Delta-Decision Procedures for Exists-Forall Problems over the Reals

Soonho Kong[1(✉)], Armando Solar-Lezama[2], and Sicun Gao[3]

[1] Toyota Research Institute, Cambridge, USA
soonho.kong@tri.global
[2] Massachusetts Institute of Technology, Cambridge, USA
asolar@csail.mit.edu
[3] University of California, San Diego, USA
sicung@ucsd.edu

Artifact Evaluation: https://dreal.github.io/CAV18

Motivation

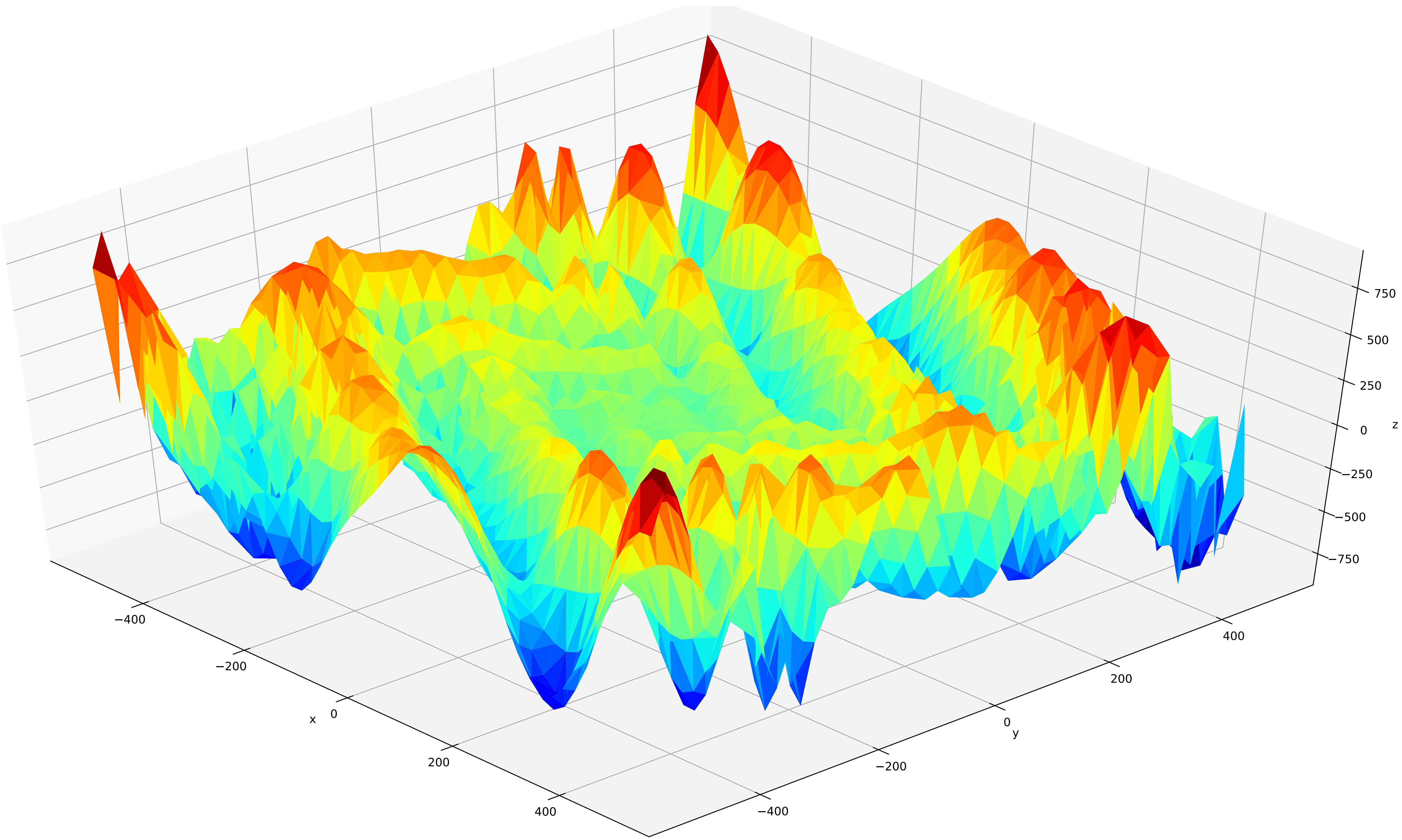
$$\exists x \,.\, \forall y \,.\, \varphi(x, y)$$

Motivation

$$\exists x \,.\, \forall y \,.\, \varphi(x, y)$$

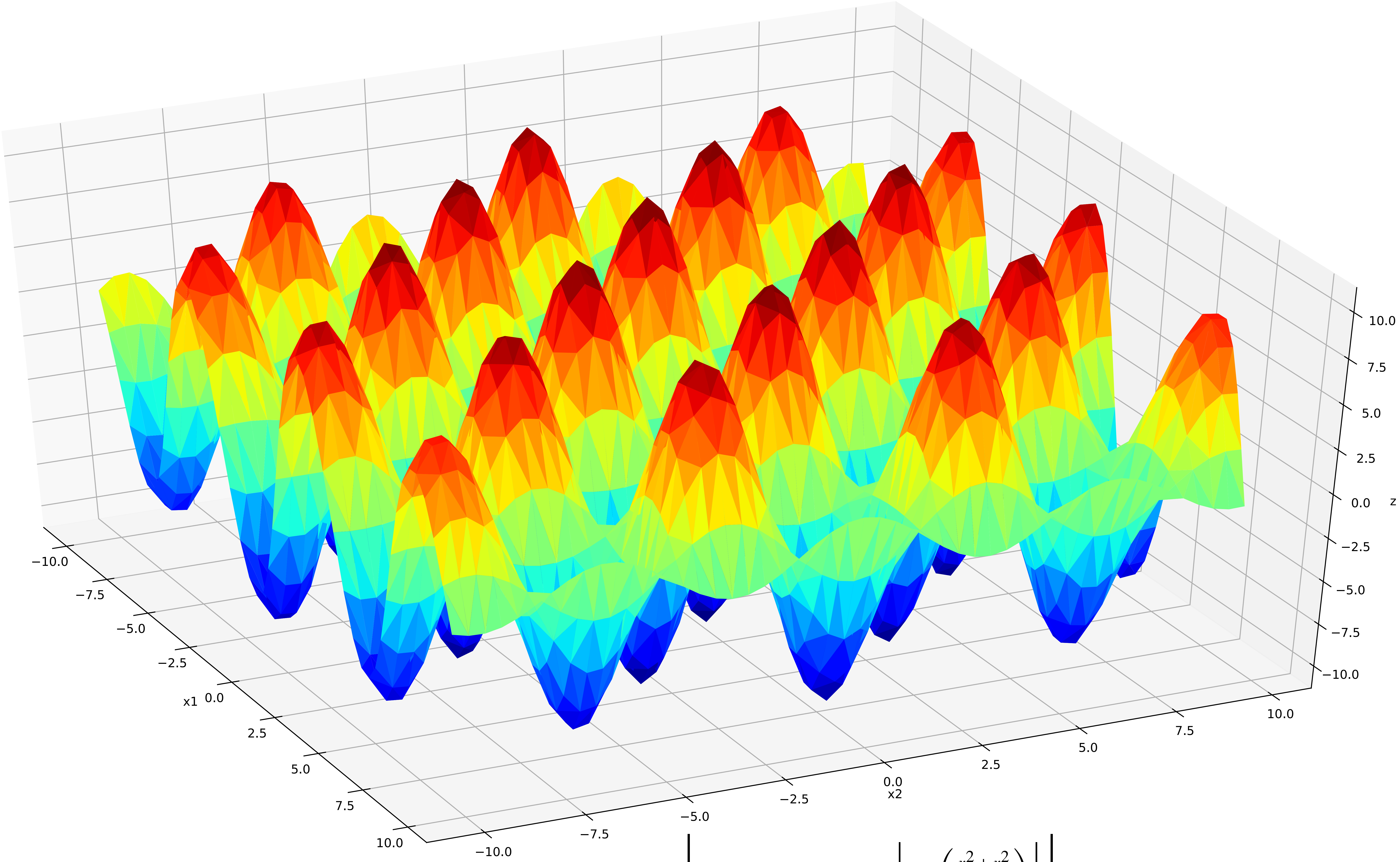
- **Optimization**
    - Global
    - Non-linear / Non-convex
    - Constrained

Motivation



Eggholder Function $f(x,y) = -(y+47)\sin\left(\sqrt{\left|\frac{x}{2}+(y+47)\right|}\right) - x\sin\left(\sqrt{|x-(y+47)|}\right) \quad -512 \le x, y \le 512$

Motivation



Testtube Holder Function $f(x_1, x_2) = -4 \left| \sin(x_1)\cos(x_2) e^{\left|\cos\left(\frac{x_1^2 + x_2^2}{200}\right)\right|} \right|$ $-10 \le x_1, x_2 \le 10$

Motivation

$$\exists x \,.\, \forall y \,.\, \varphi(x, y)$$

- **Optimization**
  - Global
  - Non-linear / Non-convex
  - Constrained
- **Synthesis**
  - Program
  - Controller (e.g. Lyapunov function, Barrier function)

Simulation-guided Lyapunov Analysis for Hybrid Dynamical Systems

James Kapinski*
Toyota Technical Center

Jyotirmoy V.Deshmukh
Toyota Technical Center

Sriram Sankaranarayanan
Univ. of Colorado

Nikos Aréchiga
Carnegie Mellon University

ABSTRACT

Lyapunov functions are used to prove stability and to obtain performance bounds on system behaviors for nonlinear and hybrid dynamical systems, but discovering Lyapunov functions is a difficult task in general. We present a technique for discovering Lyapunov functions and barrier certificates for nonlinear and hybrid dynamical systems using a search-based approach. Our approach uses concrete executions, such as those obtained through simulation, to formulate a series of linear programming (LP) optimization problems; the solution to each LP creates a candidate Lyapunov function. Intermediate candidates are iteratively improved using a global optimizer guided by the Lie derivative of the candidate Lyapunov function. The analysis is refined using counterexamples from a Satisfiability Modulo Theories (SMT) solver. When no counterexamples are found, the soundness of the analysis is verified using an arithmetic solver. The technique can be applied to a broad class of nonlinear dynamical systems, including hybrid systems and systems with polynomial and even transcendental dynamics. We present several examples illustrating the efficacy of the technique, including two automotive powertrain control examples.

ber of simulations to gain confidence in system correctness. Formal techniques provide better guarantees but are often intractable for large, complex system designs. On the other hand, simulation-based methods work well for systems of arbitrary complexity but cannot be used for verification. In this paper, we describe our effort to bridge this gap by formally addressing prominent analysis problems for hybrid systems while leveraging data obtained from simulations. In particular, we address the problems of proving stability of a system, characterizing performance bounds by computing forward invariant sets, and proving system safety by automatically synthesizing barrier certificates.

