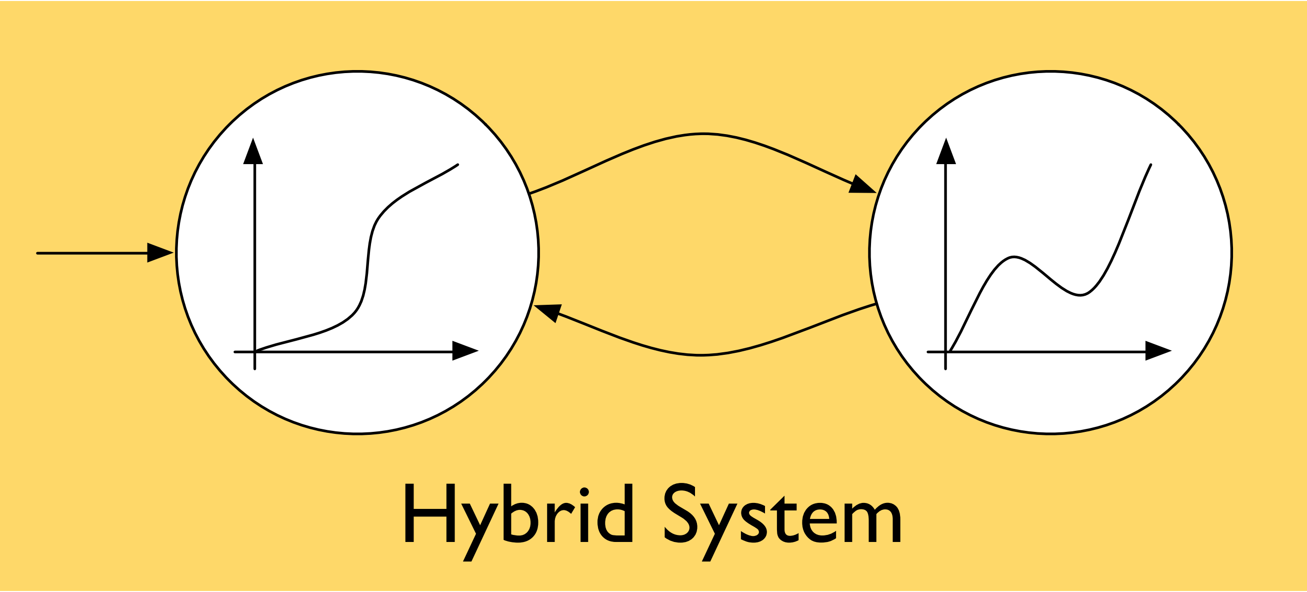
δ-Reachability Analysis for Hybrid Systems

Soonho Kong
soonhok@cs.cmu.edu
Carnegie Mellon University

Hybrid Systems



Discrete Control + Continuous Dynamics

Hybrid Systems

Discrete Control + Continuous Dynamics

Cardiac Cell Action Potential Model

$$\text{Mode 1} \underset{u<\theta_o}{\overset{u\ge\theta_o}{\rightleftarrows}} \text{Mode 2} \underset{u<\theta_w}{\overset{u\ge\theta_w}{\rightleftarrows}} \text{Mode 3} \underset{u<\theta_v}{\overset{u\ge\theta_v}{\rightleftarrows}} \text{Mode 4}$$

Mode 1

$\text{flow}_1:$

$$\frac{du}{dt} = \varepsilon - \frac{u}{\tau_{o1}}$$

$$\frac{dv}{dt} = \frac{1-v}{\tau_{v1}^-}$$

$$\frac{dw}{dt} = \frac{1 - \frac{u}{\tau_w^\infty} - w}{\tau_{w1}^- + \frac{\tau_{w2}^- - \tau_{w1}^-}{1+e^{-2k_w^-(u-u_w^-)}}}$$

$$\frac{ds}{dt} = \left(\frac{1}{1+e^{-2k_s(u-u_s)}} - s\right)\frac{1}{\tau_{s1}}$$

Mode 2

$\text{flow}_2:$

$$\frac{du}{dt} = \varepsilon - \frac{u}{\tau_{o2}}$$

$$\frac{dv}{dt} = -\frac{v}{\tau_{v2}^-}$$

$$\frac{dw}{dt} = \frac{w_\infty^* - w}{\tau_{w1}^- + \frac{\tau_{w2}^- - \tau_{w1}^-}{1+e^{-2k_w^-(u-u_w^-)}}}$$

$$\frac{ds}{dt} = \left(\frac{1}{1+e^{-2k_s(u-u_s)}} - s\right)\frac{1}{\tau_{s1}}$$

