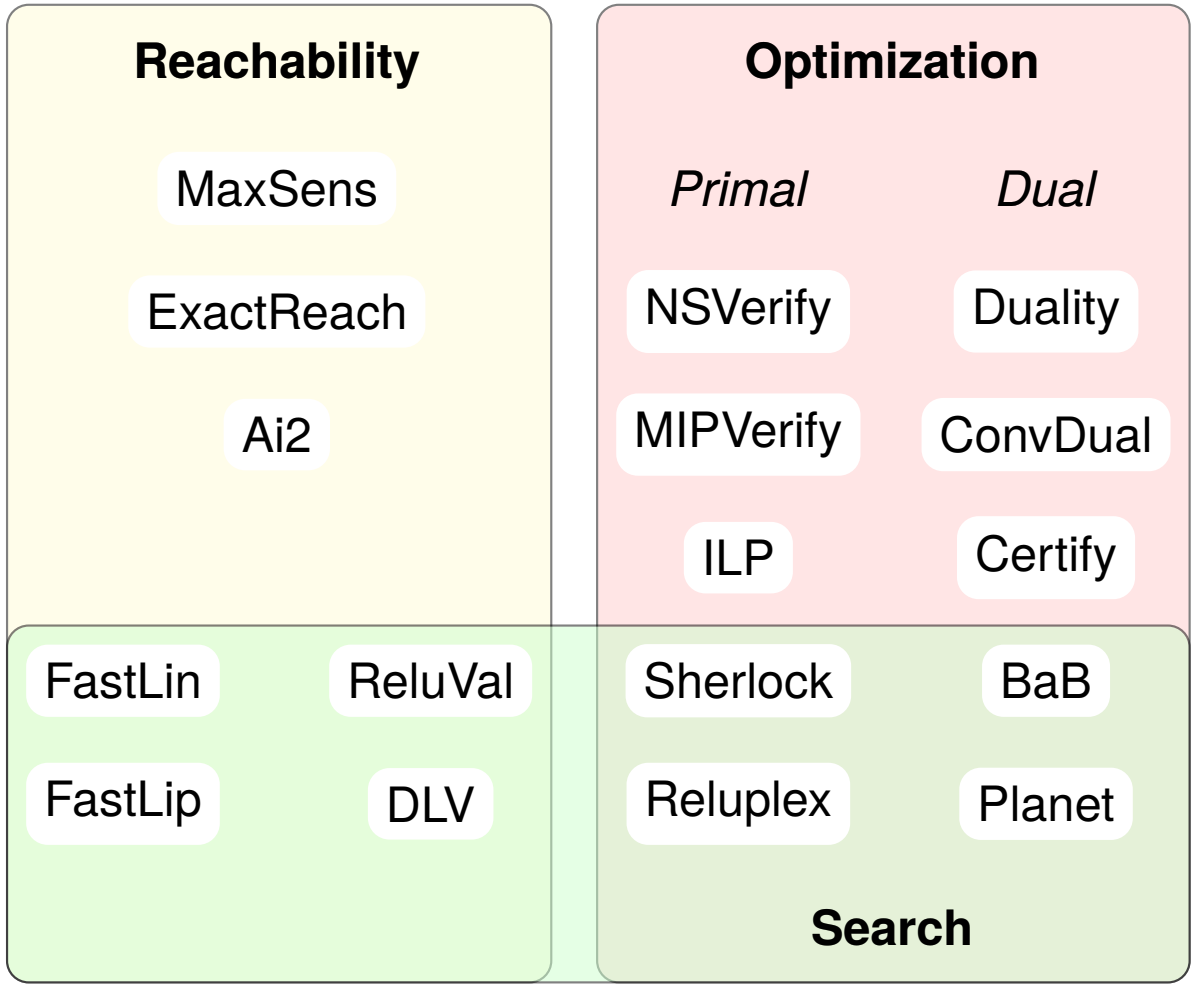
Satisfiability problem (SAT)

2. Search in input/hidden domain



Branch and Bound

Overview


Reachability
Optimization
Primal
Dual
MaxSens
ExactReach
Ai2
NSVerify
Duality
MIPVerify
ConvDual
ILP
Certify
FastLin
ReluVal
FastLip
DLV
Sherlock
BaB
Reluplex
Planet
Search