It is well-known that each of these problems can be effectively addressed if the designer is able to supply a function v that satisfies the following *Lyapunov conditions* in a given region of interest: (1) v is positive definite, and (2) the Lie derivative of v along the system dynamics is negative (semi-)definite. While the search for a Lyapunov function is widely recognized as a hard problem, sum-of-squares (SoS) optimization-based techniques have been used successfully to obtain Lyapunov functions for systems with polynomial [17, 21] , nonpolynomial [16], and hybrid [18] dynamics. While these techniques have mature tool support

Simulation-guided Lyapunov Analysis for Hybrid Dynamical Systems

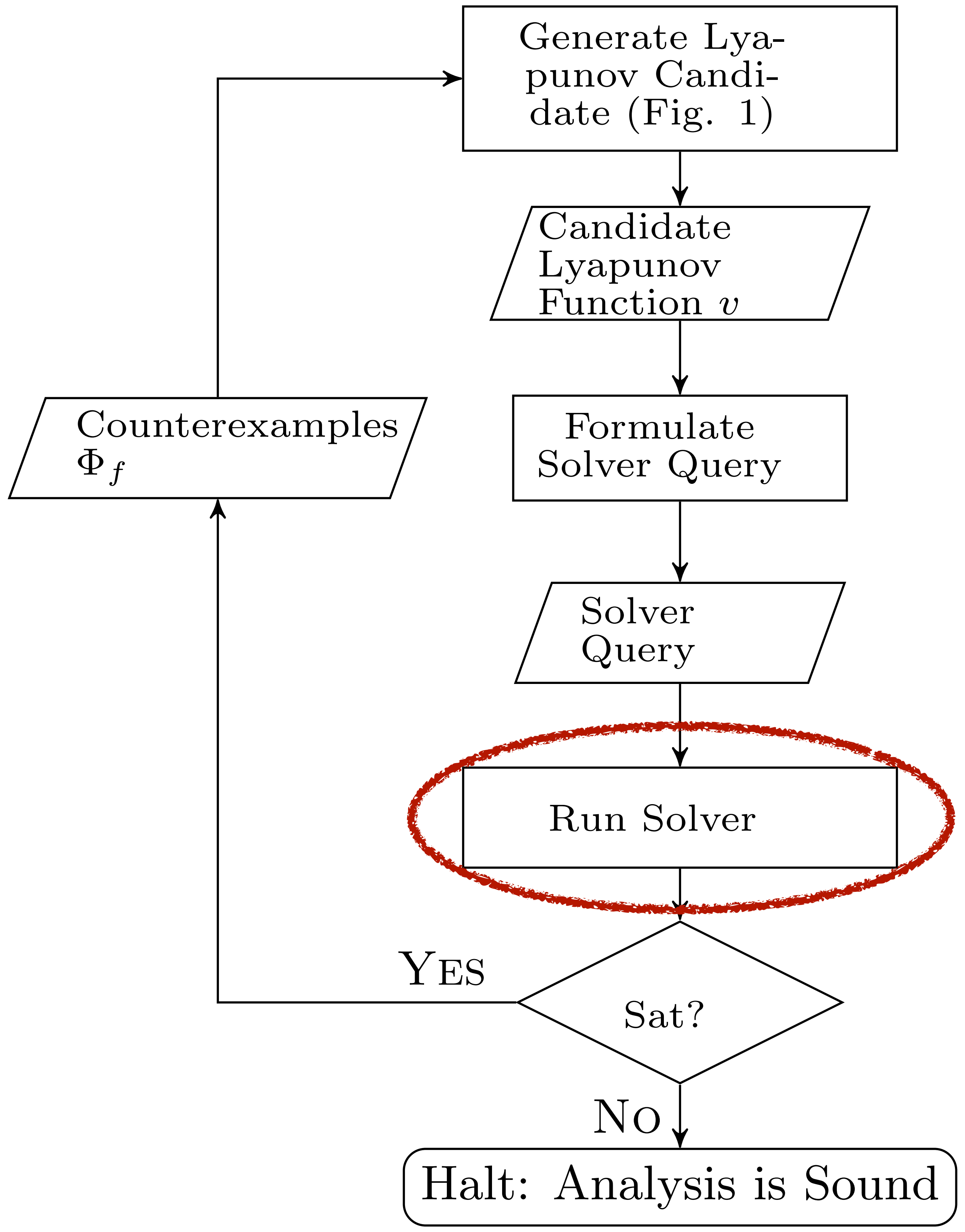


Figure 2: Incorporating solver technologies to verify soundness of the Lyapunov anlyses.

ABSTRACT

Lyapunov functi
performance bo
hybrid dynamic
tions is a difficu
for discovering
for nonlinear an
based approach
such as those o
series of linear
the solution to e
tion. Intermedia
a global optimiz
date Lyapunov f
terexamples fro
solver. When no
of the analysis
technique can b
namical systems
polynomial and
several examples illustrating the efficacy of the technique,
including two automotive powertrain control examples.

system correctness.
ntees but are often
signs. On the other
well for systems of
ed for verification.
bridge this gap by
problems for hybrid
rom simulations. In
proving stability of
unds by computing
em safety by auto-
s.
roblems can be ef-
e to supply a func-
unov conditions in
sitive definite, and
ystem dynamics is
rch for a Lyapunov
l problem, sum-of-
ques have been used
successfully to obtain Lyapunov functions for systems with
polynomial [17, 21] , nonpolynomial [16], and hybrid [18] dy-
namics. While these techniques have mature tool support

Reasoning about Safety of Learning-Enabled Components in Autonomous Cyber-physical Systems

Cumhur Erkan Tuncali[1] James Kapinski[1] Hisahiro Ito[1] Jyotirmoy V. Deshmukh[2]
[1]Toyota Research Institute of North America, Ann Arbor, MI, USA
[2] University of Southern California, Los Angeles, CA, USA
{cumhur.tuncali, jim.kapinski, hisahiro.ito}@toyota.com, jyotirmoy.deshmukh@usc.edu

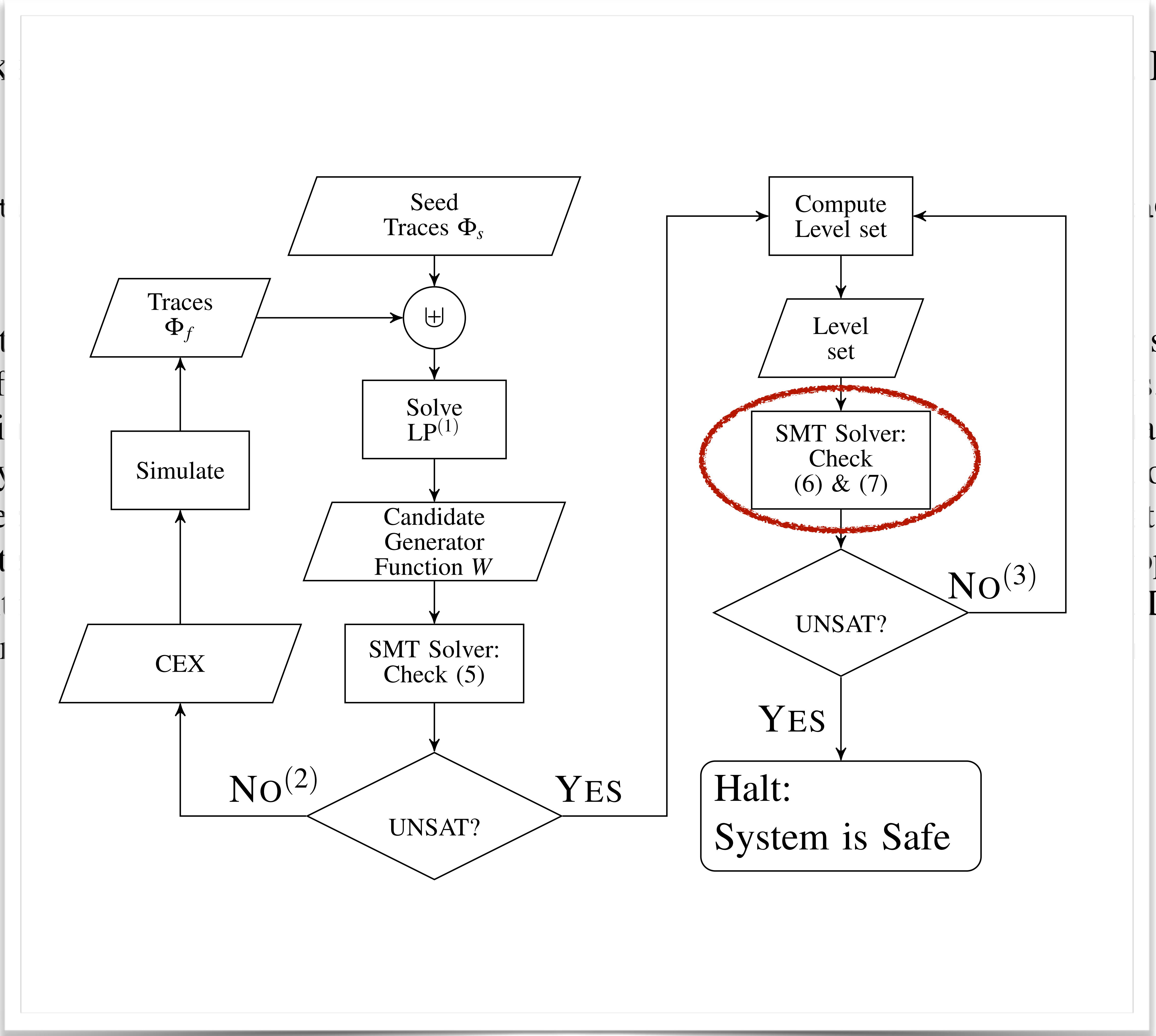
We present a simulation-based approach for generating barrier certificate functions for safety verification of cyber-physical systems (CPS) that contain neural network-based controllers. A linear programming solver is utilized to find a candidate generator function from a set of simulation traces obtained by randomly selecting initial states for the CPS model. A level set of the generator function is then selected to act as a barrier certificate for the system, meaning it demonstrates that no unsafe system states are reachable from a given set of initial states. The barrier certificate properties are verified with an SMT solver. This approach is demonstrated on a case study in which a Dubins car model of an autonomous vehicle is controlled by a neural network to follow a given path.

Reasoning about Safety of Learning-Enabled Components in Autonomous Cyber-physical Systems

Cumhur Erk ... Deshmukh[2]

{cumhur.t ... @usc.edu

We present ... safety ver-
ification of ... A linear
programmi ... tion traces
obtained by ... or function
is then sele ... no unsafe
system stat ... perties are
verified wi ... Dubins car
model of a

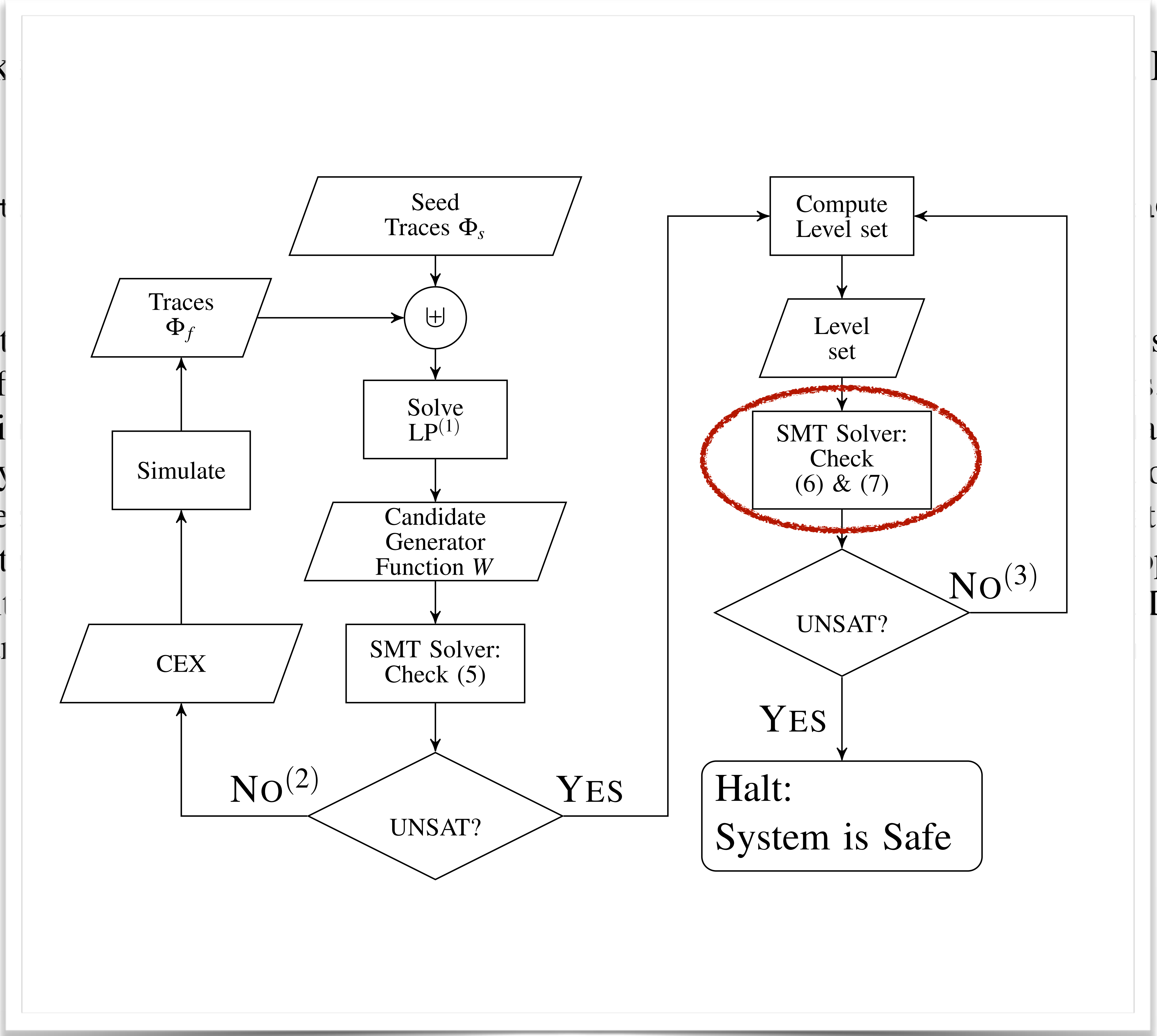


Reasoning about Safety of Learning-Enabled Components in Autonomous Cyber-physical Systems

Cumhur Erk ... Deshmukh[2]

{cumhur.t ... @usc.edu

We present ... safety ver-
ification of ... A linear
programmi ... tion traces
obtained by ... or function
is then sele ... no unsafe
system stat ... perties are
verified wi ... Dubins car
model of a



Can we handle the whole synthesis problem in a solver?