Mode 3

$\text{flow}_3:$

$$\frac{du}{dt} = \varepsilon - \frac{1}{\tau_{so1} + \frac{\tau_{so2} - \tau_{so1}}{1+e^{-2k_{so}(u-u_{so})}}} + \frac{w\cdot s}{\tau_{si}}$$

$$\frac{dv}{dt} = -\frac{v}{\tau_{v2}^-}$$

$$\frac{dw}{dt} = -\frac{w}{\tau_w^+}$$

$$\frac{ds}{dt} = \left(\frac{1}{1+e^{-2k_s(u-u_s)}} - s\right)\frac{1}{\tau_{s1}}$$

Mode 4

$\text{flow}_4:$

$$\frac{du}{dt} = \varepsilon + \frac{v(u-\theta_v)(u_u - u)}{\tau_{fi}} - \frac{1}{\tau_{so1} + \frac{\tau_{so2} - \tau_{so1}}{1+e^{-2k_{so}(u-u_{so})}}} + \frac{w\cdot s}{\tau_{si}}$$

$$\frac{dv}{dt} = -\frac{v}{\tau_{v2}^-}$$

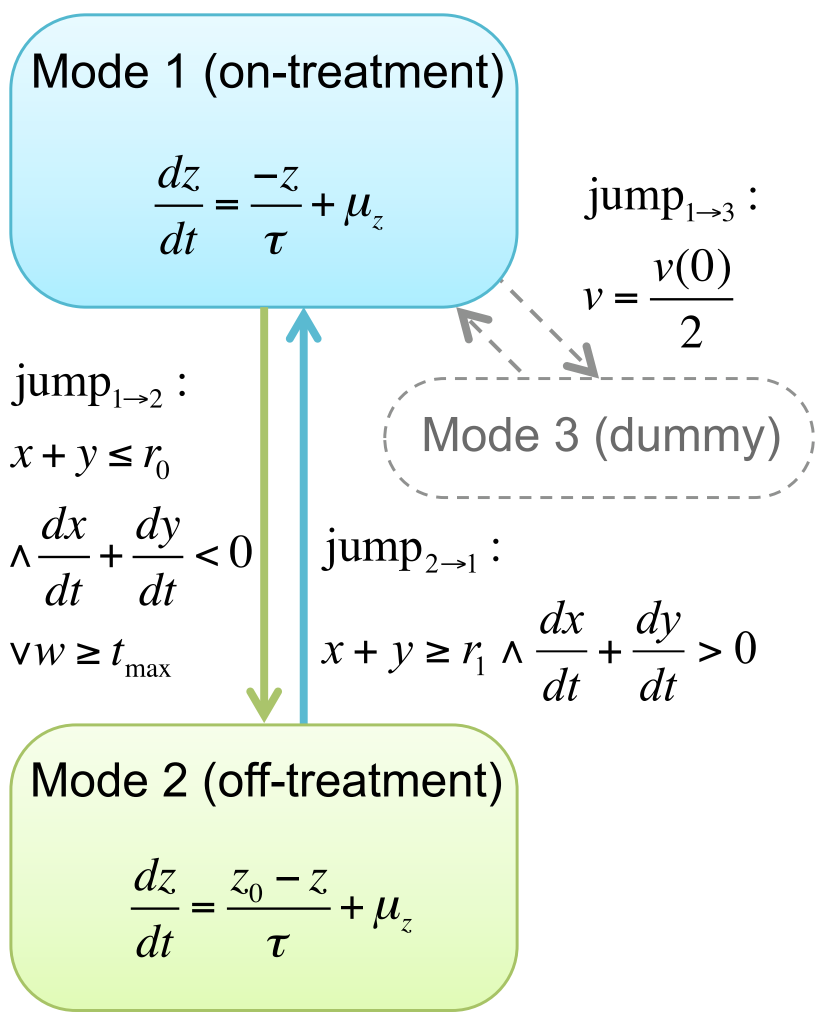
$$\frac{dw}{dt} = -\frac{w}{\tau_w^+}$$

$$\frac{ds}{dt} = \left(\frac{1}{1+e^{-2k_s(u-u_s)}} - s\right)\frac{1}{\tau_{s1}}$$

Hybrid Systems

Discrete Control + Continuous Dynamics

Prostate Cancer Treatment Model



||

$$\frac{dx}{dt} = \left(\alpha_x \left(\frac{1}{1+e^{-(z-k_1)k_2}} \right) - \beta_x \left(\frac{1}{1+e^{-(z-k_3)k_4}} \right) - m_1 \left(1 - \frac{z}{z_0} \right) - \lambda_x \right) x + \mu_x$$

$$\frac{dy}{dt} = m_1 \left(1 - \frac{z}{z_0} \right) x + \left(\alpha_y \left(1 - d\frac{z}{z_0} \right) - \beta_y \right) y$$

