Automated Reasoning over the Reals

Soonho Kong
soonhok@cs.cmu.edu

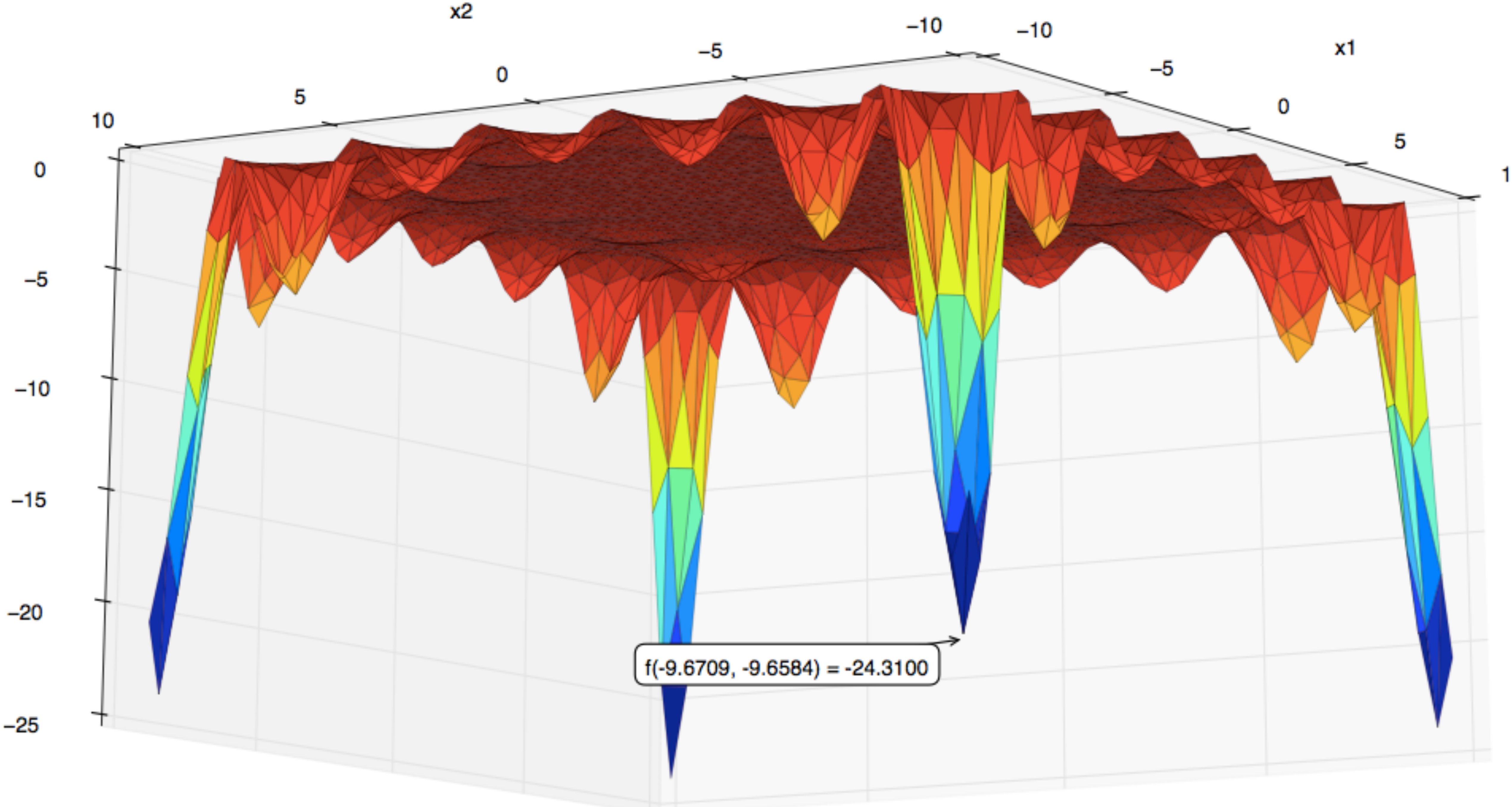
Optimization

- Used (almost) everywhere
- Classes of Polynomial-time solvable problems
  (i.e. Convex optimization, Linear programming)
- Other classes reducible to them
  (i.e. LP relaxation of MLP)

147. **Table 3 / Carrom Table Function** [58] (Continuous, Differentiable, Non-Separable, Non-Scalable, Multimodal)

$$\begin{aligned} f_{147}(\mathbf{x}) &= -[(\cos(x_1)\cos(x_2)\\ &\quad \exp|1 - [(x_1^2 + x_2^2)^{0.5}]/\pi|)^2]/30 \end{aligned}$$

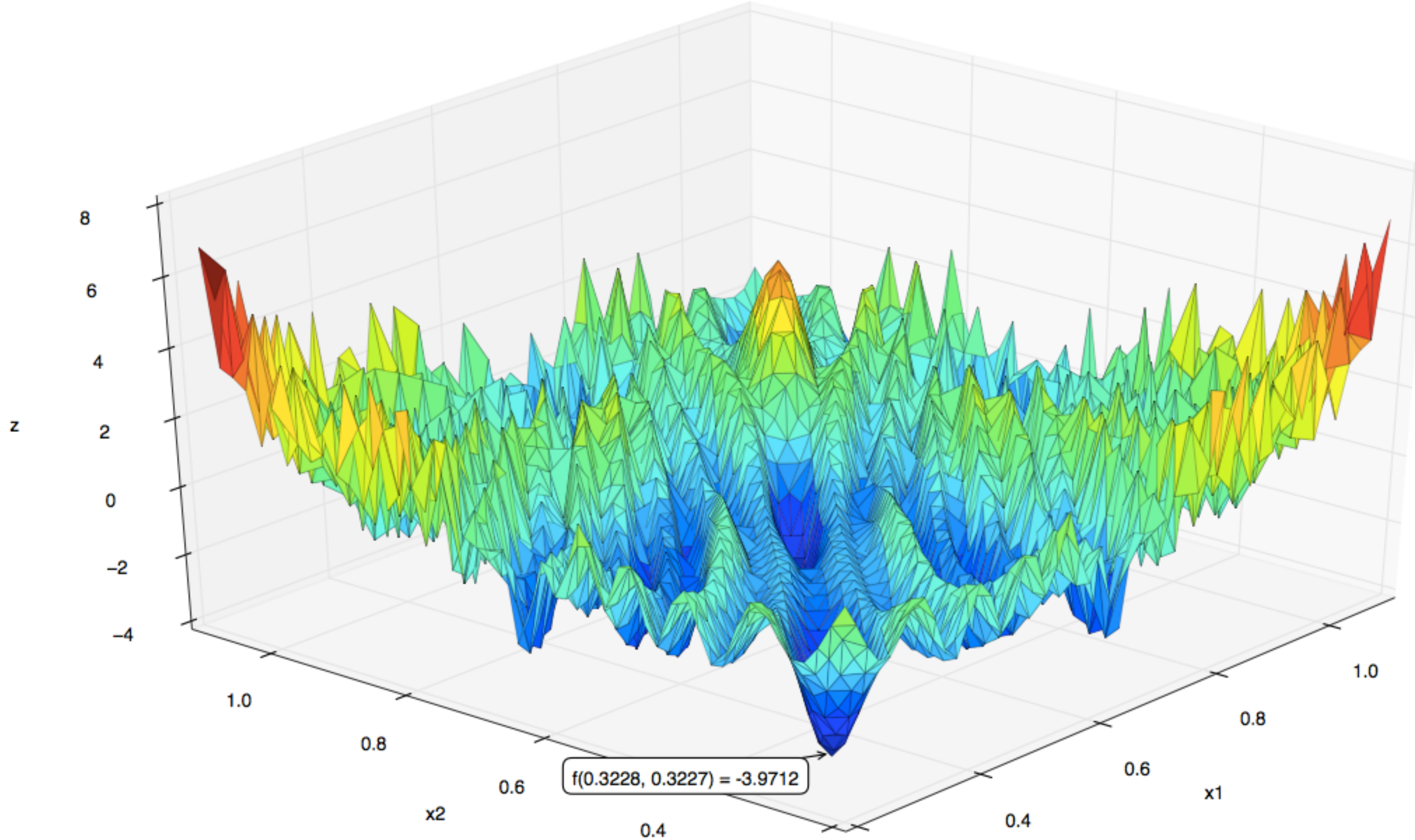
subject to $-10 \le x_i \le 10$.



Momin Jamil and Xin-She Yang, **A literature survey of benchmark functions for global optimization problems**, Int. Journal of Mathematical Modelling and Numerical Optimisation

167. **Whitley Function** [86] (Continuous, Differentiable, Non-Separable, Scalable, Multimodal)

$$f_{167}(\mathbf{x}) = \sum_{i=1}^{D}\sum_{j=1}^{D}\left[\frac{(100(x_i^2 - x_j)^2 + (1-x_j)^2)^2}{4000} - \cos\left(100(x_i^2 - x_j)^2 + (1-x_j)^2 + 1\right)\right]$$



Momin Jamil and Xin-She Yang, **A literature survey of benchmark functions for global optimization problems**, Int. Journal of Mathematical Modelling and Numerical Optimisation

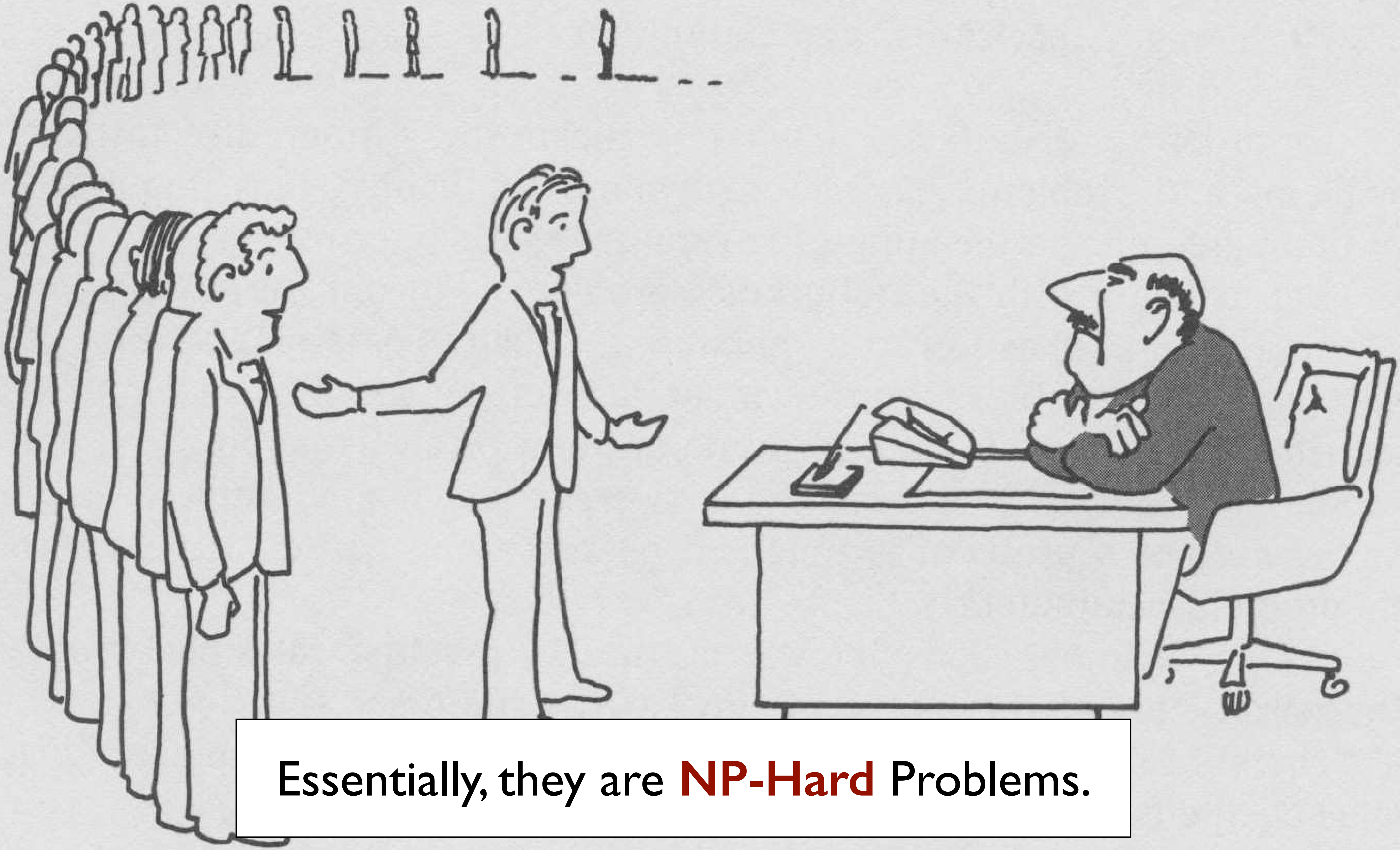
Challenges in Optimization

$$\left.\begin{array}{l}\text{Global}\\ \text{Non-convex}\\ \text{Highly Nonlinear}\\ \text{Multi-objective}\\ \text{Combinatorial (disjunctions)}\end{array}\right\}\ \text{Optimization}$$

Challenges in Optimization

$$\left.\left\{\begin{array}{l}\text{Global}\\ \text{Non-convex}\\ \text{Highly Nonlinear}\\ \text{Multi-objective}\\ \text{Combinatorial (disjunctions)}\end{array}\right.\right\}\ \text{Optimization}$$

Essentially, they are **NP-Hard** Problems.



Essentially, they are **NP-Hard** Problems.

"I can't find an efficient algorithm, but neither can all these famous people."

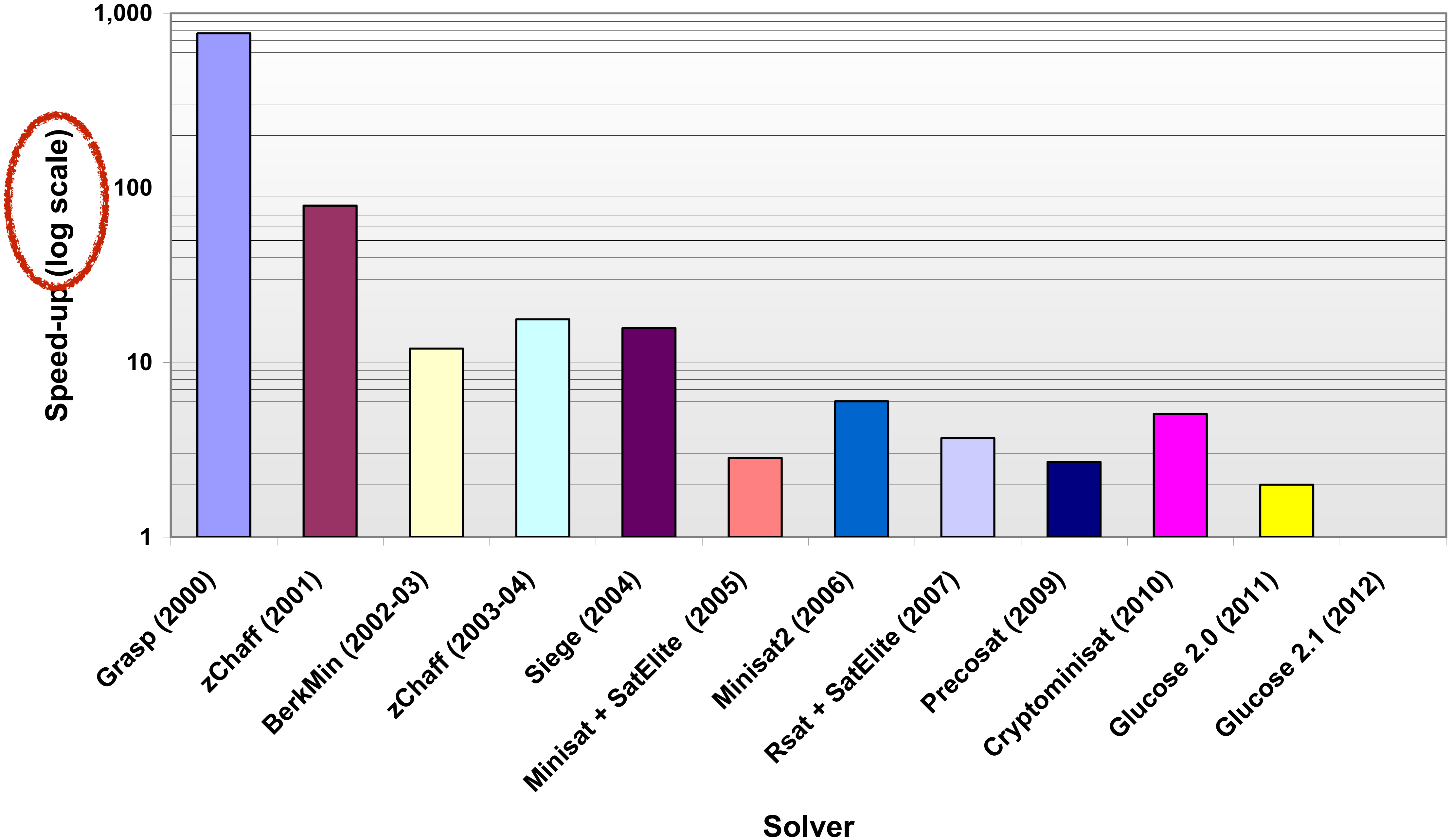
The End of Story?

Advances in Solving NP-Hard Problems

The past two decades witnessed the tremendous progress in practical algorithms for **NP-hard** problems

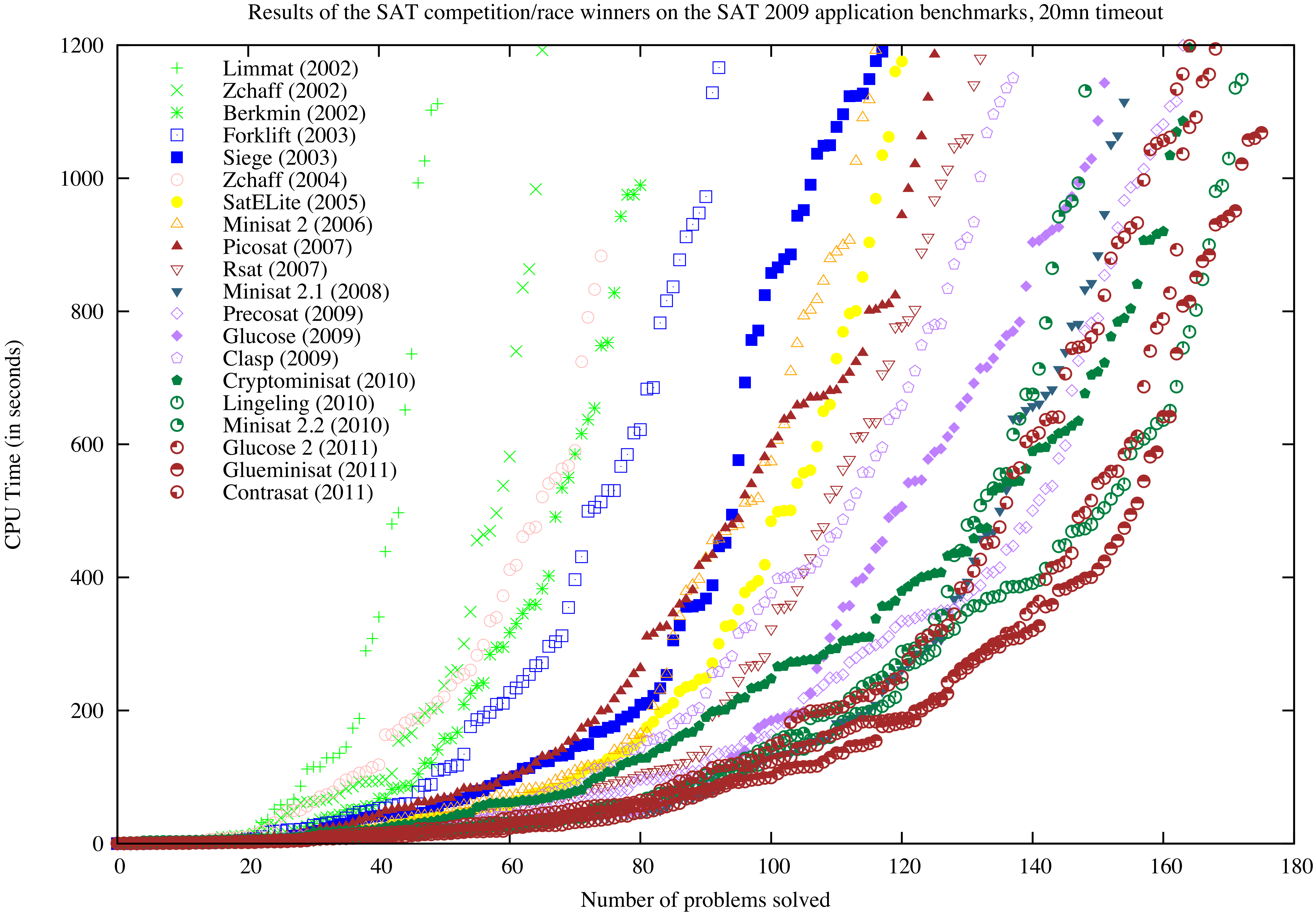
- Industrial-sized SAT problems, millions of vars
- Major driving force in HW design and SW analysis

Advances in Solving NP-Hard Problems



1000x Speed-up over 12 years!

Advances in Solving NP-Hard Problems



How to apply?

The advances in discrete domain
into hybrid (continuous/discrete) domain?

- Combinatorial structure (discrete)
- Nonlinear dynamics (concrete)

How to apply?

The advances in discrete domain
into hybrid (continuous/discrete) domain?

- Combinatorial structure (discrete)
- Nonlinear dynamics (concrete)

Feynman's Algorithm:

1. Write down the problem
2. Think real hard
3. Write down the solution

How to apply?

The advances in discrete domain
into hybrid (continuous/discrete) domain?