Reachability

Exact Reachability

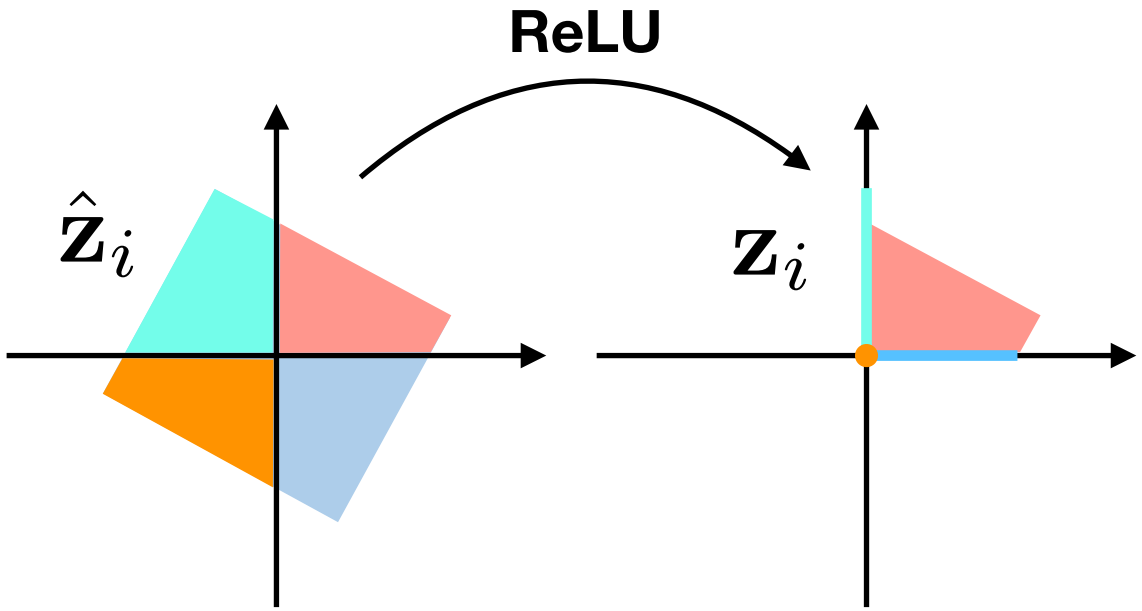
Split-and-Join

Interval Arithmetic

Symbolic Propagation

Network Approximation

ExactReach



Keep track of all polytopes

Reachability

Exact Reachability

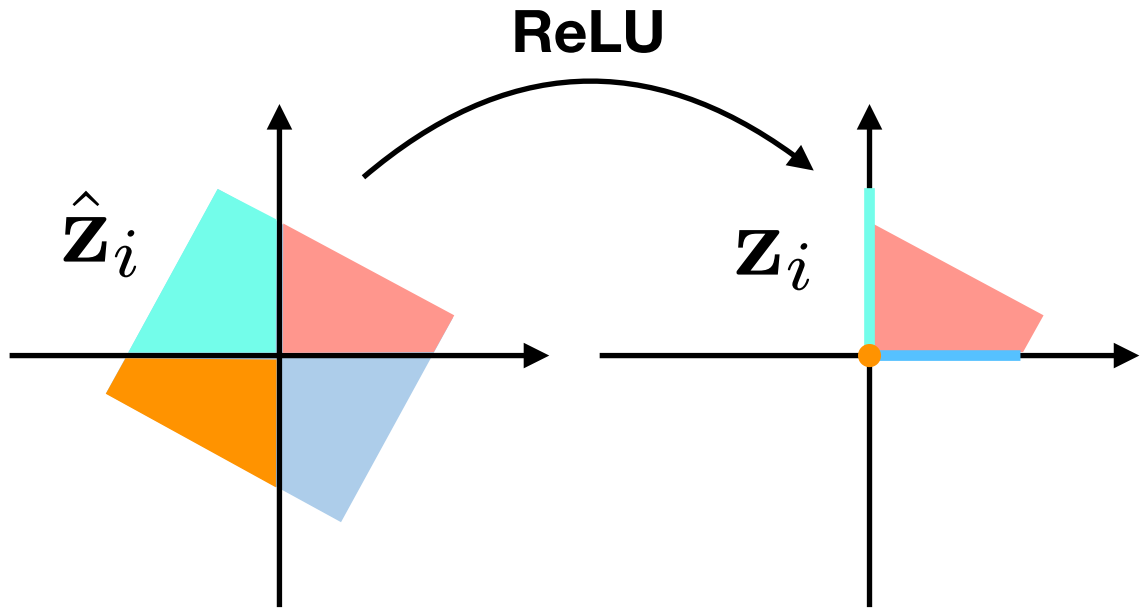
Split-and-Join

Interval Arithmetic

Symbolic Propagation

Network Approximation

Ai2



Join the polytopes for each layer

Reachability

Exact Reachability

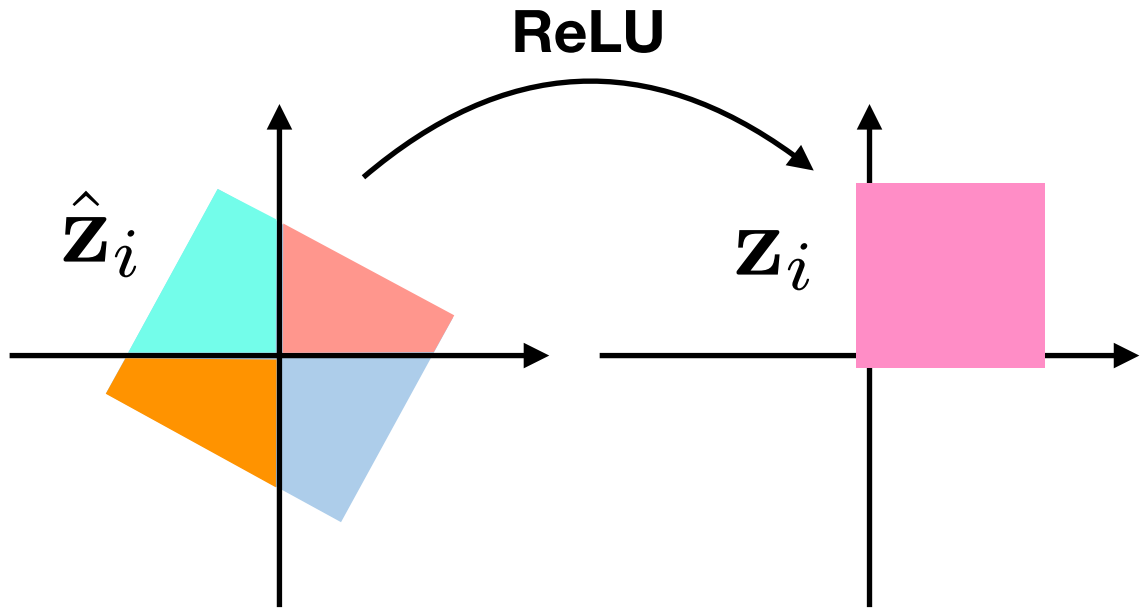
Split-and-Join

Interval Arithmetic

Symbolic Propagation

Network Approximation

Ai2



Join the polytopes for each layer

Reachability

Exact Reachability

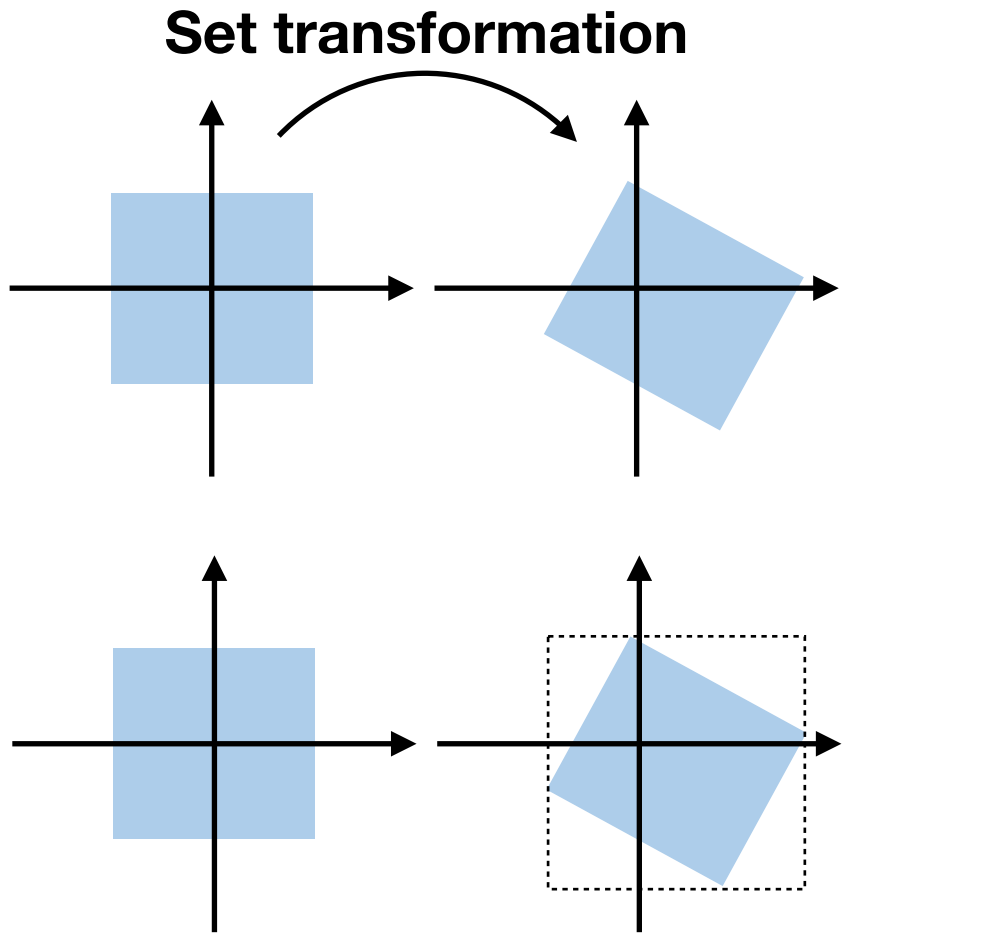
Split-and-Join

Interval Arithmetic

Symbolic Propagation

Network Approximation

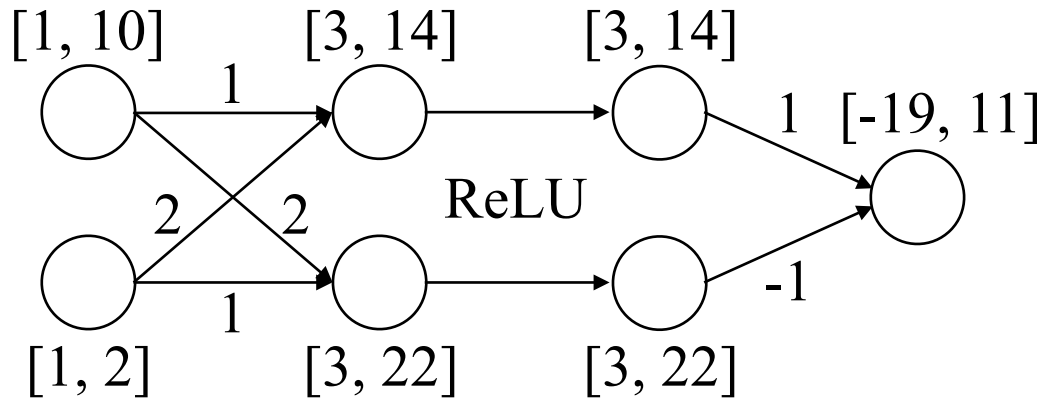
MaxSens

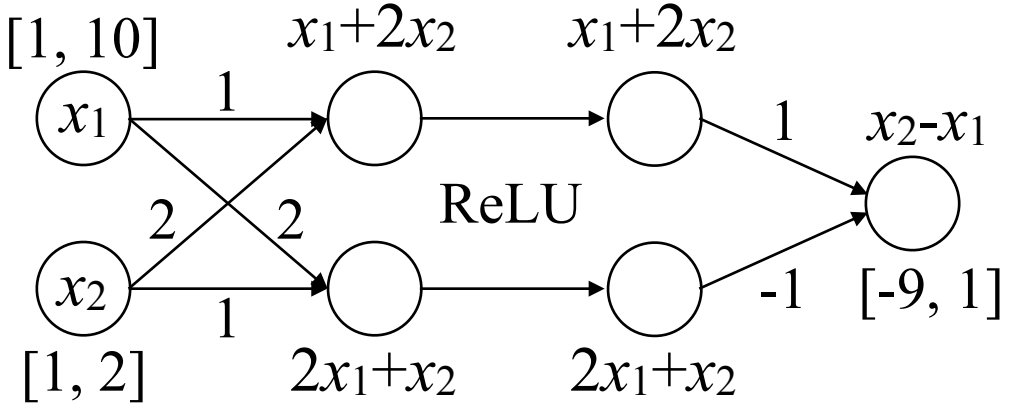


Interval arithmetic
Each axis is computed independently

Reachability

Exact Reachability
Split-and-Join
Interval Arithmetic
Symbolic Propagation
Network Approximation





ReluVal

Minimize over approximation in the hidden layers

Reachability

Exact Reachability
Split-and-Join
Interval Arithmetic
Symbolic Propagation
Network Approximation

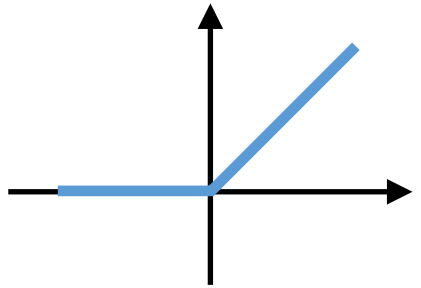
*** Dual network**

FastLin

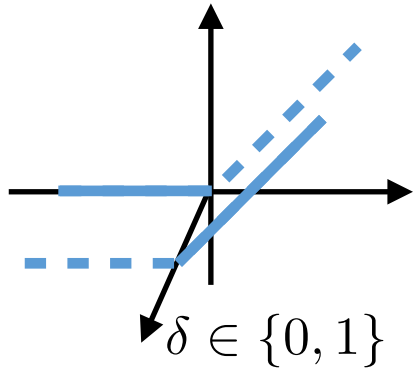
Primal Optimization

$$\min_{\mathbf{x},\mathbf{y}} J(\mathbf{x},\mathbf{y},\mathcal{X},\mathcal{Y}),$$

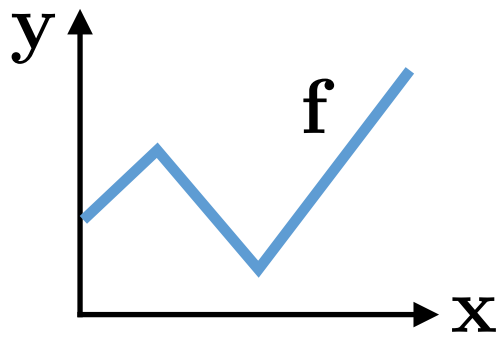
$$\text{s.t. } \mathbf{x} \in \mathcal{X}, \mathbf{y} \notin \mathcal{Y}, \mathbf{y} = \mathbf{f}(\mathbf{x}),$$

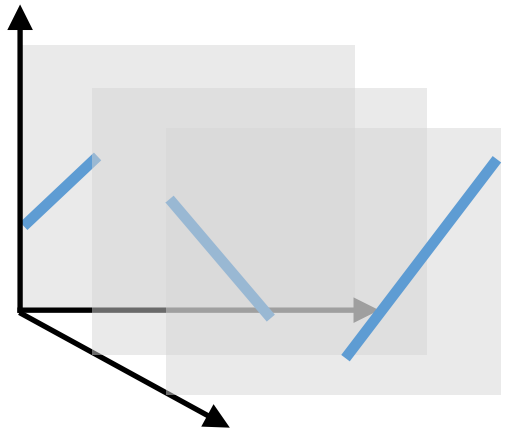


ReLU



Mixed Integer Encoding

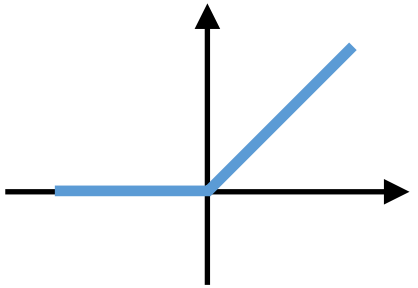




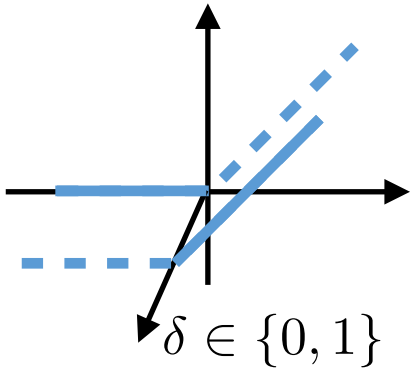
Primal Optimization

$$\min_{\mathbf{x},\mathbf{y}} J(\mathbf{x},\mathbf{y},\mathcal{X},\mathcal{Y}),$$

$$\text{s.t. } \mathbf{x} \in \mathcal{X}, \mathbf{y} \notin \mathcal{Y}, \mathbf{y} = \mathbf{f}(\mathbf{x}),$$



ReLU



Mixed Integer Encoding

NSVerify

$$z_{i,j} \geq \hat{z}_{i,j}$$
$$z_{i,j} \geq 0$$
$$z_{i,j} \leq \hat{z}_{i,j} + m\delta_{i,j}$$
$$z_{i,j} \leq m(1 - \delta_{i,j})$$

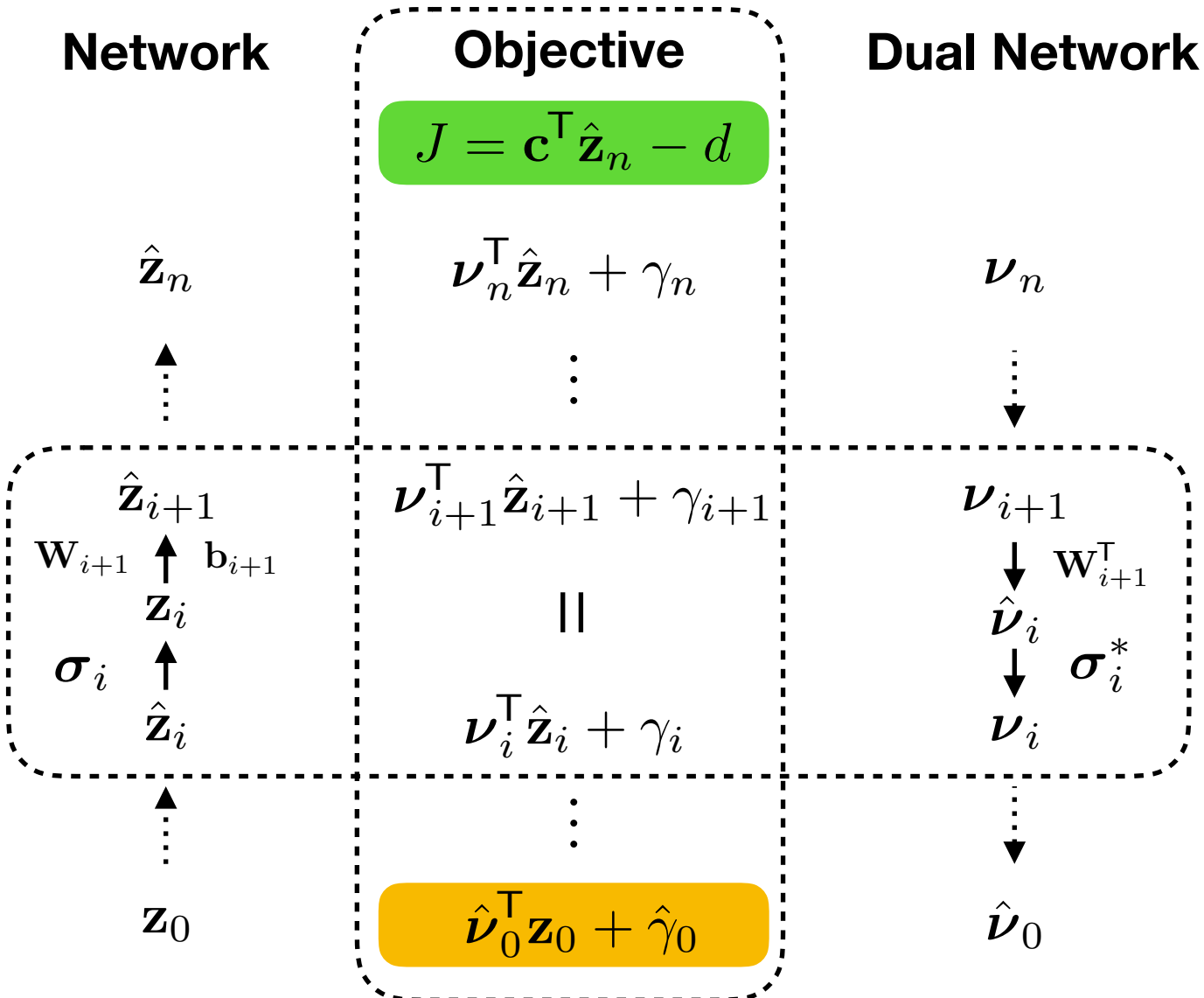
MIPVerify

$$z_{i,j} \geq \hat{z}_{i,j}$$
$$z_{i,j} \geq 0$$
$$z_{i,j} \leq \hat{u}_{i,j}\delta_{i,j}$$
$$z_{i,j} \leq \hat{z}_{i,j} - \hat{l}_{i,j}(1 - \delta_{i,j})$$

ILP

$$z_{i,j} = \delta_{i,j}\hat{z}_{i,j}$$

Dual Optimization



Three interpretations:
1. *Co-state or shadow price*
2. *Lagrangian multiplier*
3. *Back-propagation network*

$$\boldsymbol{\sigma}_i^*(\hat{\boldsymbol{\nu}}_i)^\mathsf{T}\hat{\mathbf{z}}_i \geq \hat{\boldsymbol{\nu}}_i^\mathsf{T}\boldsymbol{\sigma}_i(\hat{\mathbf{z}}_i)$$