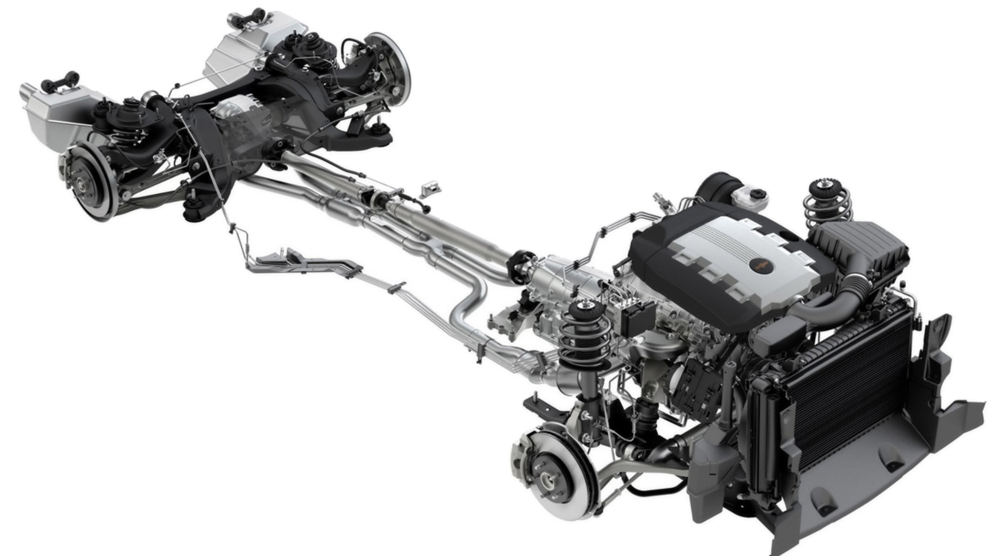
$$\frac{dv}{dt} = \left(\alpha_x \left(\frac{1}{1+e^{-(z-k_1)k_2}} \right) - \beta_x \left(\frac{1}{1+e^{-(z-k_3)k_4}} \right) - m_1 \left(1 - \frac{z}{z_0} \right) - \lambda_x \right) x + \mu_x + m_1 \left(1 - \frac{z}{z_0} \right) x + \left(\alpha_y \left(1 - d\frac{z}{z_0} \right) - \beta_y \right) y$$

Control (cancer therapy)

Plant (cancer progression)

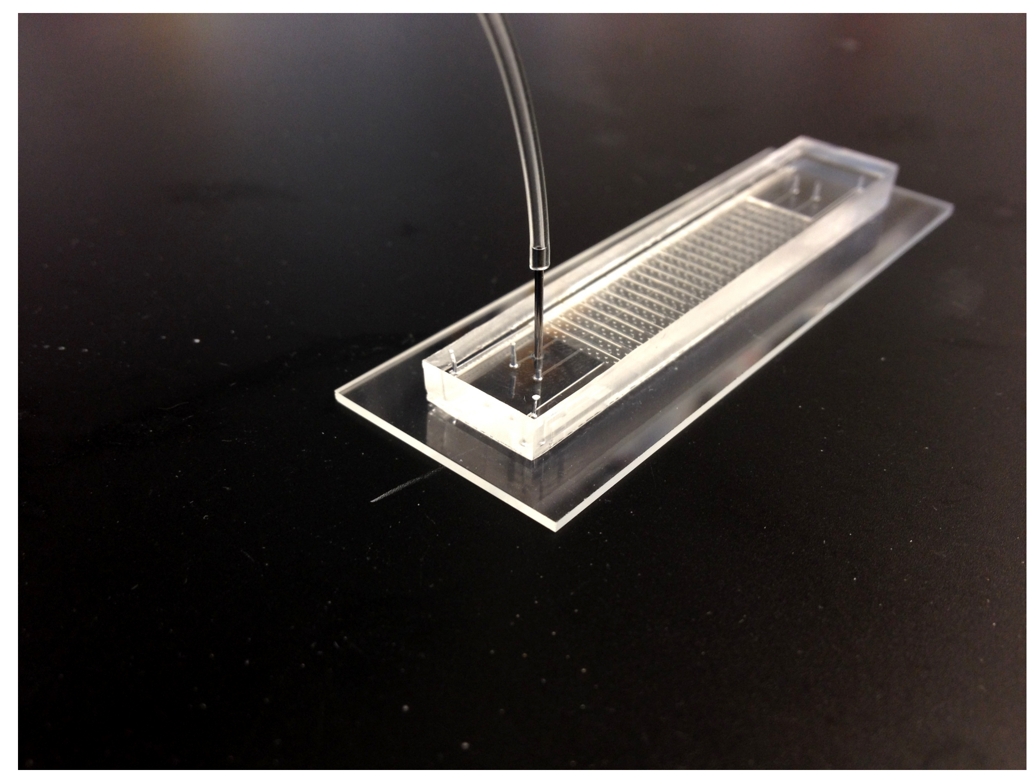
Hybrid Systems

Quadcopter Control (CMU ECE)



Power Train Control (Toyota Research)

Autonomous Vehicle (CMU ECE)



Microfluidic Chip Design (Univ. of Waterloo)

Reachability Analysis of Hybrid Systems

Can we find a set of initial values/parameters for which a cardiac cell loses excitability?