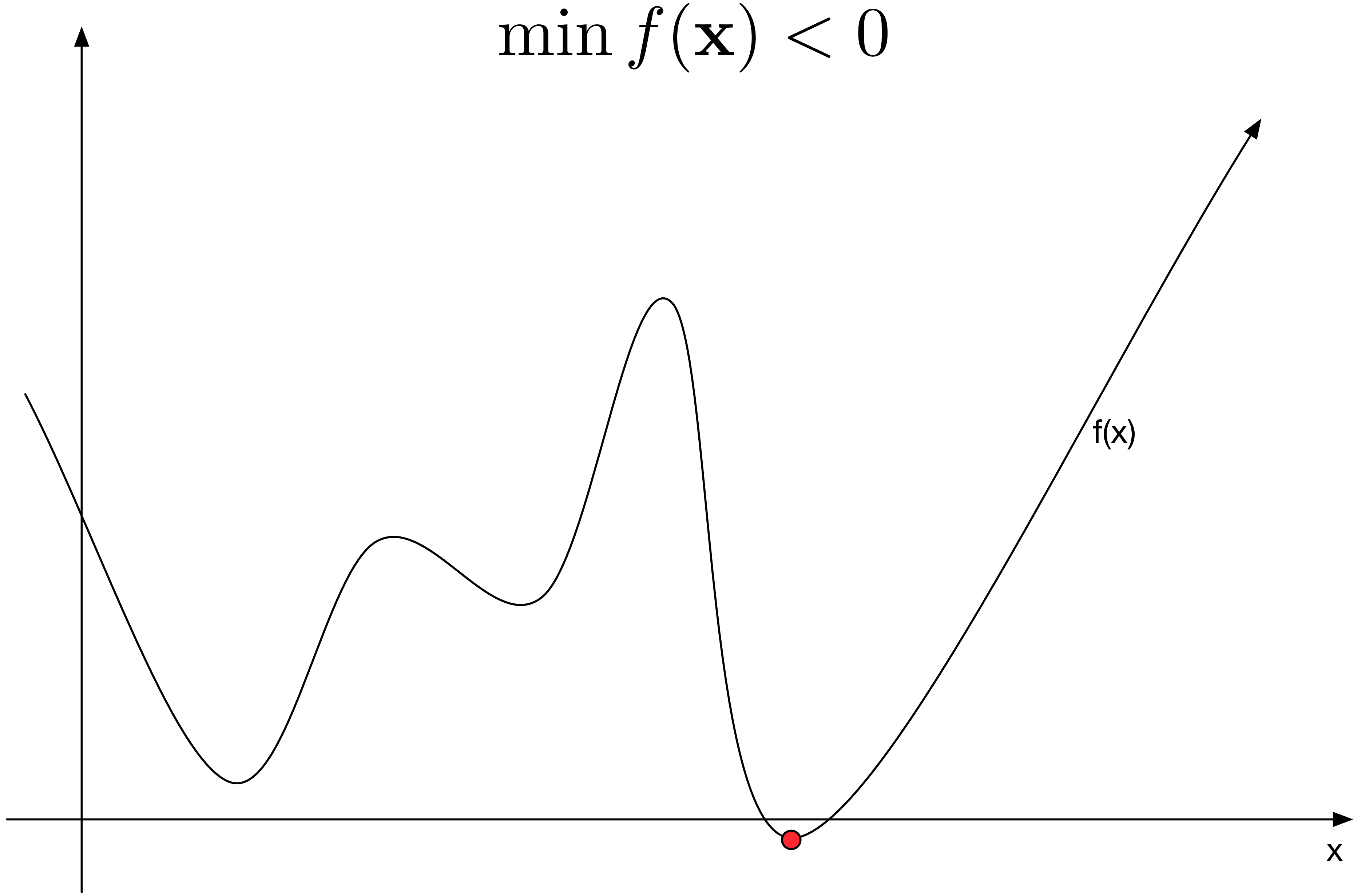
- Combinatorial structure (discrete)
- Nonlinear dynamics (concrete)

Our Approach:

1. Write down problems in first-order logic
2. Solve NP-Hard problems
3. Interpret solutions

Example: Optimization

$$\min f(\mathbf{x}) < 0$$

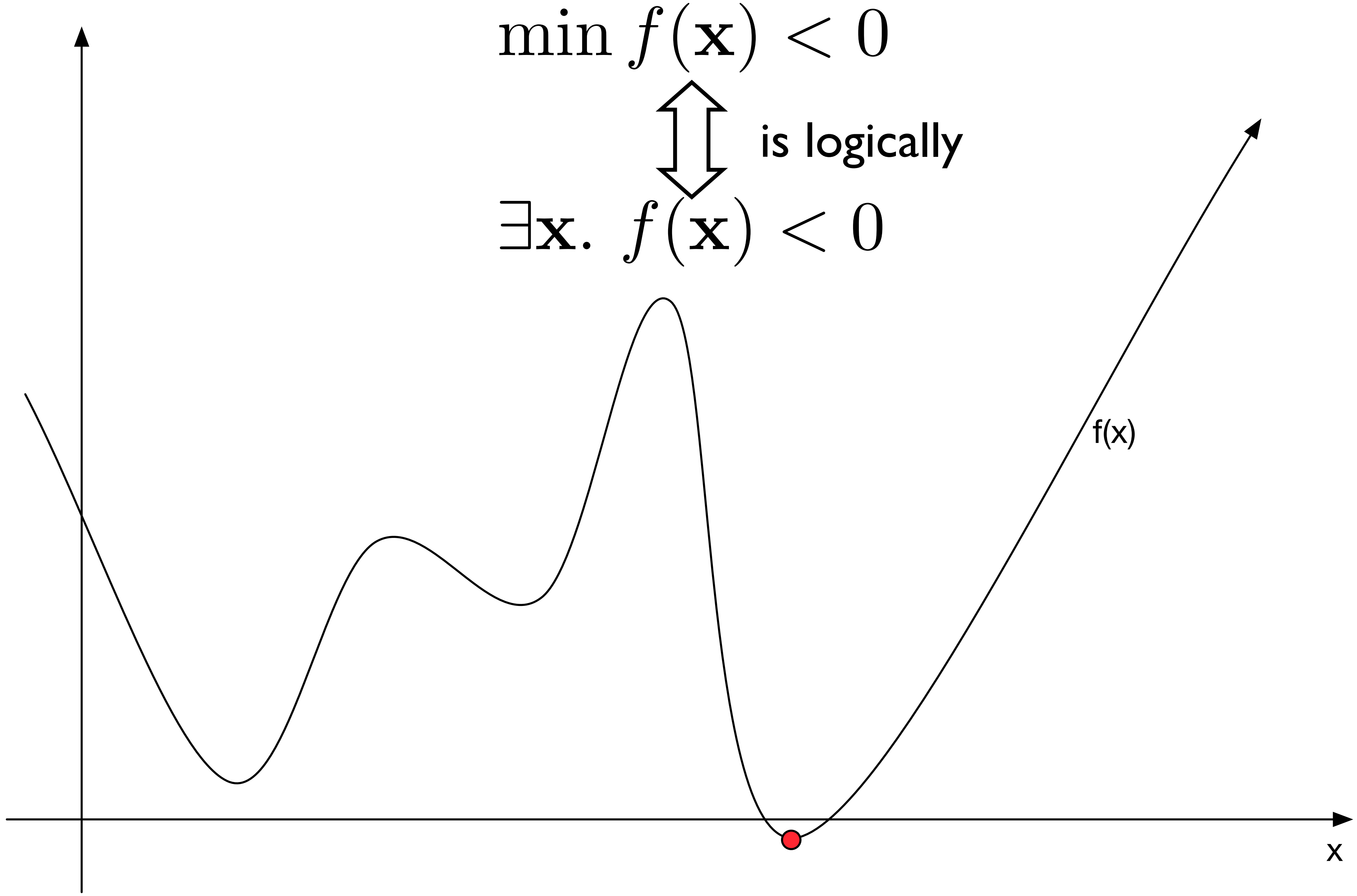


Example: Optimization

$$\min f(\mathbf{x}) < 0$$

$\Updownarrow$ is logically

$$\exists \mathbf{x}.\ f(\mathbf{x}) < 0$$



Example: Optimization

$$\min f(\mathbf{x}) < 0$$

$$\Updownarrow \quad \text{is logically}$$

$$\exists \mathbf{x}.\ f(\mathbf{x}) < 0$$

$$\min f(\mathbf{x}) \text{ s.t. } \phi(\mathbf{x})$$

Example: Optimization

$$\min f(\mathbf{x}) < 0$$

$$\Updownarrow \text{ is logically}$$

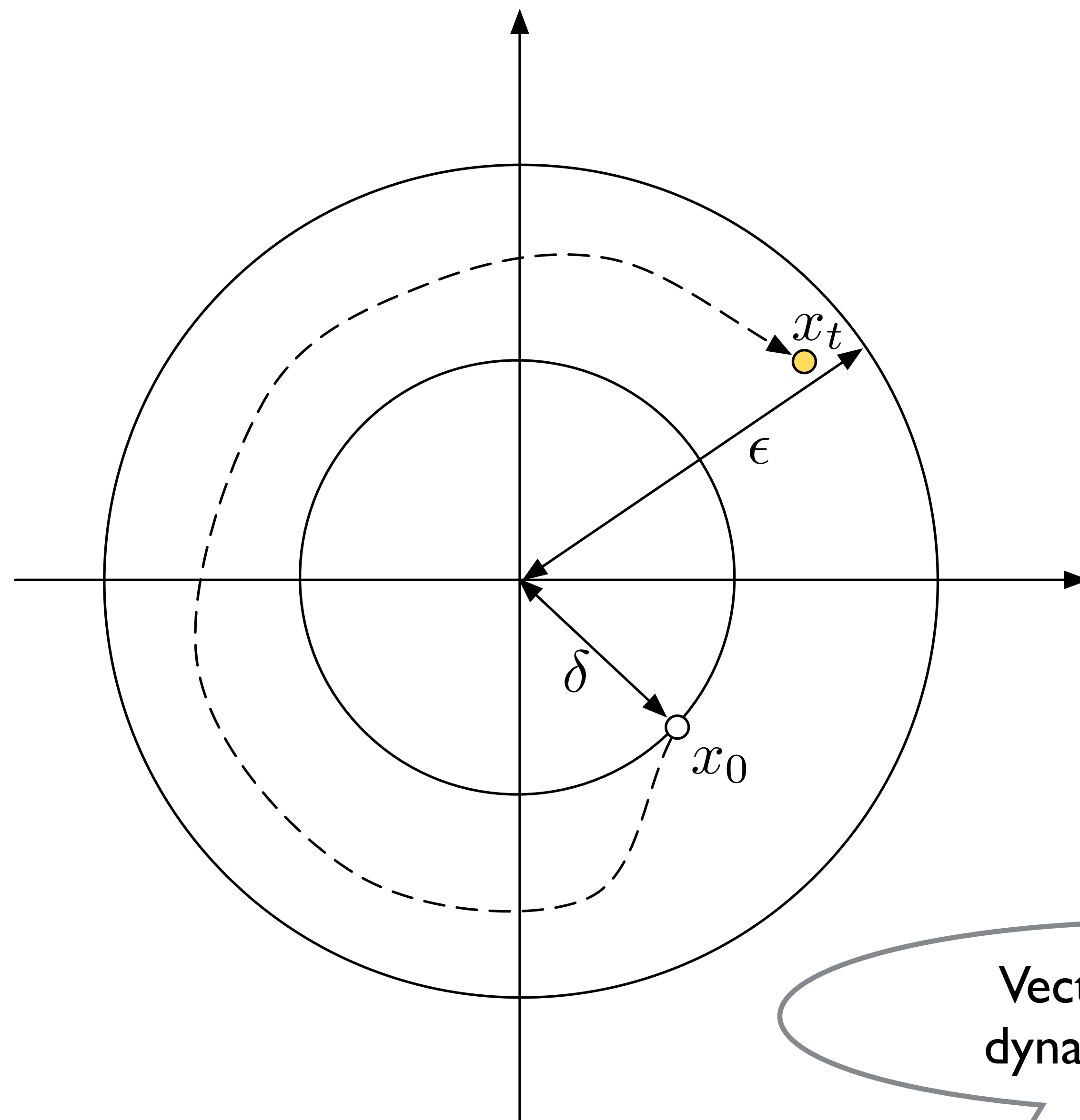
$$\exists \mathbf{x}.\ f(\mathbf{x}) < 0$$

$$\min f(\mathbf{x}) \text{ s.t. } \phi(\mathbf{x})$$

$$\Updownarrow \text{ is logically}$$

$$\exists \mathbf{x}. \forall \mathbf{y}.\ \phi(\mathbf{x}) \wedge \phi(\mathbf{y}) \rightarrow f(\mathbf{x}) \leq f(\mathbf{y})$$

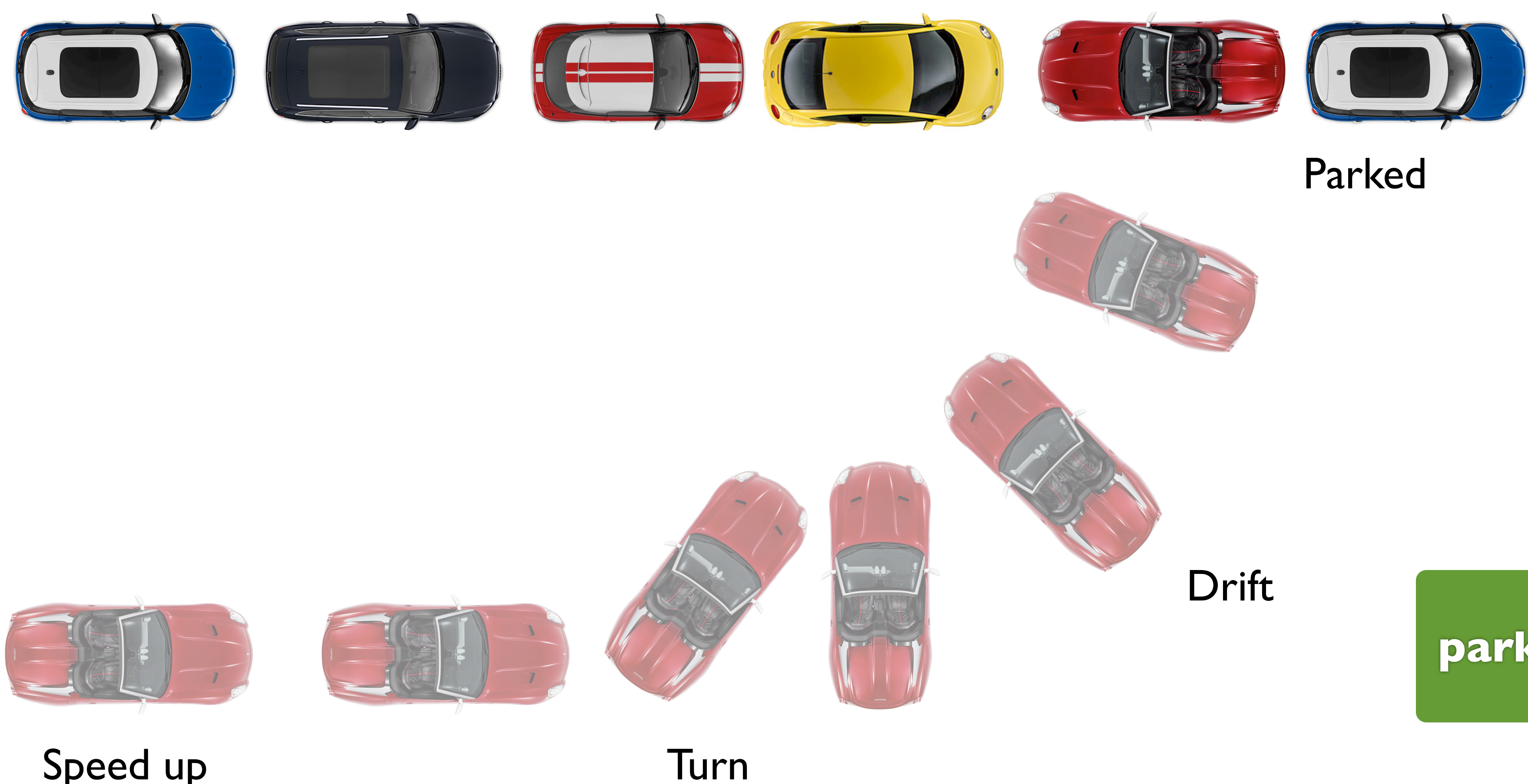
Example: Lyapunov Stability

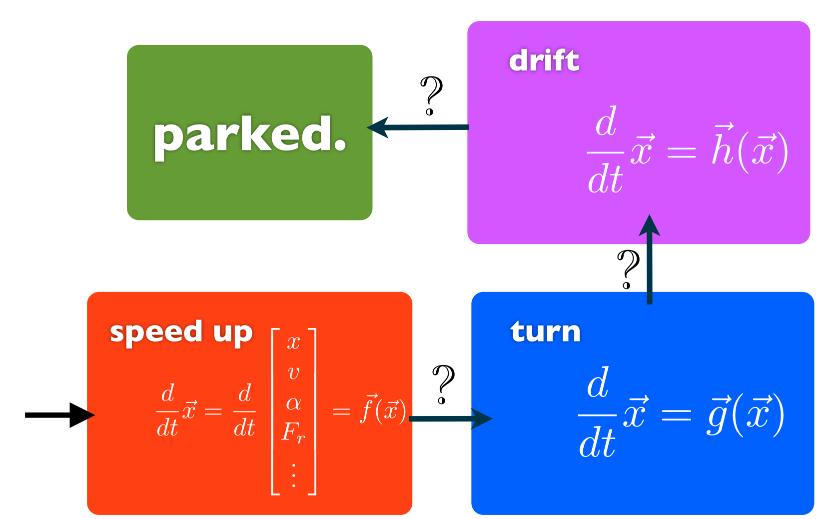


Vector field of a dynamical system

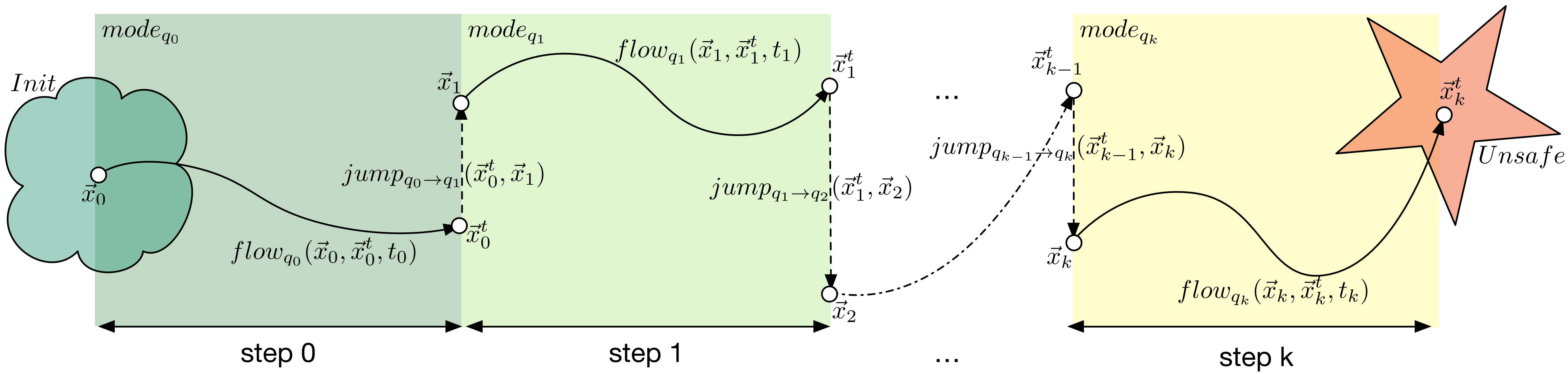
$$\forall \epsilon \exists \delta \forall x_0 \forall x_t . \ \left((||x_0|| < \delta \wedge x_t = x_0 + \int_0^t f(s)\mathrm{ds}) \rightarrow ||x_t|| < \epsilon \right)$$

Example: Planning





Example: Planning



$$\exists \vec{x}_0, \vec{x}_1, \ldots, \vec{x}_k \exists \vec{x}^t_0, \vec{x}^t_1, \ldots, \vec{x}^t_k \exists t_0, t_1, \ldots, t_k$$

$$Init(\vec{x}_0) \wedge flow_{q_0}(\vec{x}_0, \vec{x}^t_0, t_0) \wedge jump_{q_0 \to q_1}(\vec{x}^t_0, \vec{x}_1) \wedge$$

$$flow_{q_1}(\vec{x}_1, \vec{x}^t_1, t_1) \wedge jump_{q_1 \to q_2}(\vec{x}^t_1, \vec{x}_2) \wedge$$

$$\ldots$$

$$flow_{q_k}(\vec{x}_k, \vec{x}^t_k, t_k) \wedge Unsafe(\vec{x}^t_k)$$

Encode Problems in First-order Logic Formula

- Optimization

$$\exists \mathbf{x}.\forall \mathbf{y}.\ \phi(\mathbf{x}) \wedge \phi(\mathbf{y}) \rightarrow f(\mathbf{x}) \leq f(\mathbf{y})$$

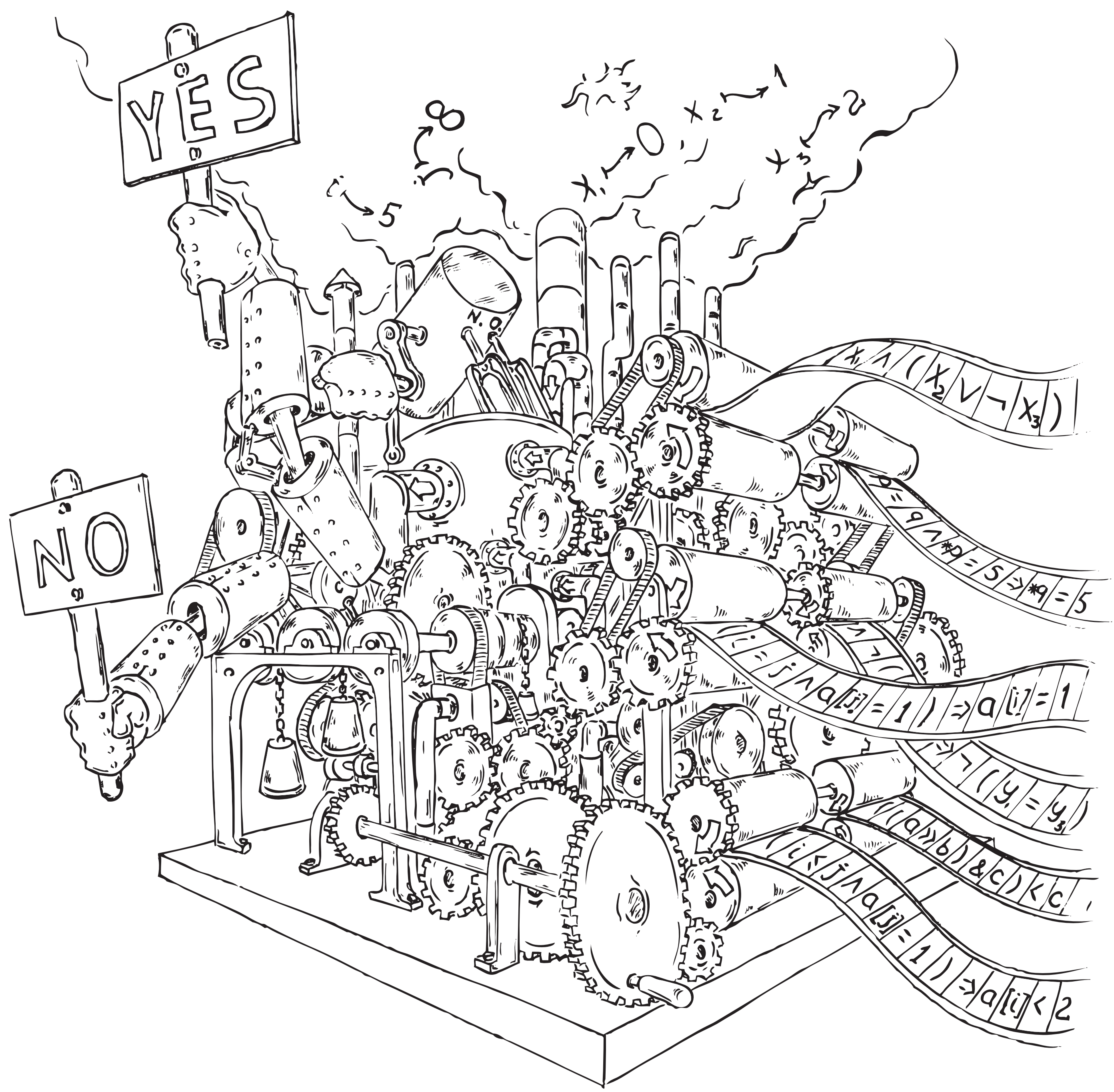
- Stability

$$\forall \epsilon \exists \delta \forall x_0 \forall x_t.\ \left((||x_0|| < \delta \wedge x_t = x_0 + \int_0^t f(s)\mathrm{ds}) \rightarrow ||x_t|| < \epsilon \right)$$