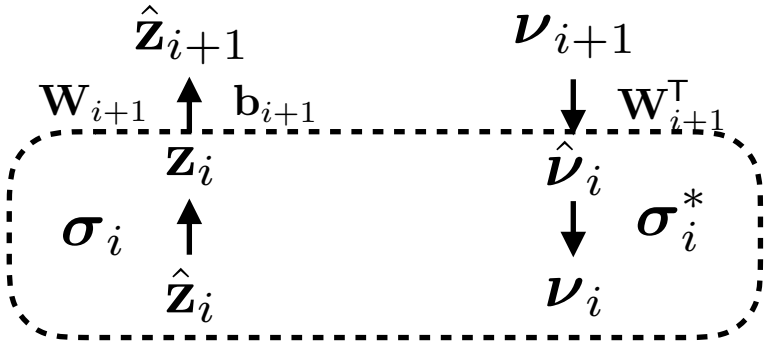
need to find minimum dual function

Dual Optimization

Lagrangian relaxation

Network relaxation

Semidefinite relaxation



$$\sigma_i^*(\hat{\mathbf{v}}_i)^{\mathsf{T}}\hat{\mathbf{z}}_i \geq \hat{\mathbf{v}}_i^{\mathsf{T}}\sigma_i(\hat{\mathbf{z}}_i)$$

find minimum dual function

Duality

$$\min_{\hat{\boldsymbol{\nu}}_i, \boldsymbol{\nu}_i} \quad \max_{\hat{\mathbf{l}}_i \leq \hat{\mathbf{z}}_i \leq \hat{\mathbf{u}}_i} \quad \hat{\boldsymbol{\nu}}_i^{\mathsf{T}}\boldsymbol{\sigma}_i(\hat{\mathbf{z}}_i) - \boldsymbol{\nu}_i^{\mathsf{T}}\hat{\mathbf{z}}_i$$

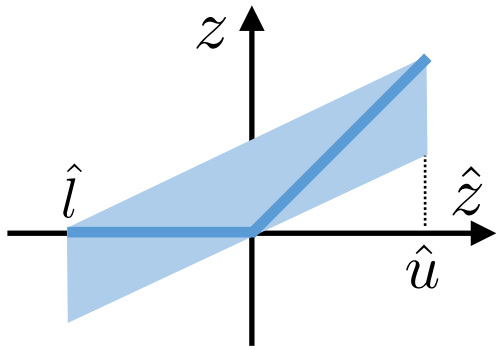
Dual Optimization

Lagrangian relaxation

Network relaxation

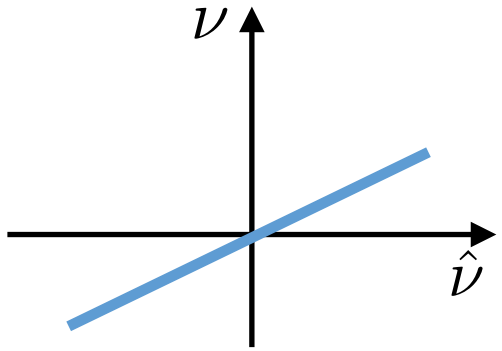
Semidefinite relaxation

ConvDual



Primal

$$\frac{\hat{u}}{\hat{u}-\hat{l}}\hat{z} \leq z \leq \frac{\hat{u}}{\hat{u}-\hat{l}}(\hat{z}-\hat{l})$$



Dual

$$\hat{\nu} z \leq \underbrace{\frac{\hat{u}}{\hat{u}-\hat{l}}}_{\nu}\hat{\nu}\,\hat{z}$$

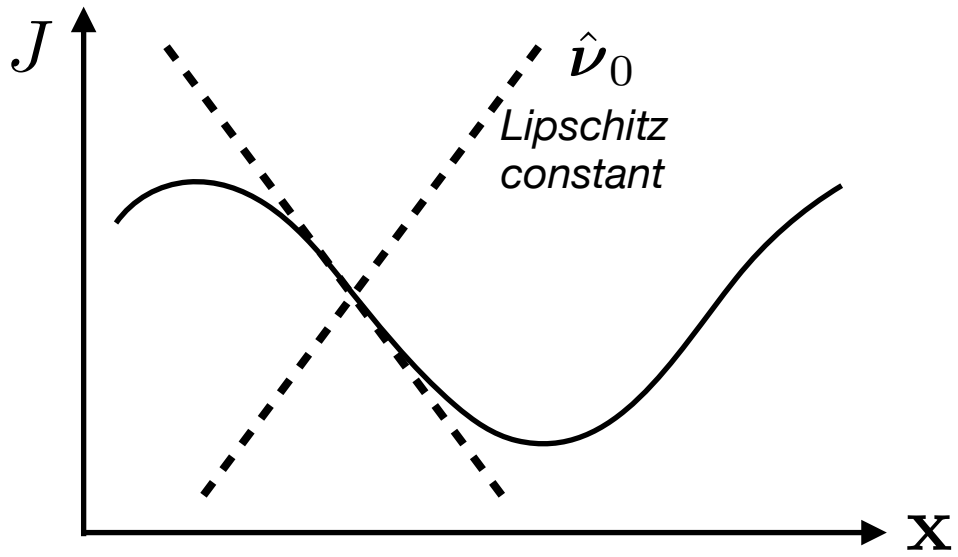
**** The dual network is linear!***

Dual Optimization

Lagrangian relaxation

Network relaxation

Semidefinite relaxation



Certify

$$\max_{\|\mathbf{x}-\mathbf{x}_0\|_\infty \le \epsilon} J(\mathbf{x}) \le J(\mathbf{x}_0) + \frac{\epsilon}{4} \max_{\mathbf{P}\succeq\mathbf{0}, P_{jj}\le 1} \langle \mathbf{M}(\mathbf{c},\mathbf{W}), \mathbf{P}\rangle$$

Only works for NN with one hidden layer

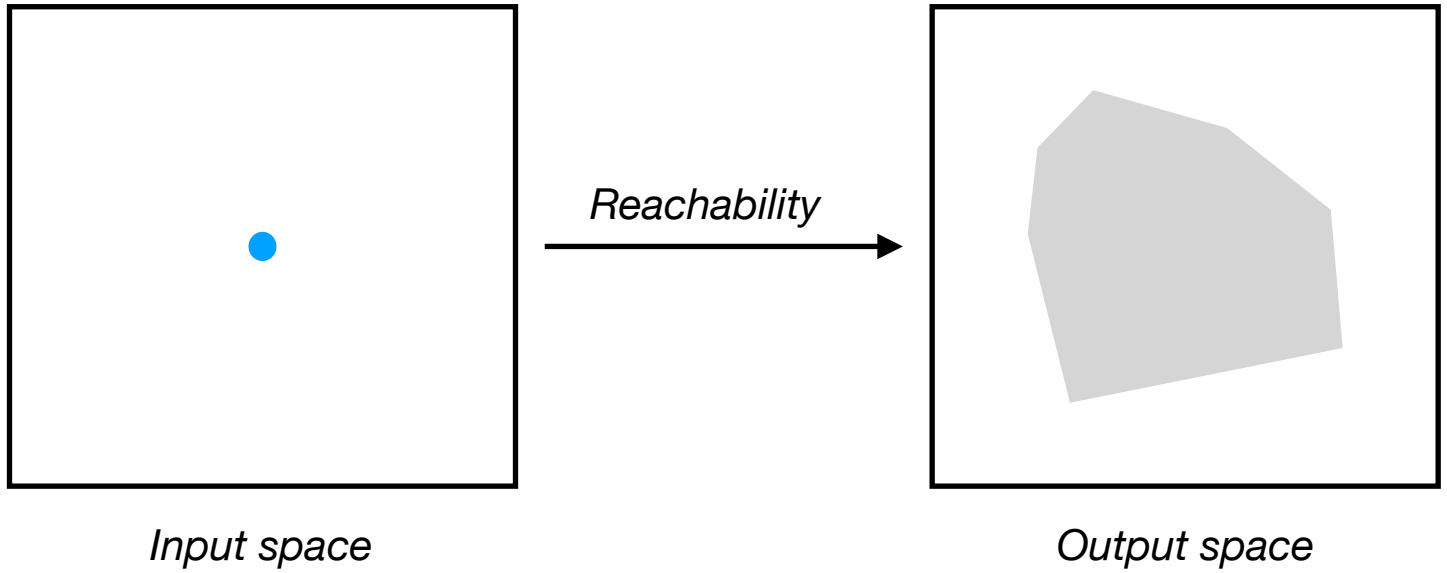
Search + Reachability

FastLin	**Binary search of desired input radius** **Reachability by network approximation**
FastLip	**FastLin + Lipchitz estimation**
ReluVal	**Search input domain by interval split** **Reachability by symbolic propagation**
DLV	**Search layer-by-layer in hidden layers**