Decision Problem over the Real

Given an arbitrary first-order logic formula with **computable real** functions

$$\varphi = \exists^{[l_1,u_1]} x_1 \forall^{[l_2,u_2]} x_2 \ldots Q_n^{[l_n,u_n]} x_n . \bigwedge_i \left(\bigvee_j f_{i,j}(\overrightarrow{x}) > 0 \vee \bigvee_k f_{i,k}(\overrightarrow{x}) \geq 0 \right)$$

decide whether ϕ is **satisfiable** or **not**.

Decision Problem over the Real

Given an arbitrary first-order logic formula with **computable real** functions

$$\varphi = \exists^{[l_1,u_1]}x_1 \forall^{[l_2,u_2]}x_2 \ldots Q_n^{[l_n,u_n]}x_n . \bigwedge_i \left(\bigvee_j f_{i,j}(\overrightarrow{x}) > 0 \vee \bigvee_k f_{i,k}(\overrightarrow{x}) \geq 0 \right)$$

decide whether ϕ is **satisfiable** or **not**.

Complexity results for the existential problems:

- **Doubly exponential lower bound** for fragment with only polynomials [Davenport 1988]

- **Undecidable** with “sine” [Tarski 1950s]

Delta-Decision Problem

Idea: Allow bounded **numerical errors** in logical decision

Instead of solving ϕ,

$$\varphi = \exists^{[l_1,u_1]} x_1 \forall^{[l_2,u_2]} x_2 \ldots Q_n^{[l_n,u_n]} x_n . \bigwedge_i \left(\bigvee_j f_{i,j}(\overrightarrow{x}) > 0 \vee \bigvee_k f_{i,k}(\overrightarrow{x}) \geq 0 \right)$$

decide the **delta-weakening** of ϕ defined as follows

$$\varphi^{-\delta} = \exists^{[l_1,u_1]} x_1 \forall^{[l_2,u_2]} x_2 \ldots Q_n^{[l_n,u_n]} x_n . \bigwedge_i \left(\bigvee_j f_{i,j}(\overrightarrow{x}) > -\delta \vee \bigvee_k f_{i,k}(\overrightarrow{x}) \geq -\delta \right)$$

Delta-Decision Problem

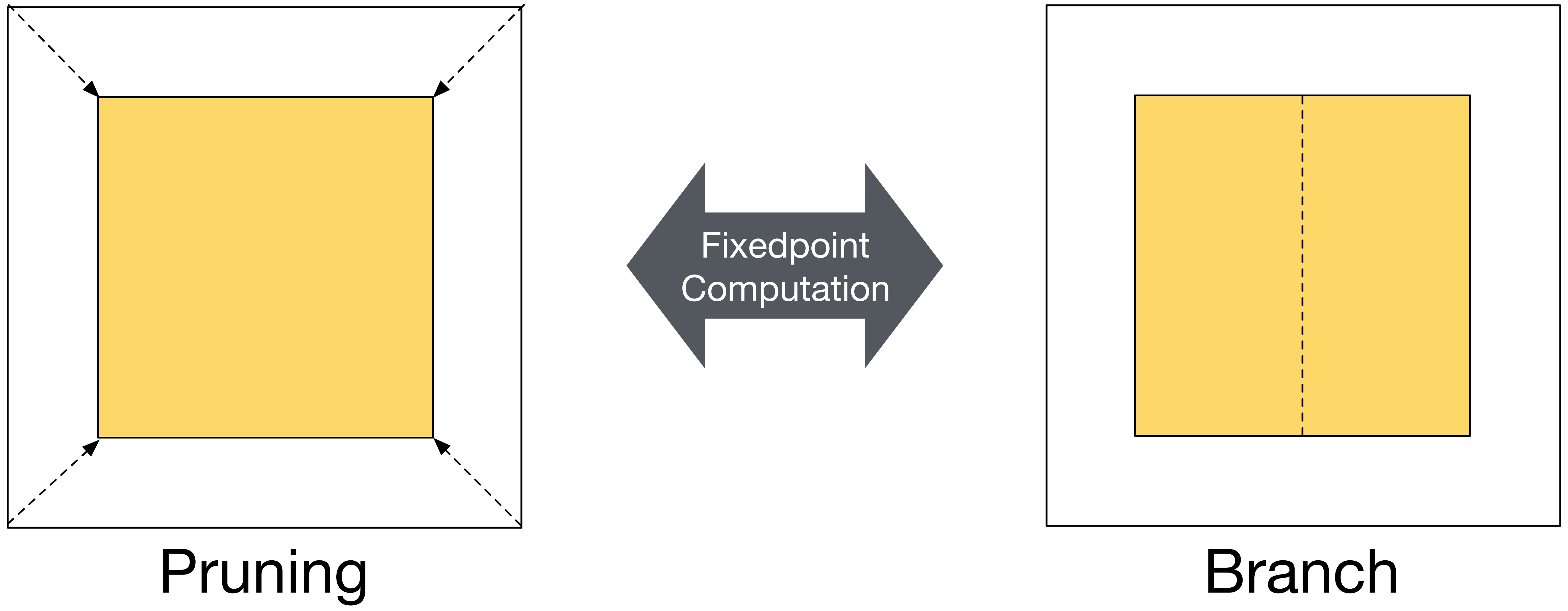
- Key Results from [LICS12]
  - The delta-decision problem is **decidable**.
  - The **complexity** of the problem is **not higher than Boolean logic** when considering P-computable functions.
    - Existential Problem (∃) → Σ_1^P (=NP)
    - Exists-forall Problem (∃∀) → Σ_2^P
    - ...

Algorithm: DPLL<ICP>

- SAT Solver: Find a conjunction of theory literals.

- Theory Solver: Checks if a given conjunction of theory literals is satisfiable under the theory.

  - ICP: Interval Constraint Propagation



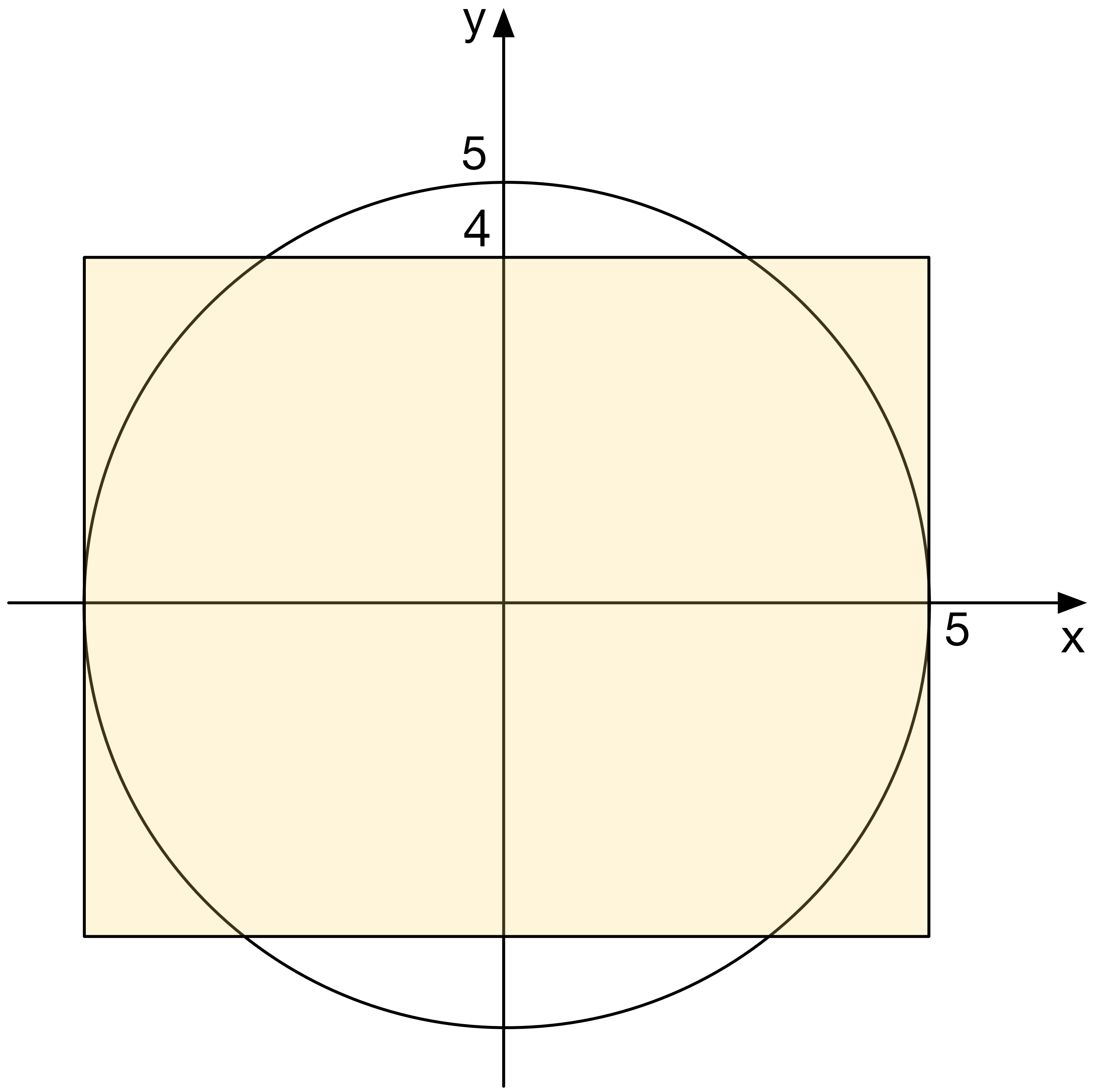
Reduce a search space without removing solutions

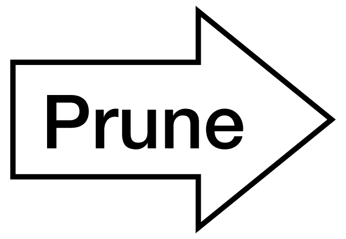
Partition a search space into two sub-spaces