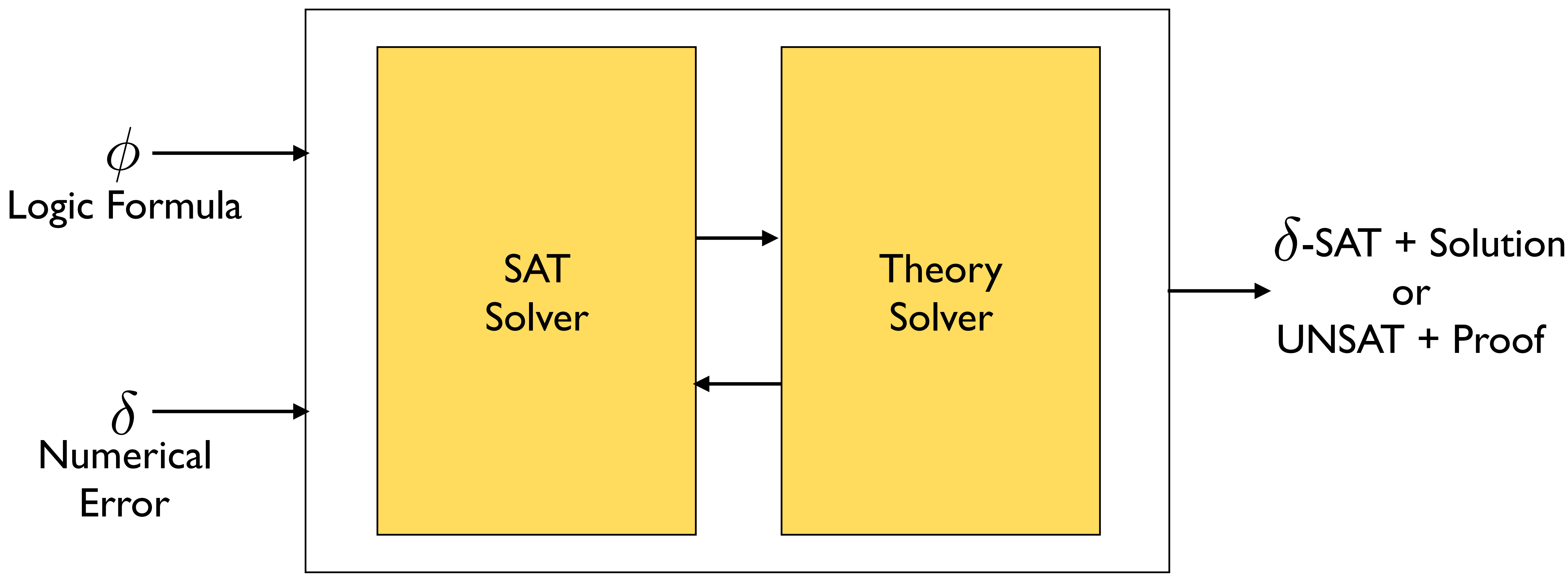
- Planning

$$\exists \mathbf{x}.\ \bigvee_i \bigwedge_j f_{i,j}(\mathbf{x})$$

Solving Logic Formula

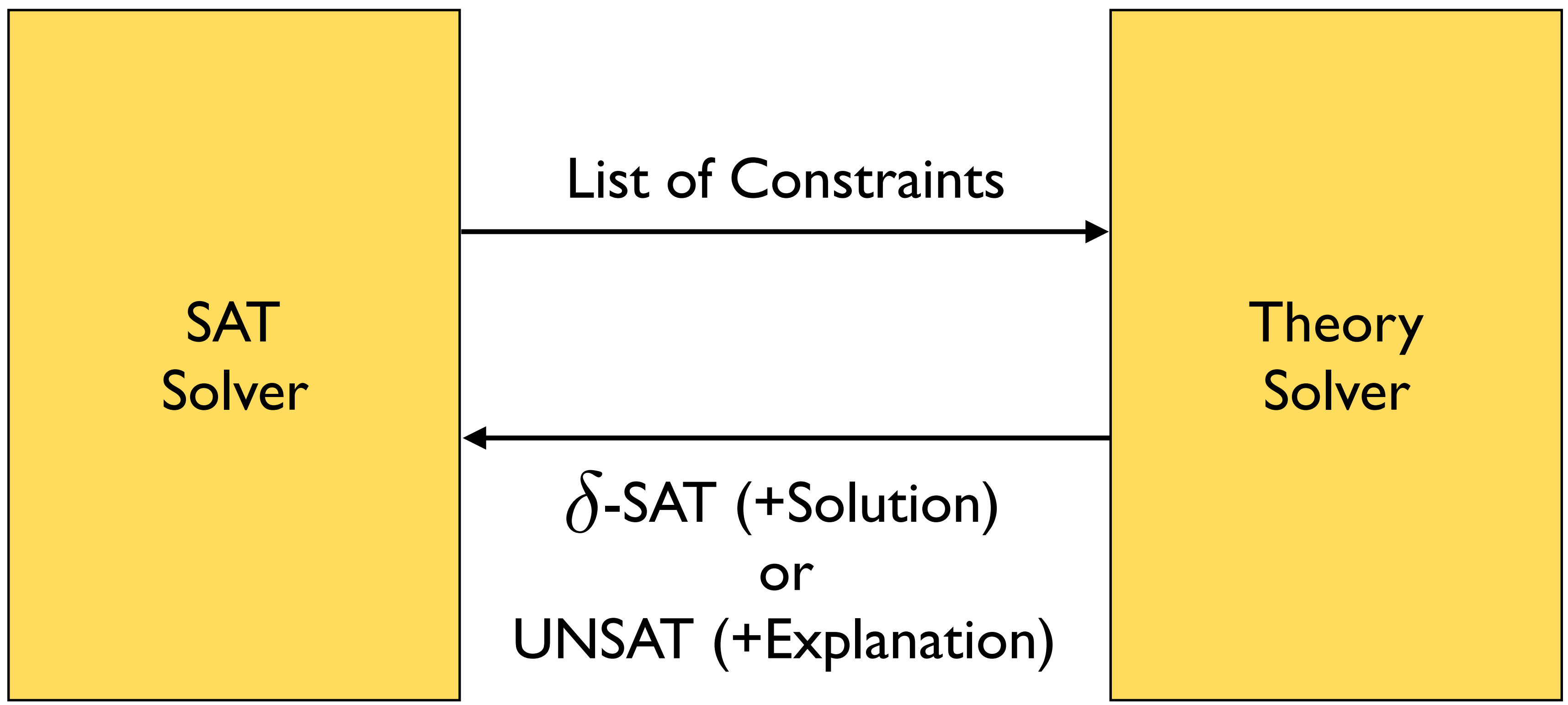


Big Picture



- **SAT solver** finds a satisfying **Boolean** assignment
- **Theory solver** checks whether the assignment is feasible under the first order theory of **Real**

Big Picture

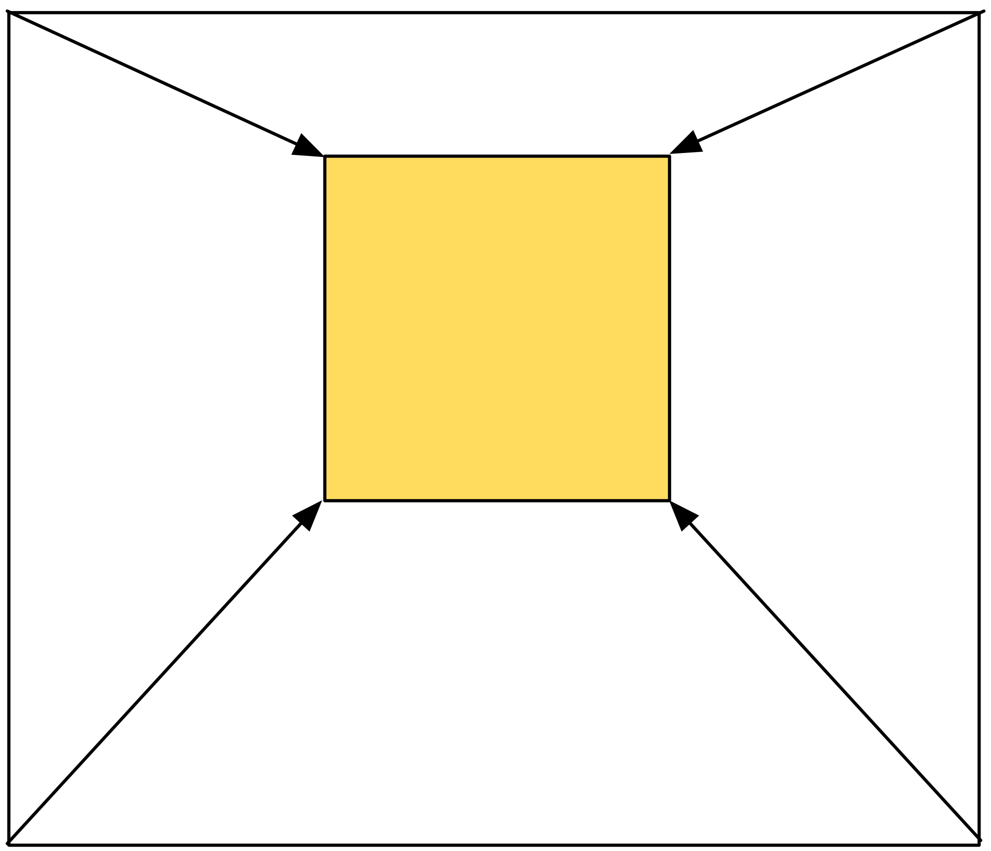


Boolean Search
Non-chronological Backtracking
Learning
...

(Discrete Domain)

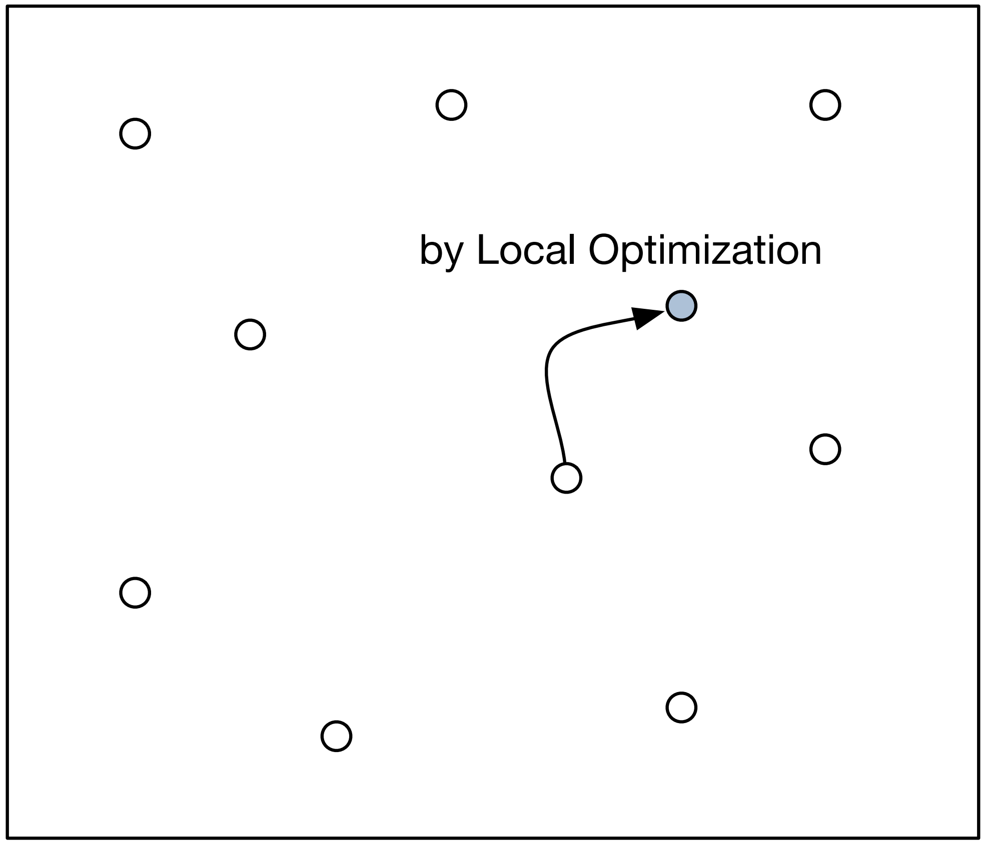
Constraints Solving
Validated Numerics
Optimization
Simulation/Sampling
...
(Continuous Domain)

Top-down/Bottom Approaches in Theory Solver



Top-Down Approach

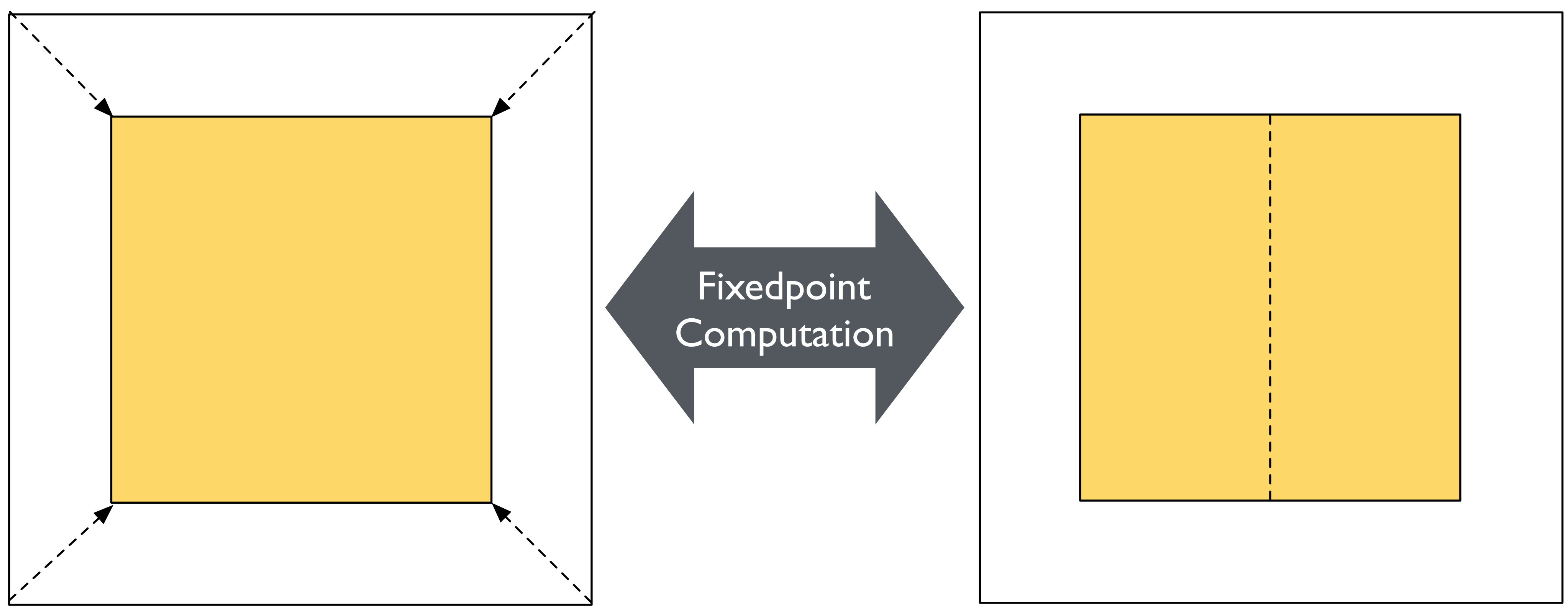
Maintain a set of possible solutions
Useful to show **UNSAT**
Validated Numerics
(i.e. Interval-based methods)



Bottom-Up Approach

Sample points and test them
Useful to show **SAT**
Use local-optimization to improve

An Algorithm in Theory Solver: ICP(Interval Constraint Propagation)

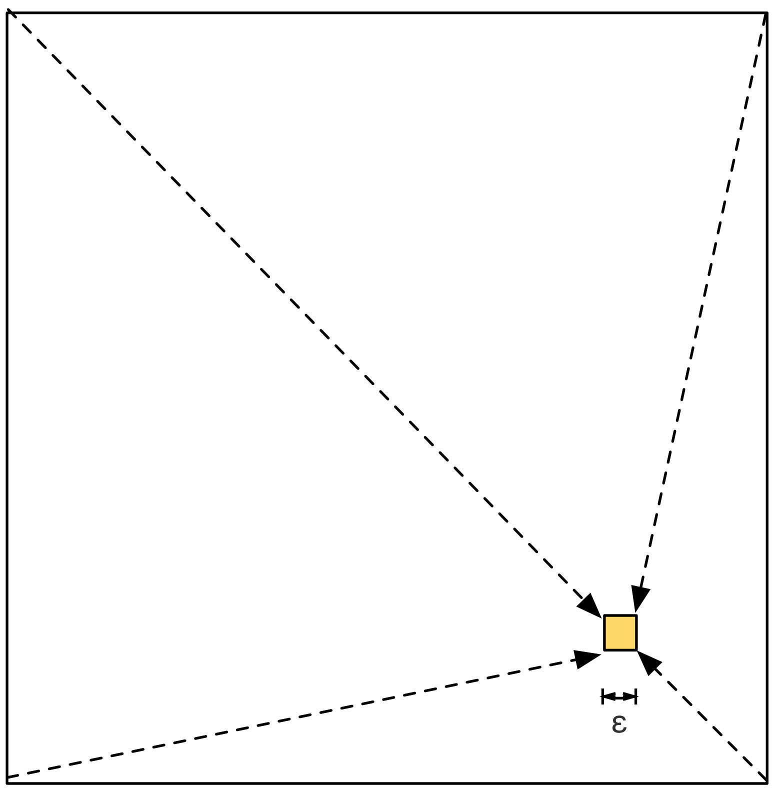


Pruning | Branch

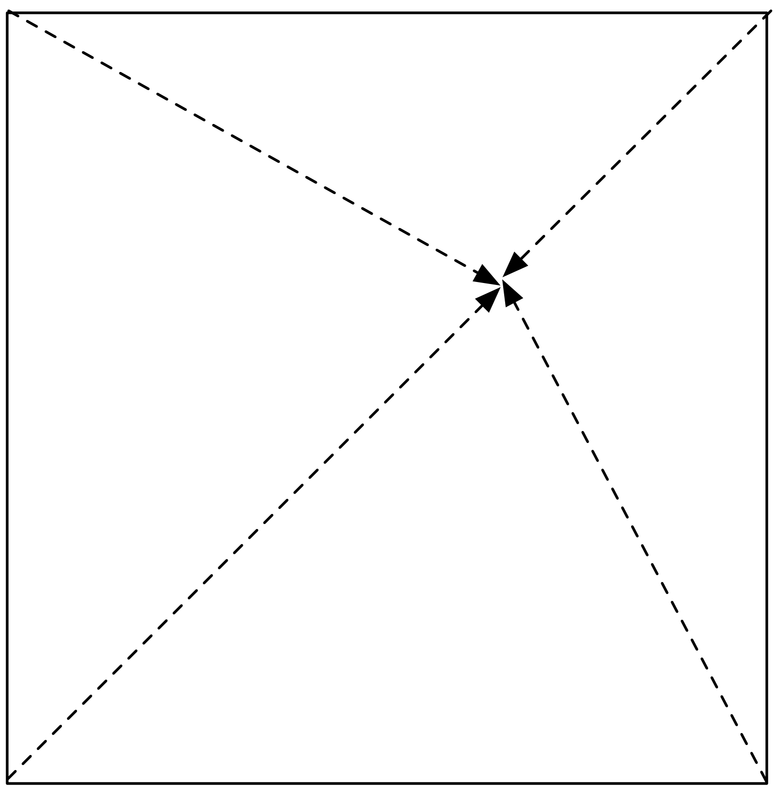
Safely **reduce** a search space without removing solutions