Search + Reachability

FastLin

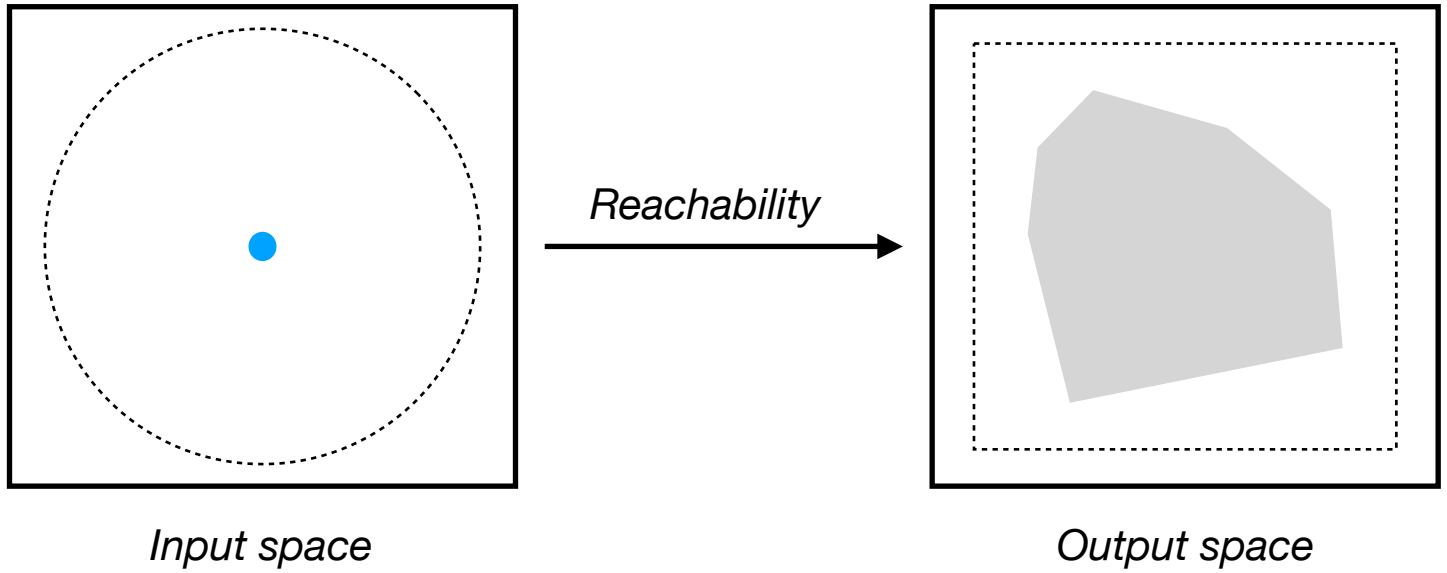
FastLip



Search + Reachability

FastLin

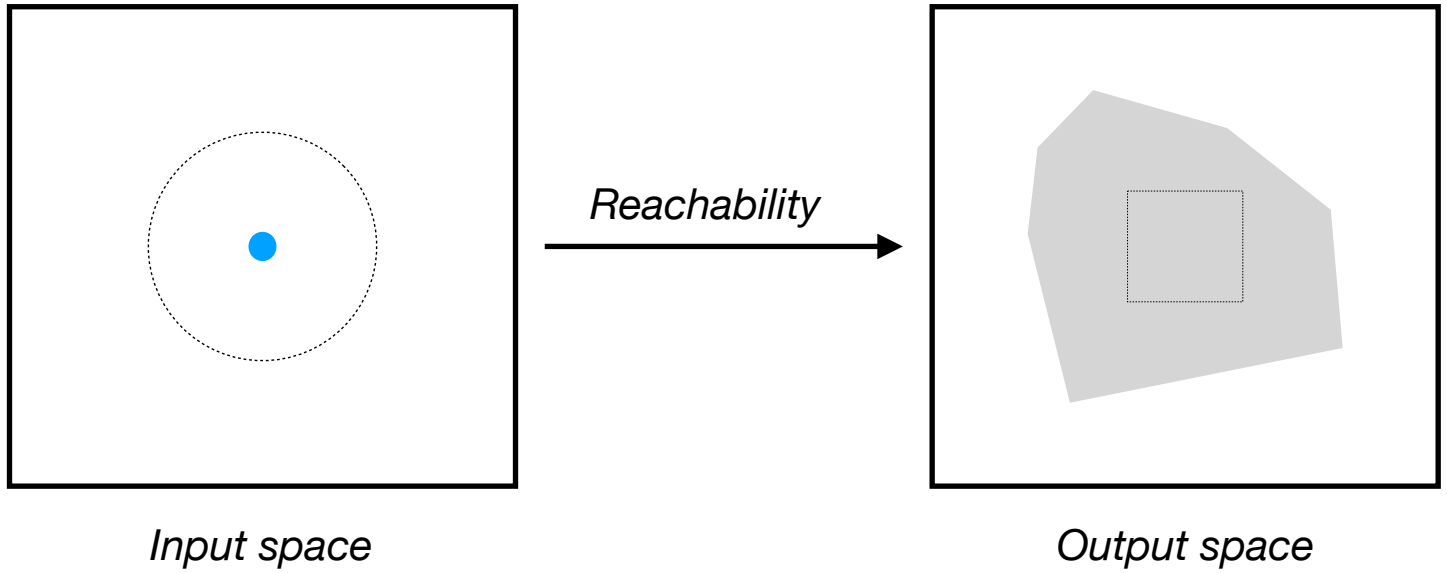
FastLip



Search + Reachability

FastLin

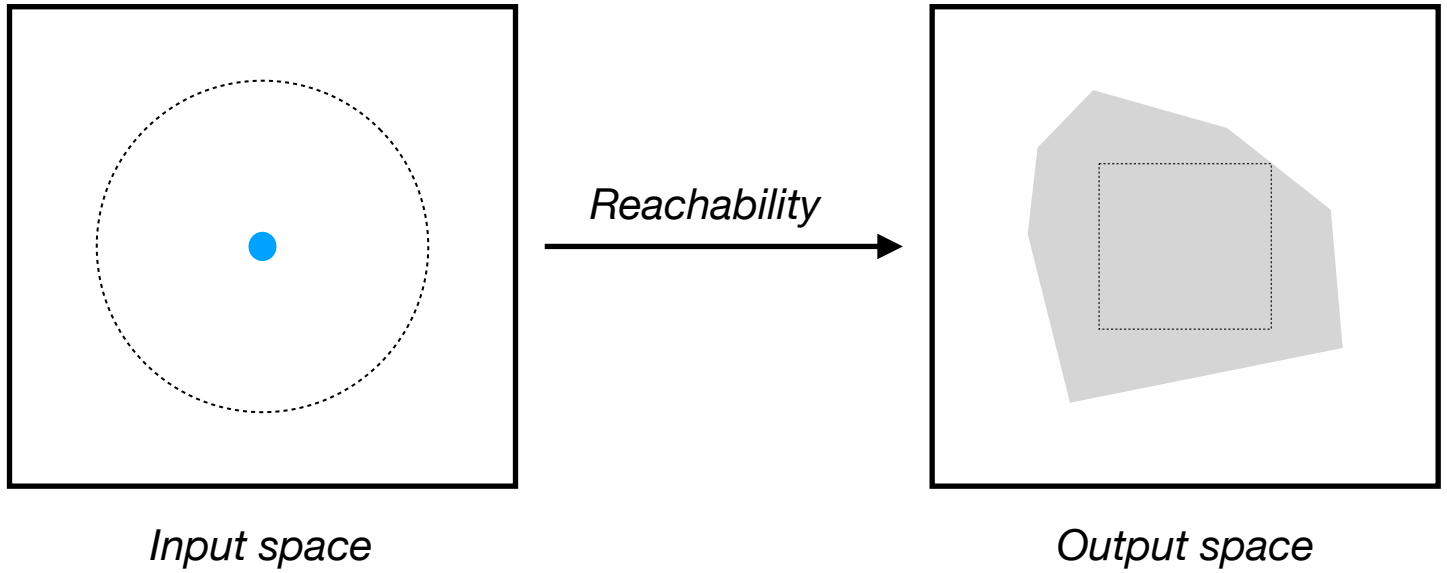
FastLip



Search + Reachability

FastLin

FastLip



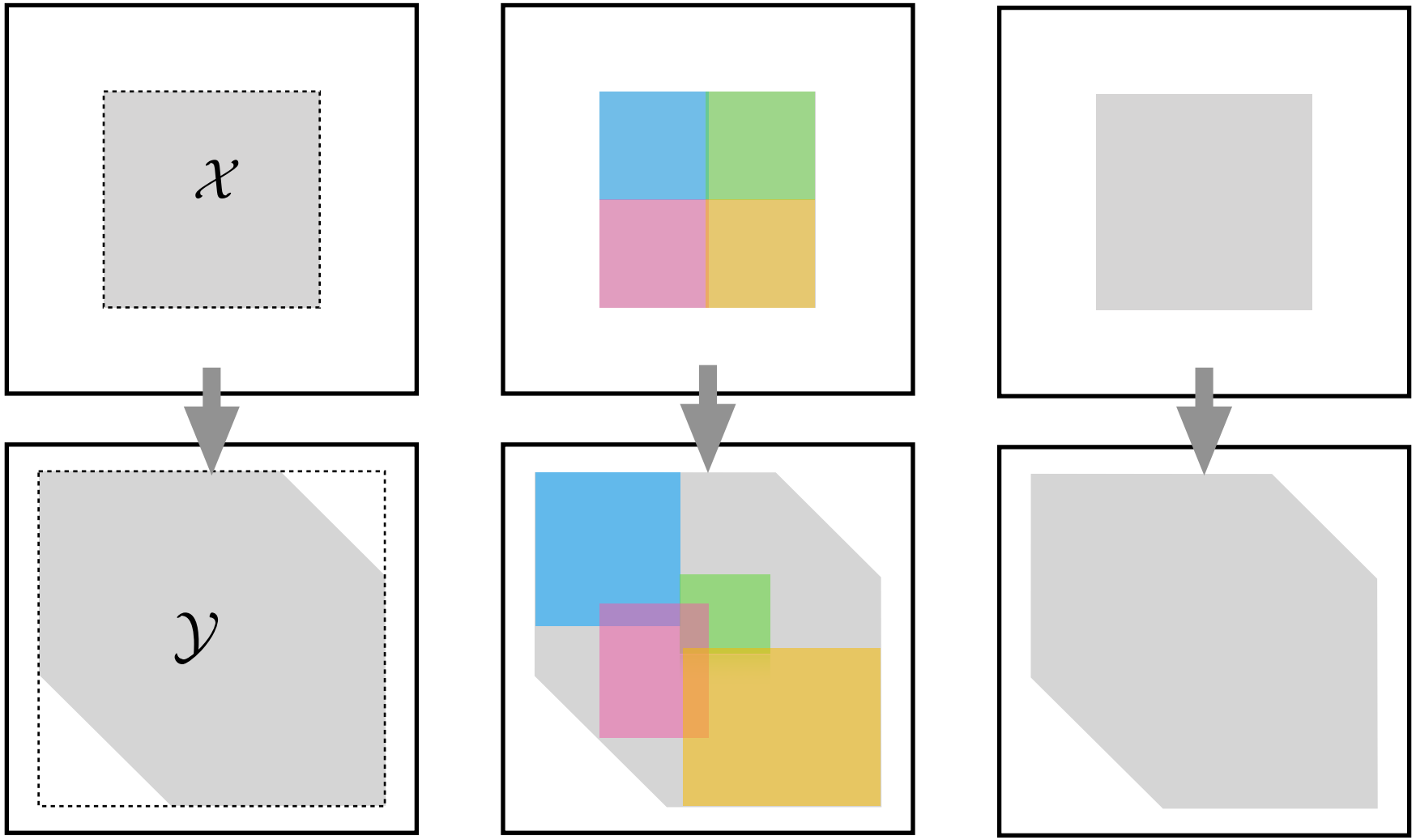
Search + Reachability

ReluVal

Search + Reachability

ReluVal

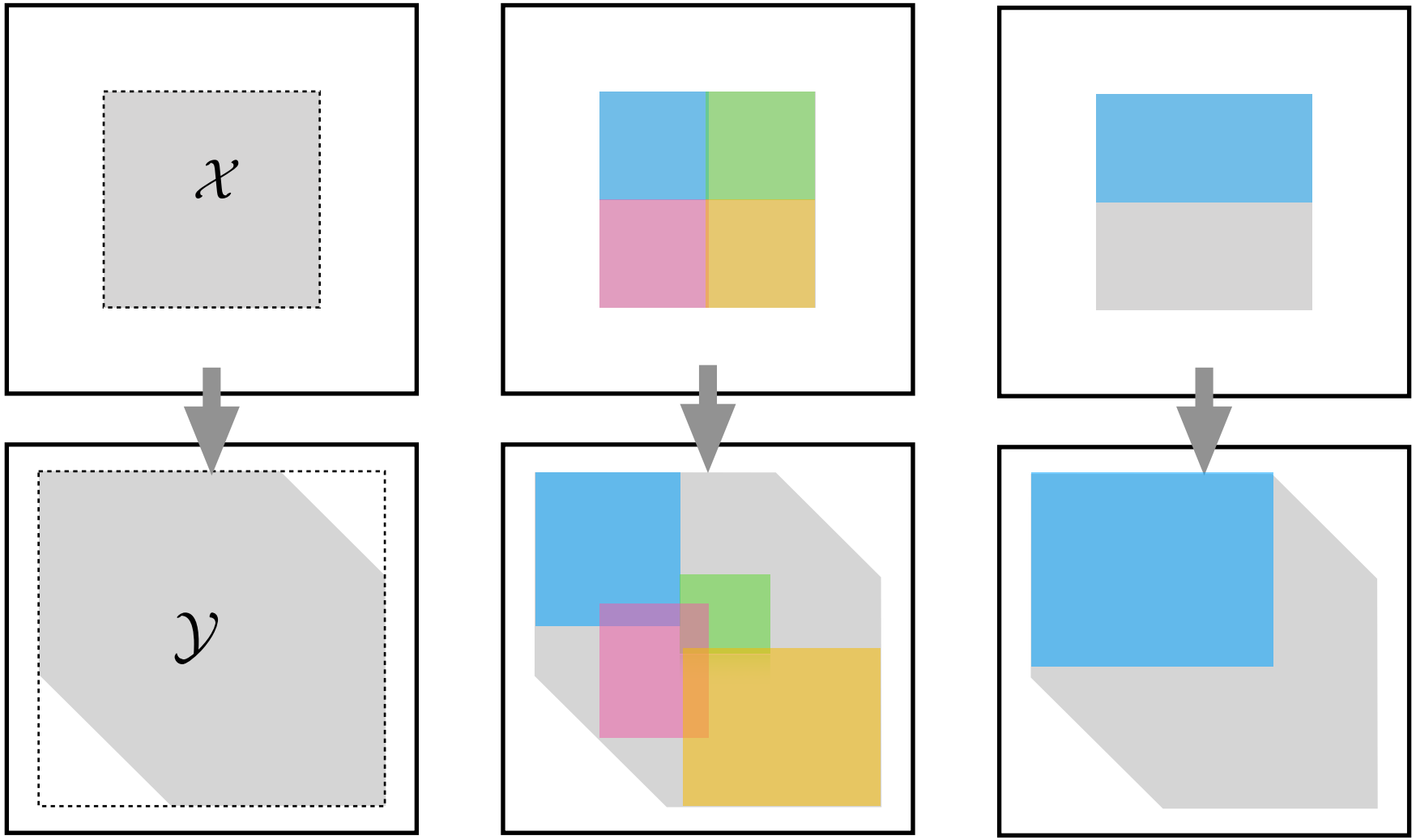
Iterative interval refinement



Search + Reachability

ReluVal

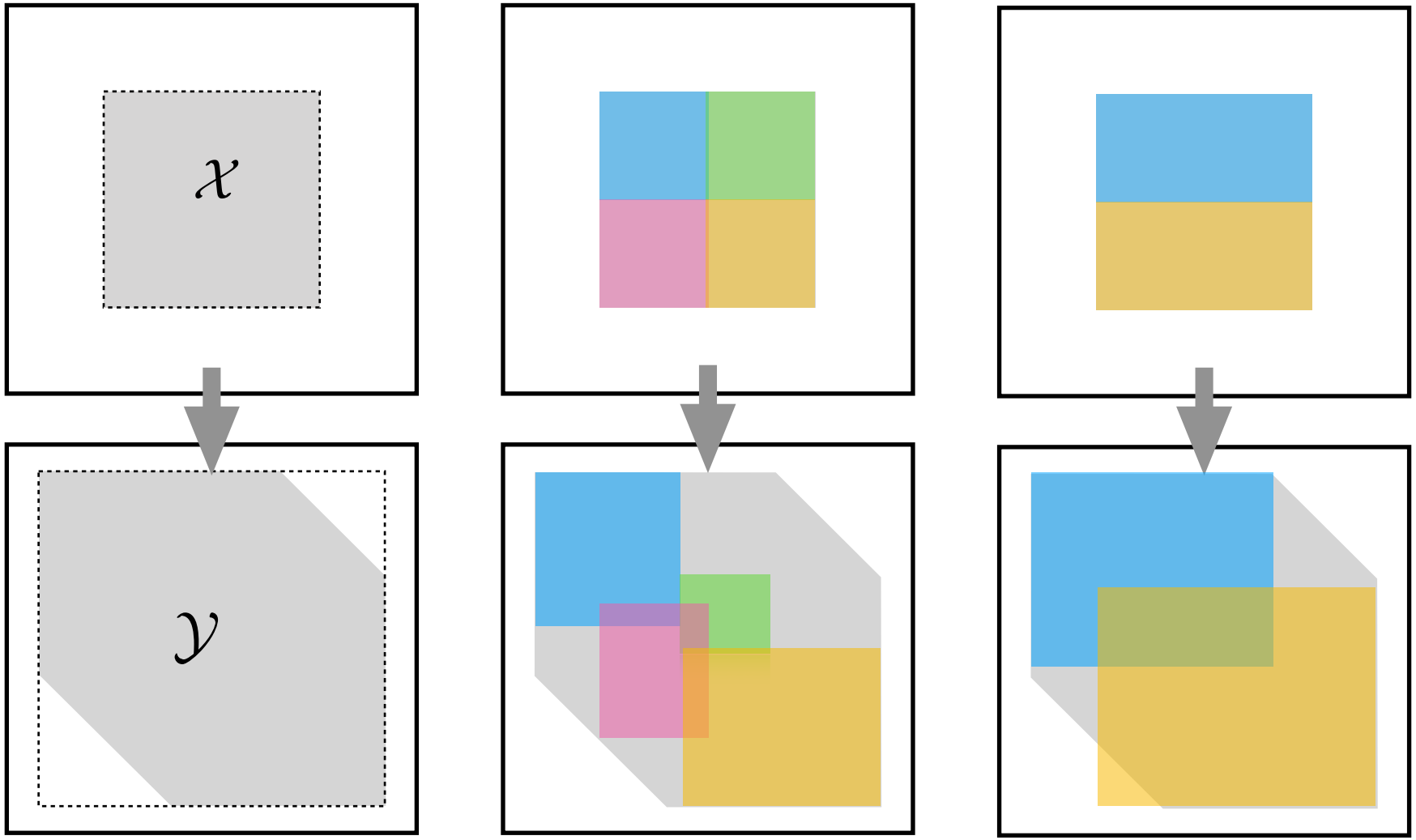
Iterative interval refinement



Search + Reachability

ReluVal

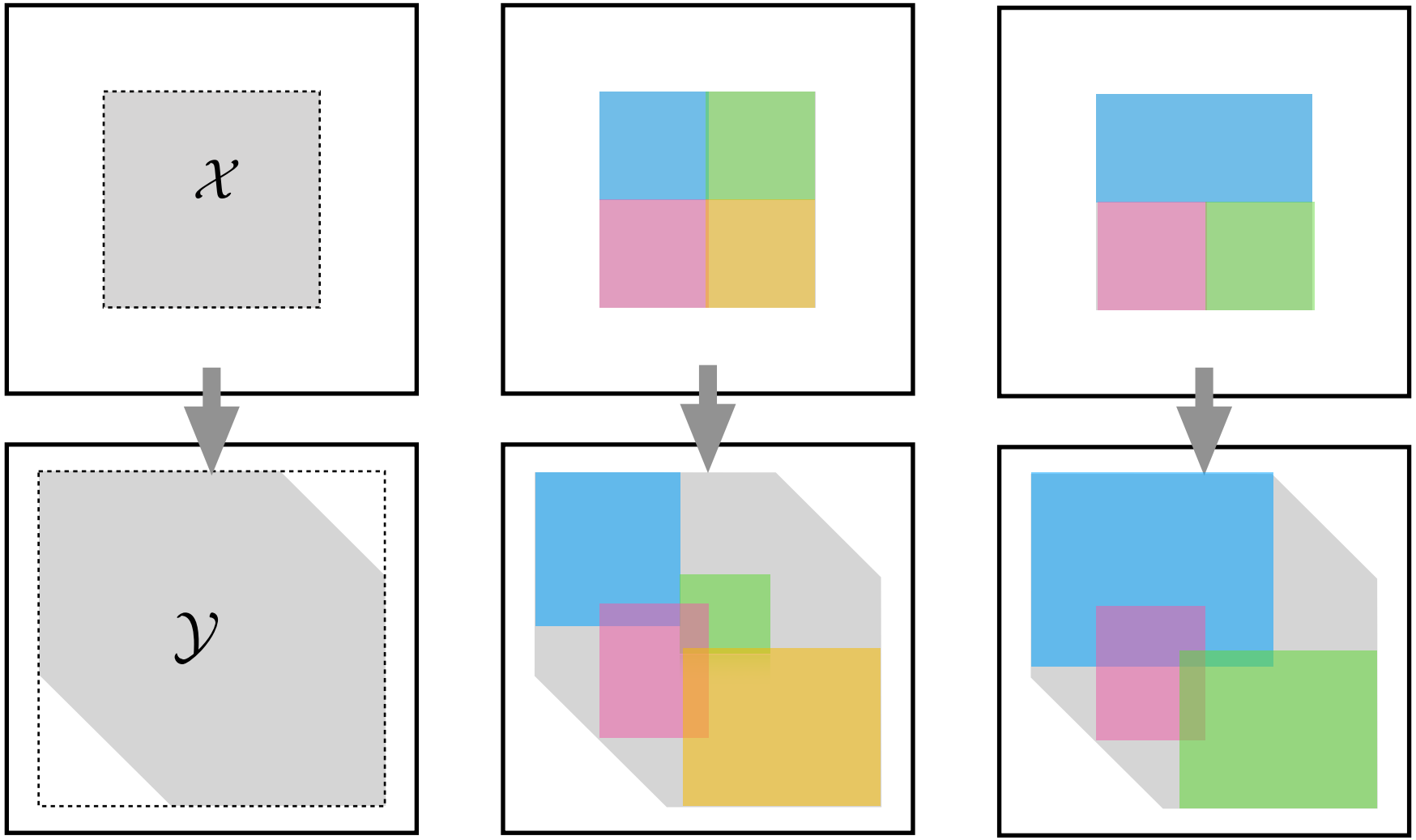
Iterative interval refinement



Search + Reachability

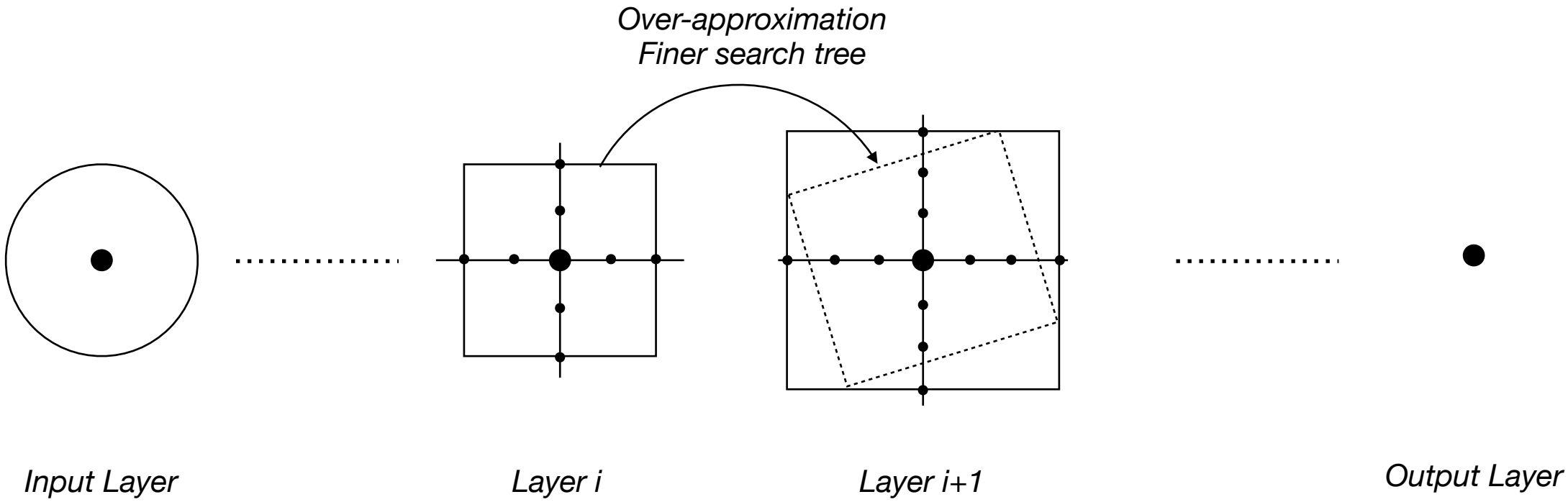