Cardiac Cell Action Potential

Mode 1

flow$_1$:

$$\frac{du}{dt} = \varepsilon - \frac{u}{\tau_{o1}}$$

$$\frac{dv}{dt} = \frac{1-v}{\tau_{v1}^{-}}$$

$$\frac{dw}{dt} = \frac{1 - \frac{u}{\tau_w^{\infty}} - w}{\tau_{w1}^{-} + \frac{\tau_{w2}^{-} - \tau_{w1}^{-}}{1+e^{-2k_w^{-}(u-u_w^{-})}}}$$

$$\frac{ds}{dt} = \left(\frac{1}{1+e^{-2k_s(u-u_s)}} - s\right)\frac{1}{\tau_{s1}}$$

Mode 1 → Mode 2: $u \geq \theta_o$; Mode 2 → Mode 1: $u < \theta_o$

Mode 2

flow$_2$:

$$\frac{du}{dt} = \varepsilon - \frac{u}{\tau_{o2}}$$

$$\frac{dv}{dt} = -\frac{v}{\tau_{v2}^{-}}$$

$$\frac{dw}{dt} = \frac{w_{\infty}^{*} - w}{\tau_{w1}^{-} + \frac{\tau_{w2}^{-} - \tau_{w1}^{-}}{1+e^{-2k_w^{-}(u-u_w^{-})}}}$$

$$\frac{ds}{dt} = \left(\frac{1}{1+e^{-2k_s(u-u_s)}} - s\right)\frac{1}{\tau_{s1}}$$

Mode 2 → Mode 3: $u \geq \theta_w$; Mode 3 → Mode 2: $u < \theta_w$

Mode 3

flow$_3$:

$$\frac{du}{dt} = \varepsilon - \frac{1}{\tau_{so1} + \frac{\tau_{so2} - \tau_{so1}}{1+e^{-2k_{so}(u-u_{so})}}} + \frac{w \cdot s}{\tau_{si}}$$

$$\frac{dv}{dt} = -\frac{v}{\tau_{v2}^{-}}$$

$$\frac{dw}{dt} = -\frac{w}{\tau_w^{+}}$$

$$\frac{ds}{dt} = \left(\frac{1}{1+e^{-2k_s(u-u_s)}} - s\right)\frac{1}{\tau_{s1}}$$

Mode 3 → Mode 4: $u \geq \theta_v$; Mode 4 → Mode 3: $u < \theta_v$

Mode 4

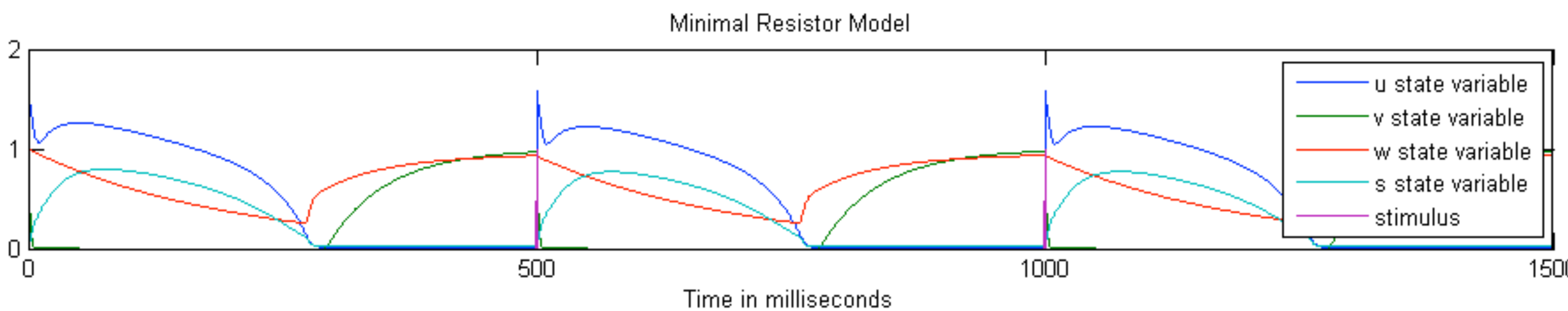
flow$_4$:

$$\frac{du}{dt} = \varepsilon + \frac{v(u-\theta_v)(u_u - u)}{\tau_{fi}} - \frac{1}{\tau_{so1} + \frac{\tau_{so2} - \tau_{so1}}{1+e^{-2k_{so}(u-u_{so})}}} + \frac{w \cdot s}{\tau_{si}}$$