Partition a search space into two sub-spaces

Two Termination Conditions of ICP

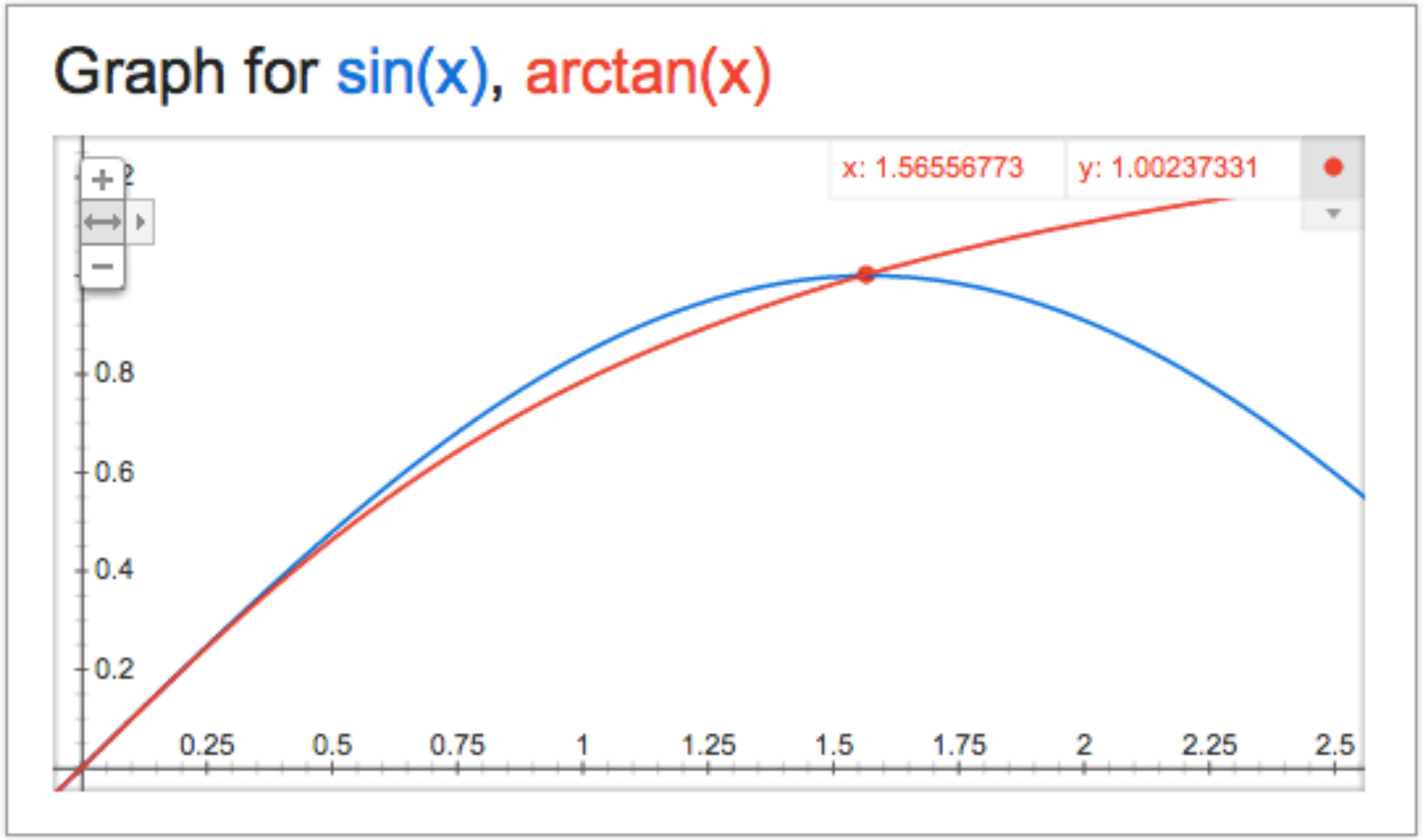


δ-sat



Unsat

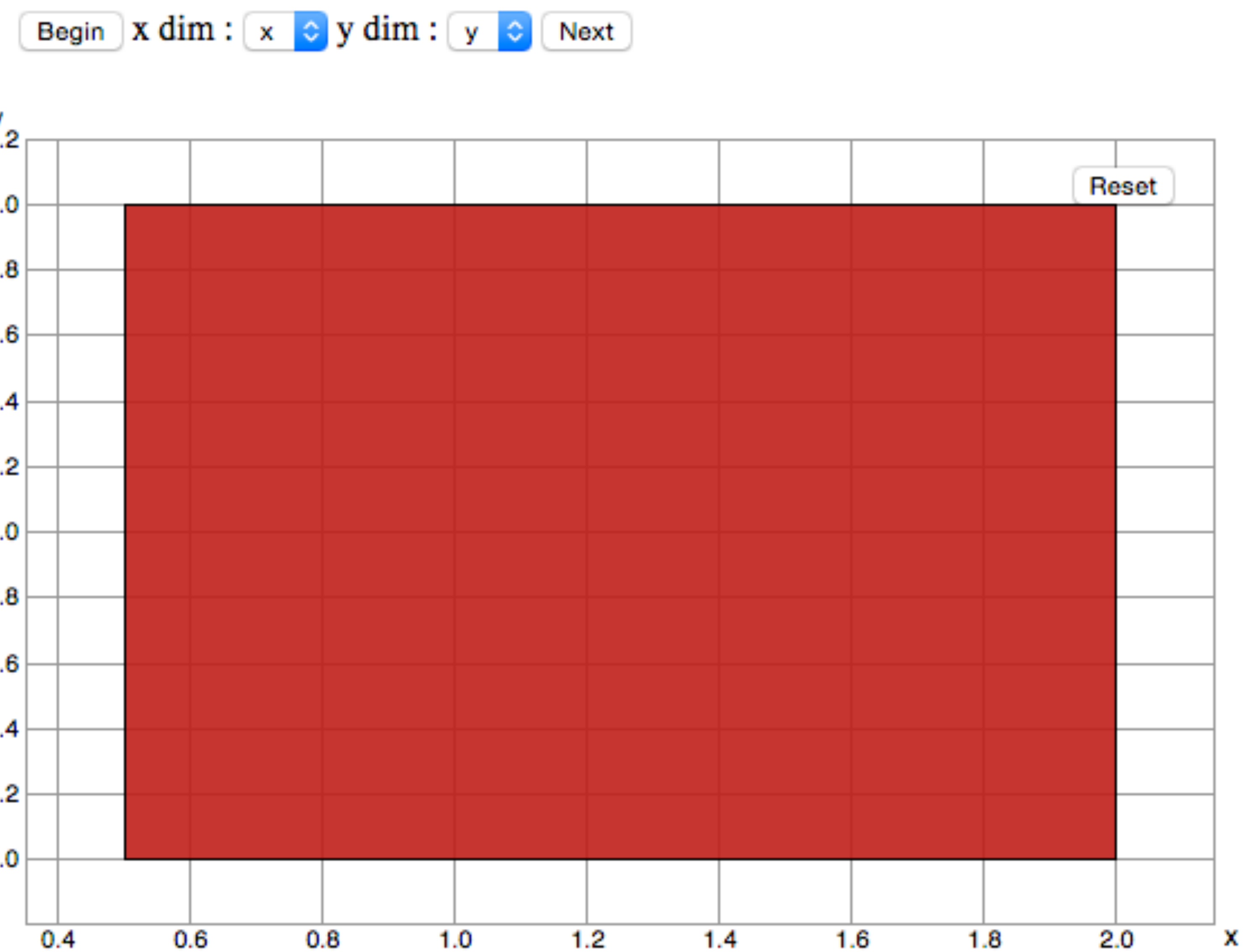
Example of ICP



ANSWER: **SAT**

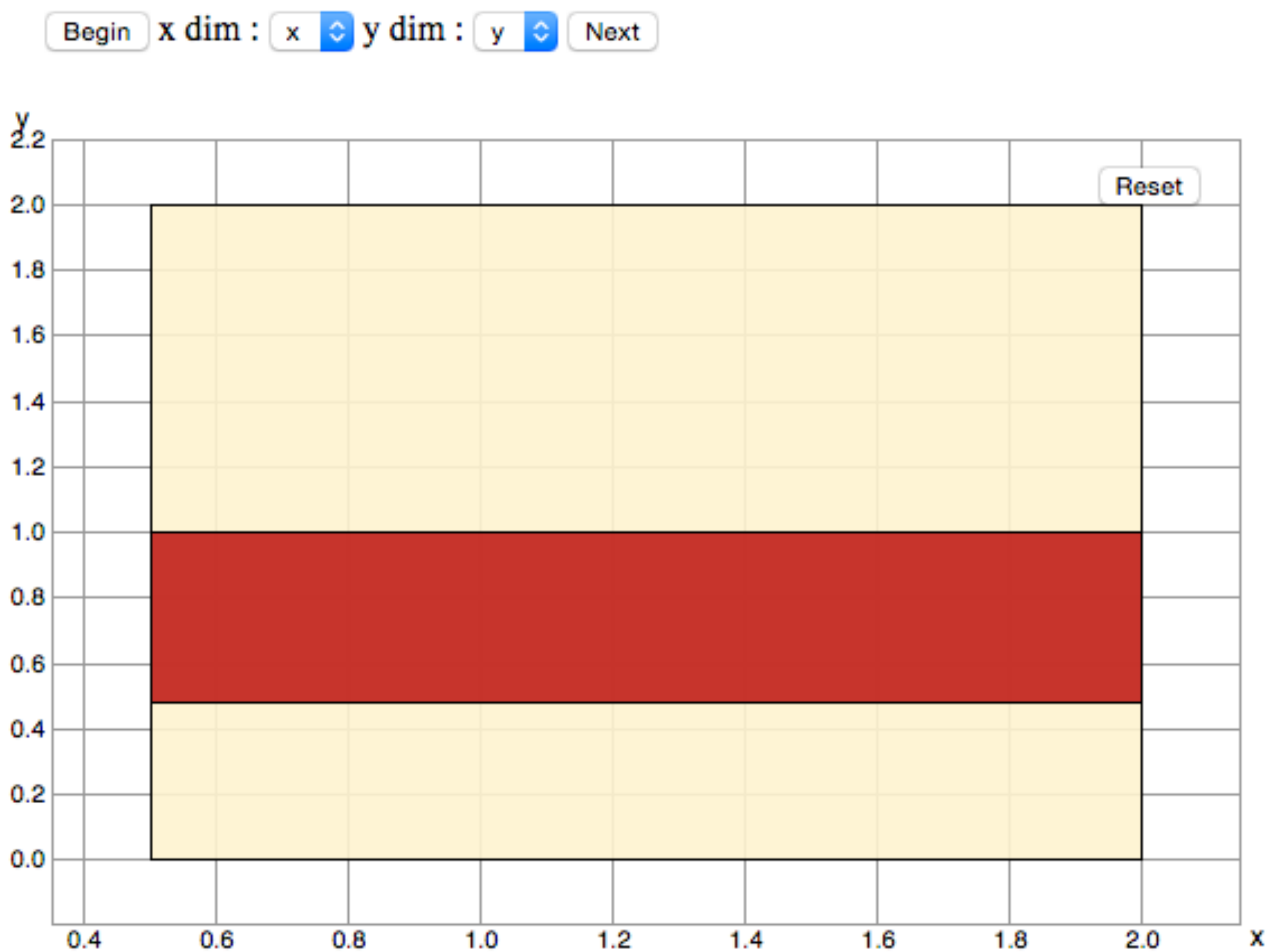
Example of ICP

$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \mathrm{atan}(x)$$



Example of ICP

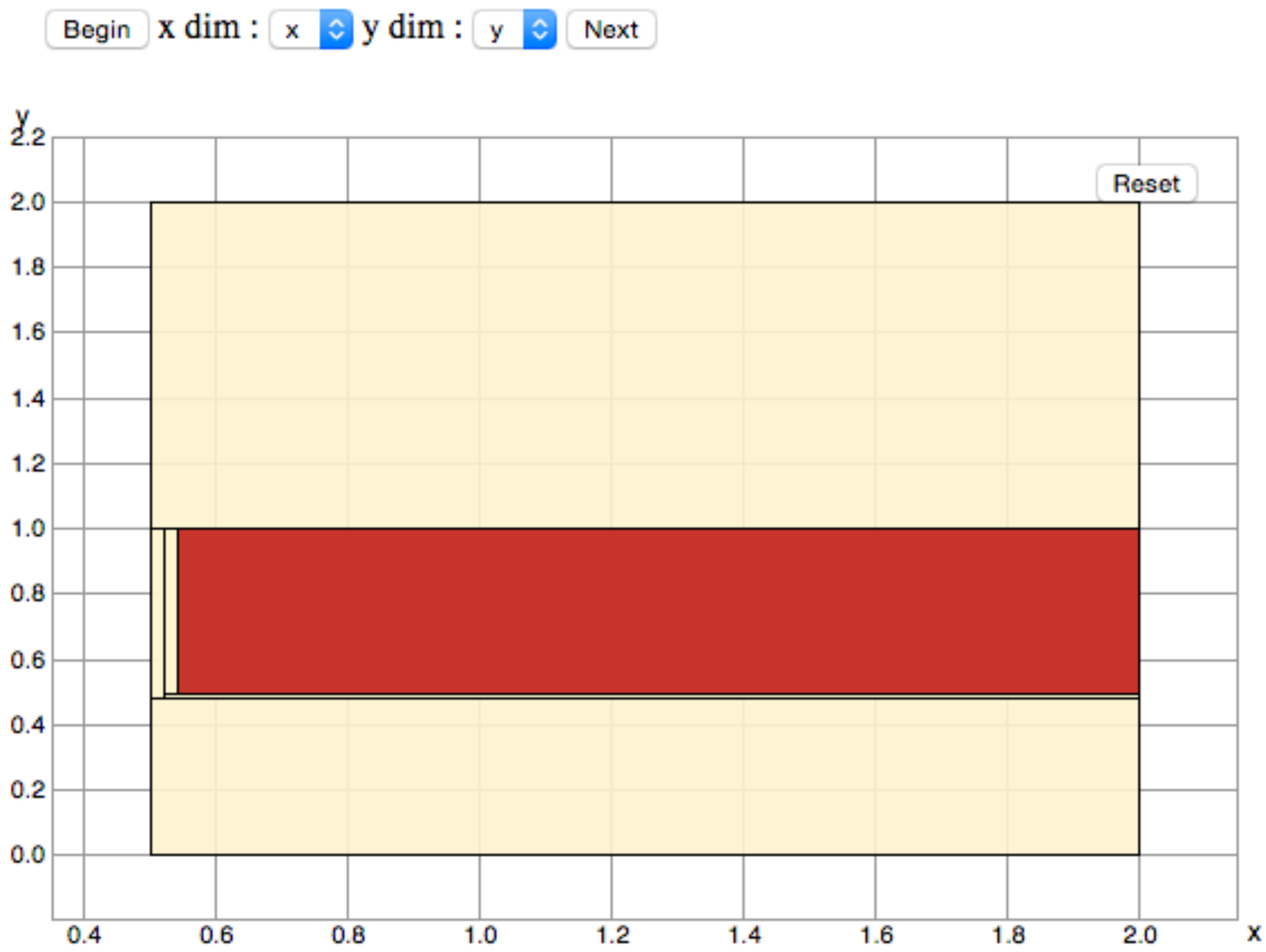
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : \boxed{y = \sin(x)} \wedge y = \mathrm{atan}(x)$$



Pruning Applied

Example of ICP

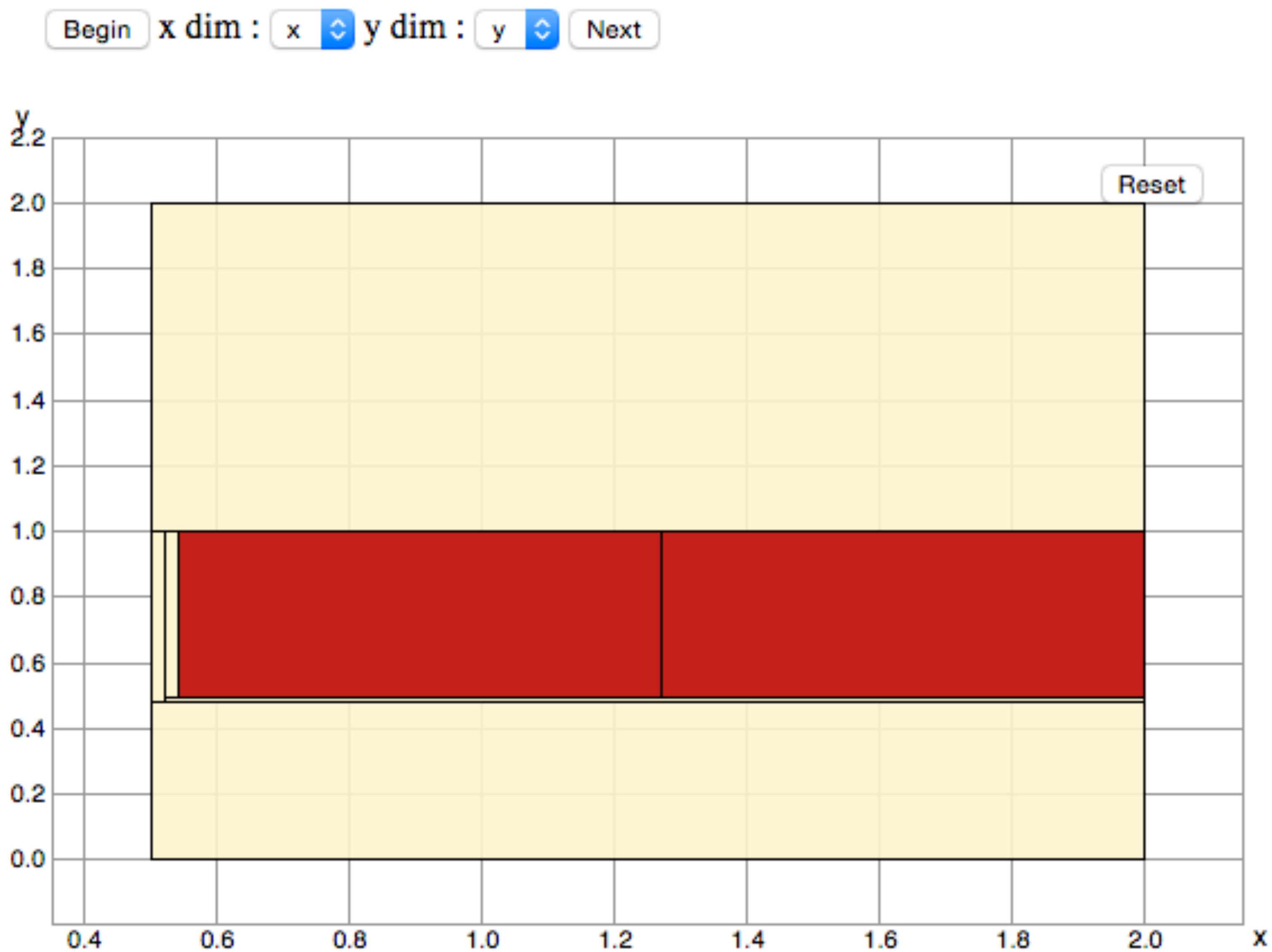
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



After steps, pruning reaches a fixed point.

Example of ICP

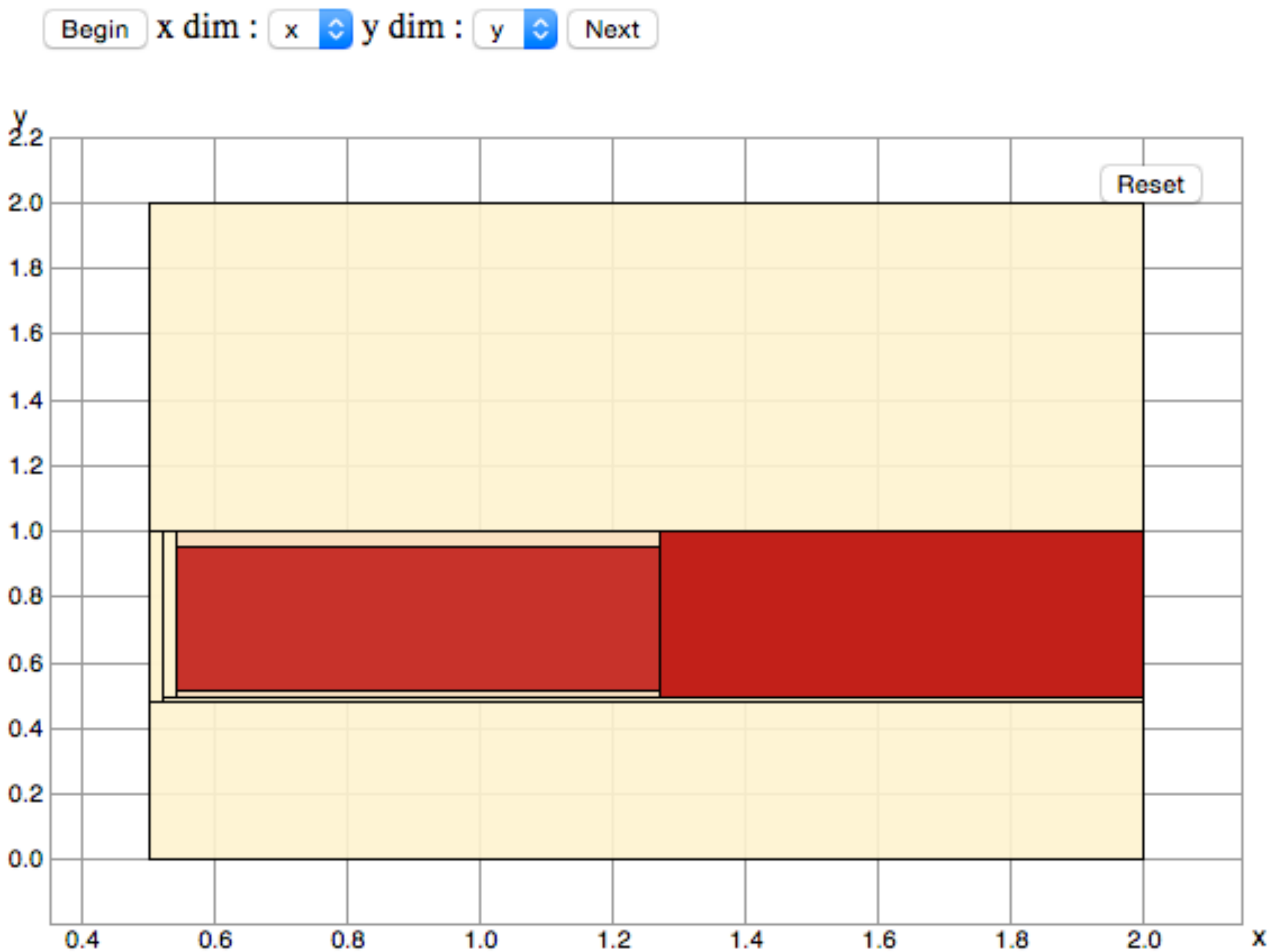
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



Branching on X

Example of ICP

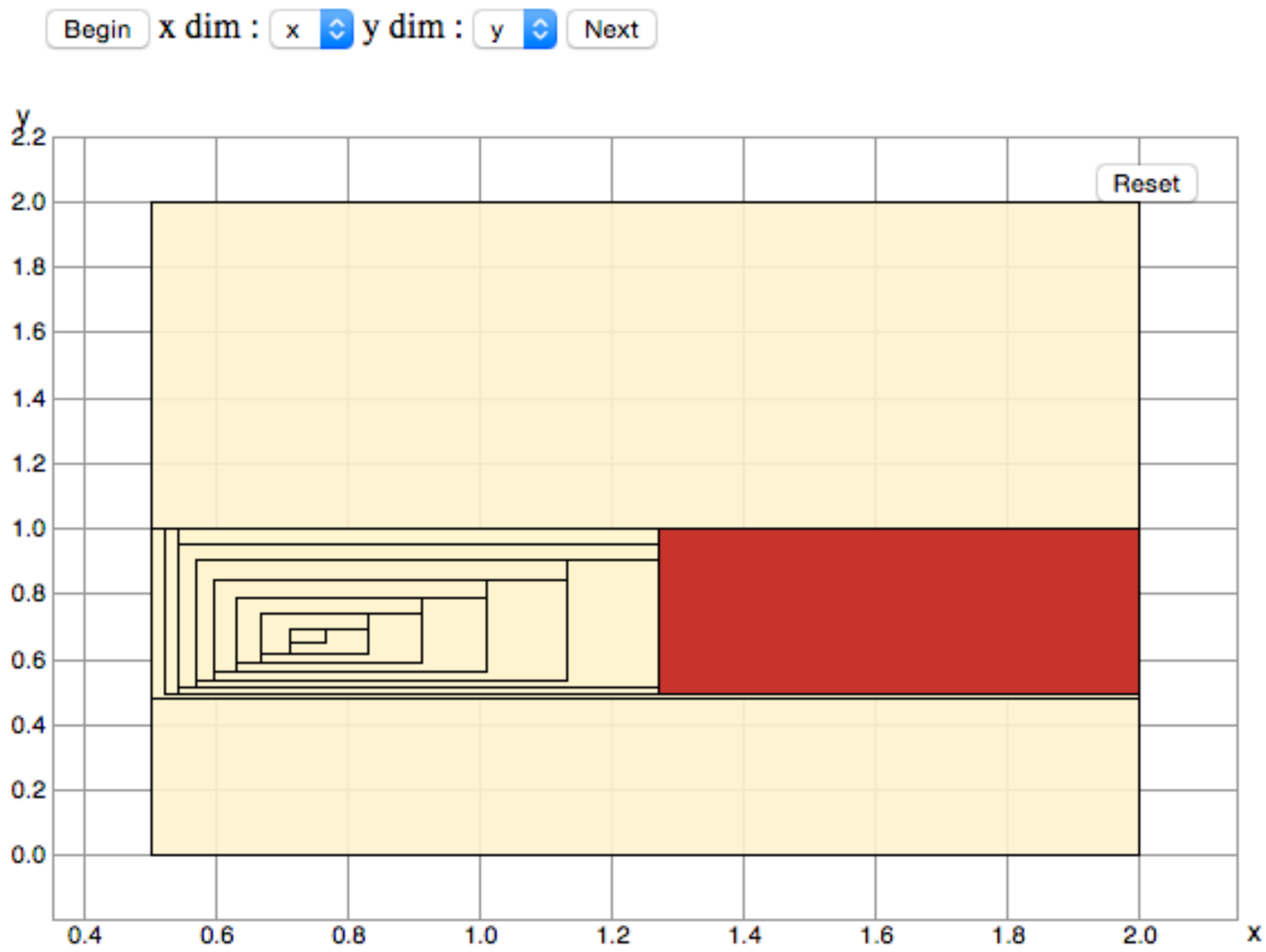
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



Apply Pruning on the Left-hand Box

Example of ICP

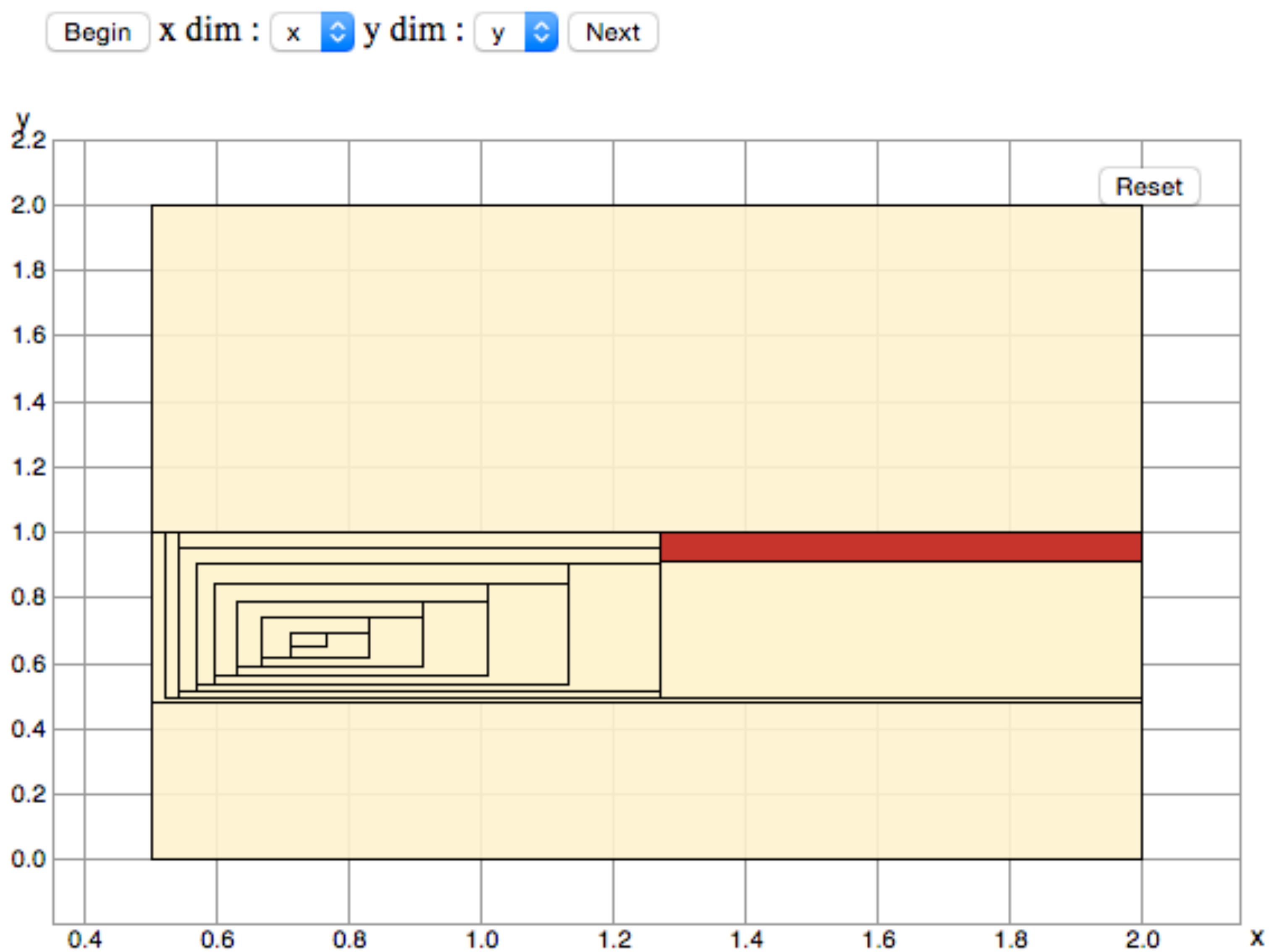
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



After pruning steps,
it shows that left-hand box contains NO solution.