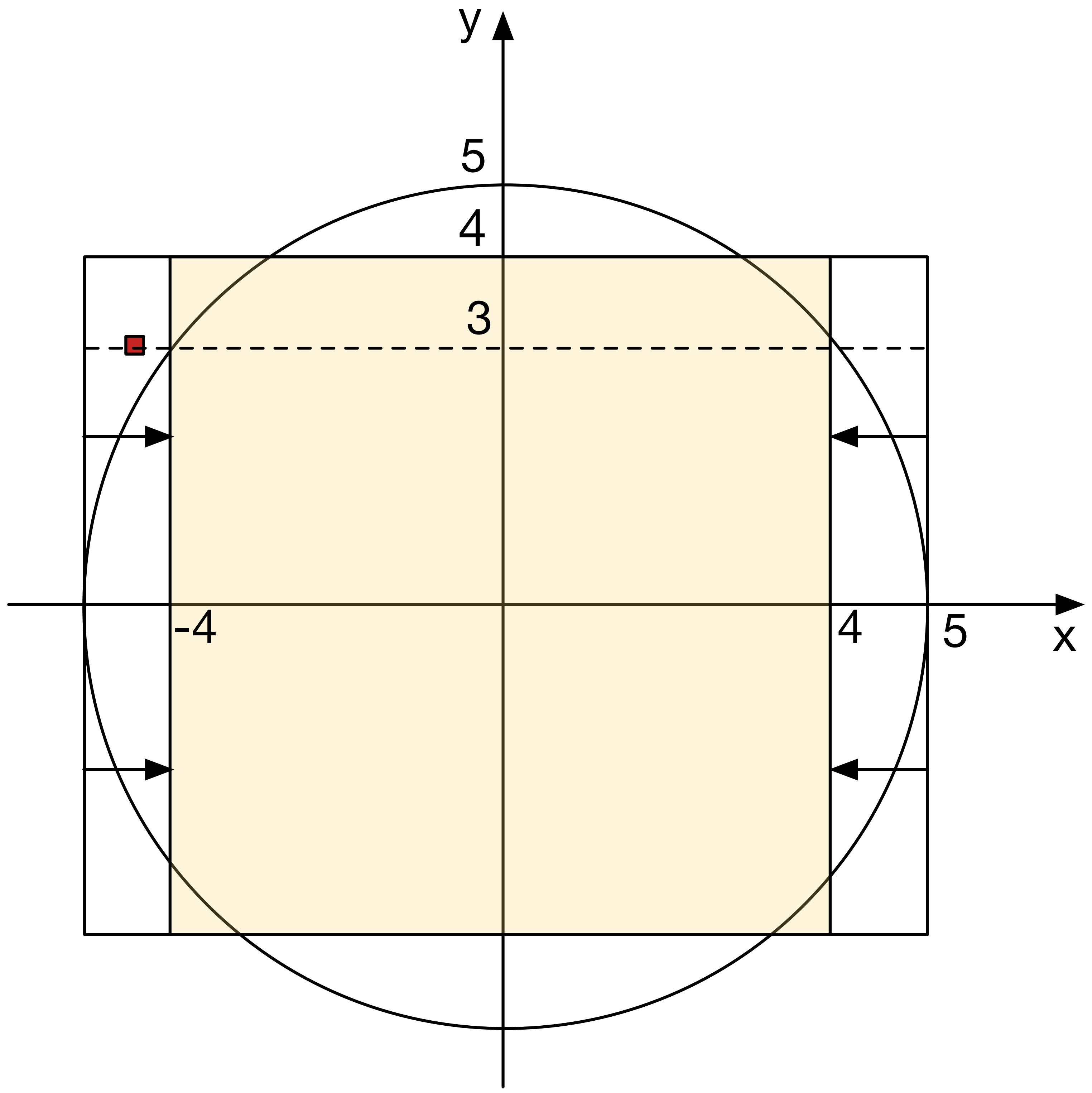
How to Design Delta-Decision Procedures for Exists-forall Problems?

⇒ How to Design a **pruning operator** for **forall** constraints

Example: $\exists^{[-5,5]} x \,.\, \forall^{[-4,4]} y \,.\, x^2 + y^2 \leq 5^2$





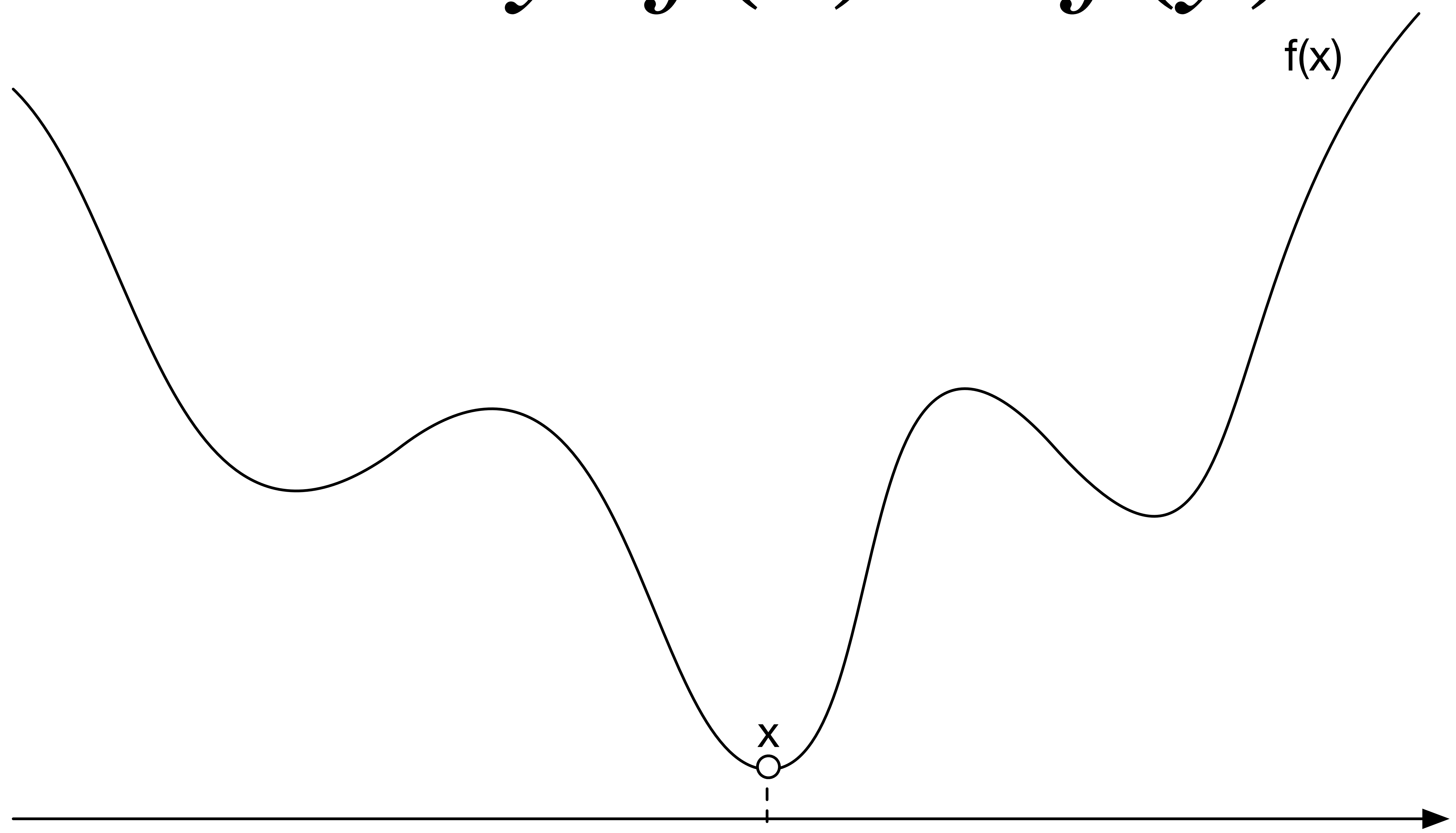


Simple Case: Unconstrained Global Optimization

$$\min f(x)$$

$$\Downarrow$$

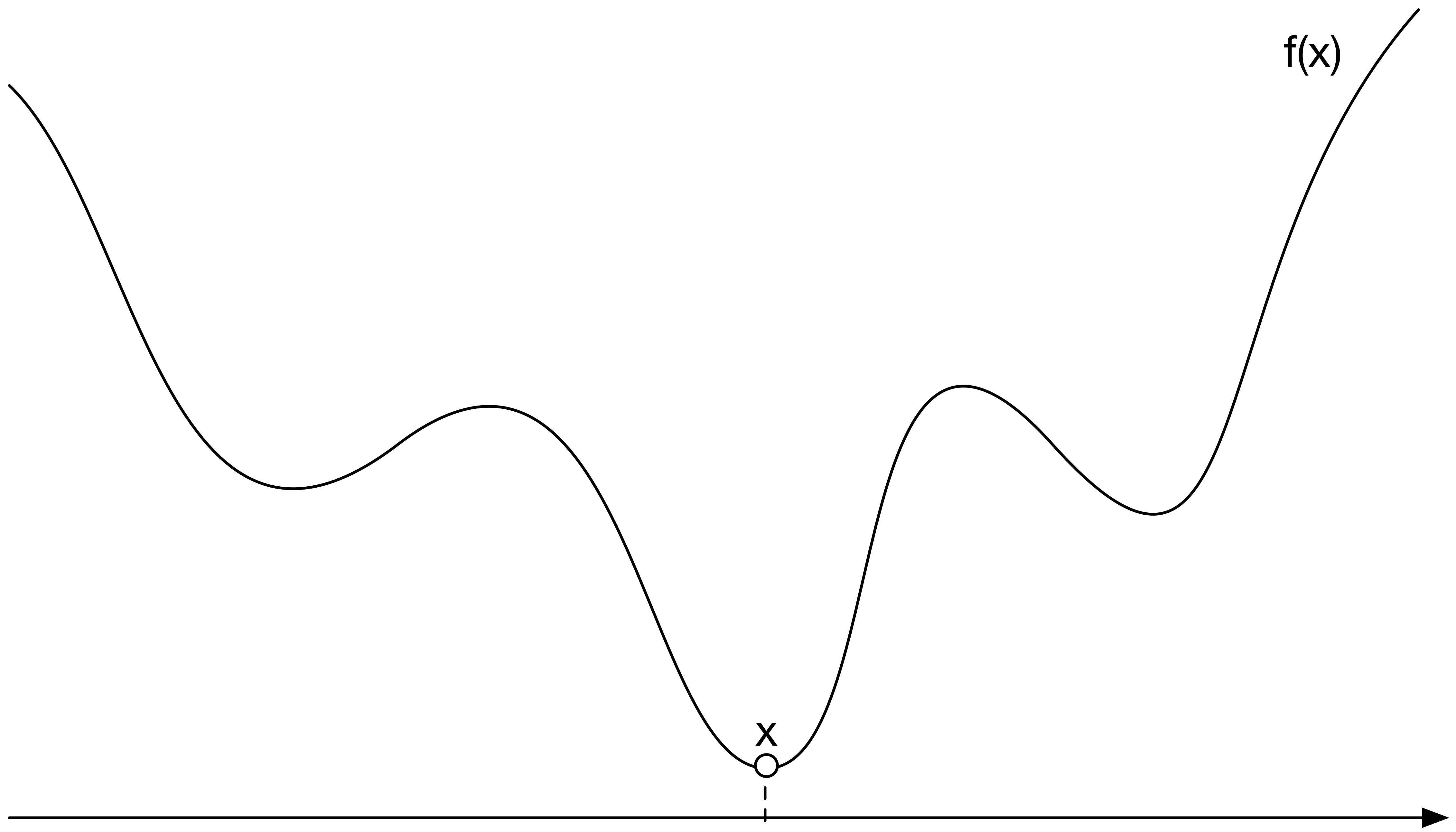
$$\exists x \,.\, \forall y \,.\, f(x) \leq f(y)$$



Simple Case: Unconstrained Global Optimization

$$\exists x \,.\, \forall y \,.\, f(x) \leq f(y)$$

Finding the **exact** global optimum is **undecidable** when we allow functions such as sin, cos.