$$\frac{dv}{dt} = -\frac{v}{\tau_{v2}^{-}}$$

$$\frac{dw}{dt} = -\frac{w}{\tau_w^{+}}$$

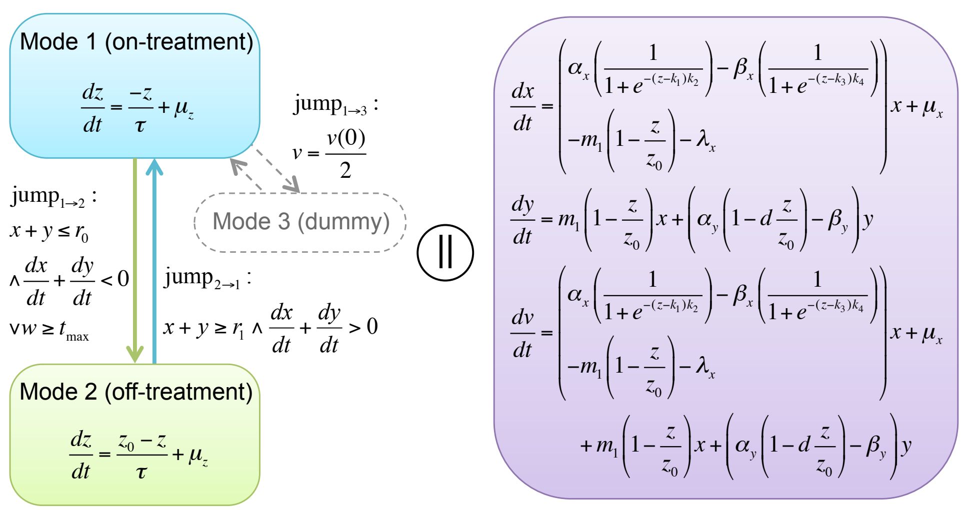
$$\frac{ds}{dt} = \left(\frac{1}{1+e^{-2k_s(u-u_s)}} - s\right)\frac{1}{\tau_{s1}}$$



Reachability Analysis of Hybrid Systems

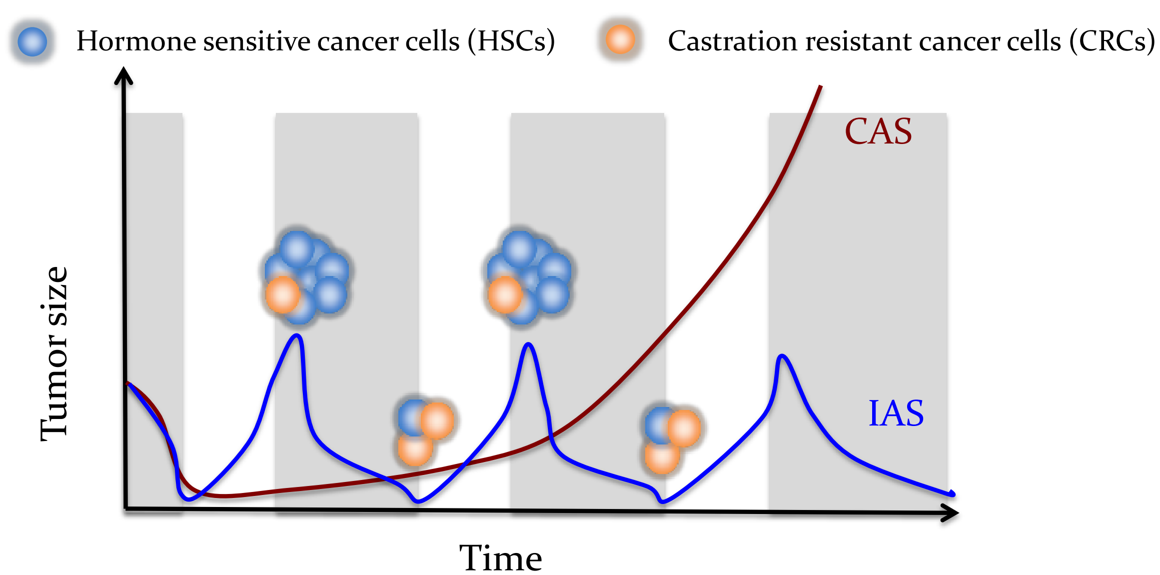
Can we find a personalized treatment model which prevents the cancer recurrence in 5 years??

Prostate Cancer Treatment Model



Control (cancer therapy)

Plant (cancer progression)