Example of ICP

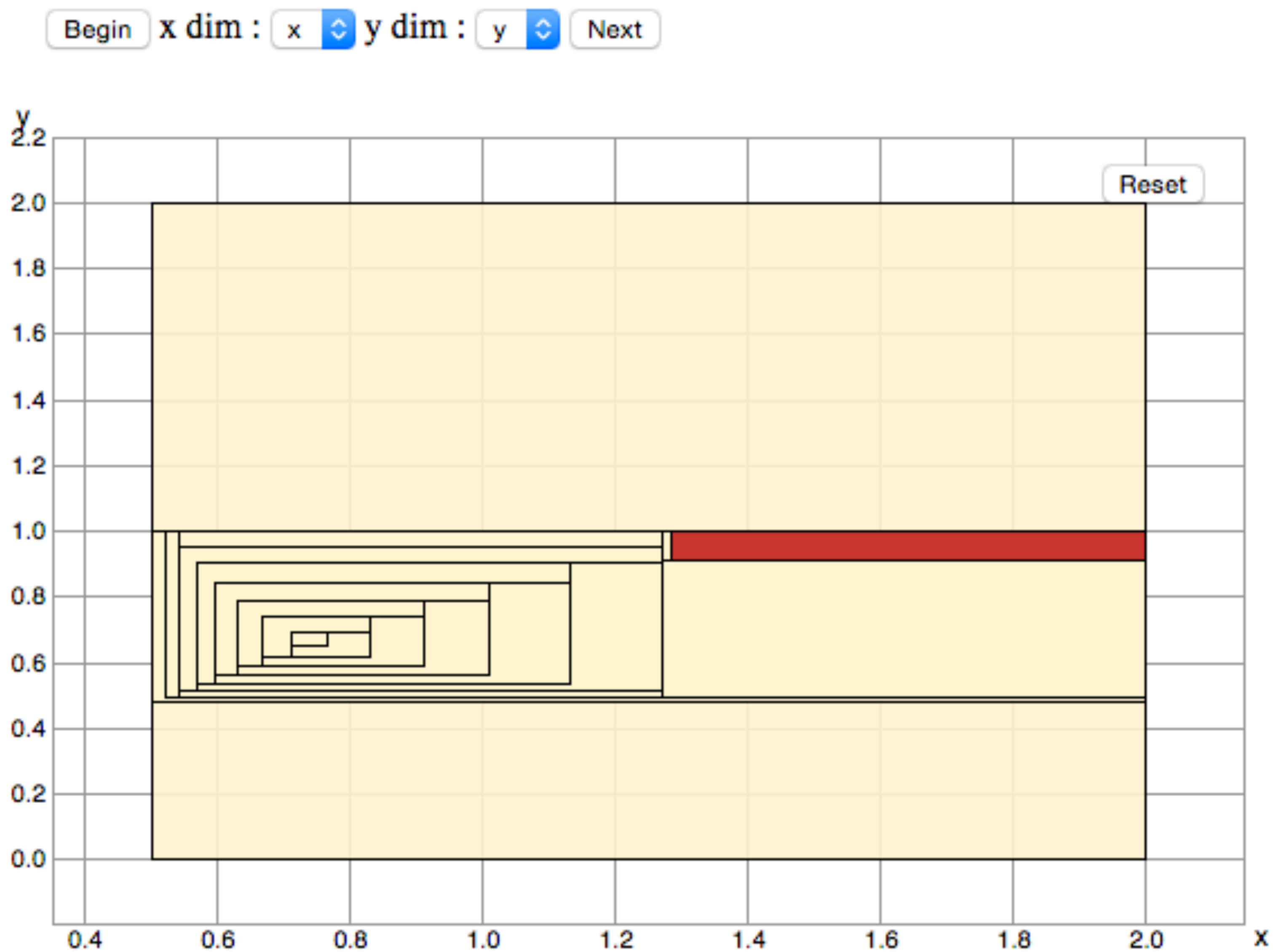
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



Apply Pruning on the Right-hand Box

Example of ICP

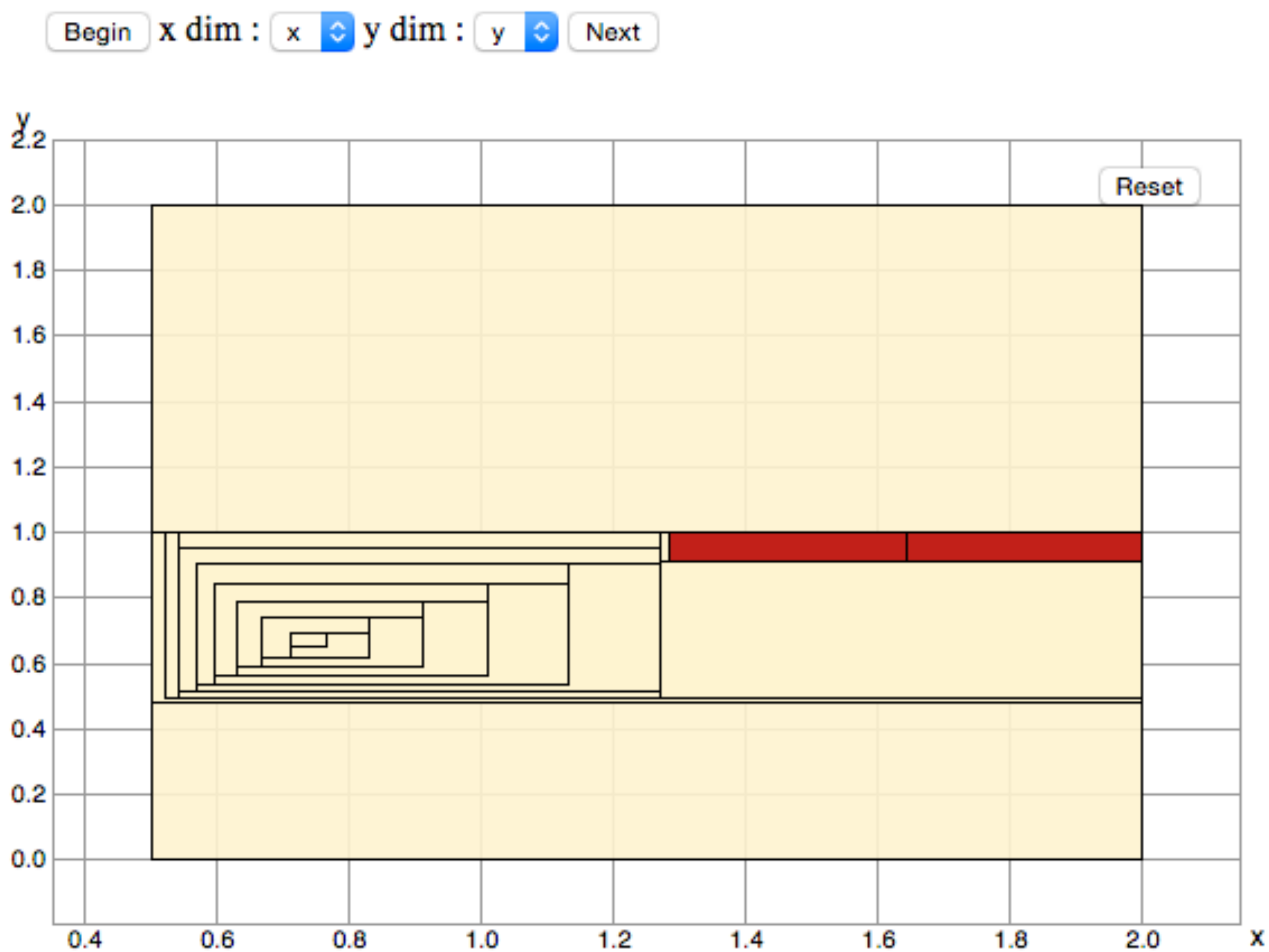
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



Apply Pruning on the Right-hand Box

Example of ICP

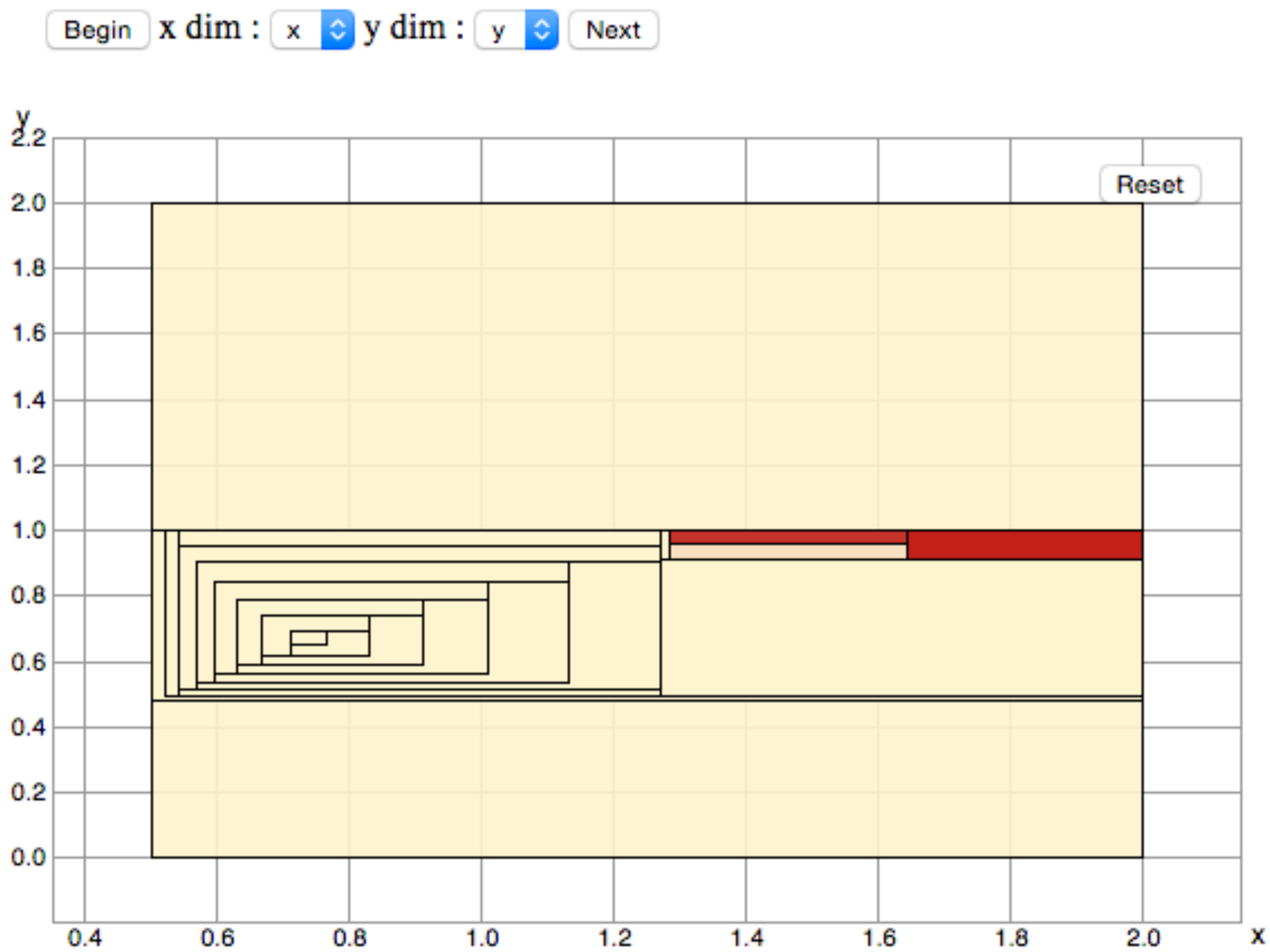
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \operatorname{atan}(x)$$



Branching on X

Example of ICP

$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$

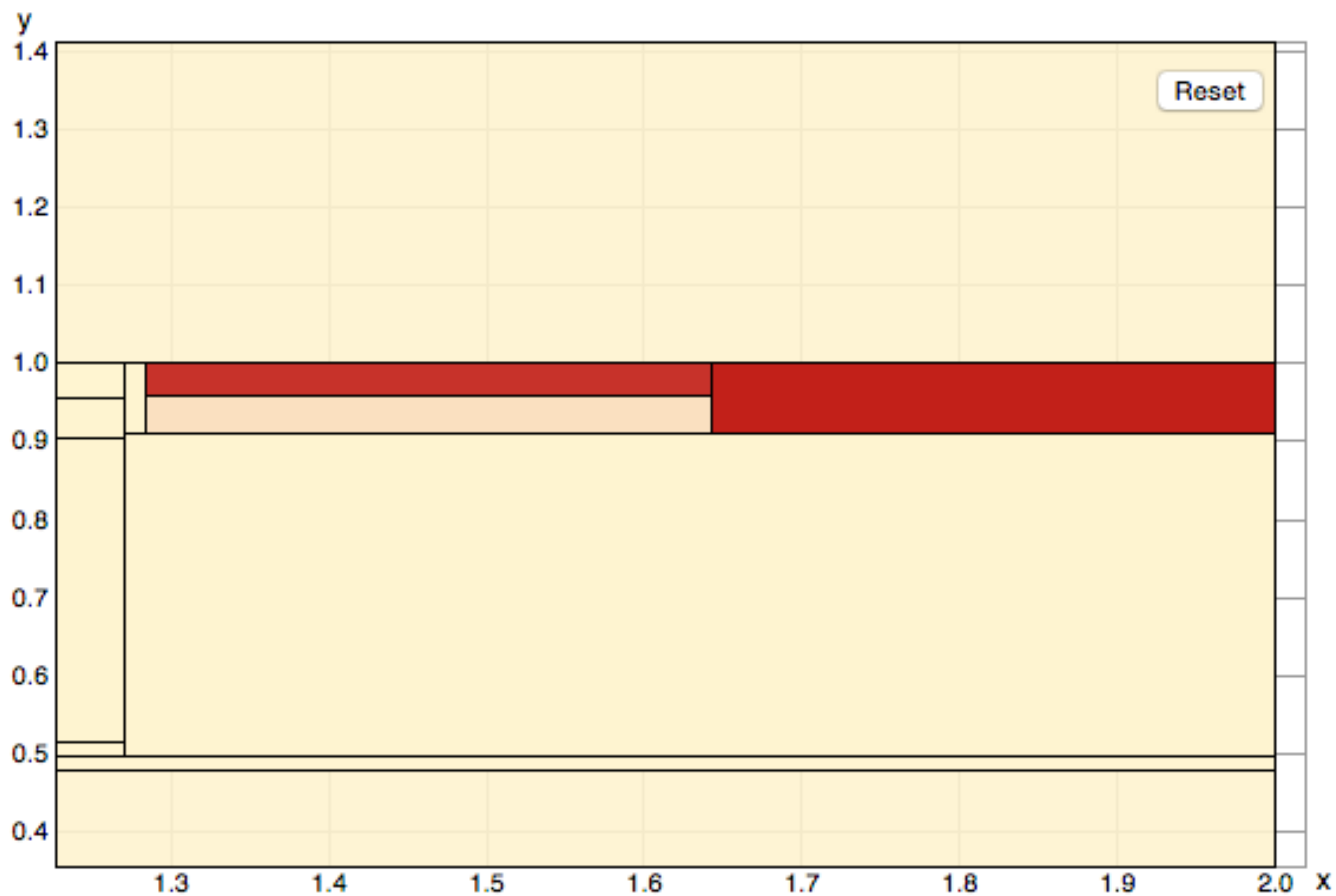


Apply Pruning on the Right-hand Box

Example of ICP

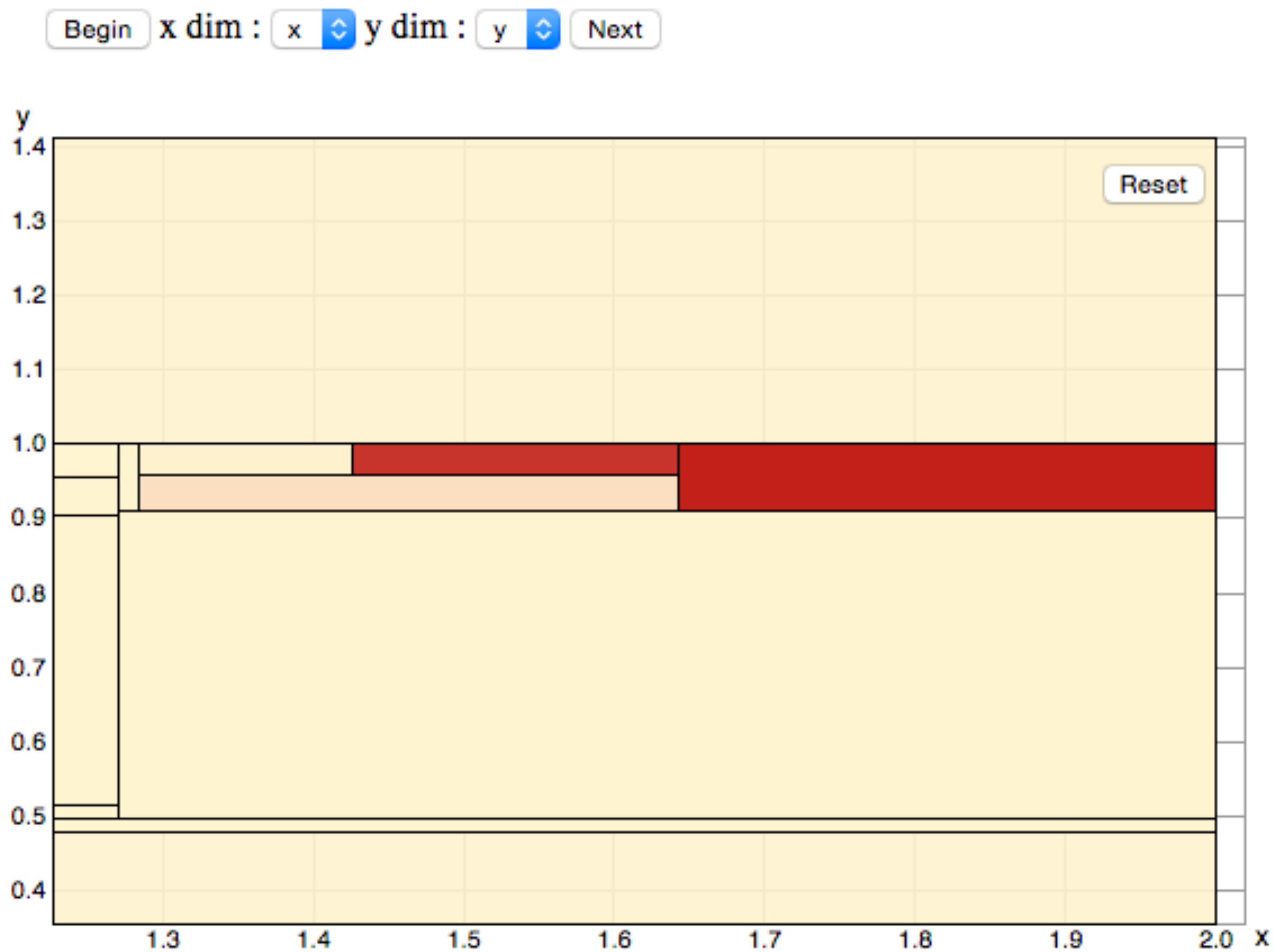
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$