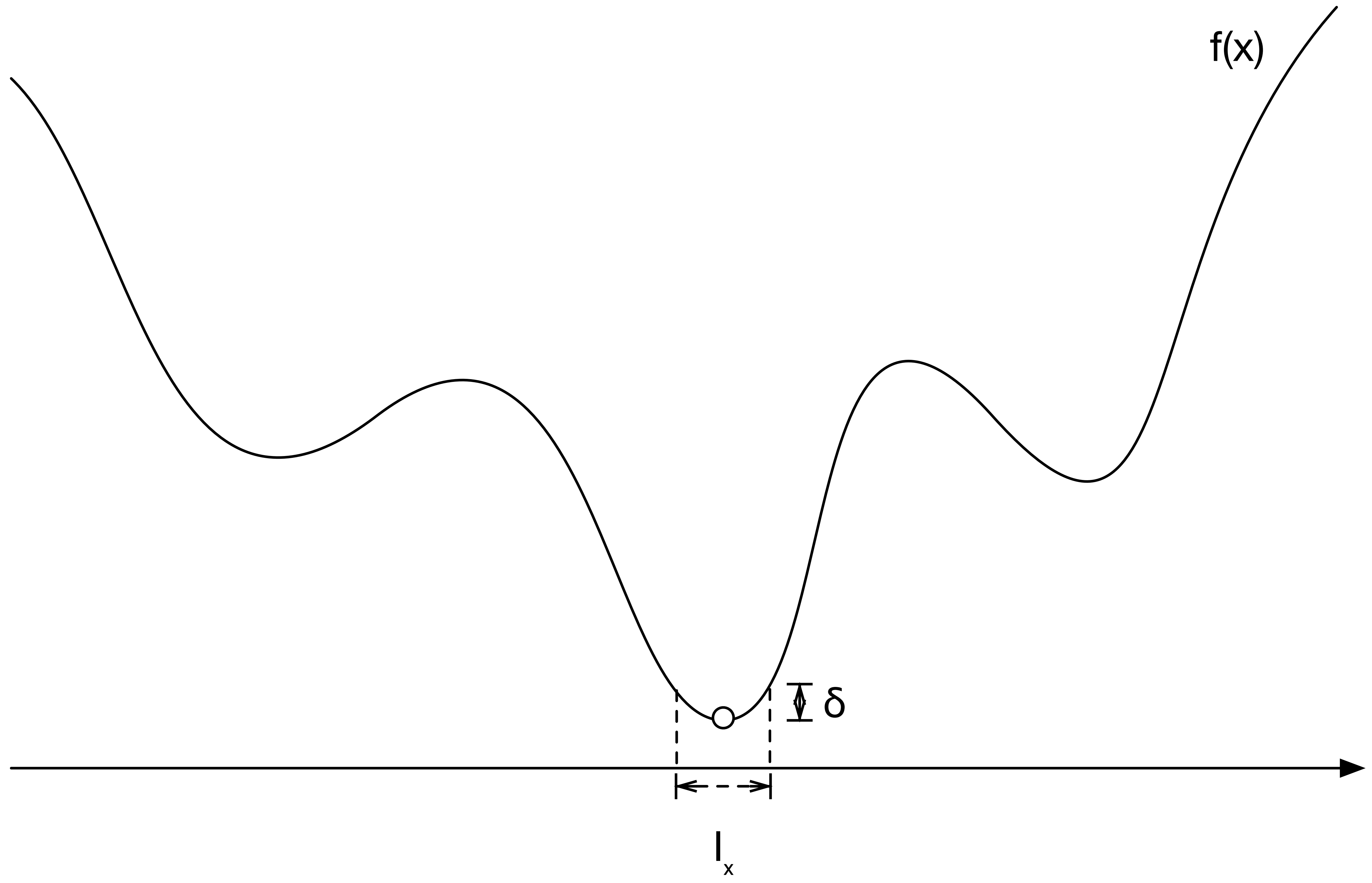


Simple Case:
Unconstrained Global Optimization

Instead, we want to find an interval I_x such that for all $x \in I_x$:

$$\forall y \,.\, f(x) \leq f(y) + \delta$$

Note that this problem is **decidable** (Σ_2^P)



Idea: Counterexample Refinement

Find a **counterexample** b such that for an a in I_x^1

$$f(a) > f(b)$$

and use it to reduce I_x^1 to I_x^2.

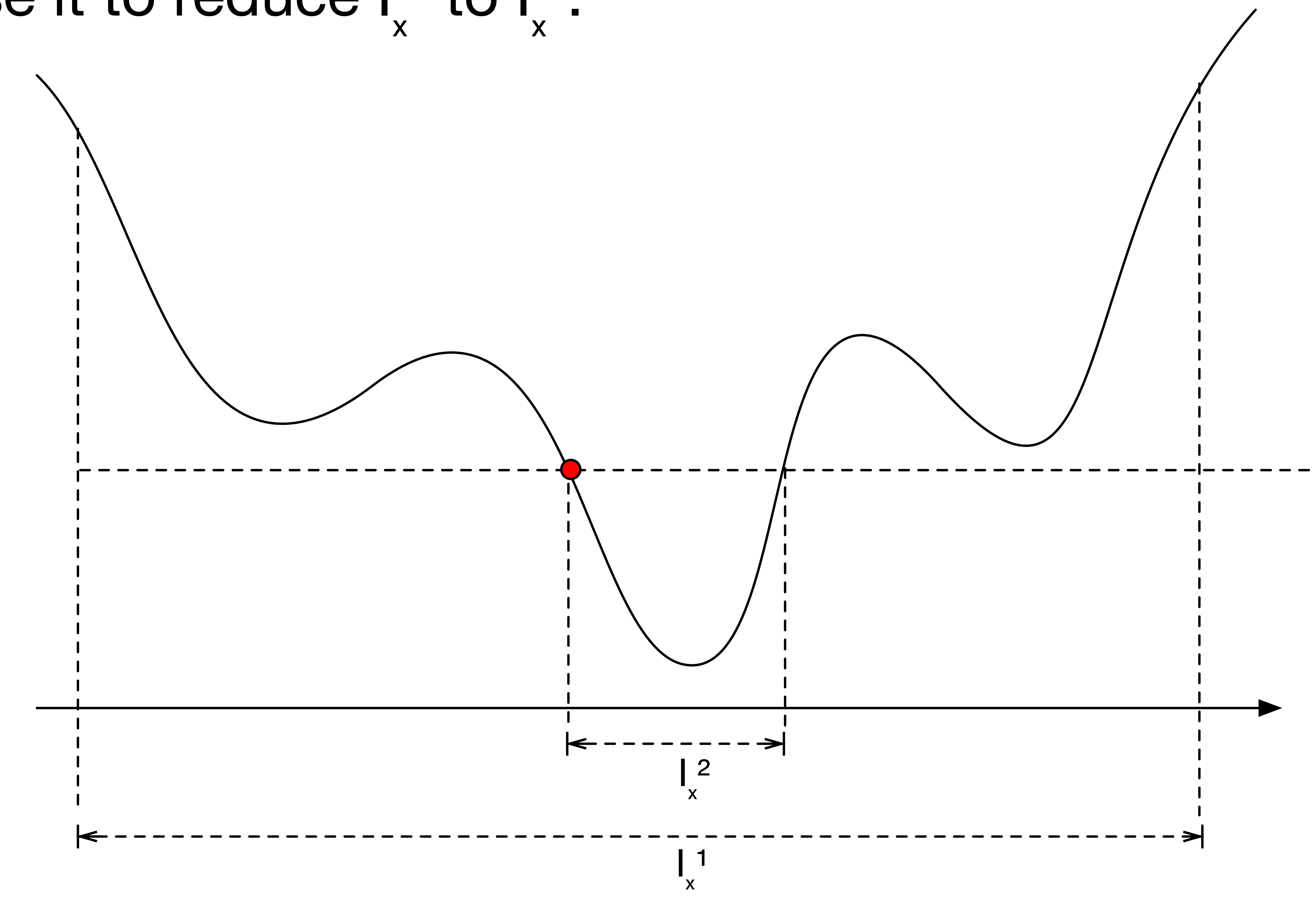
$$\exists x \in I_x^1 \,.\, \forall y \,.\, f(x) \leq f(y)$$

Instantiate **y = b**

Counterexample for y!

$$\exists x \in I_x^1 \,.\, f(x) \leq f(b)$$

Reuse the pruning operators for ∃-Problems.



Finding a Counterexample

$$f(x) > f(y)$$

How do we find such y?
Note that the problem is in general **undecidable** again.

We use **delta-decision solver** to find a counterexample:

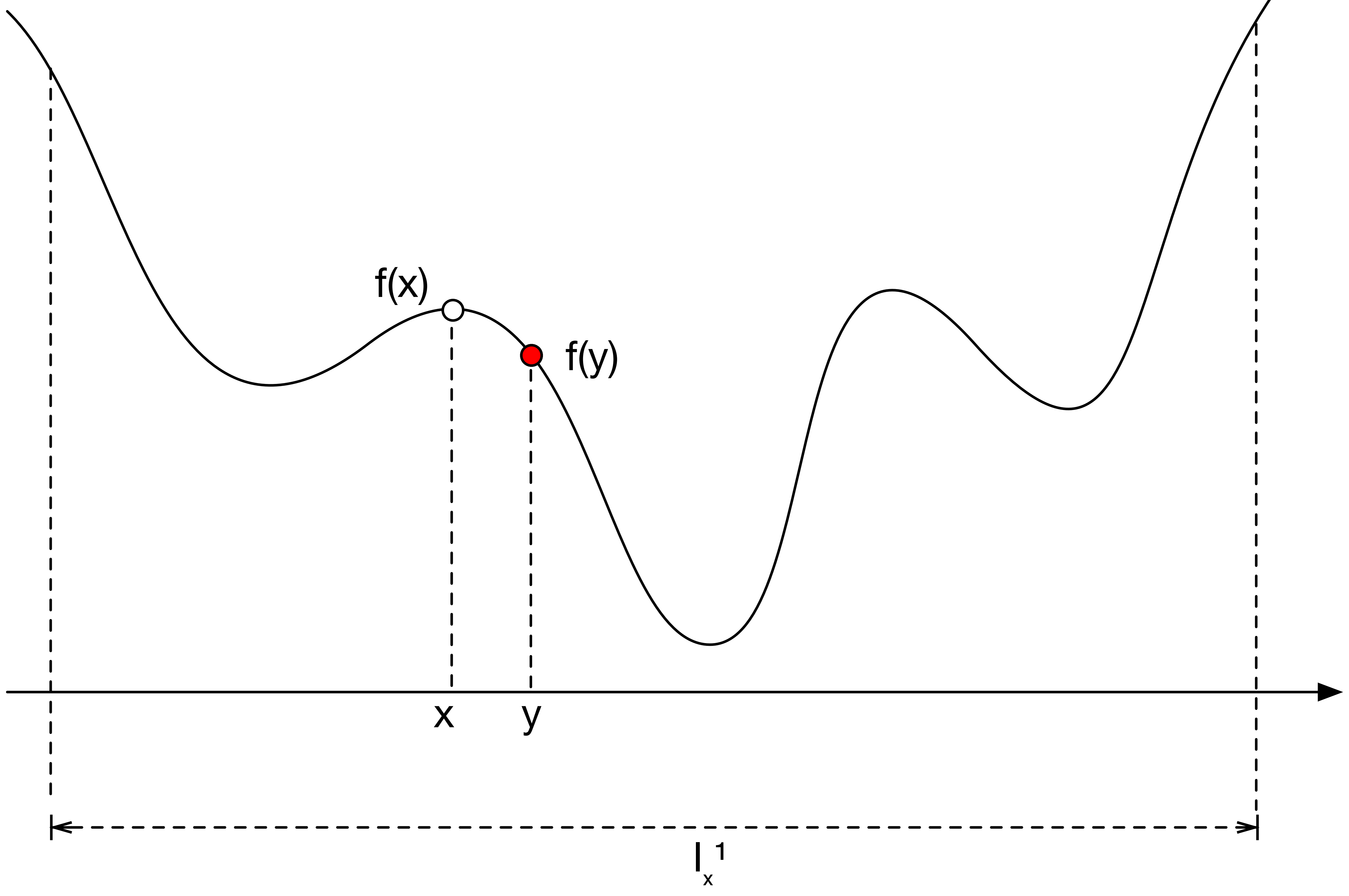
$$\text{Solve}(f(x) > f(y), \delta')$$

How to pick this?