ReluVal

Iterative interval refinement



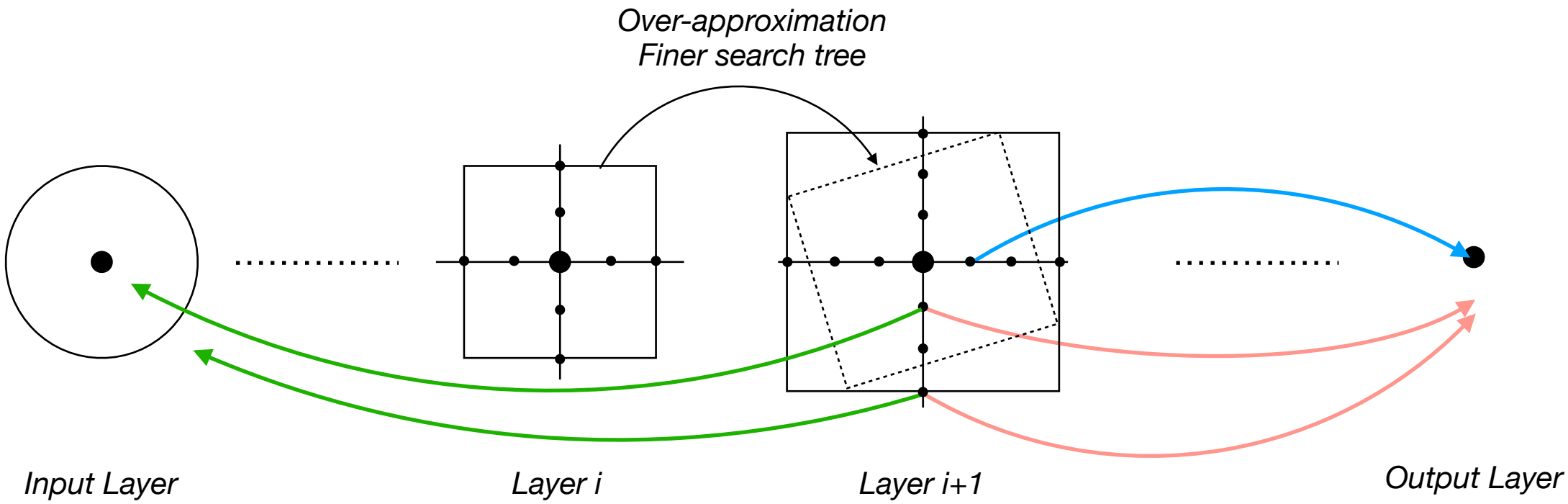
Search + Reachability

DLV



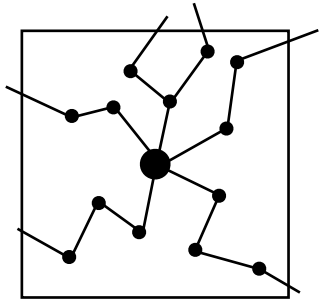
Search + Reachability

DLV

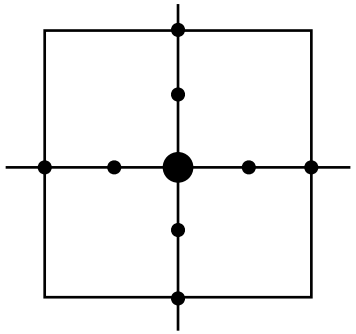


Search + Reachability

DLV

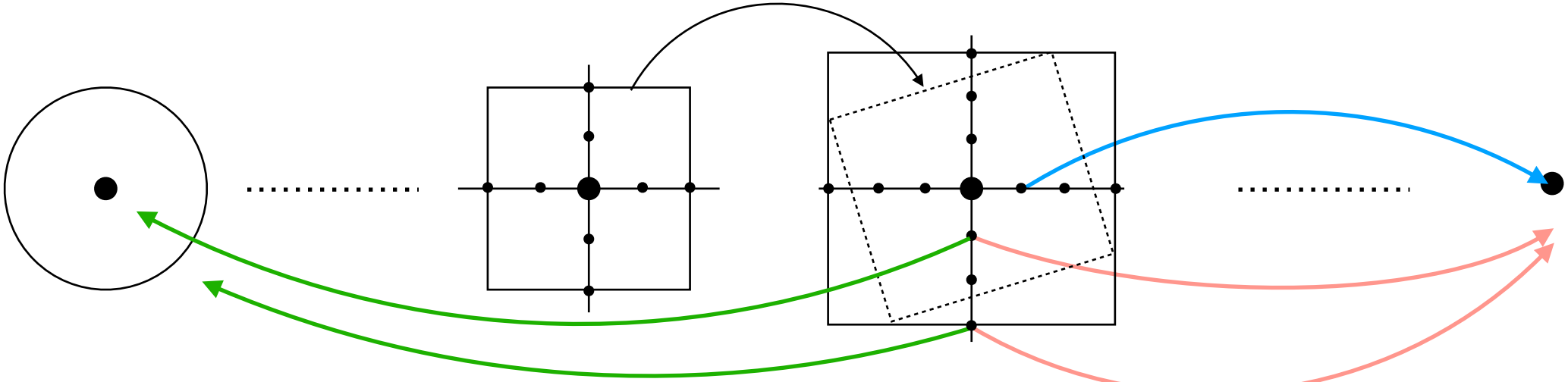


High-dimensional search
Complicated O(2^n)



Low-dimensional search
Easy O(2n)

Over-approximation
Finer search tree



Input Layer

Layer i

Layer i+1

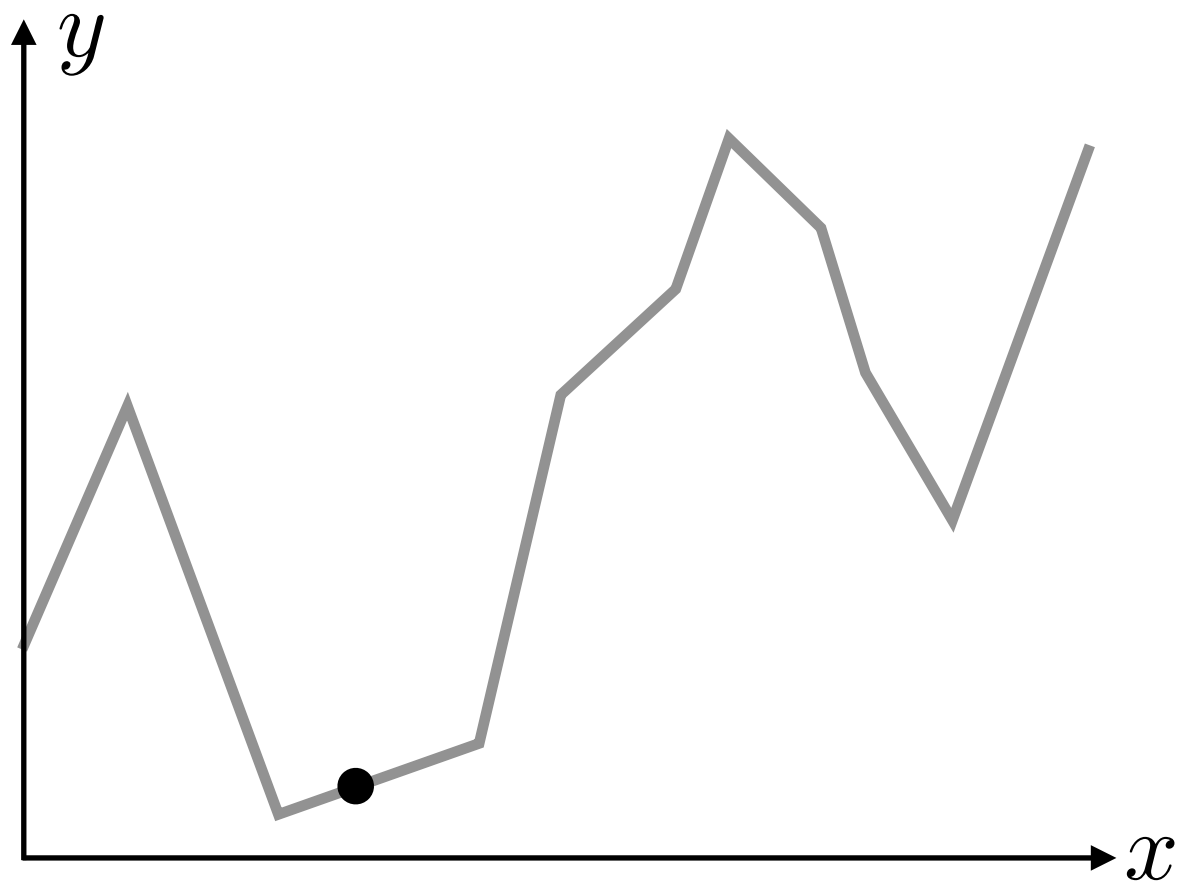
Output Layer

Search + Optimization

Sherlock	Search input domain Under local constraints or global constraints
BaB	Search input domain by splitting to smaller intervals
Planet	Search for feasible activation patterns
Reluplex	Search for feasible activation patterns