Begin x dim : x y dim : y Next



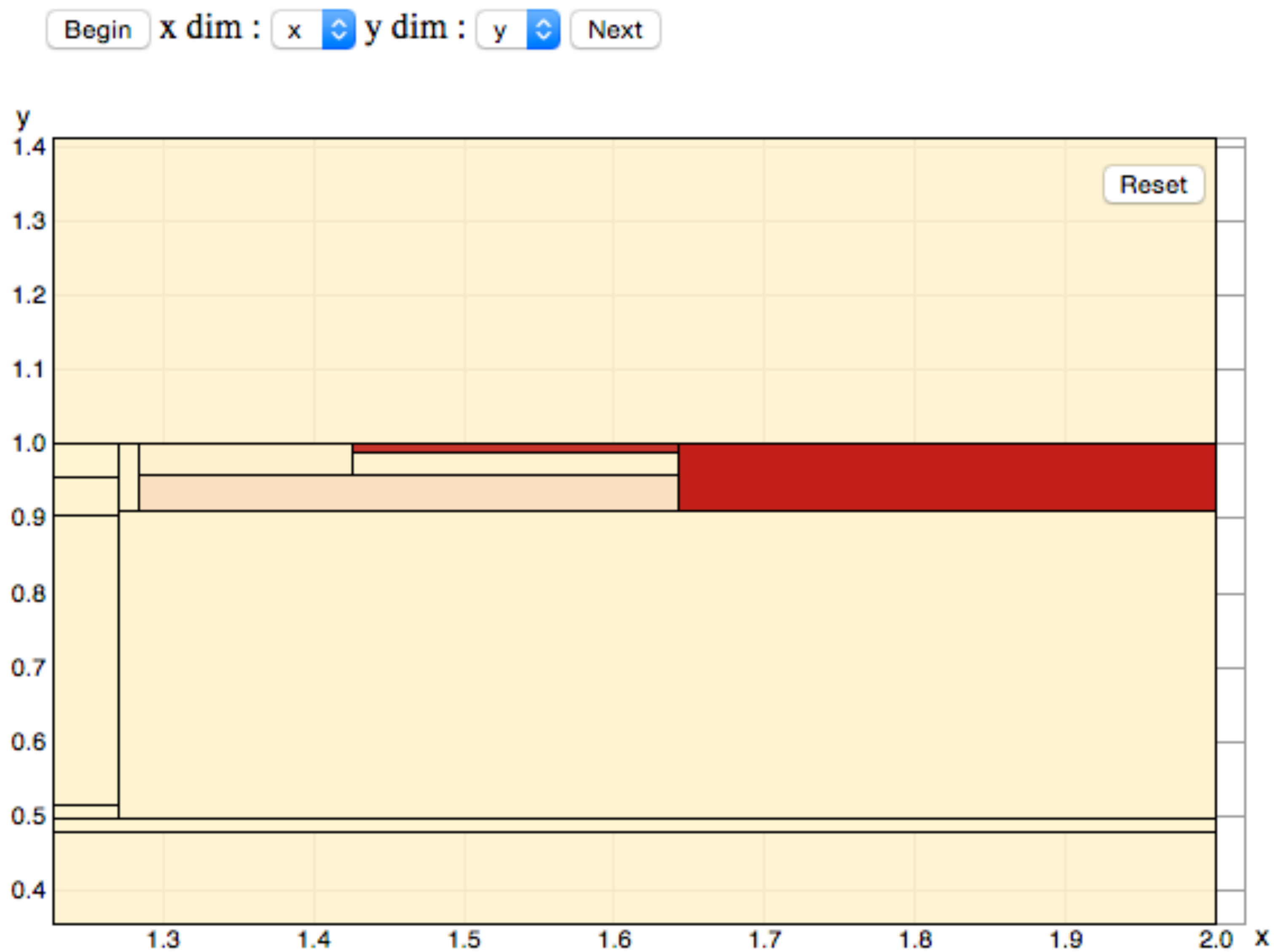
Example of ICP

$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



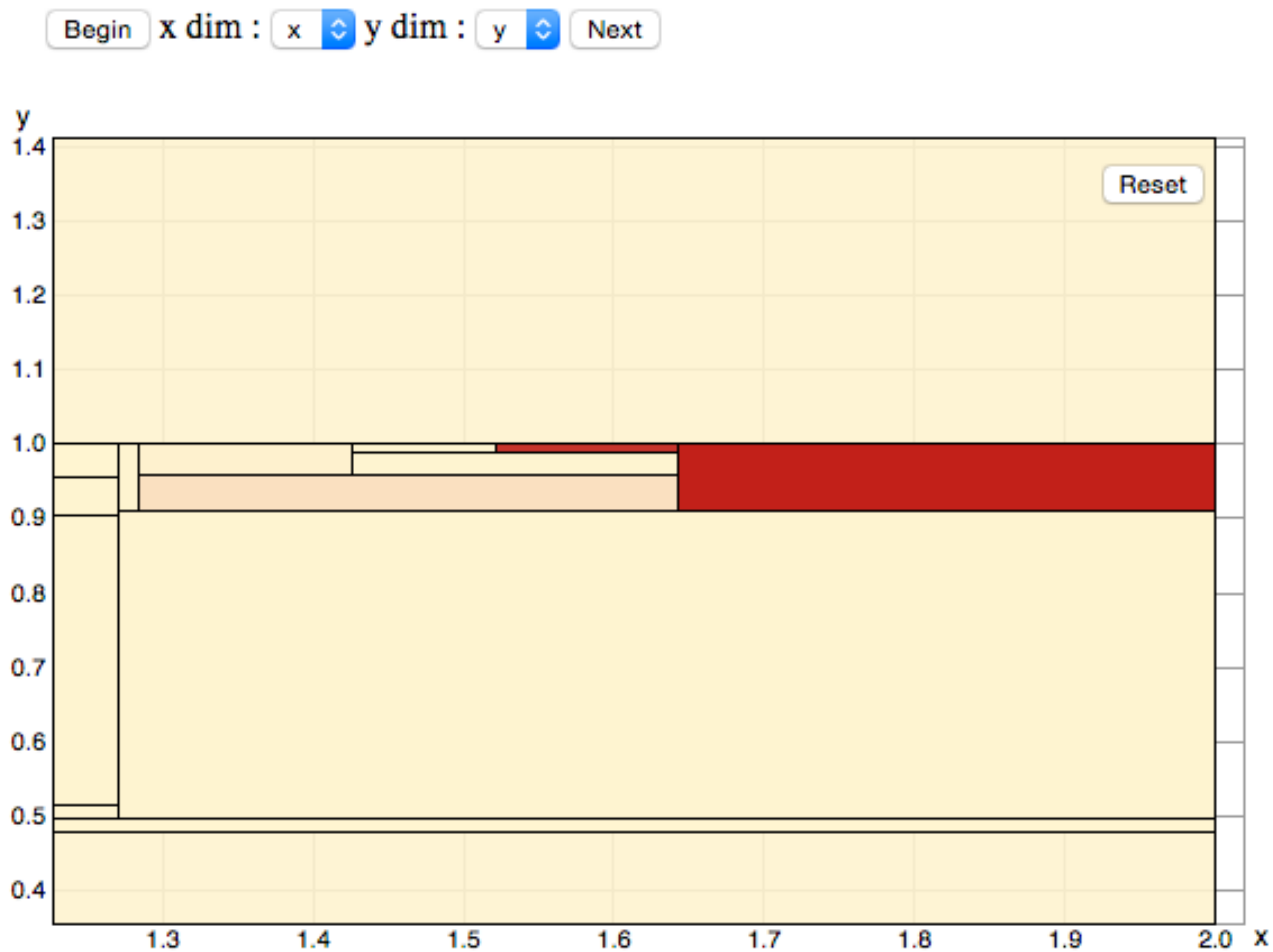
Example of ICP

$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \text{atan}(x)$$



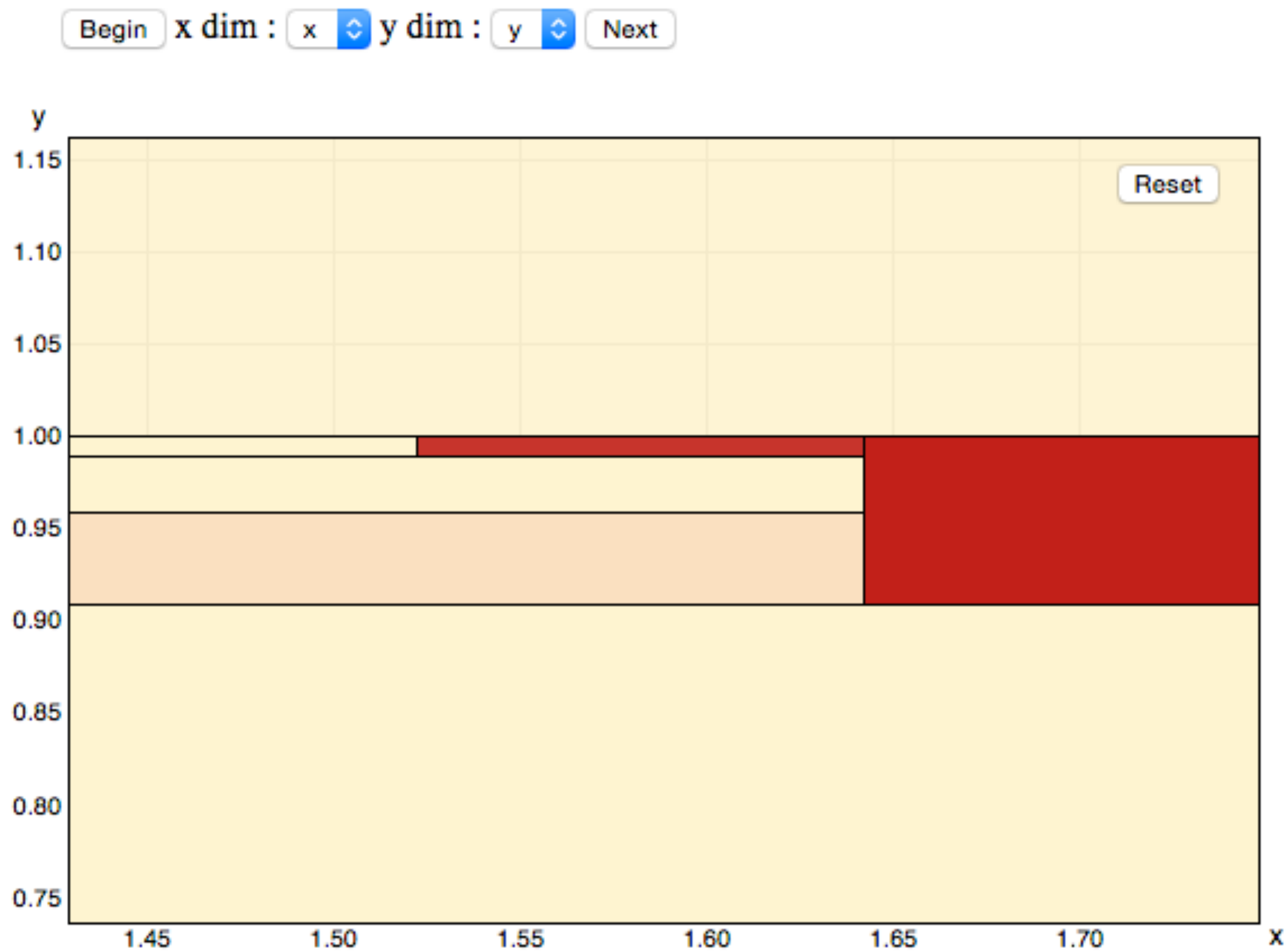
Example of ICP

$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \operatorname{atan}(x)$$



Example of ICP

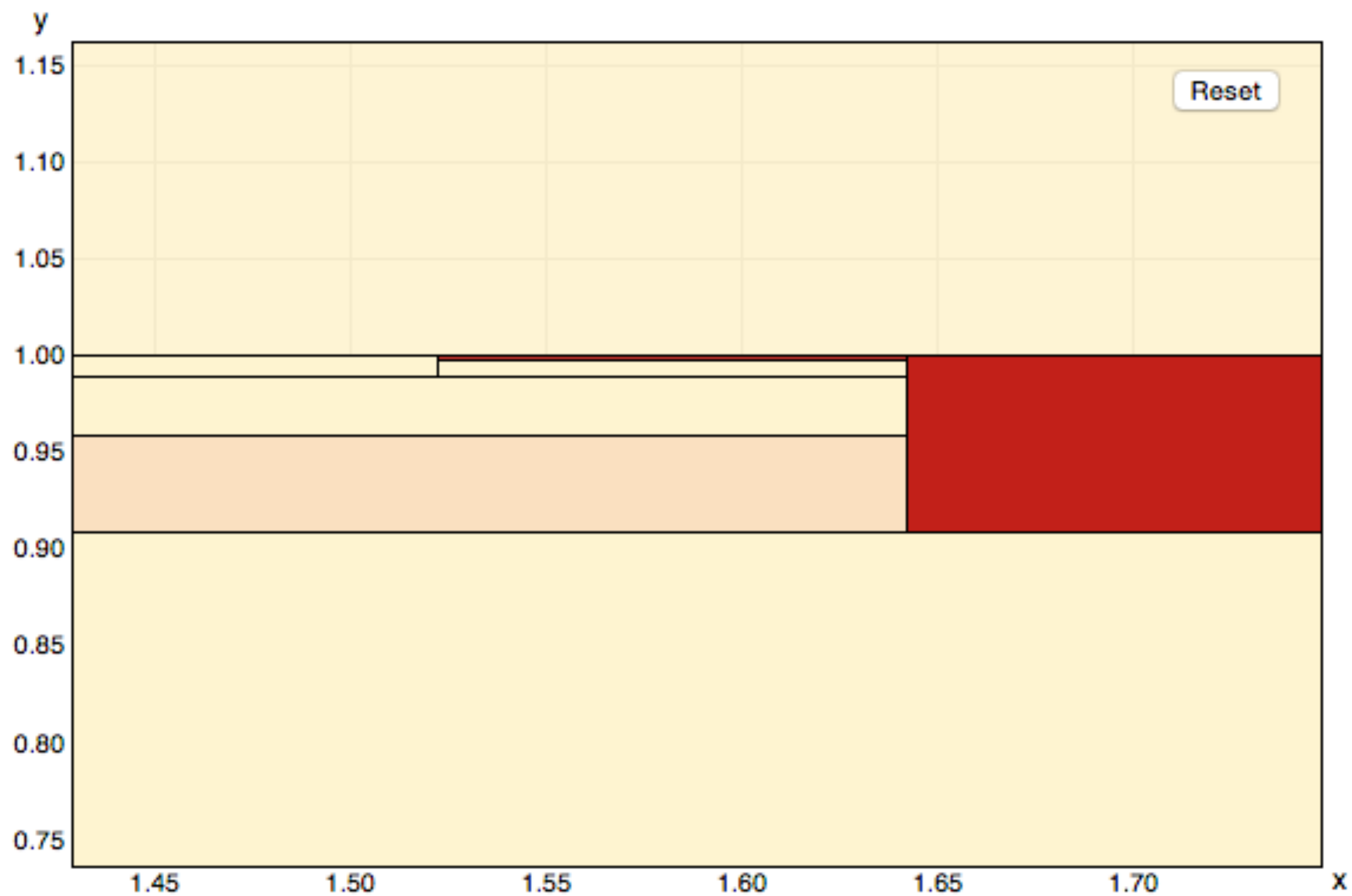
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



Example of ICP

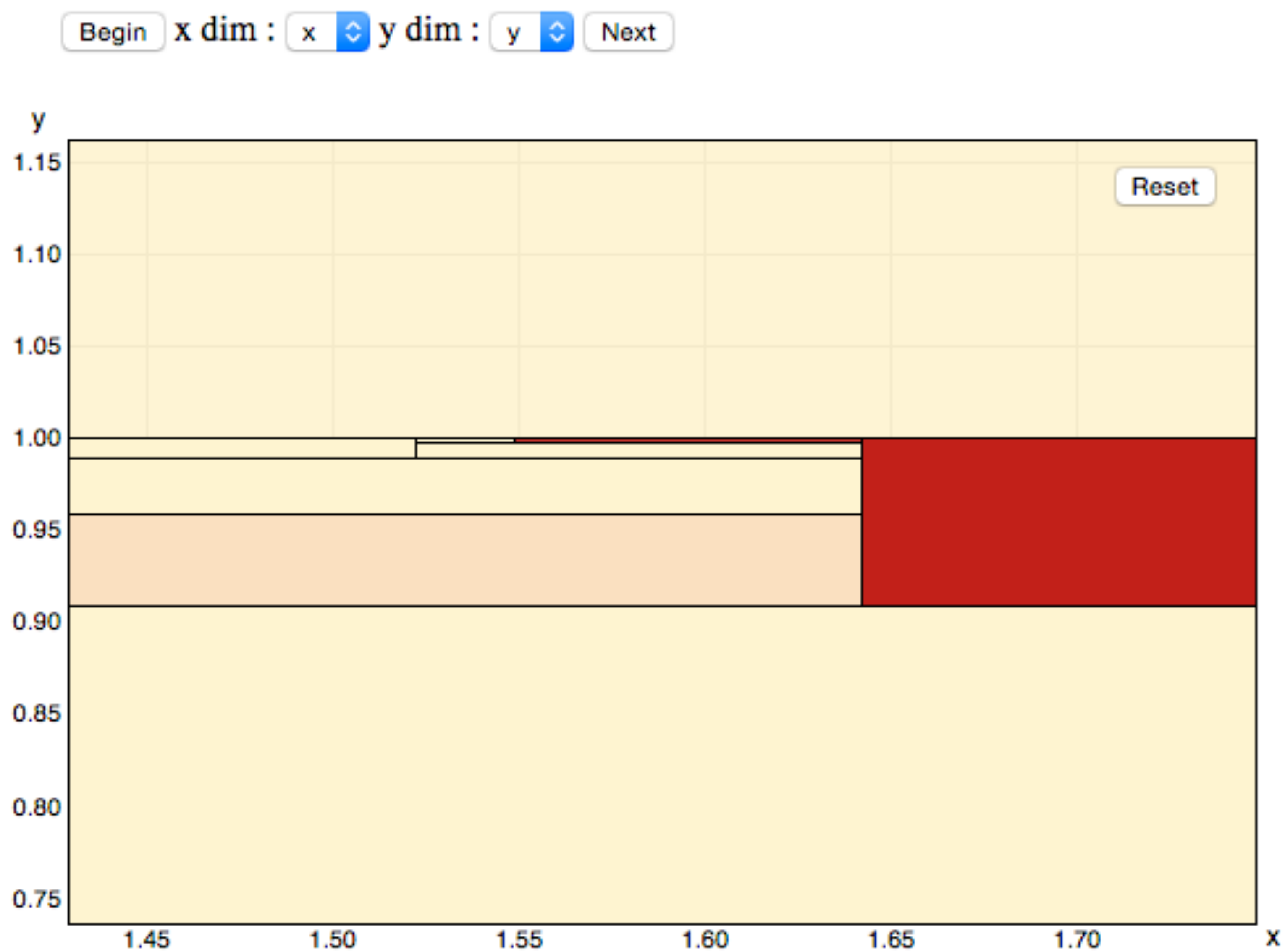
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$

Begin x dim : x y dim : y Next



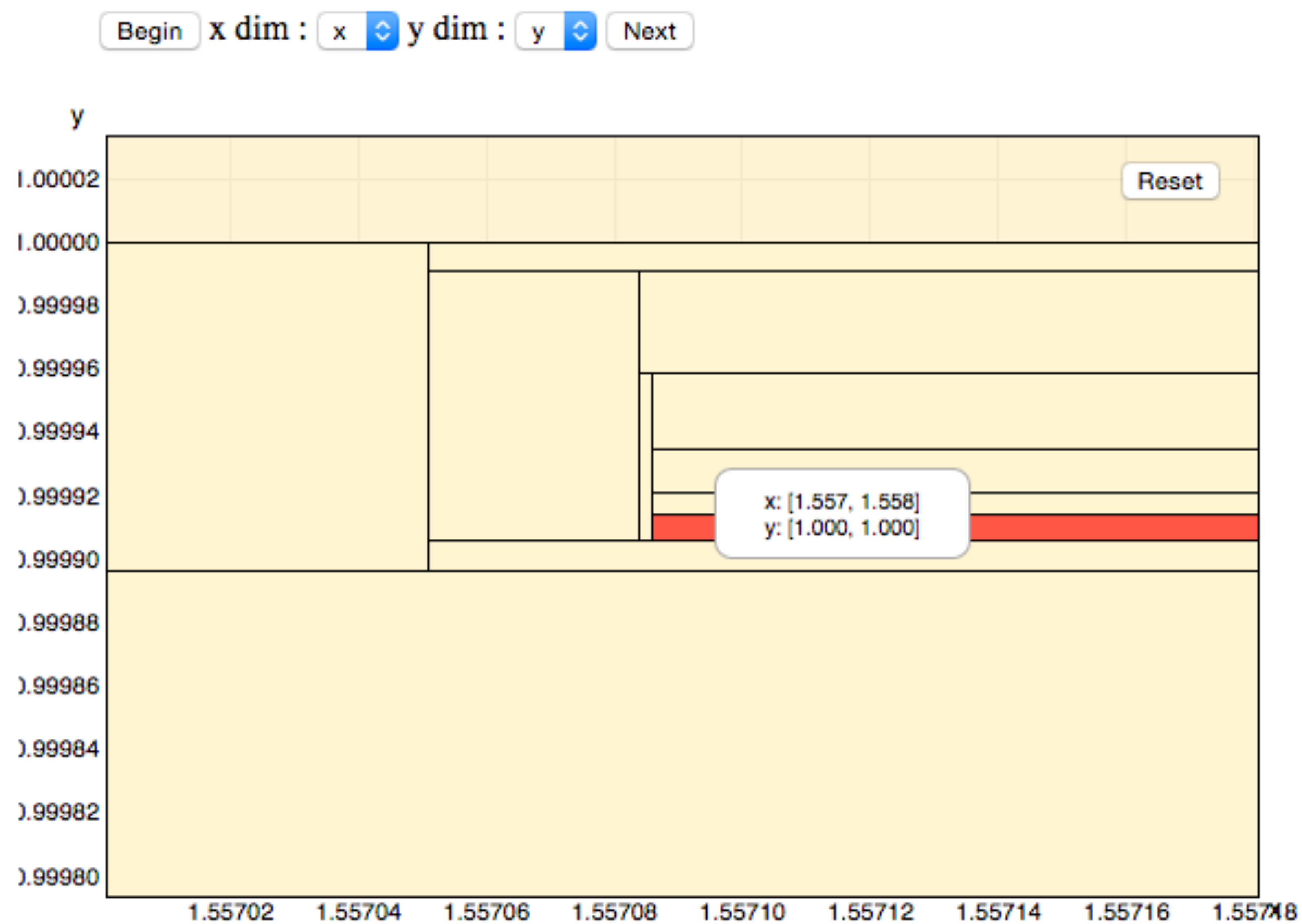
Example of ICP

$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



Example of ICP

$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \land y = \mathrm{atan}(x)$$



Found a small enough Box (width <= 0.001)

Answer: delta-SAT

Algorithm of ICP

Algorithm 1: Theory Solving in DPLL$\langle$ICP$\rangle$

input : A conjunction of theory atoms, seen as constraints, $c_1(x_1, ..., x_n), ..., c_m(x_1, ..., x_n)$, the initial interval bounds on all variables $B^0 = I_1^0 \times \cdots \times I_n^0$, box stack $S = \emptyset$, and precision $\delta \in \mathbb{Q}^+$.

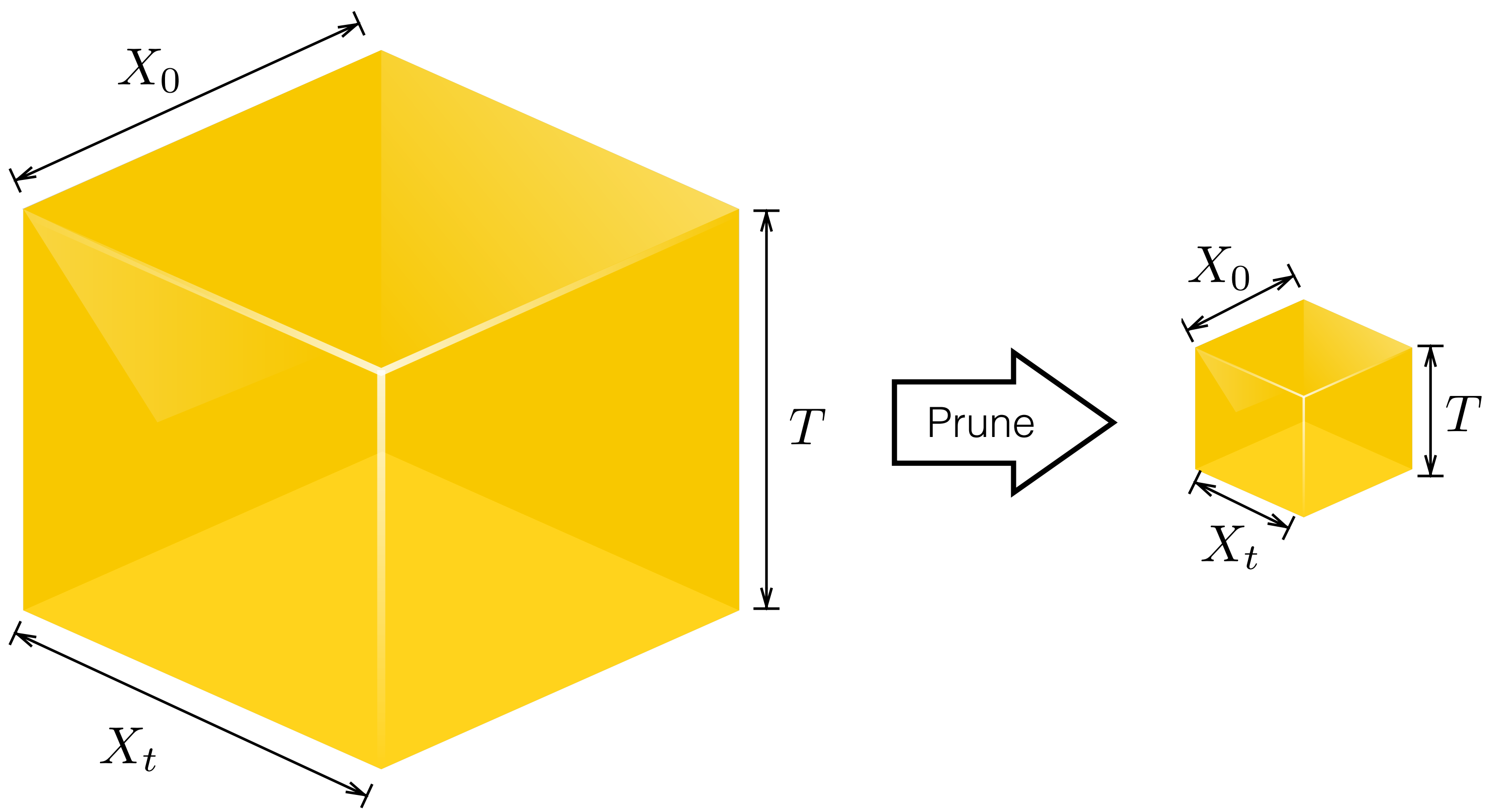
output: δ-sat, or unsat with learned conflict clauses.

S.push(B_0);
while $S \neq \emptyset$ **do**
    $B \leftarrow S$.pop() ;
    while $\exists 1 \leq i \leq m, B \neq \text{Prune}(B, c_i)$ **do**
        `//Pruning without branching, used as the assert() function.`
        $B \leftarrow \text{Prune}(B, c_i)$;
    end
    `//The` ε `below is computed from` δ `and the Lipschitz constants of functions beforehand.`
    if $B \neq \emptyset$ **then**
        if $\exists 1 \leq i \leq n, |I_i| \geq \varepsilon$ **then**
            $\{B_1, B_2\} \leftarrow \text{Branch}(B, i)$; `//Splitting on the intervals`
            S.push($\{B_1, B_2\}$);
        else
            return δ-sat; `//Complete check() is successful.`
        end
    end
end
return unsat;

Pruning

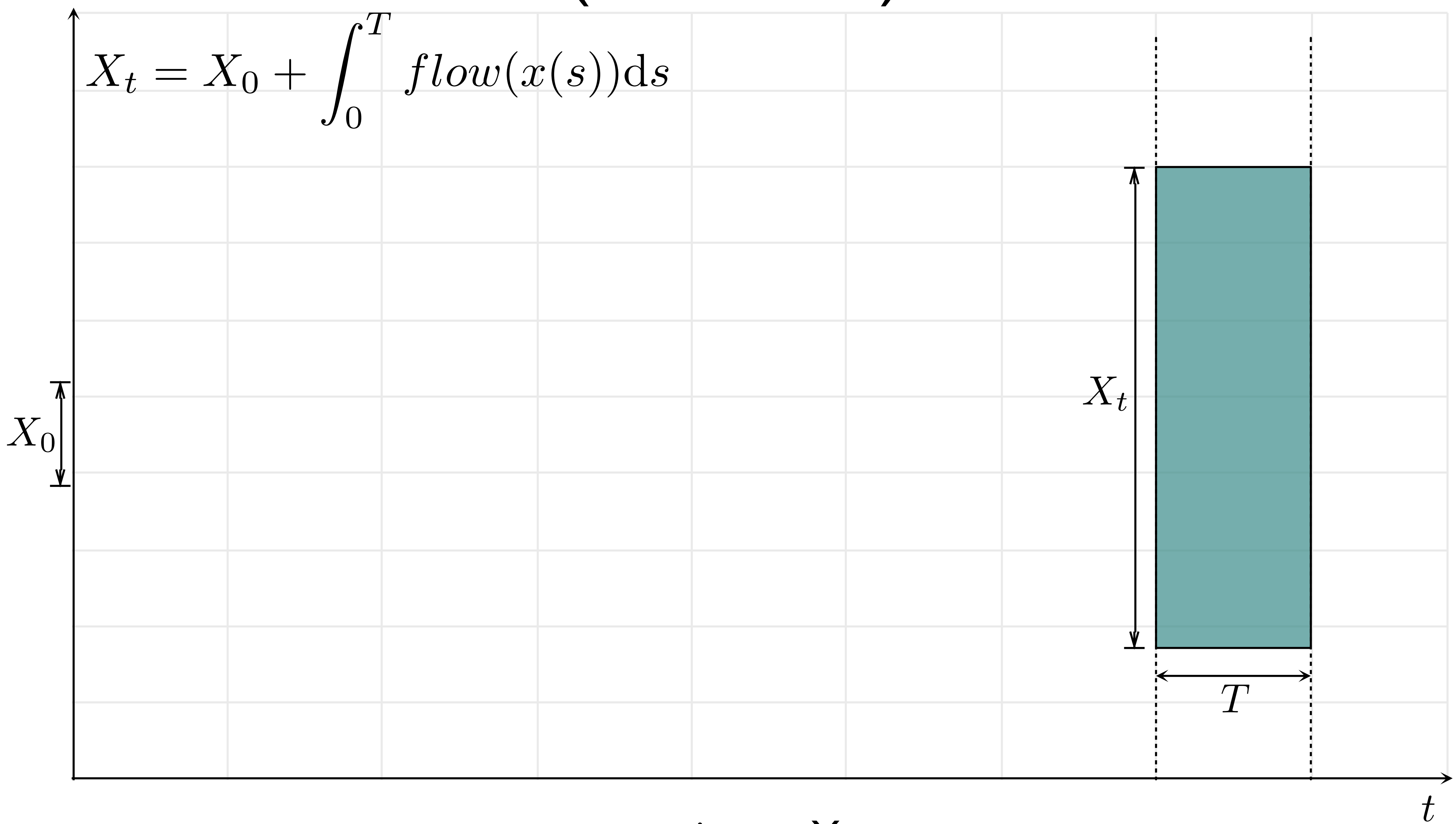
Branching

Pruning using ODEs



$$X_t = X_0 + \int_0^T flow(x(s))\mathrm{d}s$$

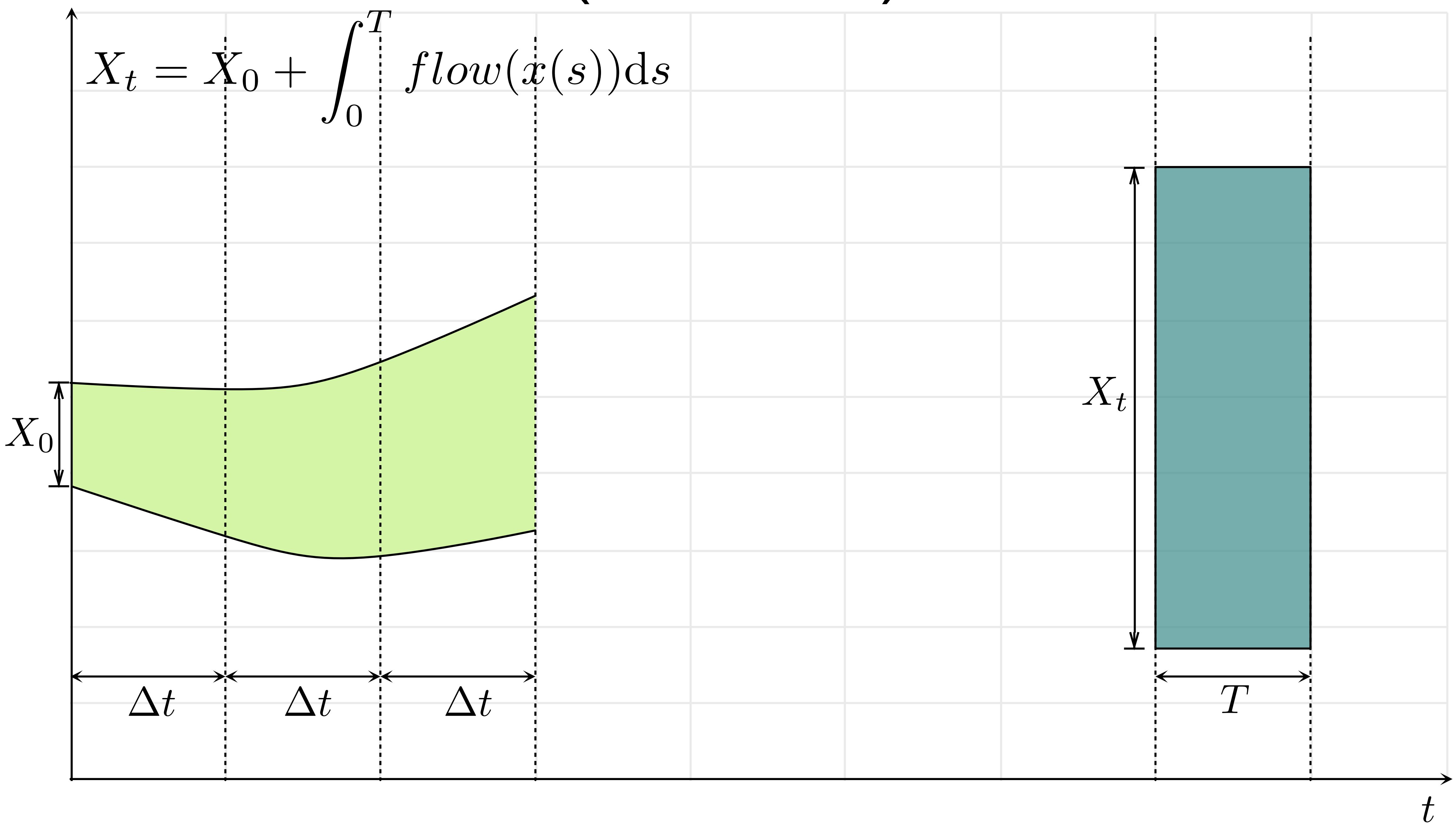
Pruning using ODEs (Forward)