Problem of spurious counterexamples

$\text{Solve}(f(x) > f(y), \delta')$ finds (x, y) such that:

$$f(x) > f(y)$$

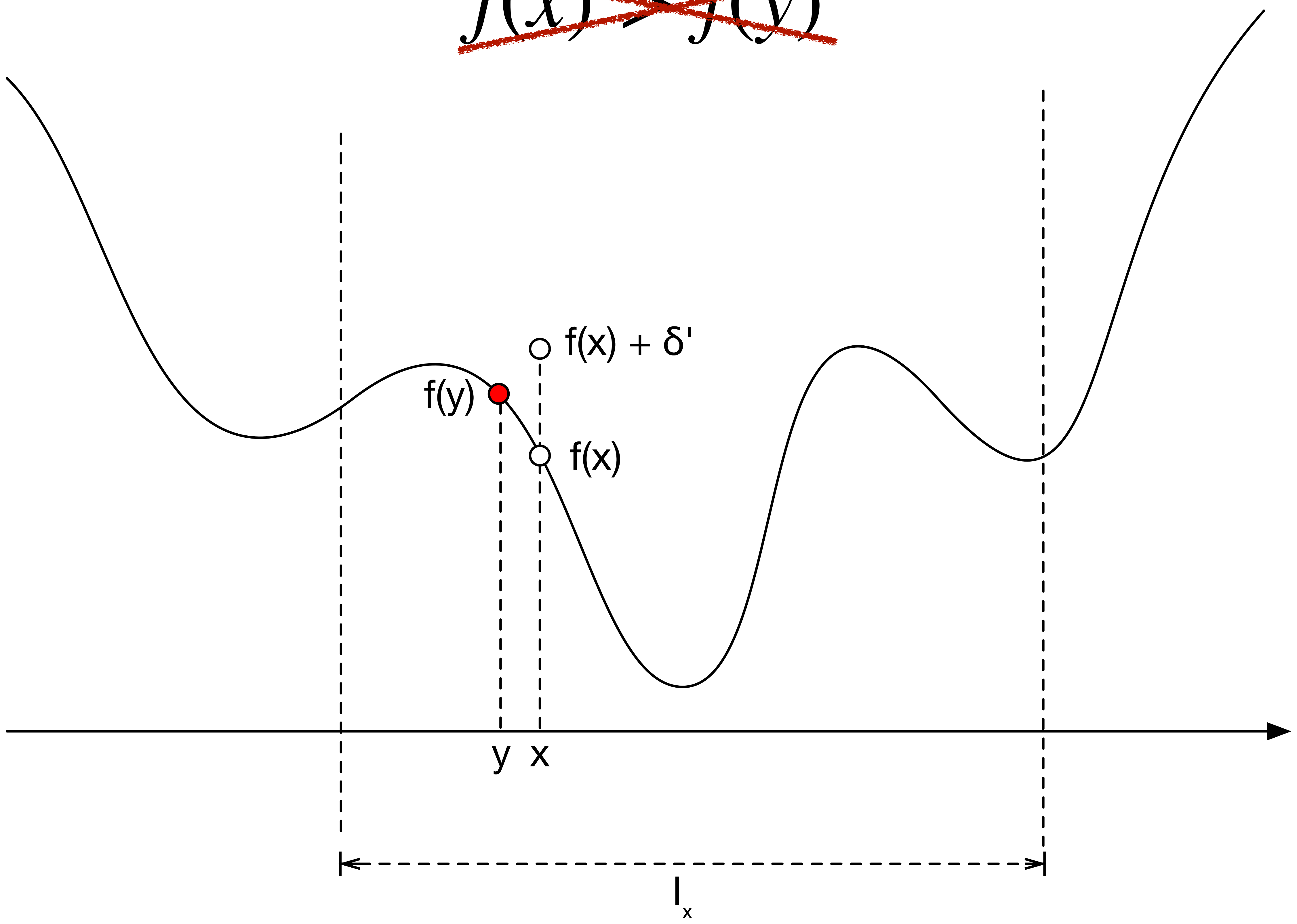


Problem of spurious counterexamples

$\text{Solve}(f(x) > f(y), \delta')$ finds (x, y) such that:

$$f(x) + \delta' > f(y)$$

$$\cancel{f(x) > f(y)}$$



Spurious counterexamples give **NO** pruning power.

Solution: Double-sided Error Control

Key Idea

Strengthen the counterexample query to **avoid** spurious counterexamples.

Solution: Double-sided Error Control

Instead of solving the following to counterexample:

$$\mathrm{Solve}(f(x) > f(y), \delta')$$

Solve the following **ε-strengthened the CE query**:

$$\mathrm{Solve}(f(x) > f(y) + \epsilon, \delta')$$

Q: Given δ, how to pick **δ'** and **ε**?

Solution: Double-sided Error Control

Q: Given δ, how to pick **δ'** and **ε**?

$$\mathrm{Solve}(f(x) > f(y) + \epsilon, \delta')$$

UNSAT CASE: It shows that

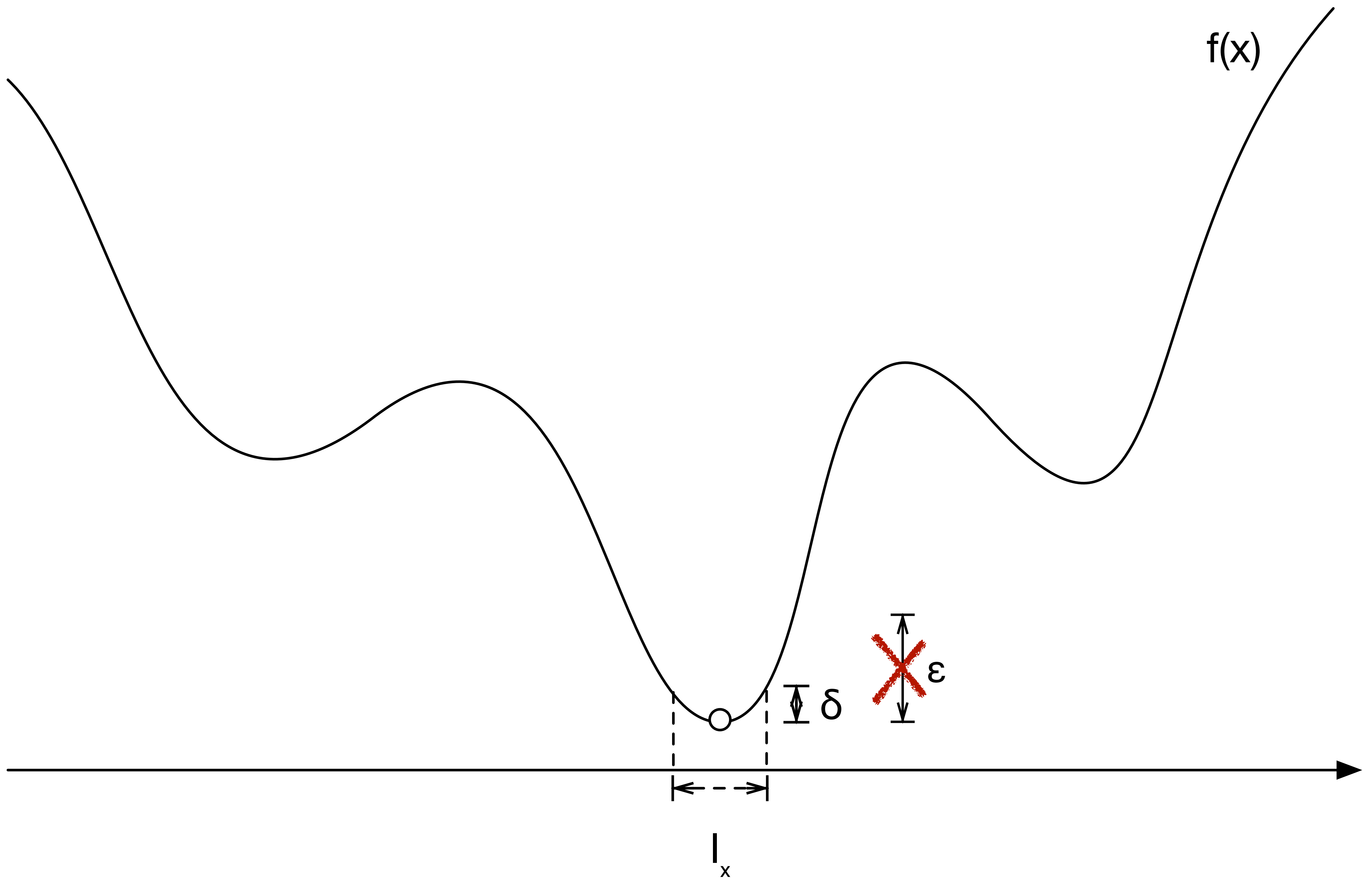
$$\forall y \,.\, f(x) \leq f(y) + \epsilon$$

Note that we wanted to satisfy:

$$\forall y \,.\, f(x) \leq f(y) + \delta$$

So we have:

$$\epsilon < \delta$$



Solution: Double-sided Error Control

Q: Given δ, how to pick **δ'** and **ε**?

$$\text{Solve}(f(x) > f(y) + \epsilon, \delta')$$

δ-SAT CASE: We have (x, y) such that:

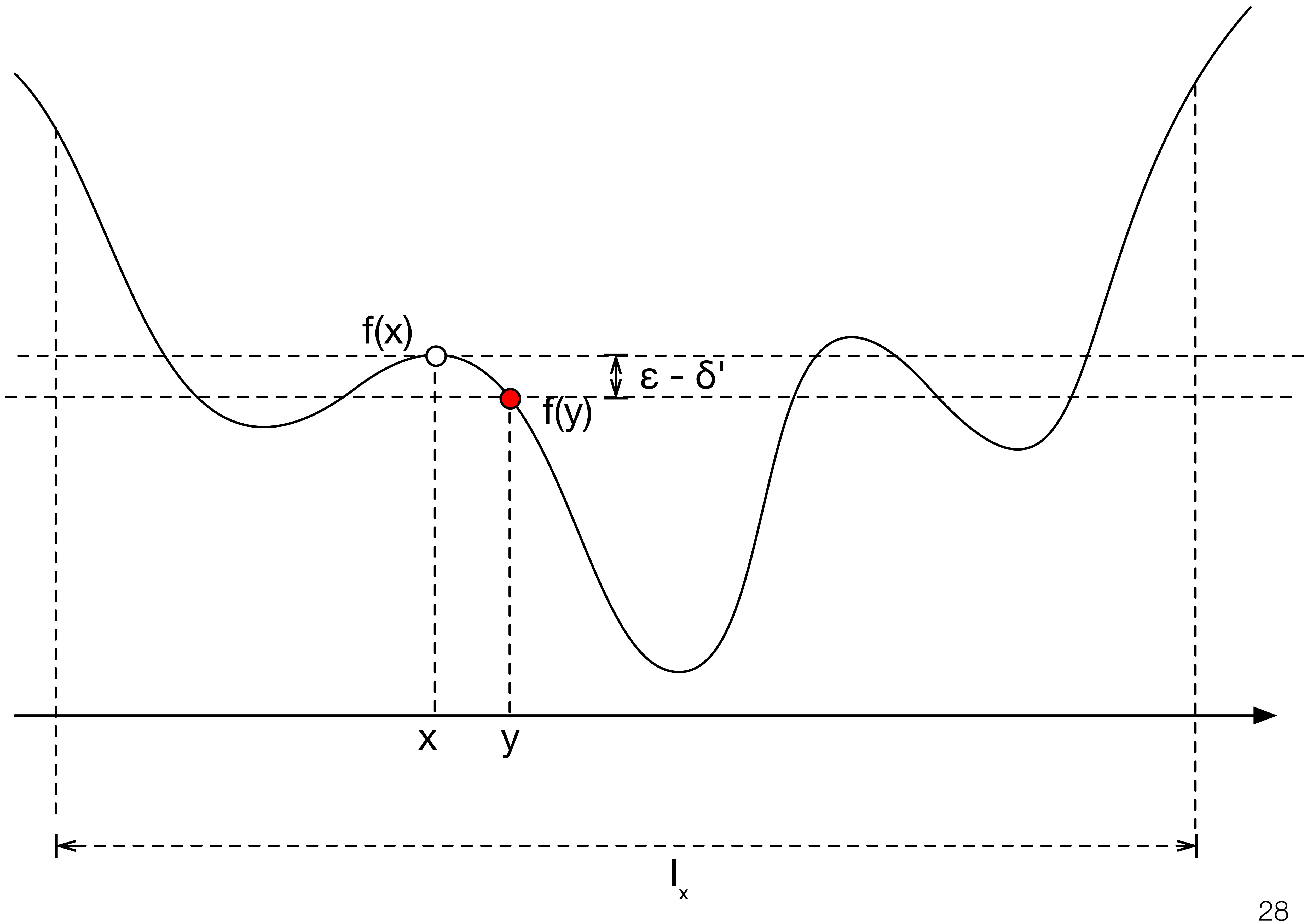
$$f(x) > f(y) + (\epsilon - \delta')$$

Since y should be a **true counterexample**:

$$\epsilon - \delta' > 0$$

That is,

$$\delta' < \epsilon$$



Solution: Double-sided Error Control

Given δ, how to pick **δ'** and **ε**?