Search + Optimization

Sherlock



Global search

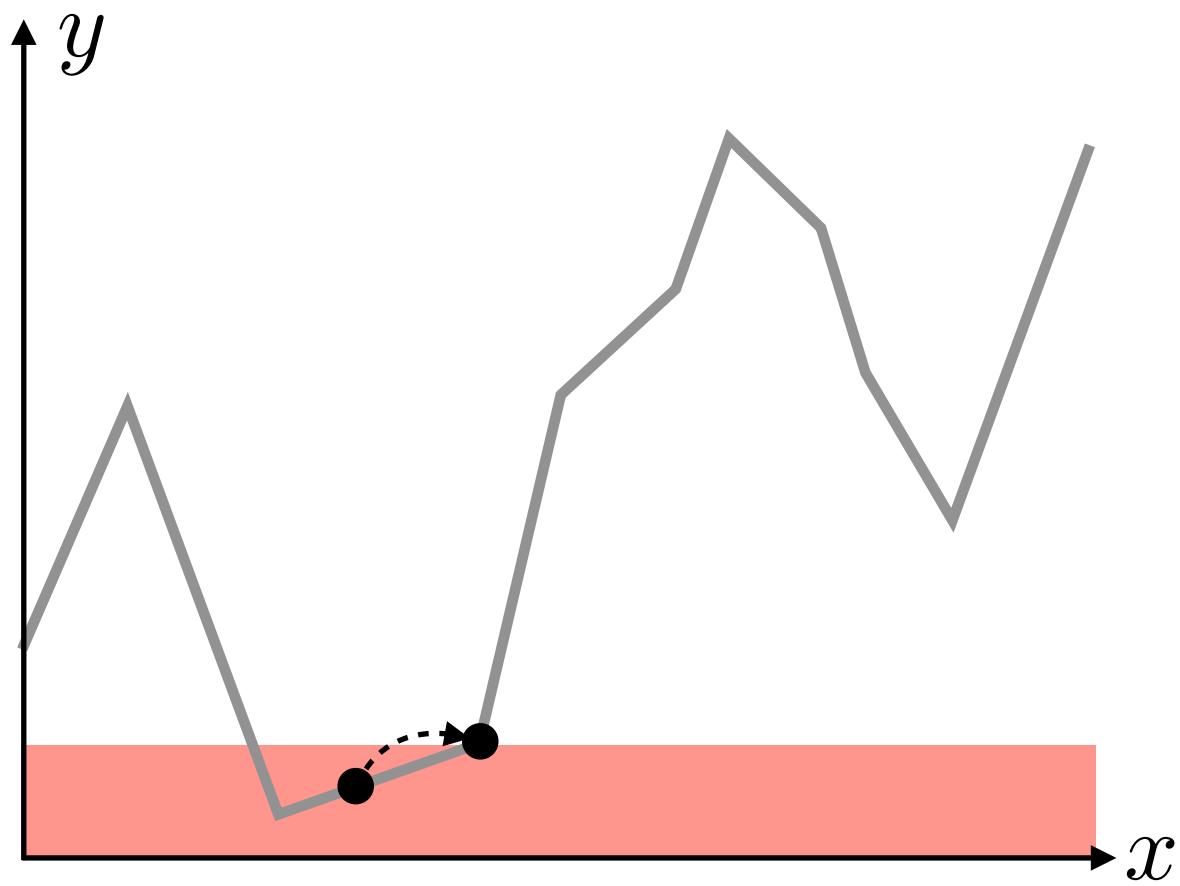
Primal optimization with MILP encoding

Local search

Primal optimization with LP encoding

Search + Optimization

Sherlock



Global search

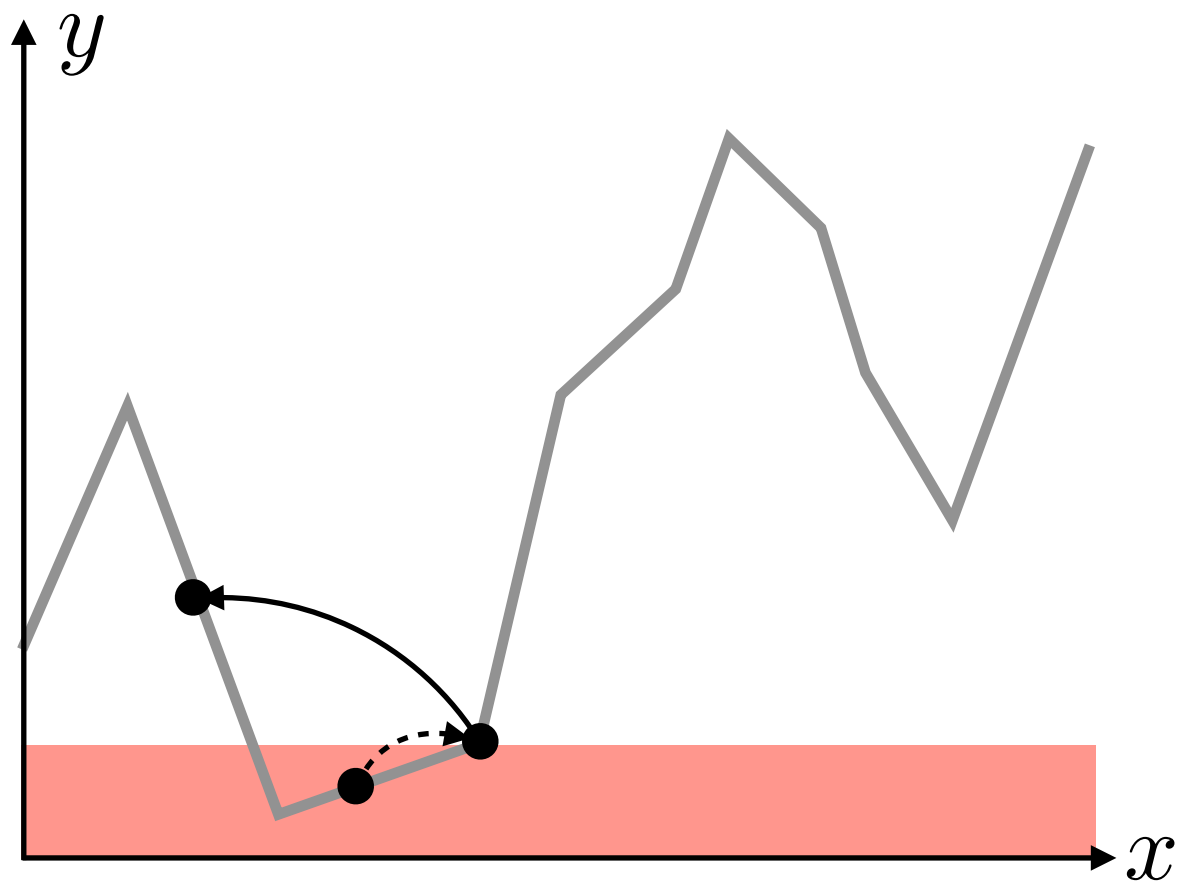
Primal optimization with MILP encoding

Local search

Primal optimization with LP encoding

Search + Optimization

Sherlock



Global search

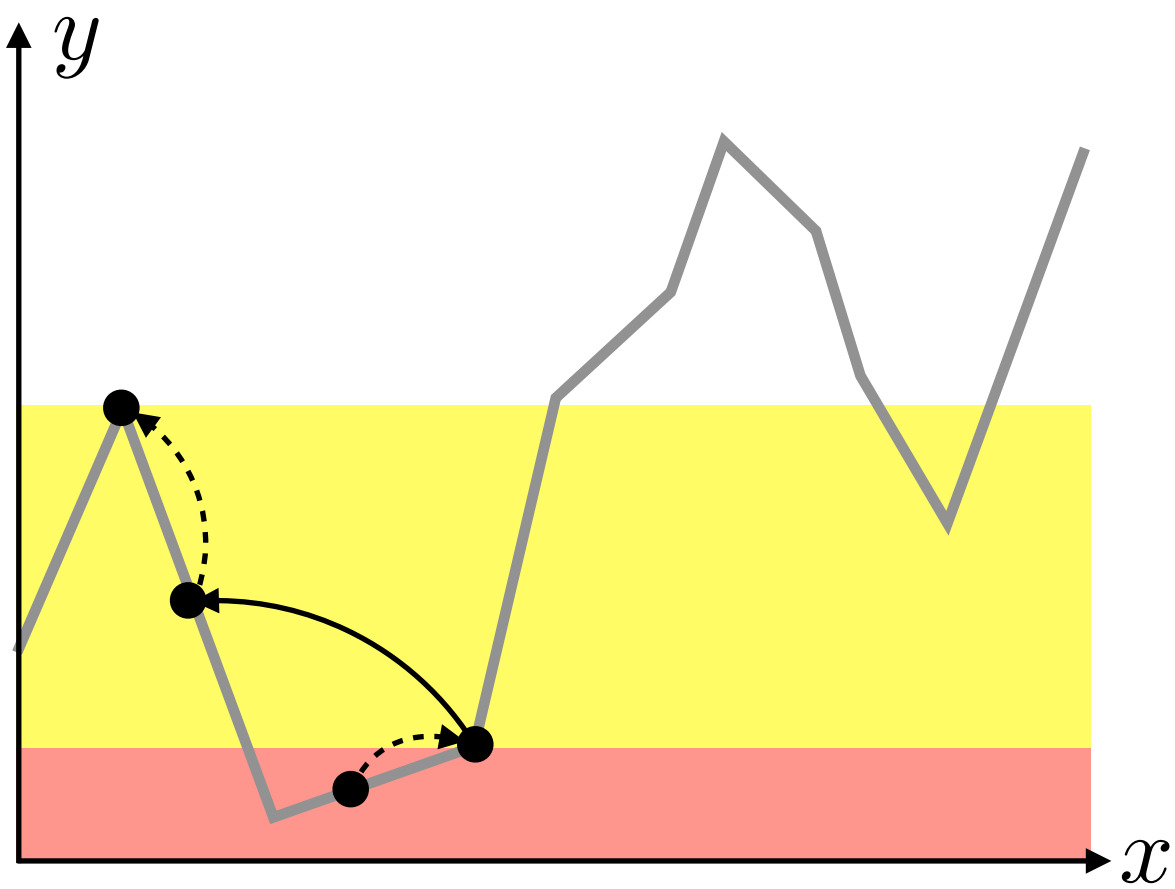
Primal optimization with MILP encoding

Local search

Primal optimization with LP encoding

Search + Optimization

Sherlock



Global search

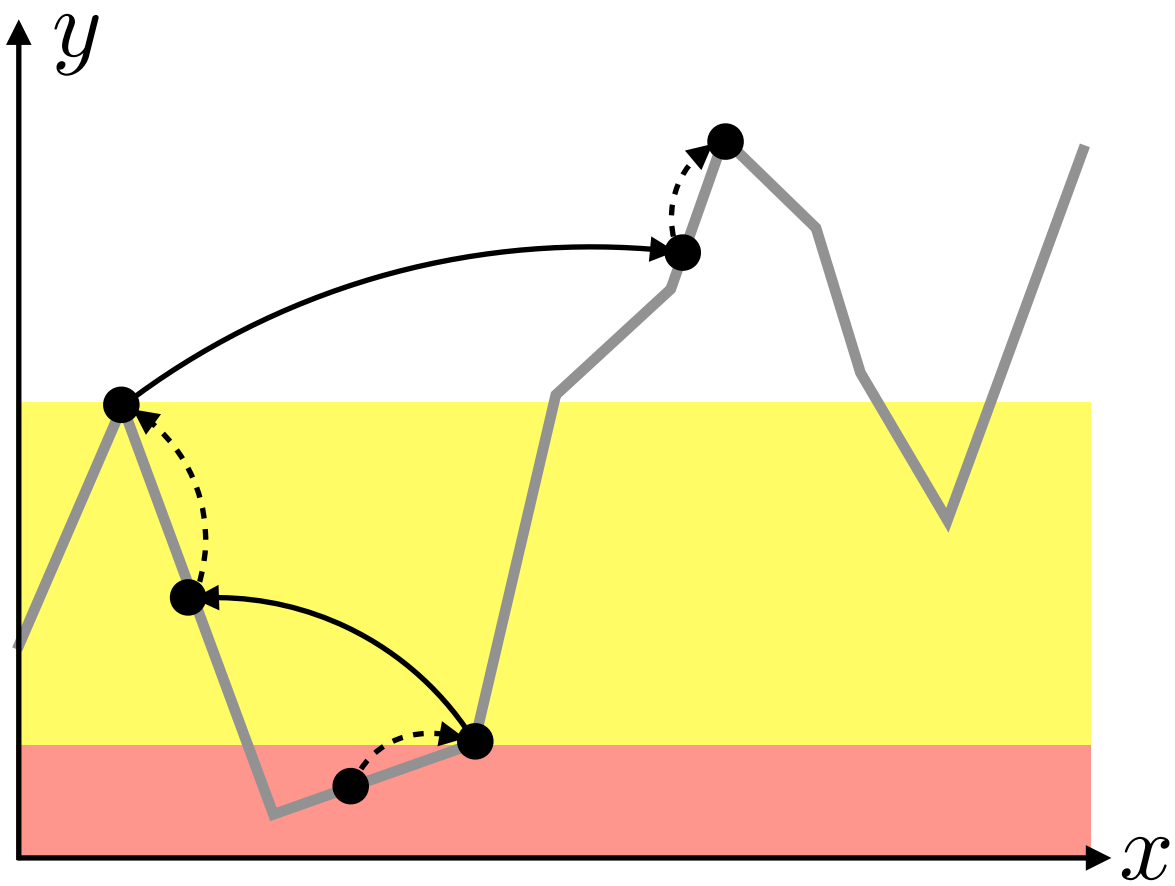
Primal optimization with MILP encoding

Local search

Primal optimization with LP encoding

Search + Optimization

Sherlock



Global search

Primal optimization with MILP encoding

Local search