pruning on Xt

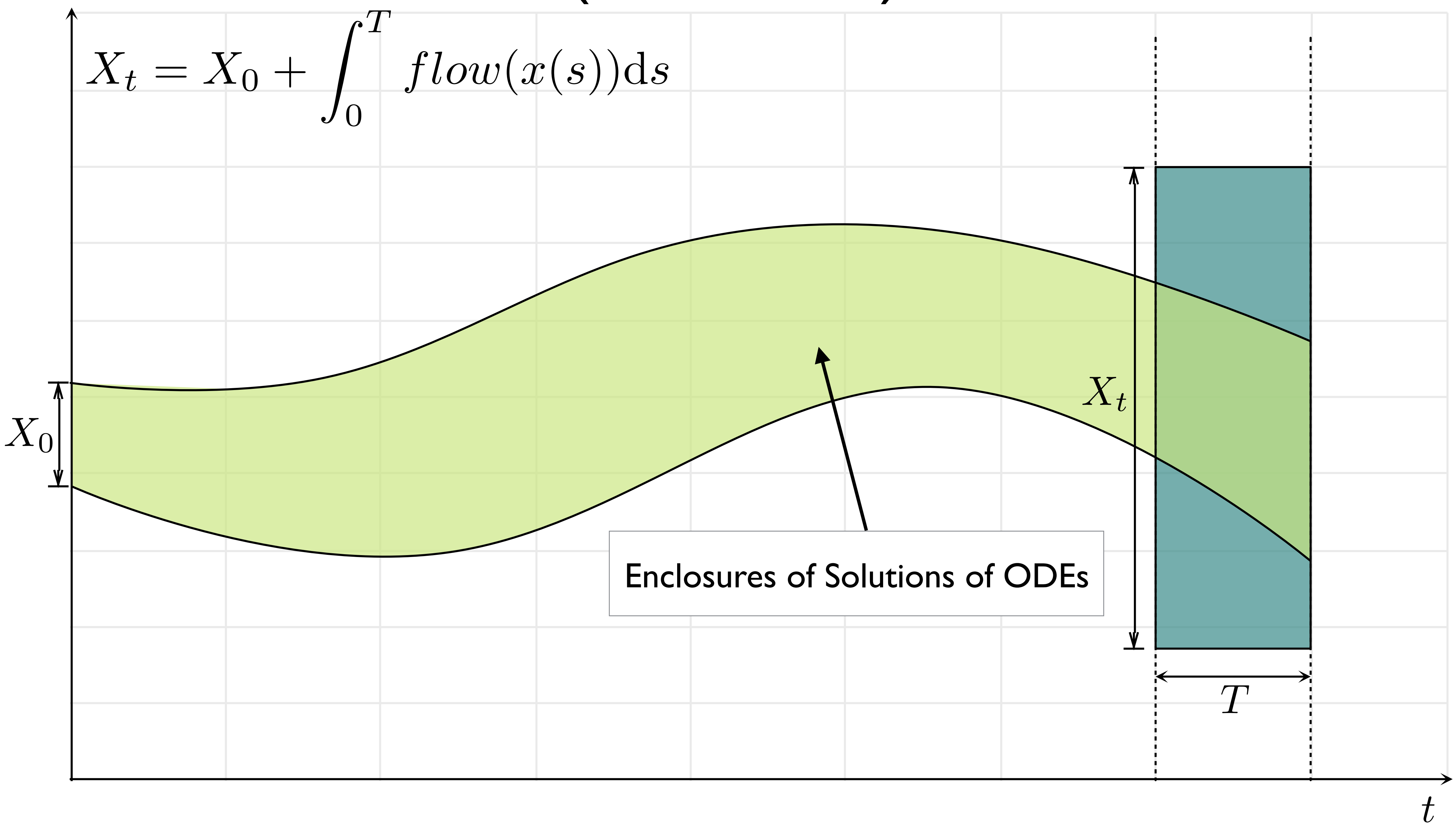
How can we prune X_t?

Pruning using ODEs (Forward)



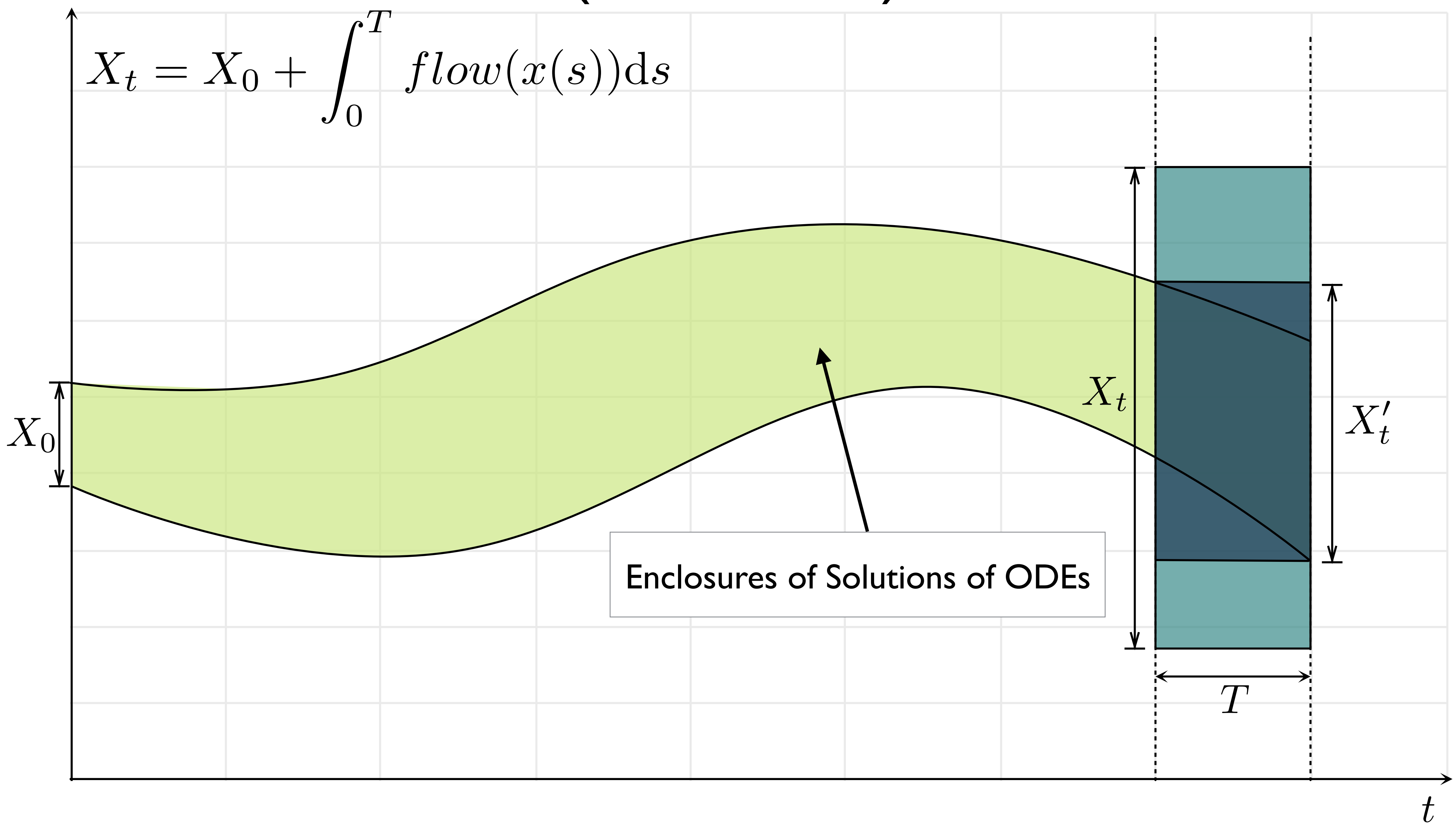
(numerically) Compute the enclosures of the solutions of ODE

Pruning using ODEs (Forward)



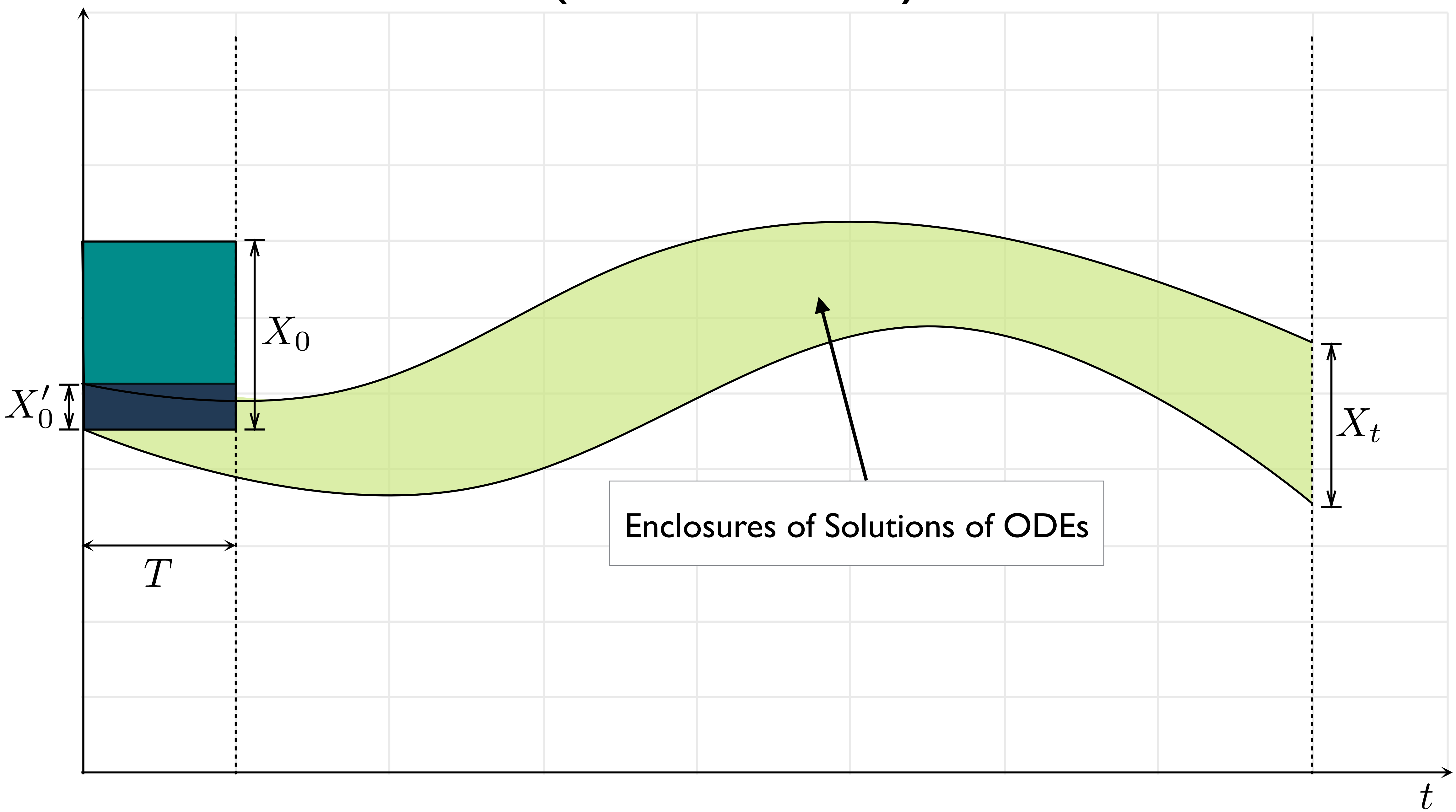
(numerically) Compute the enclosures of the solutions of ODE

Pruning using ODEs (Forward)



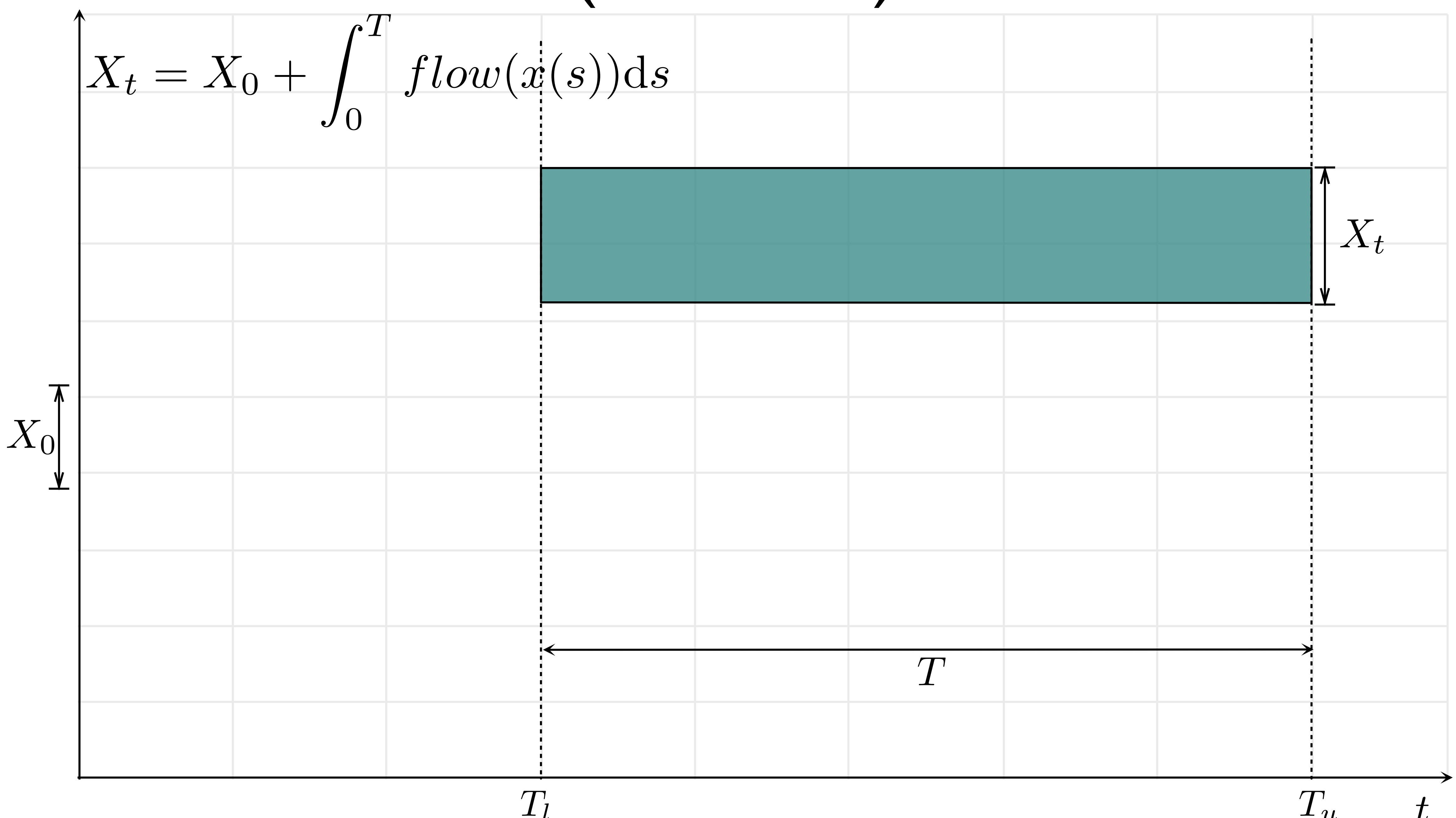
Take the intersection between the Enclosure and Xt

Pruning using ODEs (Backward)



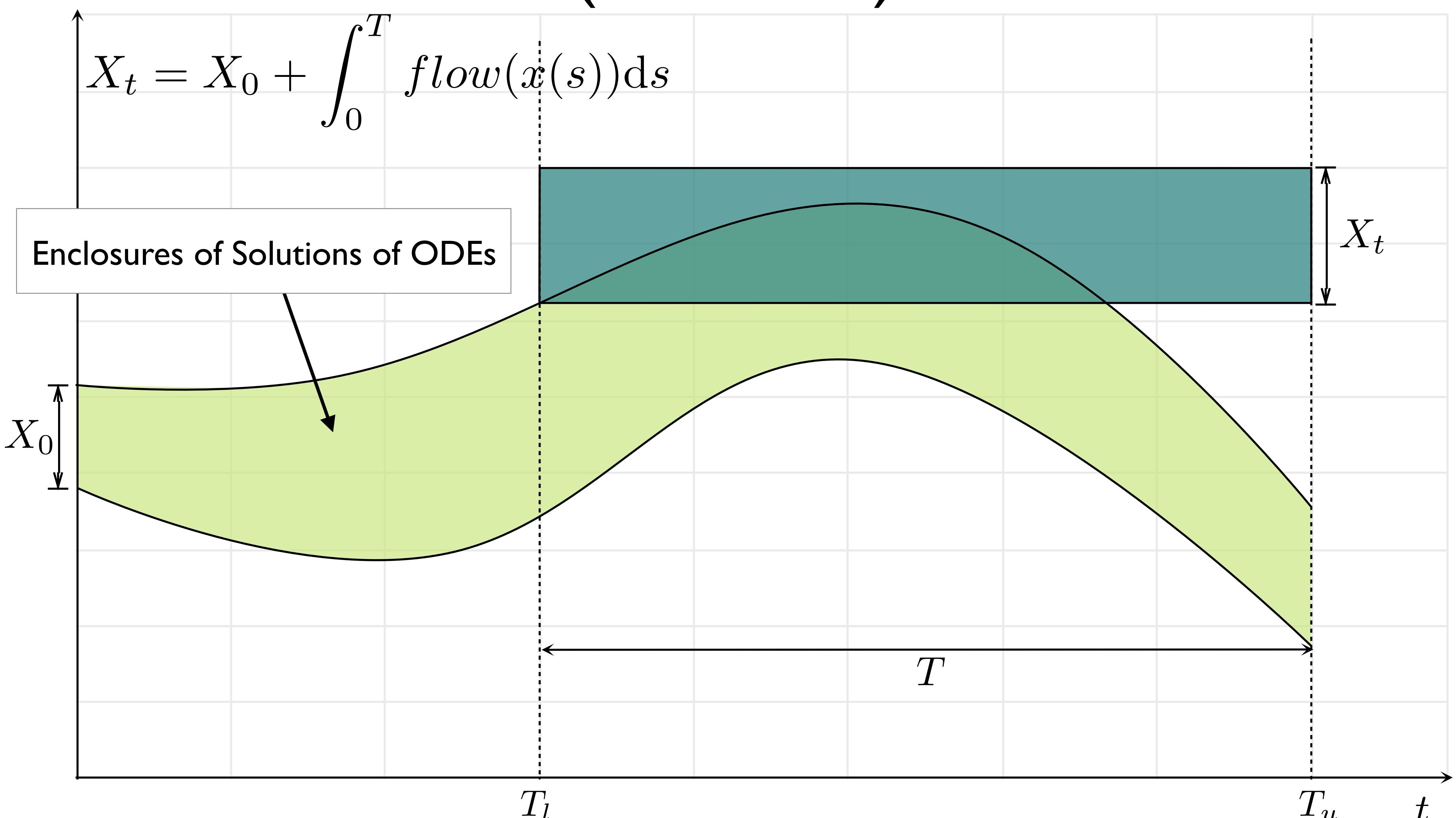
Pruning on X0

Pruning using ODEs (on Time)



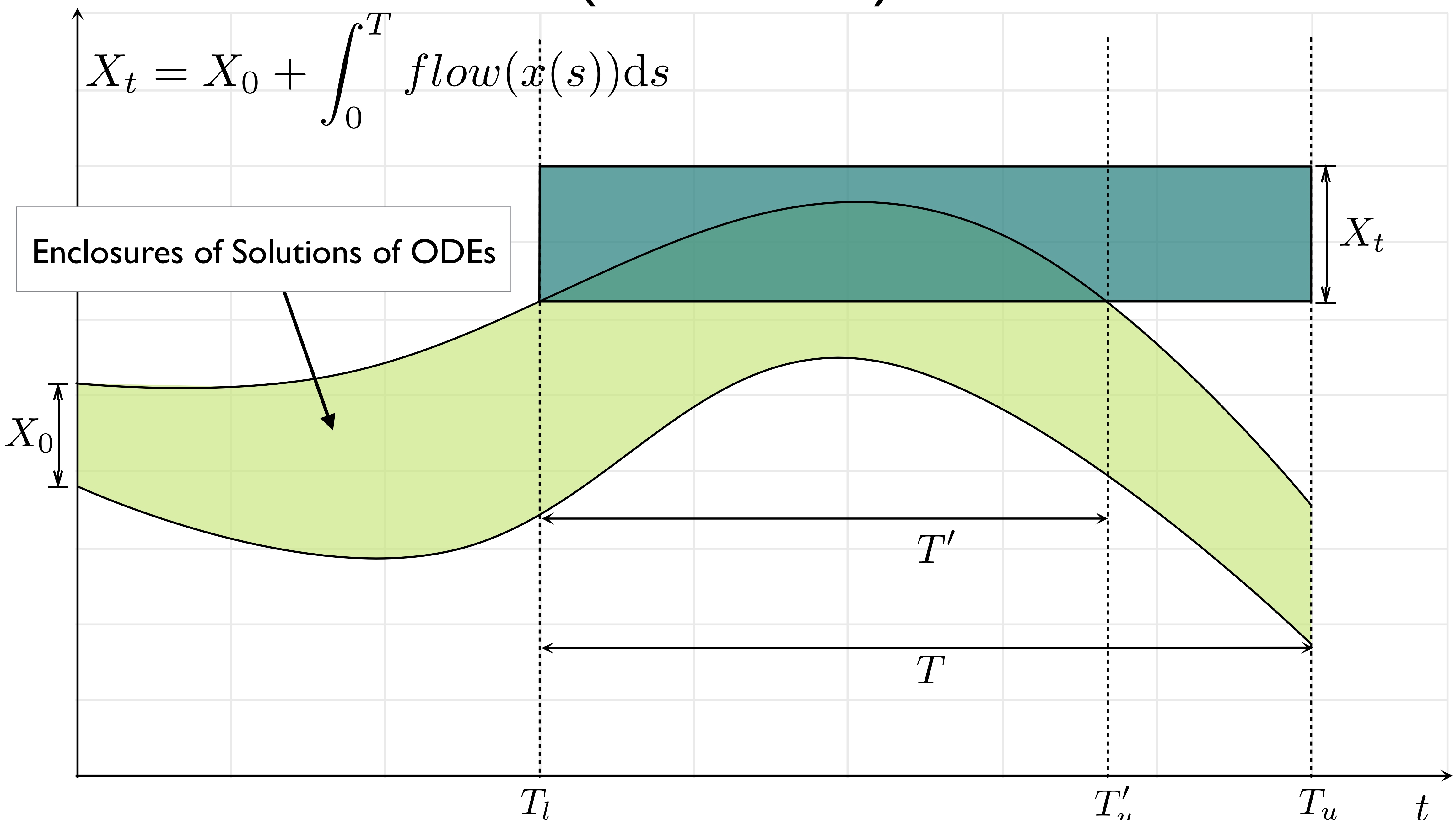
Pruning on T

Pruning using ODEs (on Time)