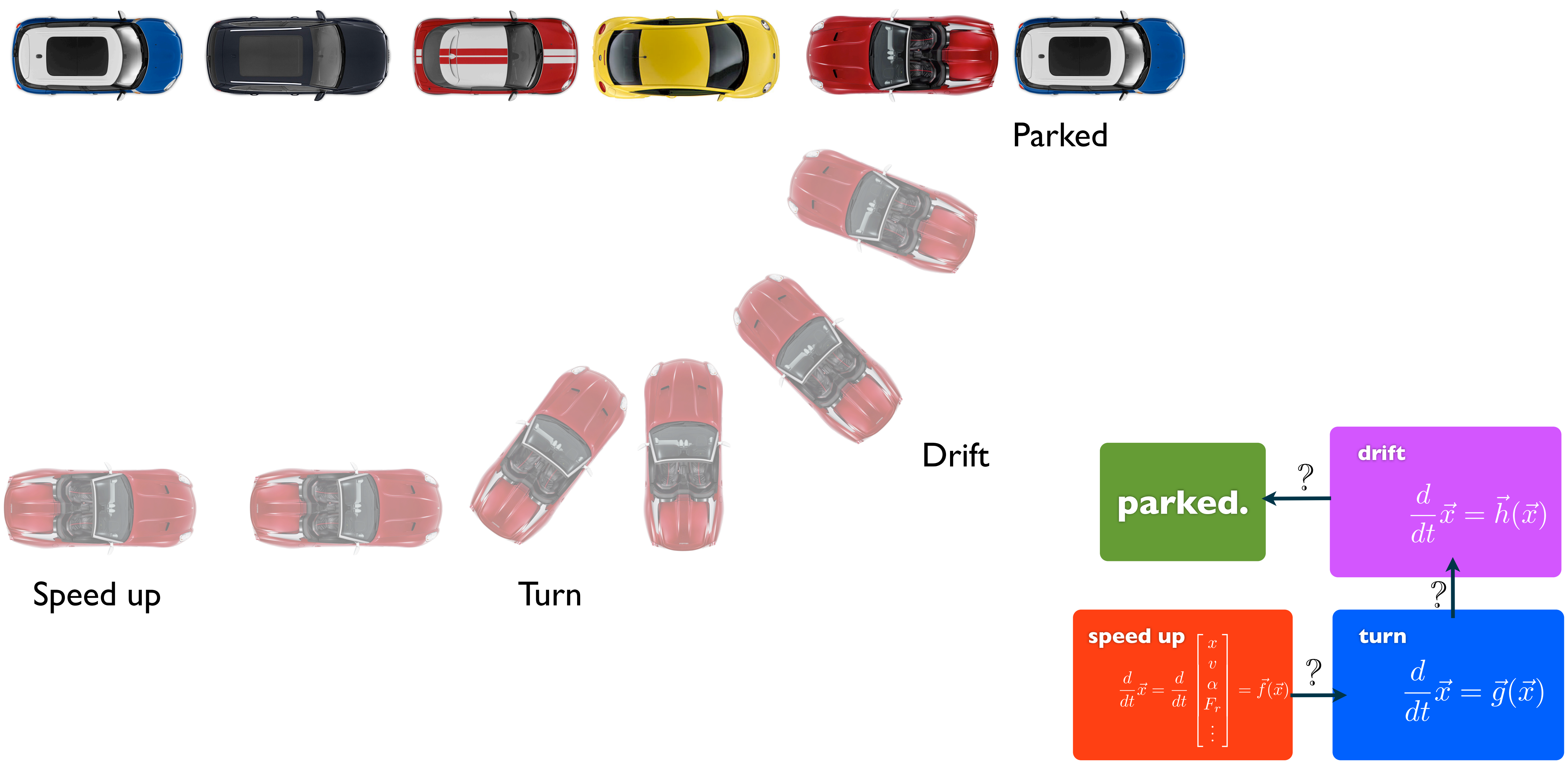


Reachability Analysis of Hybrid Systems

Can we automate a non-trivial parking?

Reachability Analysis of Hybrid Systems

Can we automate a non-trivial parking?