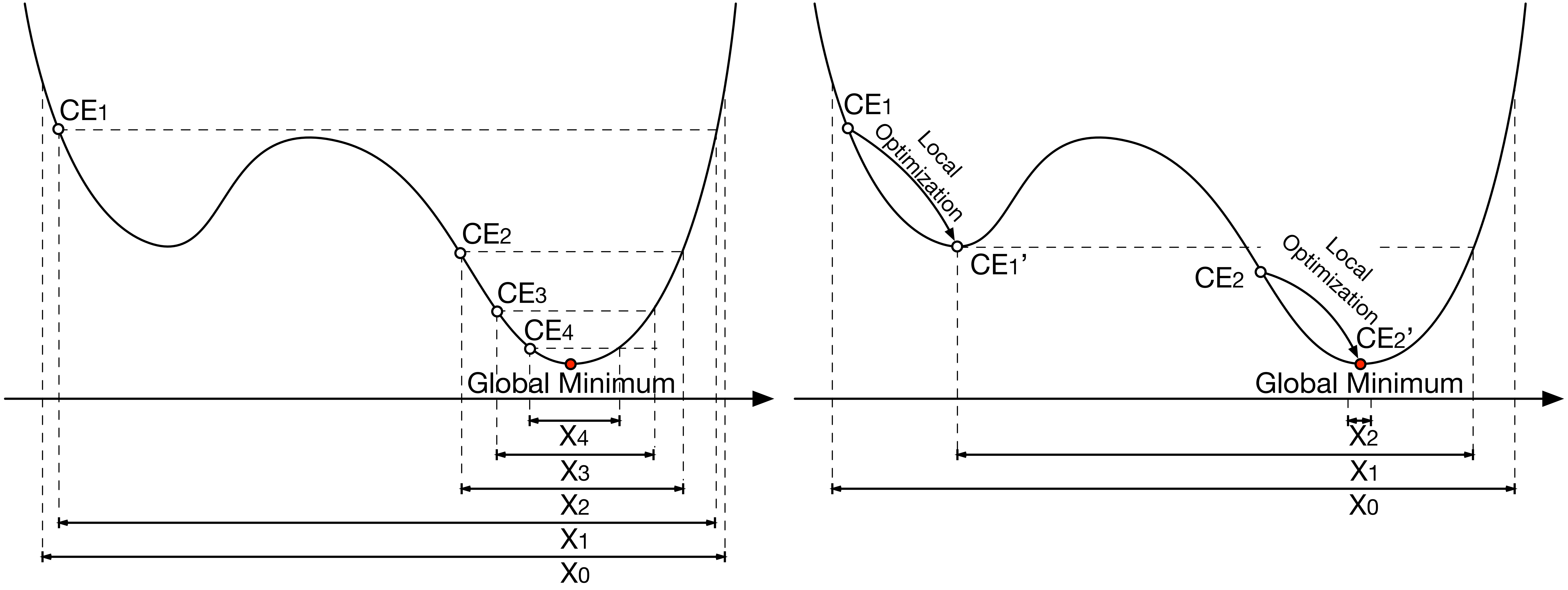
$$\delta' < \epsilon < \delta$$

Pruning Algorithm

Algorithm 2 $\forall$-Clause Pruning

function PRUNE(B_x, B_y, $\forall y \bigvee_{i=0}^{k} f_i(x,y) \geq 0$, δ', ε, δ)
    repeat
        $B_x^{\text{prev}} \leftarrow B_x$
        $\psi \leftarrow \bigwedge_i f_i(x,y) < 0$
        $\psi^{+\varepsilon} \leftarrow \mathsf{Strengthen}(\psi, \varepsilon)$
        $b \leftarrow \mathsf{Solve}(y, \psi^{+\varepsilon}, \delta')$ ▷ $0 < \delta' < \varepsilon < \delta$ should hold.
        if $b = \emptyset$ **then**
            return B_x ▷ No counterexample found, stop pruning.
        end if
        for $i \in \{0, ..., k\}$ **do**
            $B_i \leftarrow B_x \cap \mathsf{Prune}\Big(B_x, f_i(x,b) \geq 0\Big)$
        end for
        $B_x \leftarrow \bigsqcup_{i=0}^{k} B_i$
    until $B_x \neq B_x^{\text{prev}}$
    return B_x
end function

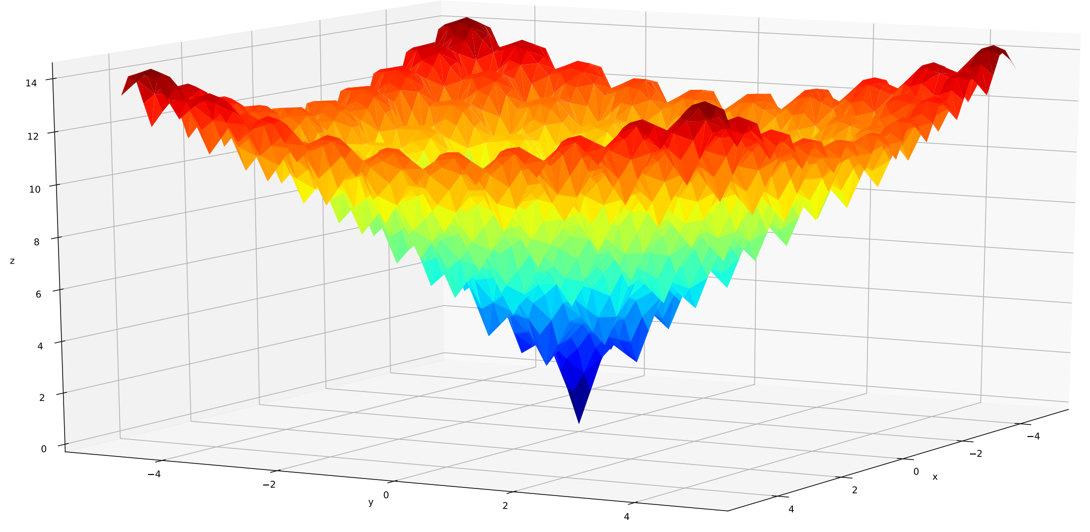
Local Optimization



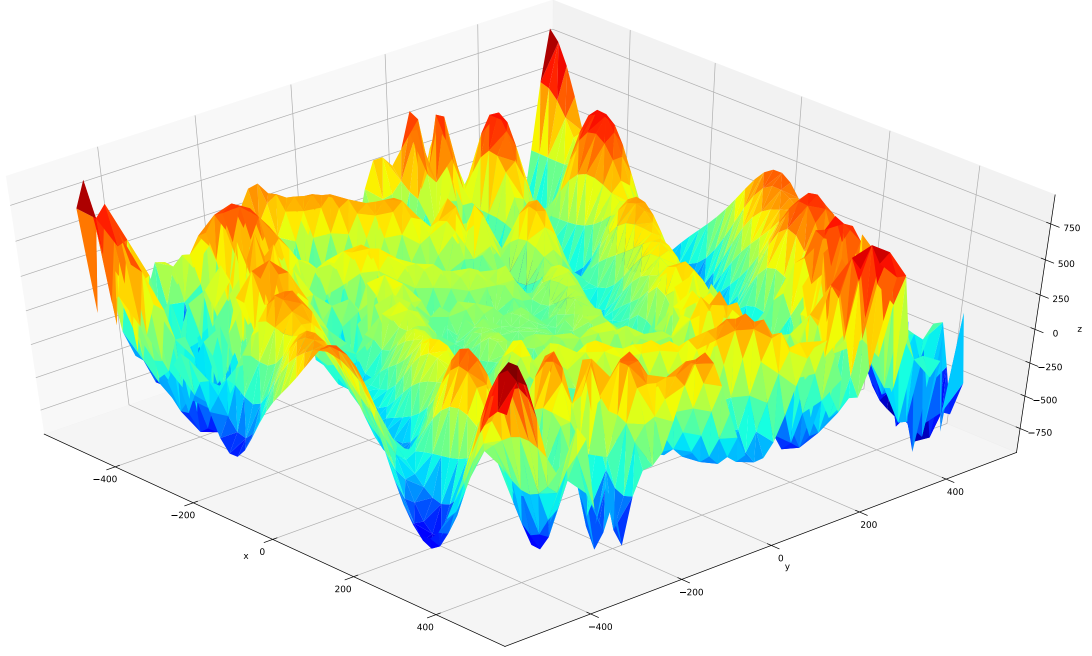
(a) Without local optimization. (b) With local optimization.

Fig. 1: Illustrations of the pruning algorithm for CNF$^{\forall}$-formula with and without using local optimization.

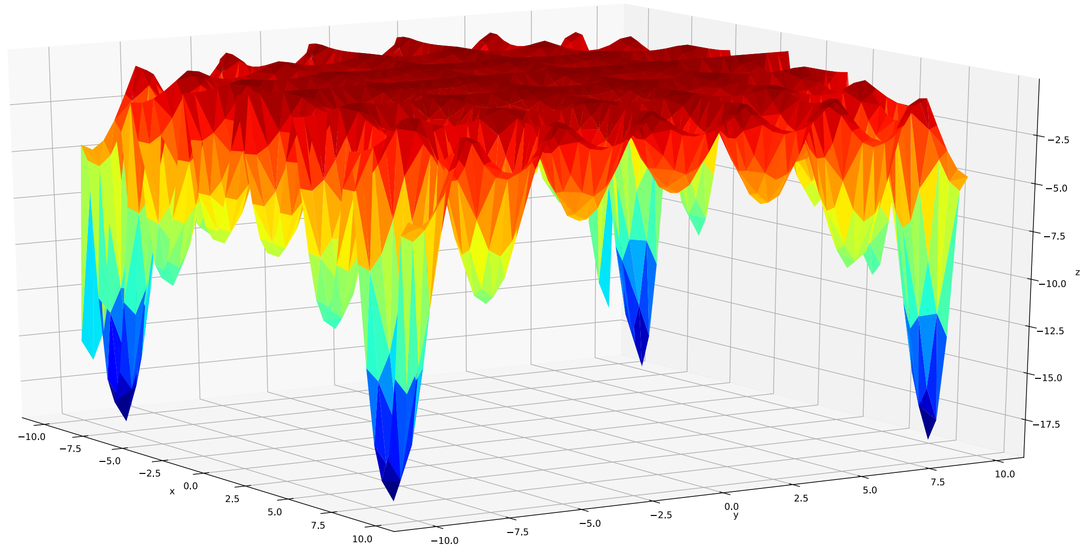
Case Study: Nonlinear Global Optimization