Primal optimization with LP encoding

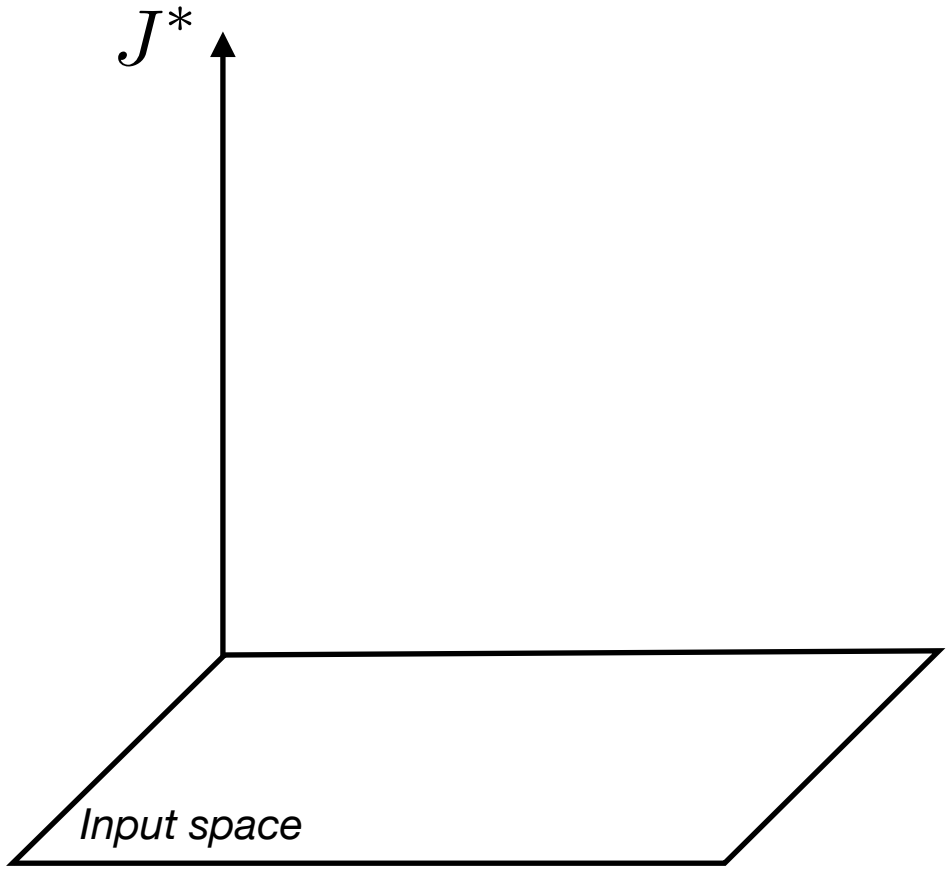
Search + Optimization

BaB

$$J^* = \max J$$

For every domain

- *Compute upper bound by optimization using network relaxation*
- *Compute lower bound by sampling*



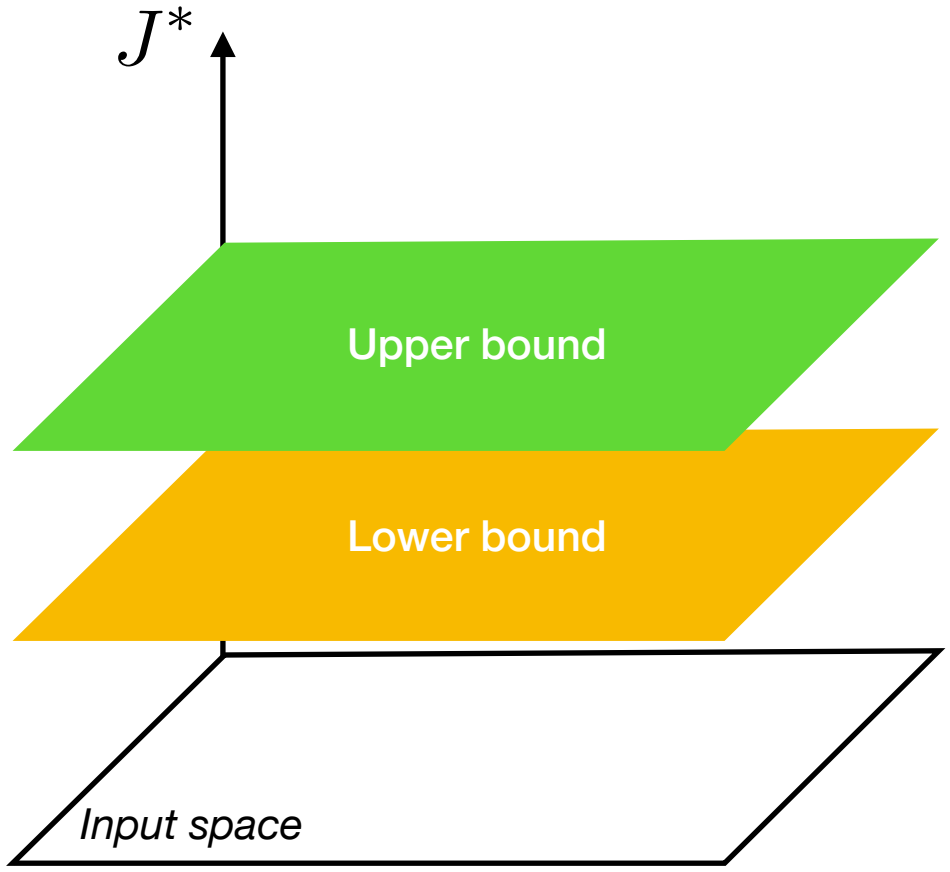
Search + Optimization

BaB

$$J^* = \max J$$

For every domain

- *Compute upper bound by optimization using network relaxation*
- *Compute lower bound by sampling*



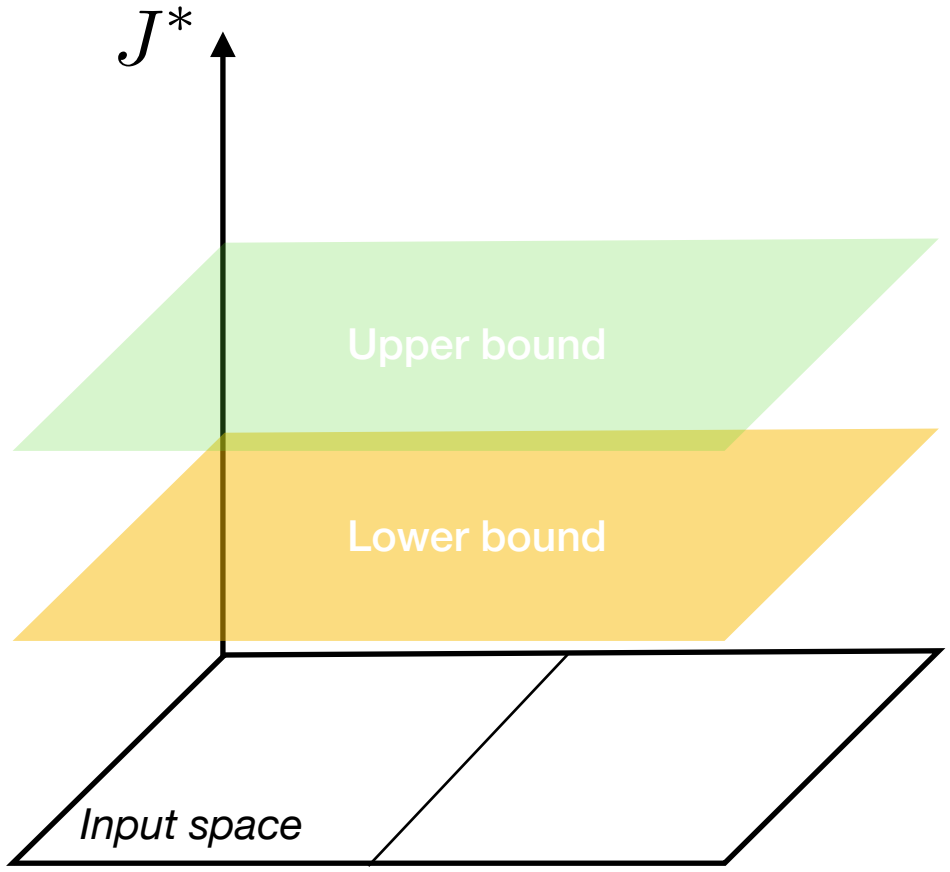
Search + Optimization

BaB

$$J^* = \max J$$

For every domain

- *Compute upper bound by optimization using network relaxation*
- *Compute lower bound by sampling*



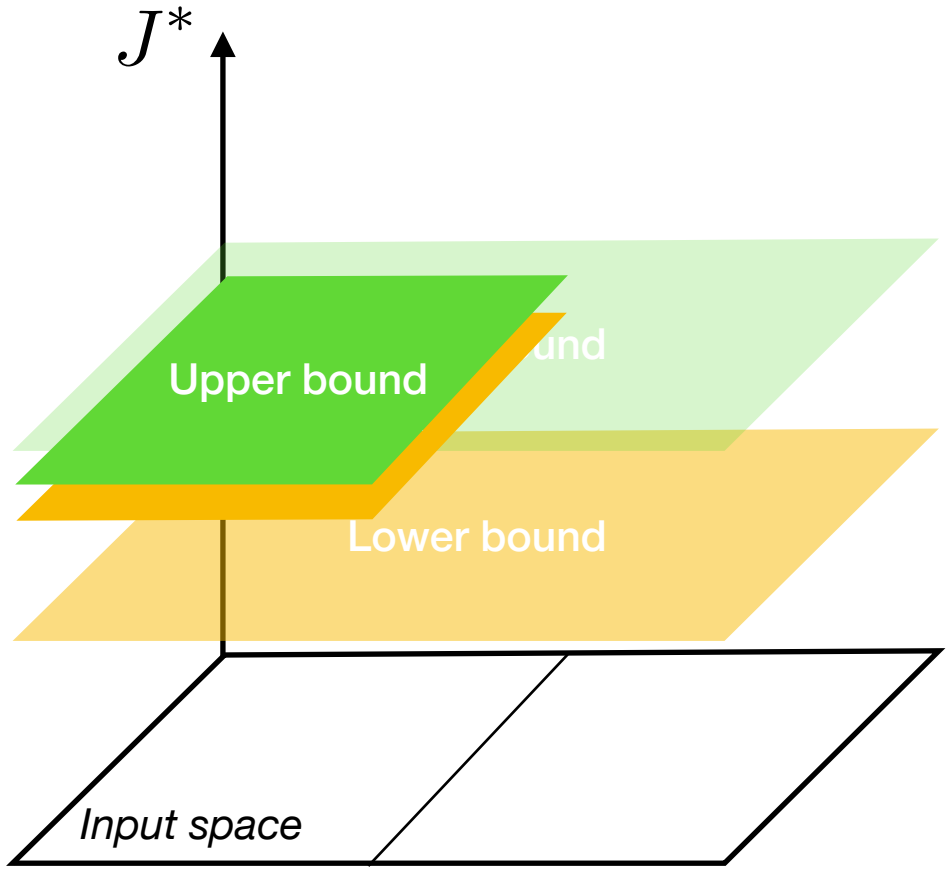
Search + Optimization

BaB

$$J^* = \max J$$

For every domain

- *Compute upper bound by optimization using network relaxation*
- *Compute lower bound by sampling*