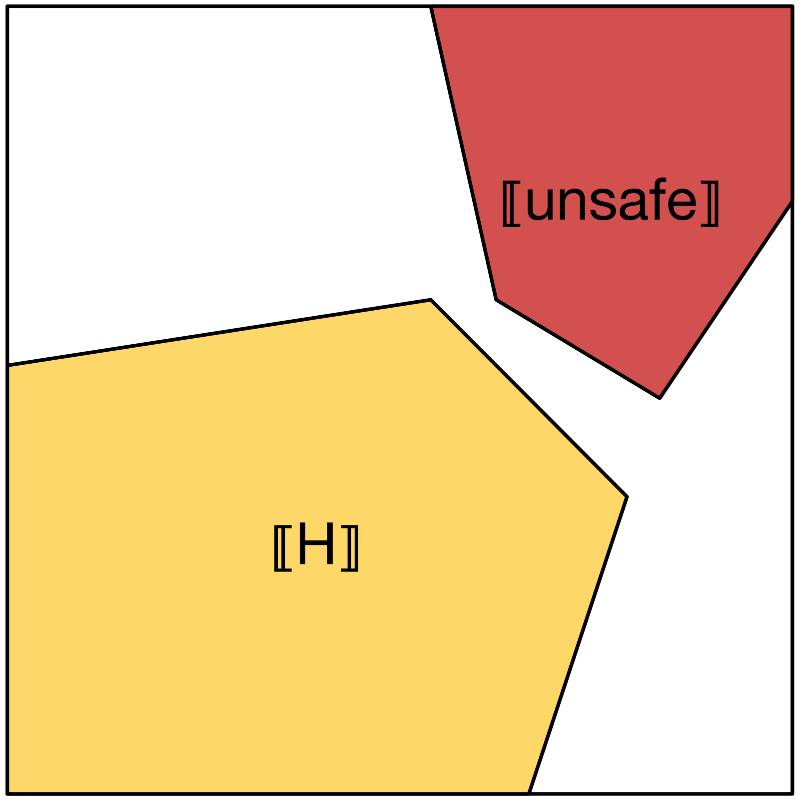


Reachability Analysis of Hybrid Systems

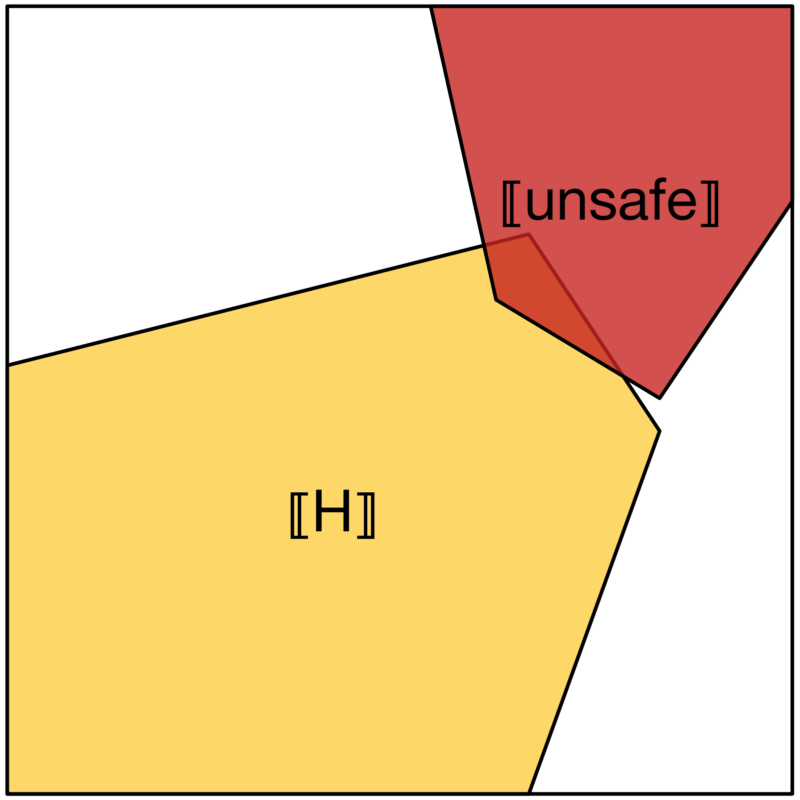
Can a hybrid system run into an **unsafe** region of its state space?

Reachability Analysis of Hybrid Systems

Can a hybrid system run into an **unsafe** region of its state space?



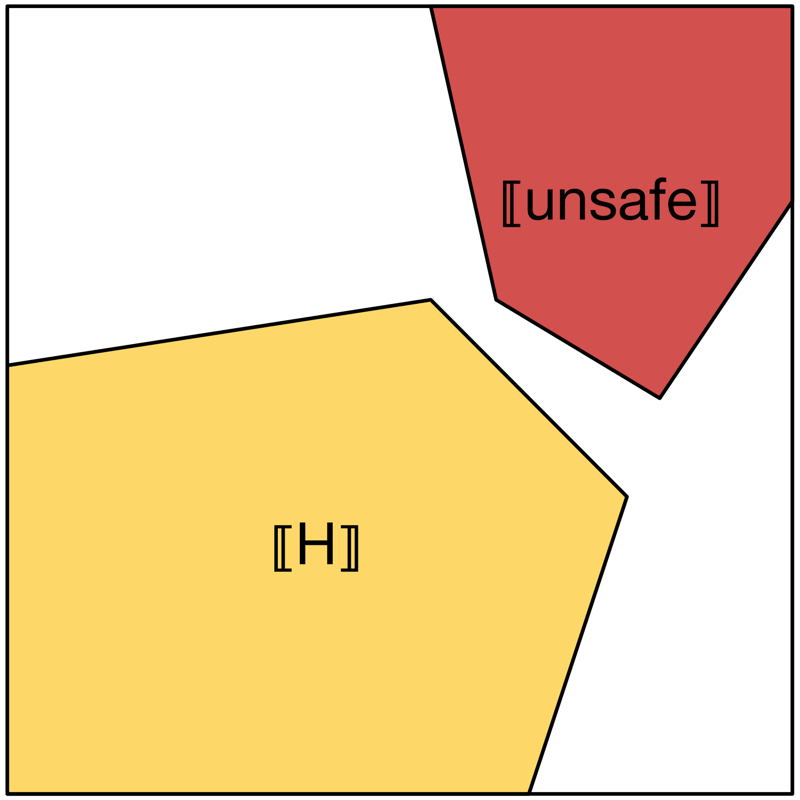
Unreachable
(Safe)