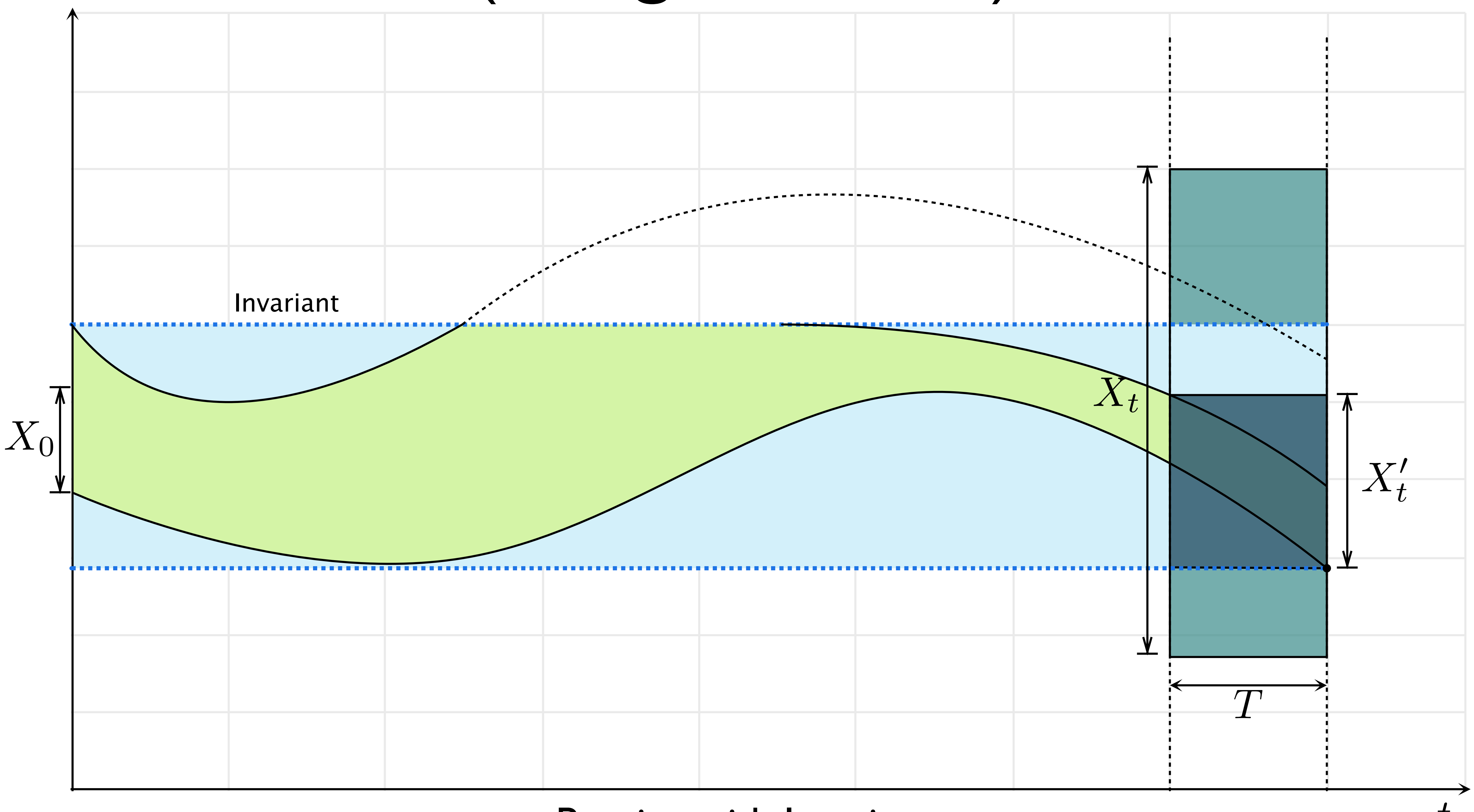
Pruning on T

Pruning using ODEs (on Time)



Pruning on T

Pruning using ODEs (using Invariant)



Pruning with Invariant

Implementation: dReal

Fork me on GitHub

- Automated reasoning tool over the Reals
- Support **nonlinear real functions** such a sin, cos, tan, arcsin, arccos, arctan, log, exp, …
- Support **ODEs** (Ordinary Differential Equations)
- Generating **proofs for UNSAT** cases [experimental]
- **Open-source**: https://dreal.github.io

Applications

* Power-train Control (Toyota) [HSCC'14, ACC'15]
* Autonomous Driving (Penn) [SAE'16]
* Planning (CMU,SIFT) [AAAI'15]
* Security (MIT, TAMU, QCRI) [CDC'15]
* Atrial Fibrillation (Stony Brook, TU, CMU) [HSCC'15, CMSB'14]
* Diabetes (Penn) [ADHS'15]
* Prostate Cancer (Pitt, CMU) [HSCC'15]
* Microfluid Chip Design (Waterloo)

…

Application: Powertrain Control

$$
\begin{aligned}
\dot{p} &= c_1\left(2\hat{u}_1\sqrt{\frac{p}{c_{11}} - \left(\frac{p}{c_{11}}\right)^2} - (c_3 + c_4c_2p + c_5c_2p^2 + c_6c_2^2p)\right) \\
\dot{r} &= 4\left(\frac{c_3 + c_4c_2p + c_5c_2p^2 + c_6c_2^2p}{c_{13}(c_3 + c_4c_2p_{est}^2 + c_5c_2p_{est}^2 + c_6c_2^2p_{est})(1 + i + c_{14}(r - c_{16}))} - r\right) \\
\dot{p}_{est} &= c_1\left(2\hat{u}_1\sqrt{\frac{p}{c_{11}} - \left(\frac{p}{c_{11}}\right)^2} - c_{13}\left(c_3 + c_4c_2p_{est} + c_5c_2p_{est}^2 + c_6c_2^2p_{est}\right)\right) \\
\dot{i} &= c_{15}(r - c_{16})
\end{aligned}
\qquad (21)
$$

Figure 5: System dynamics for the Powertrain Control System.

recently in the literature [11]. There has been interest in adopting MPC in the automotive industry, but several hurdles remain, such as the ability to prove safety properties of the closed-loop system. A technique that provides a means to, for example, prove stability or to provide guarantees on performance bounds would help to ease the way for this new technology to find application in industry. Below, we apply our technique to prove this system is stable by discovering a discrete-time Lyapunov function that is valid over a given domain.

The purpose of the MPC system for this application is to regulate the manifold pressure (MAP) and exhaust gas recirculation (EGR) rate. The MAP affects the amount of air injected into the cylinder for the combustion phase of the engine; accurately controlling the MAP directly affects

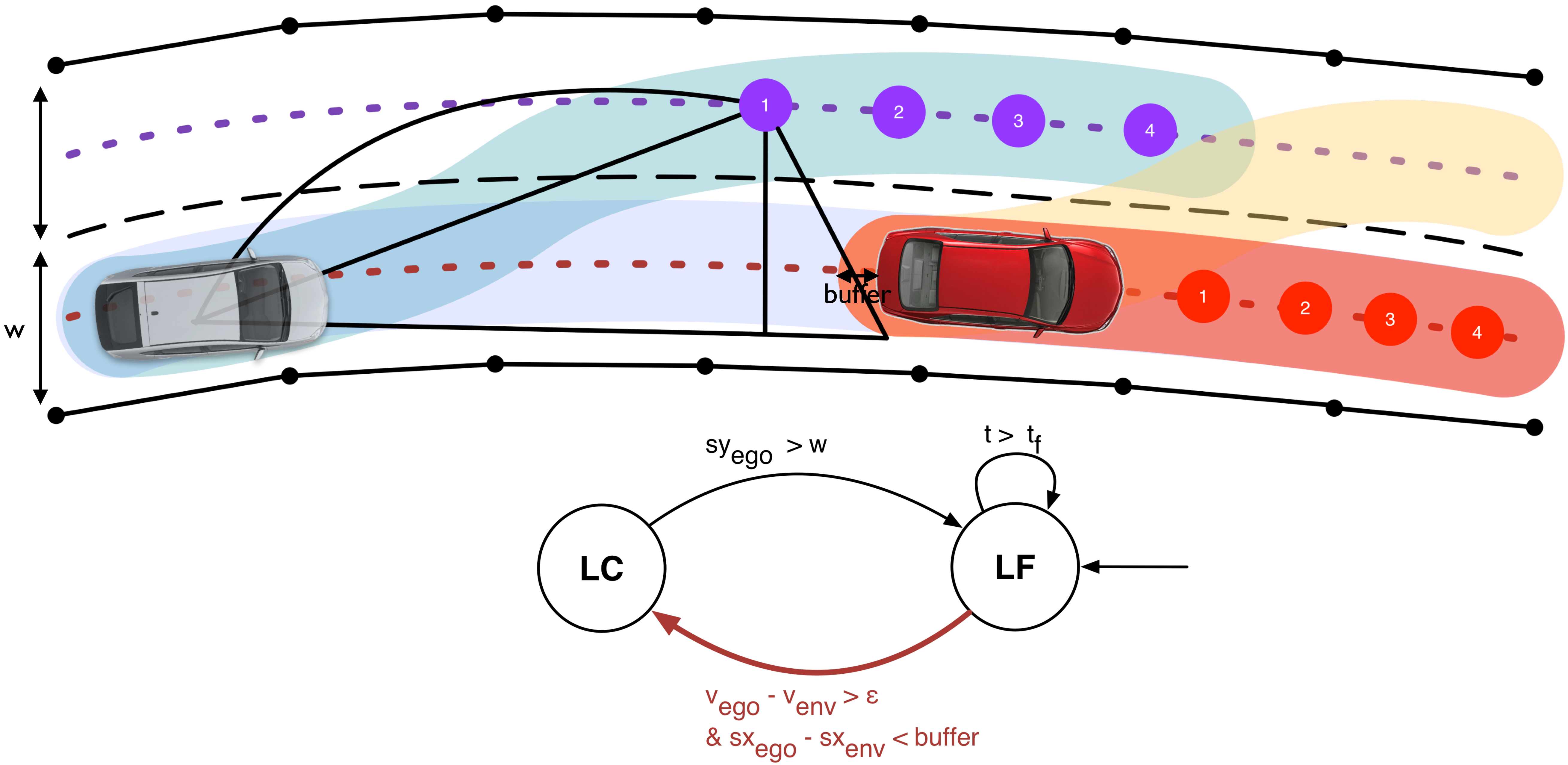
We use a quadratic Lyapunov template and define the domain as the ball of radius 20.0 centered at the origin. The search procedure produces the following Lyapunov candidate in 107.29 seconds:

$$
\mathbf{P} = \begin{bmatrix} 1.625 & -0.309 & 0.740 \\ -0.309 & 0.886 & 0.208 \\ 0.740 & 0.208 & 1.688 \end{bmatrix}.
$$

A query to dReal takes 133 seconds to prove that the resulting candidate Lyapunov function is a proper Lyapunov function over the domain. This provides a proof of stability as well as a mechanism to produce forward invariant sets for the MPC system.

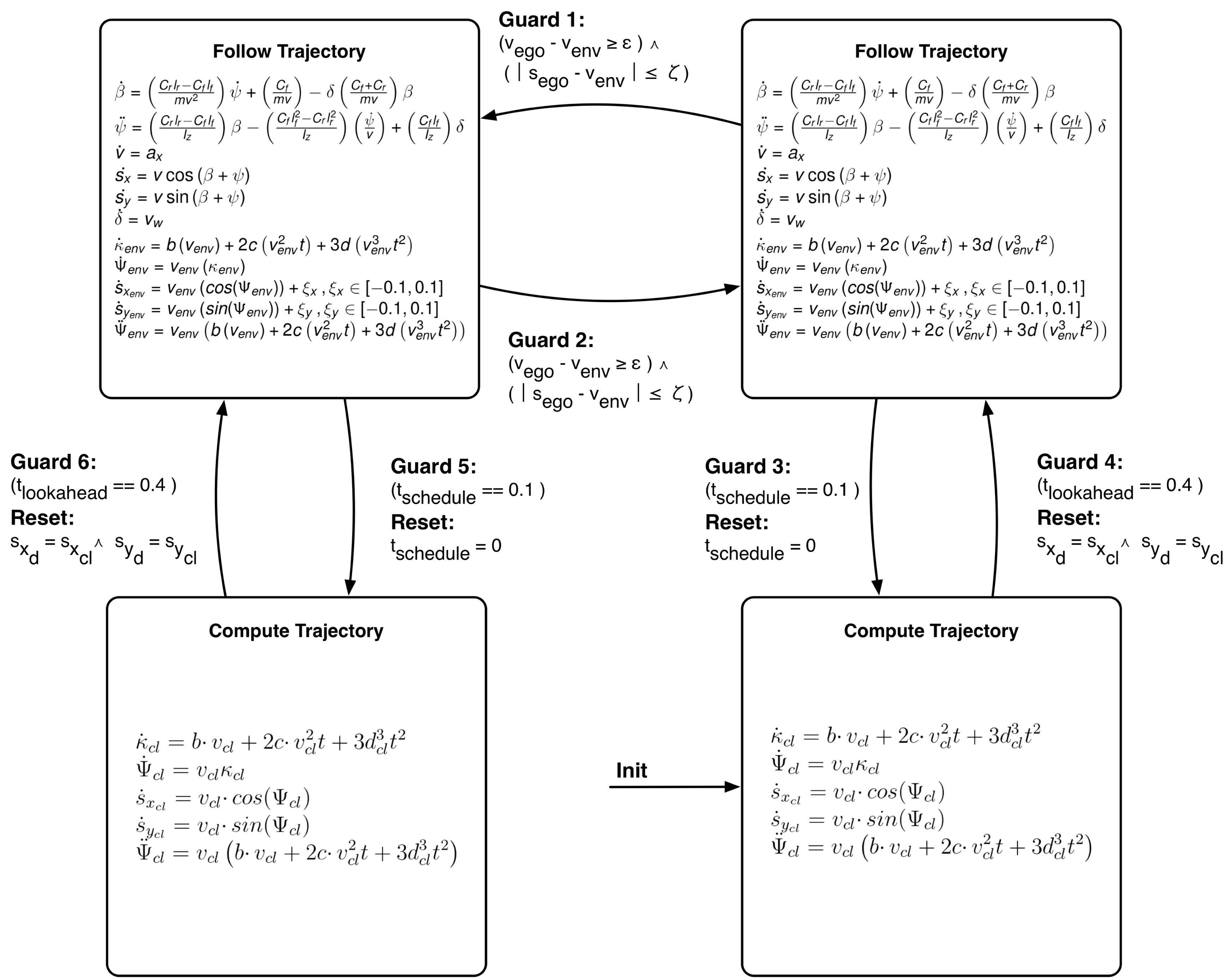
James Kapinski, Jyotirmoy V. Deshmukh, Sriram Sankaranarayanan, Nikos Aréchiga, "Simulation-guided Lyapunov Analysis for Hybrid Dynamical Systems" Hybrid Systems: Computation and Control 2014

Application: Validated Planning



Matthew O'Kelly, Houssam Abbas, Sicun Gao, Shin'ichi Shiraishi, Shinpei Kato, and Rahul Mangharam ,
"APEX: A Tool for Autonomous Vehicle Plan Verification and Execution", In Society of Automotive Engineers (SAE) World Congress and Exhibition 2016

Application: Validated Planning



Matthew O'Kelly, Houssam Abbas, Sicun Gao, Shin'ichi Shiraishi, Shinpei Kato, and Rahul Mangharam , "APEX: A Tool for Autonomous Vehicle Plan Verification and Execution", In Society of Automotive Engineers (SAE) World Congress and Exhibition 2016

Tools based on dReal

* APEX: A Tool for Autonomous Vehicle Plan Verification and Execution (Toyota/UPenn)
* BioPSy: Parameter set synthesis on biological models (Univ. of Newcastle)
* dReach: Reachability analysis tool for hybrid system (CMU)
* Osmosis: Semantic importance sampling for statistical model checking (CMU SEI)
* ProbReach: Probabilistic reachability analysis of hybrid systems (Univ. of Newcastle)
* SReach: Bounded model checker for stochastic hybrid systems (CMU)
* Sigma: Probabilistic programming language (MIT)

Conclusion

- First-order logic = Language to express general problems
- Handle combinatorial structure + nonlinear dynamics
- Existing optimization/simulation algorithms/techniques can be integrated
- Possible to generate proofs for verification
- Open-source Implementation is available
  https://github.com/dreal/dreal3

Any Questions?

FAQs

Q1. How to pick ε from a given $\delta \in \mathbb{Q}^+$ in ICP Algorithm?

A1:

- For all f_i, find ε_i such that

$$\forall \vec{x}, \vec{y} \in B, ||x - y|| < \epsilon_i \implies |f_i(\vec{x}) - f_i(\vec{y})| < \delta$$

- Fix ε be the minimum of ε_is

$$\epsilon = \min(\epsilon_1, \ldots, \epsilon_n)$$

FAQs

Q2. Lipschitz continuity?

A2: A Lipschitz continuous function is limited in how fast it can change: there exists a definite real number K such that, for every pair of points on the graph of this function, the absolute value of the slope of the line connecting them is not greater than this real number; this bound is called a "Lipschitz constant" of the function (or "modulus of uniform continuity").

$$\frac{d_Y(f(x_1), f(x_2))}{d_X(x_1, x_2)} \leq K.$$

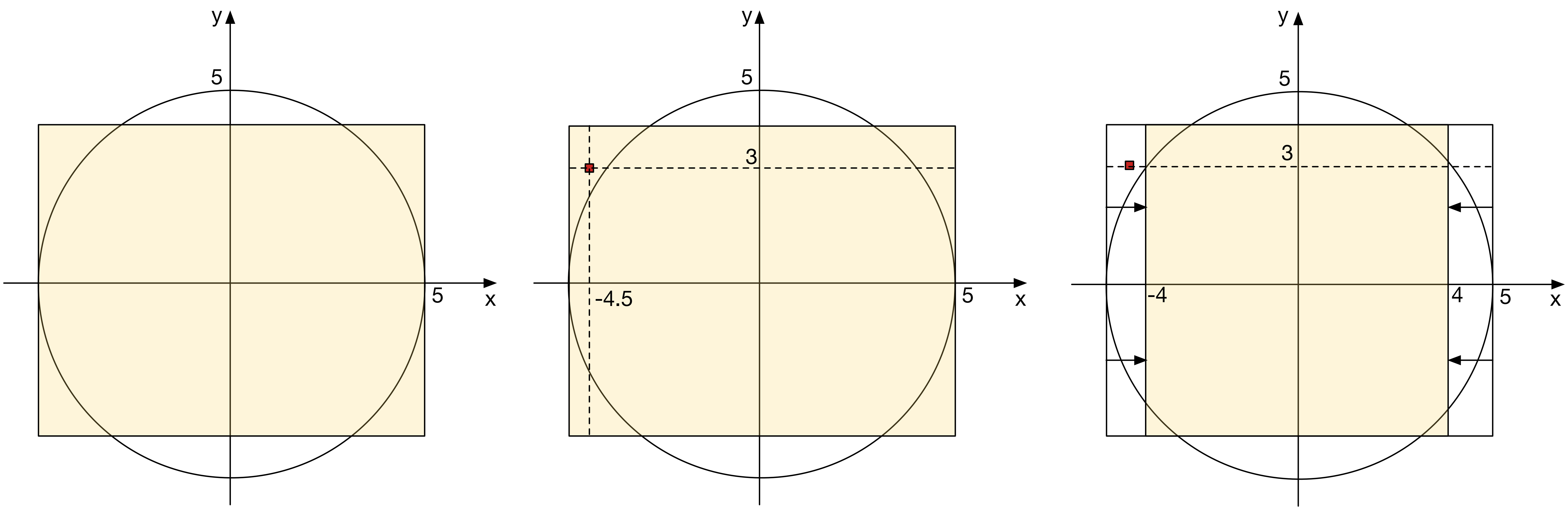
FAQs

Q3. Optimization?

A3: Check the next two slides

Counterexample-guided Global Optimization (If time permits)

$$\exists^{[-5,5]}x.\forall^{[-4,4]}y.\ x^2 + y^2 <= 5^2$$



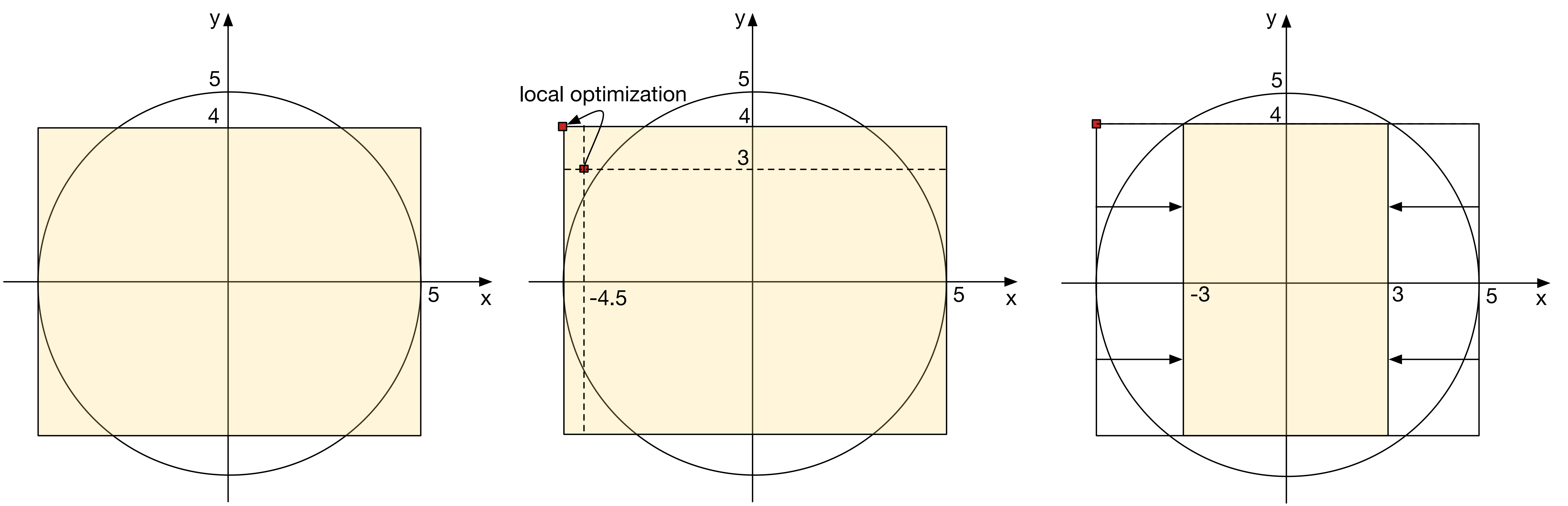
(A) Initial Search Space: x = [-5, 5]

(B) Find a counterexample

(C) Prune x using the counterexample

Counterexample-guided Global Optimization using Local Optimization (If time permits)

$$\exists^{[-5,5]}x.\forall^{[-4,4]}y.\ x^2 + y^2 <= 5^2$$



(A) Initial Search Space: x = [-5, 5]

(B) Find a counterexample and improve it using local optimization

(C) Prune x using the counterexample