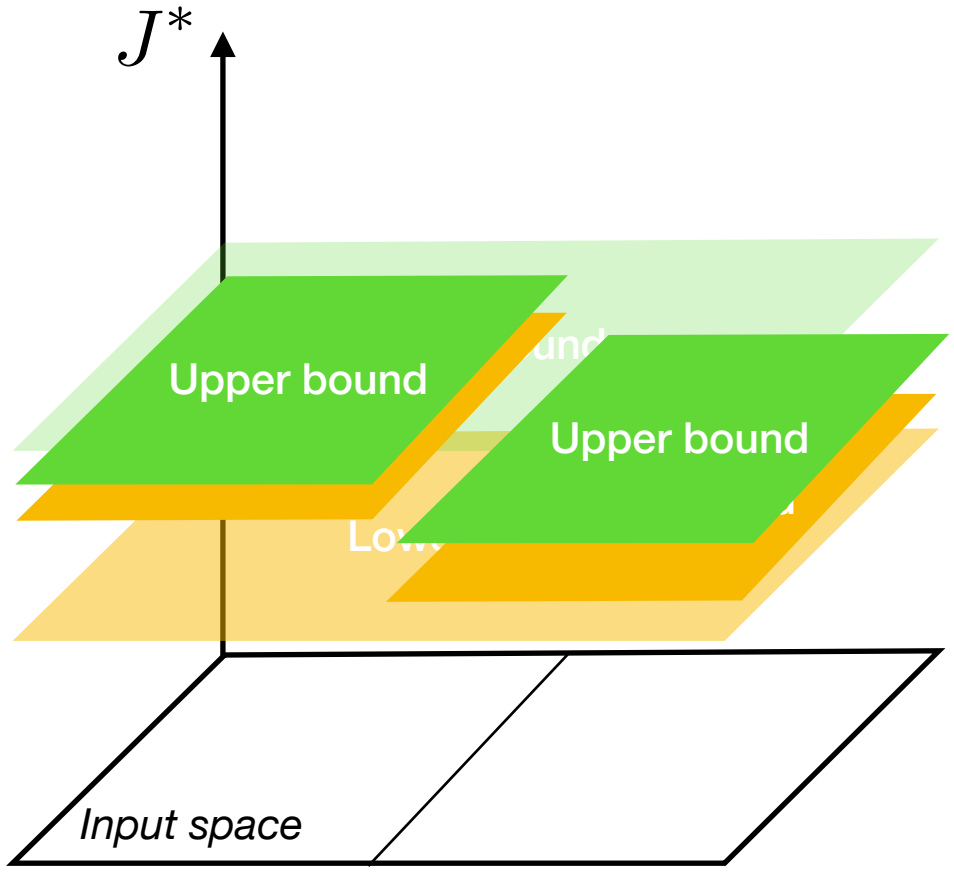
Search + Optimization

BaB

$$J^* = \max J$$

For every domain

- *Compute upper bound by optimization using network relaxation*
- *Compute lower bound by sampling*



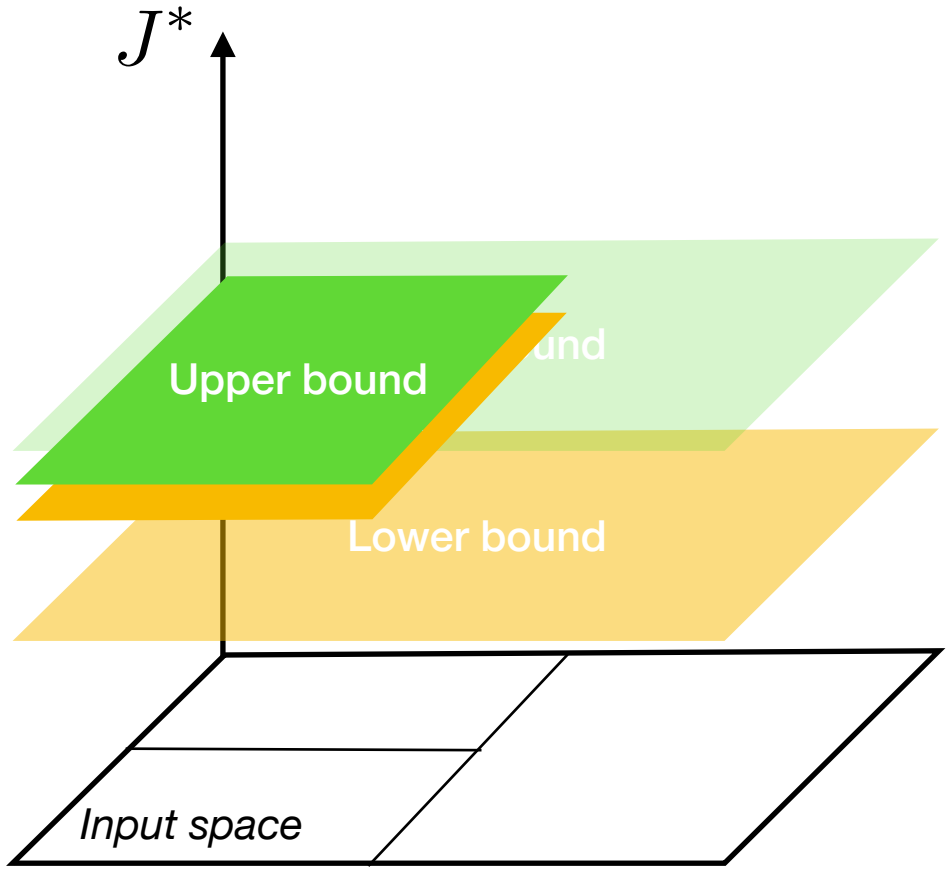
Search + Optimization

BaB

$$J^* = \max J$$

For every domain

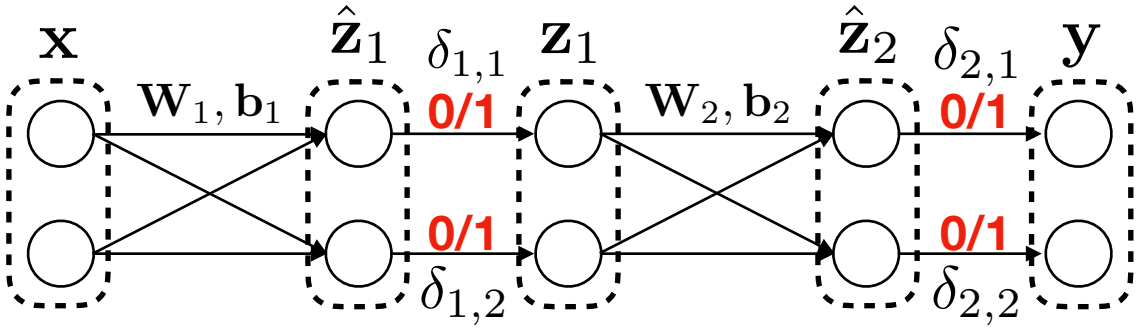
- *Compute upper bound by optimization using network relaxation*
- *Compute lower bound by sampling*



Search + Optimization

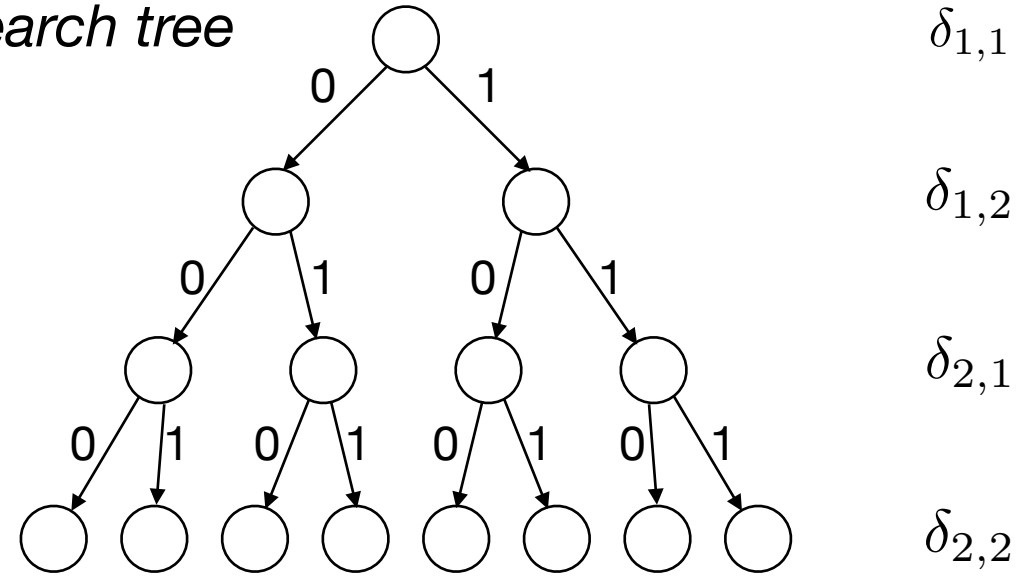
Planet

Reluplex



Search tree

Pruning by optimization-based feasibility check



Summary

Method Name	Activation	Approach	Input/Output	Completeness
ExactReach [46]	ReLU	Exact Reachability	HP/HP(bounded)[1]	✓
AI2 [14]	Piecewise Linear	Split and Join	HP/HP(bounded)[1]	×
MaxSens [44]	Any	Interval Arithmatic	HP/HP(bounded)[1]	×
NSVerify [23]	ReLU	Naive MILP	HR/PC[2]	✓
MIPVerify [38]	ReLU and Max	MILP with bounds	HR/PC[2]	✓
ILP [4]	ReLU	Iterative LP	HR/PC[2]	×
Duality [12]	Any	Lagrangian Relaxation	HR(uniform)/HS	×
ConvDual [43]	ReLU	Convex Relaxation	HR(uniform)/HS	×
Certify [31]	Differentiable	Semidefinite Relaxation	HR/HS	×
Fast-Lin [41]	ReLU	Network Relaxation	HR/HS	×
Fast-Lip [41]	ReLU	Lipschitz Estimation	HR/HS	×
ReluVal [40]	ReLU	Symbolic Interval	HR/HR	✓
DLV [18]	Any	Search in Hidden Layers	HR/HR(1-D)[3]	✓*
Sherlock [10]	ReLU	Local and Global Search	HR/HR(1-D)[3]	×
BaB [7]	Piecewise Linear	Branch and Bound	HR/HR(1-D)[3]	×
Planet [13]	Piecewise Linear	Satisfiability (SAT)	HR/PC[2]	✓
Reluplex [20]	ReLU	Simplex	HR/PC[2]	✓

NeuralVerification.jl

github.com/sisl/NeuralVerification.jl

Testing	Coverage	Documentation
build passing	coverage 88%	docs latest

NeuralVerification.jl

This library contains implementations of various methods to soundly verify deep neural networks. In general, we verify whether a neural network satisfies certain input-output constraints. The verification methods are divided into five categories:

- *Reachability methods:* ExactReach, MaxSens, Ai2,
- *Primal optimization methods:* NSVerify, MIPVerify, ILP
- *Dual optimization methods:* Duality, ConvDual, Certify
- *Search and reachability methods:* ReluVal, DLV, FastLin, FastLip
- *Search and optimization methods:* Sherlock, BaB, Planet, Reluplex

NeuralVerification.jl

Choose a solver

```
using NeuralVerification

solver = BaB()
```

Set up the problem

```
nnet = read_nnet("examples/networks/small_nnet.nnet")
input_set  = Hyperrectangle(low = [-1.0], high = [1.0])
output_set = Hyperrectangle(low = [-1.0], high = [70.0])
problem = Problem(nnet, input_set, output_set)
```

Solve

```
julia> result = solve(solver, problem)
CounterExampleResult(:violated, [1.0])

julia> result.status
:violated
```

Comparison

Algorithm	small_nnet	mnist1	mnist2	mnist3	mnist4	acas
NSVerify	0.000437805	0.561558217	1.23185184	0.234906589	-	-
MIPVerify	0.422803839	0.236930468	1.055669556	46.36556919	-	-
ILP	0.423599637	0.374052375	0.65545287	0.951015935*	4.866014534*	0.108586473*
convDual	0.001725546	0.374052375	0.02184014	0.001638679	0.005836024	0.002674586
Duality	0.001336414*	21.60260384*	130.7149674*	37.3058392*	52.87126159*	0.020148564*
FastLin	2.088460281	0.477470278	0.002369427	0.007551539	0.01564011	0.166508046**
FastLip	9.179938173	0.130989072	0.059900531	0.051348471**	0.091895366**	0.001703324**
MIPVerify	9.940695891	0.256747563	1.23054911	46.77128226	-	-
ILP	0.347950783	0.367925285	0.665364297	0.890090777*	4.664959506*	0.00834114*
Planet	0.000247003	0.211006714	0.301637916	0.186482861	-	-
Reluplex	0.012245108	125.0347088	62582.48883	-	-	2952.784643
Reluval	0.000031296	0.646829741	0.008979679	0.008030365	0.824760803*	0.196084517*

Conclusion

- A unified mathematical framework was introduced to verify satisfiability of a neural network given certain input and output constraints.
- Three basic verification methods were identified: reachability, optimization, and search.
- Trade-off between completeness of a verification algorithm and its scalability.
- Pedagogical implementations of all these methods were provided in Julia.

Future Directions

- Beyond piece-wise linear activations
- Beyond feedforward structures - RNNs
- Completeness vs. scalability
- Beyond universal approximation theorem (what is exactly the function space for a given network topology?)

Acknowledgement

- This work is partially supported by the Center for Automotive Research at Stanford (CARS).
- We would like to thank many of the authors of the referenced papers for their help in clarifying their algorithms and reviewing early drafts of this survey: Weiming Xiang, Taylor Johnson, Hoang-Dung Tran, Martin Vechev, Gagandeep Singh, Alessio Lomuscio, Michael Akintunde, Osbert Bastani, Zico Kolter, Shiqi Wang, Huan Zhang, Xiaowei Huang, Rudy Bunel, Reudiger Ehlers, and Guy Katz.
- We would also like to thank Christian Schilling and Marcelo Forets, the authors of LazySets.jl, for their implementation support.