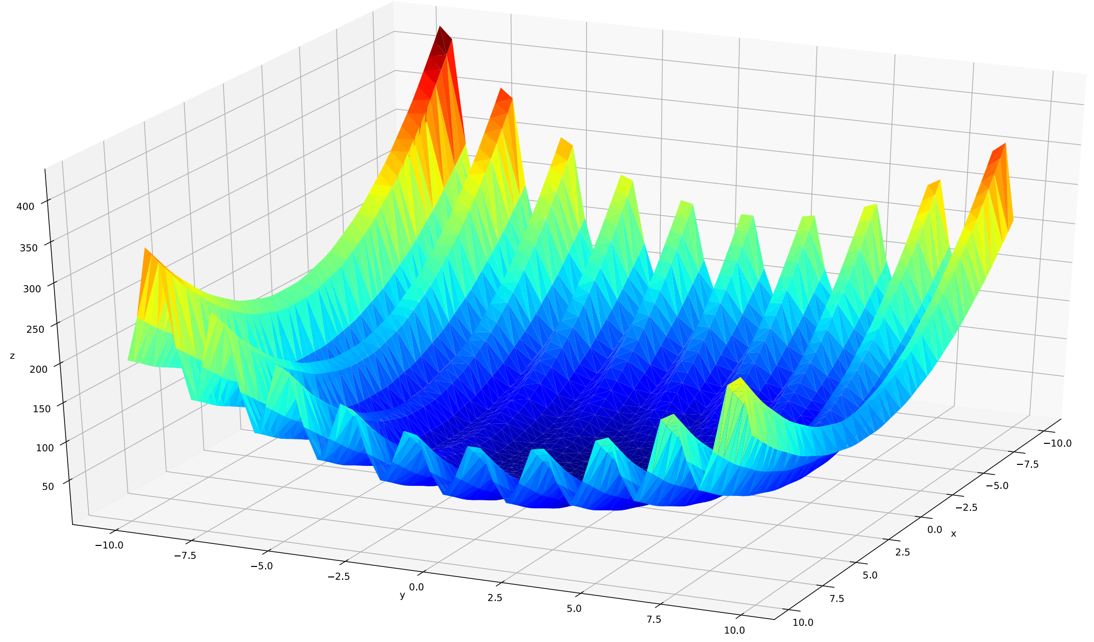
(a) Ackley Function.



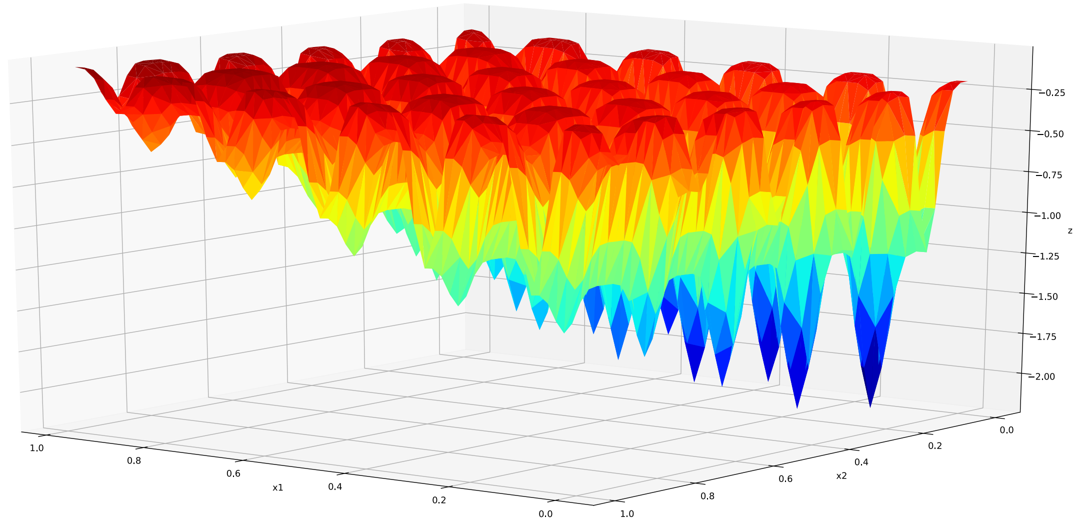
(b) EggHolder Function.



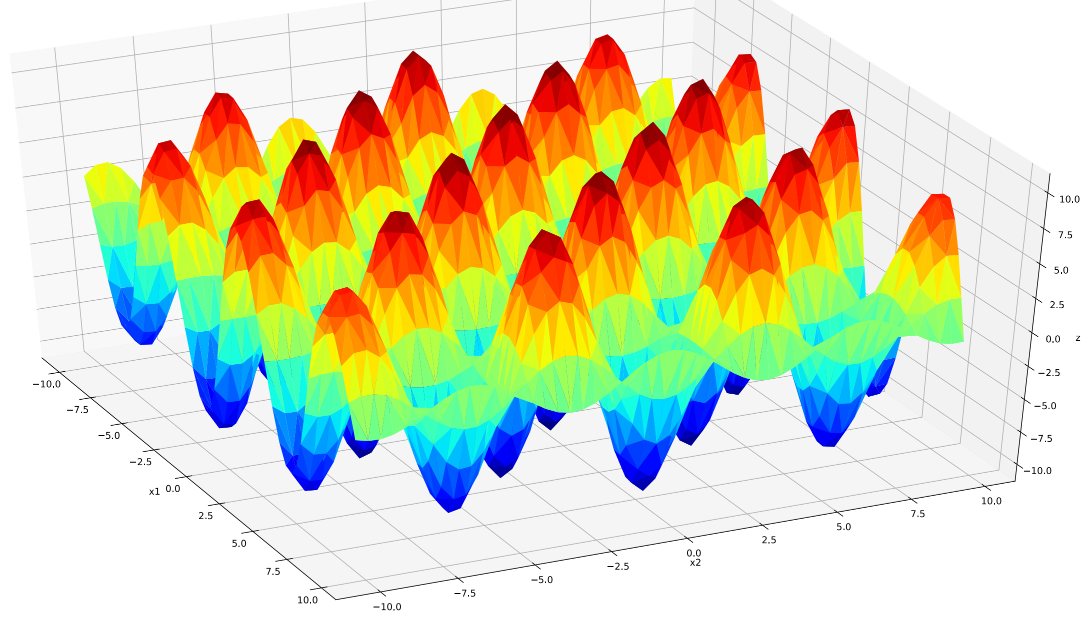
(c) Holder Table2 Function.



(d) Levi N13 Function.



(e) Ripple 1 Function.



(f) Testtube Holder Function.

Case Study: Nonlinear Global Optimization

$\delta = 1e^{-4}$	Name	Solution			Time (sec)		
		Global	No L-Opt.	L-Opt.	No L-Opt.	L-Opt.	Speed up
Unconstrained	Ackley 2D	0.00000	0.00000	0.00000	0.0579	0.0047	12.32
	Ackley 4D	0.00000	0.00005	0.00000	8.2256	0.1930	42.62
	Aluffi Pentini	−0.35230	−0.35231	−0.35239	0.0321	0.1868	0.17
	Beale	0.00000	0.00003	0.00000	0.0317	0.0615	0.52
	Bohachevsky1	0.00000	0.00006	0.00000	0.0094	0.0020	4.70
	Booth	0.00000	0.00006	0.00000	0.5035	0.0020	251.75
	Brent	0.00000	0.00006	0.00000	0.0095	0.0017	5.59
	Bukin6	0.00000	0.00003	0.00003	0.0093	0.0083	1.12
	Cross in tray	−2.06261	−2.06254	−2.06260	0.5669	0.1623	3.49
	Easom	−1.00000	−1.00000	−1.00000	0.0061	0.0030	2.03
	EggHolder	−959.64070	−959.64030	−959.64031	0.0446	0.0211	2.11
	Holder Table 2	−19.20850	−19.20846	−19.20845	52.9152	41.7004	1.27
	Levi N13	0.00000	0.00000	0.00000	0.1383	0.0034	40.68
	Ripple 1	−2.20000	−2.20000	−2.20000	0.0059	0.0065	0.91
	Schaffer F6	0.00000	0.00004	0.00000	0.0531	0.0056	9.48
	Testtube holder	−10.87230	−10.87227	−10.87230	0.0636	0.0035	18.17
	Trefethen	−3.30687	−3.30681	−3.30685	3.0689	1.4916	2.06
	W Wavy	0.00000	0.00000	0.00000	0.1234	0.0138	8.94
	Zettl	−0.00379	−0.00375	−0.00379	0.0070	0.0069	1.01
Constrained	Rosenbrock Cubic	0.00000	0.00005	0.00002	0.0045	0.0036	1.25
	Rosenbrock Disk	0.00000	0.00002	0.00000	0.0036	0.0028	1.29
	Mishra Bird	−106.76454	−106.76449	−106.76451	1.8496	0.9122	2.03
	Townsend	−2.02399	−2.02385	−2.02390	2.6216	0.5817	4.51
	Simionescu	−0.07262	−0.07199	−0.07200	0.0064	0.0048	1.33

Case Study: Synthesizing Lyapunov Function

Problem: Find a Lyapunov function for a dynamical system, v : X → $\mathbb{R}^+$, which satisfies the following condition:

$$\forall \boldsymbol{x} \in X \setminus \mathbf{0} \; v(\boldsymbol{x})(\mathbf{0}) = 0$$
$$\forall \boldsymbol{x} \in X \;\; \nabla v(\boldsymbol{x}(t))^T \cdot f_i(\boldsymbol{x}(t)) \leq 0.$$

where the system is described by a system of ODEs:

$$\dot{\boldsymbol{x}}(t) = f_i(\boldsymbol{x}(t)), \quad \forall \boldsymbol{x}(t) \in X_i.$$

Case Study: Synthesizing Lyapunov Function

Damped Mathieu System Mathieu dynamics are time-varying and defined by the following ODEs:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_2 - (2 + \sin(t))x_1 \end{bmatrix}.$$

Using a quadratic template for a Lyapunov function $v(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} = c_1 x_1 x_2 + c_2 x_1^2 + c_3 x_2^2$, we can encode this synthesis problem into the following $\exists\forall$-formula:

$$\begin{aligned} \exists c_1 c_2 c_3 \; \forall x_1 x_2 t \; & [(50 x_1 x_2 c_2 + 50 x_1^2 c_1 + 50 x_2^2 c_3 > 0) \wedge \\ & (100 c_1 x_1 x_2 + 50 x_2 c_2 + (-x_2 - x_1(2 + \sin(t)))(50 x_1 c_2 + 100 x_2 c_3) < 0) \\ & \vee \neg((0.01 \leq x_1^2 + x_2^2) \wedge (0.1 \leq t) \wedge (t \leq 1) \wedge (x_1^2 + x_2^2 \leq 1))] \end{aligned}$$

Our prototype solver takes 26.533 seconds to synthesize the following function as a solution to the problem for the bound $\|\boldsymbol{x}\| \in [0.1, 1.0]$, $t \in [0.1, 1.0]$, and $c_i \in [45, 98]$ using $\delta = 0.05$:

$$V = 54.6950 x_1 x_2 + 90.2849 x_1^2 + 50.5376 x_2^2.$$

Conclusion

- To handle exist-forall problems in the delta-decision framework, we have designed **pruning operators for ∀-constraints**.
  - Finding (good) **counterexamples** is the key:
    - **Double-sided error control** is necessary to avoid spurious counterexamples.
    - Using **local-optimization** techniques can accelerate the solving process.
  - Proved the **correctness** of the algorithm (See theorem 1 in the paper)
  - The **recursive** nature of the algorithm (dReal calls dReal) **matches** with the **problem's time complexity** (Σ_2^P).
- Demonstrated the effectiveness of the procedures on various **global optimization** and **Lyapunov function synthesis** problems.
- The tool is available at https://github.com/dreal/dreal4 (released under Apache 2.0)