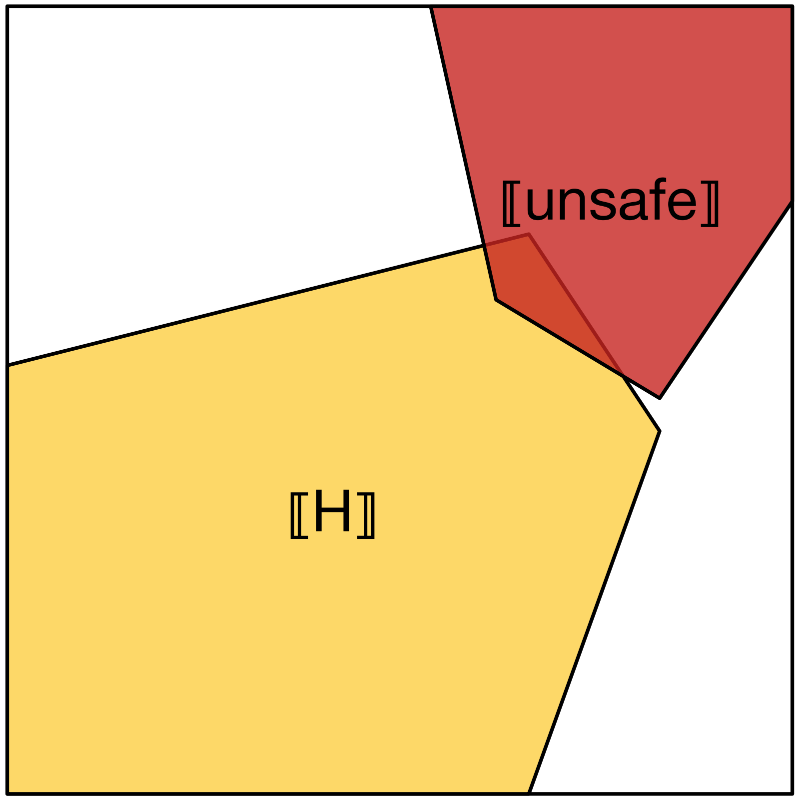
Reachable
(Unsafe)

Reachability Analysis of Hybrid Systems

Can a hybrid system run into an **unsafe** region of its state space?



Unreachable
(Safe)



Reachable
(Unsafe)

The standard bounded reachability problems for simple hybrid systems are **undecidable**[Alur et al, 1992].

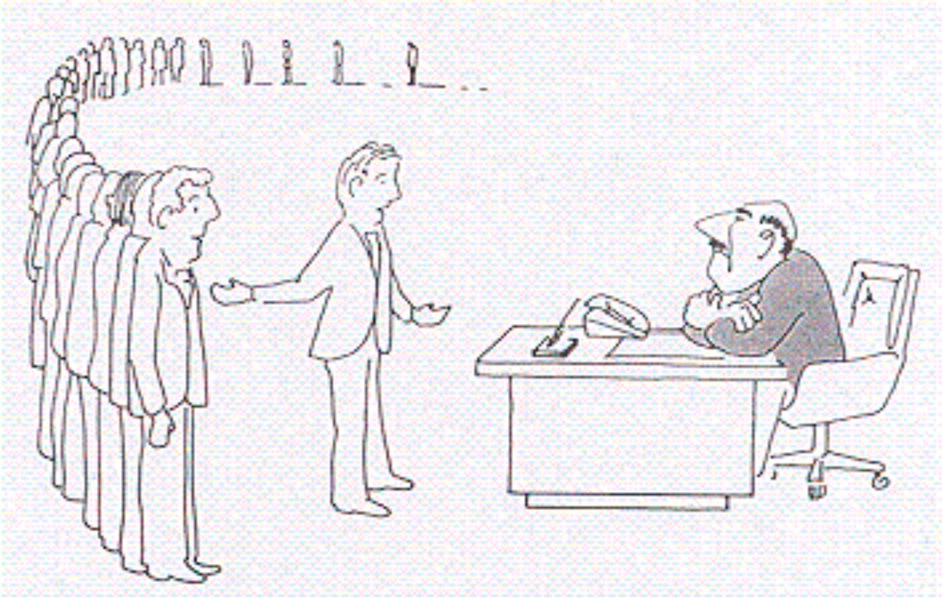
Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are **undecidable**.

Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are **undecidable**.

1. Give up



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

Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are **undecidable**.

1. Give up

2. Don't give Up

Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are **undecidable**.

1. Give up

2. Don't give Up

   A. Find a decidable fragment and solve it

Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are **undecidable**.

1. Give up

2. Don't give Up

   A. Find a decidable fragment and solve it

   B. Use approximation

Reachability Analysis of Hybrid Systems

The standard bounded reachability problems for simple hybrid systems are **undecidable**.

1. Give up

2. Don’t give Up

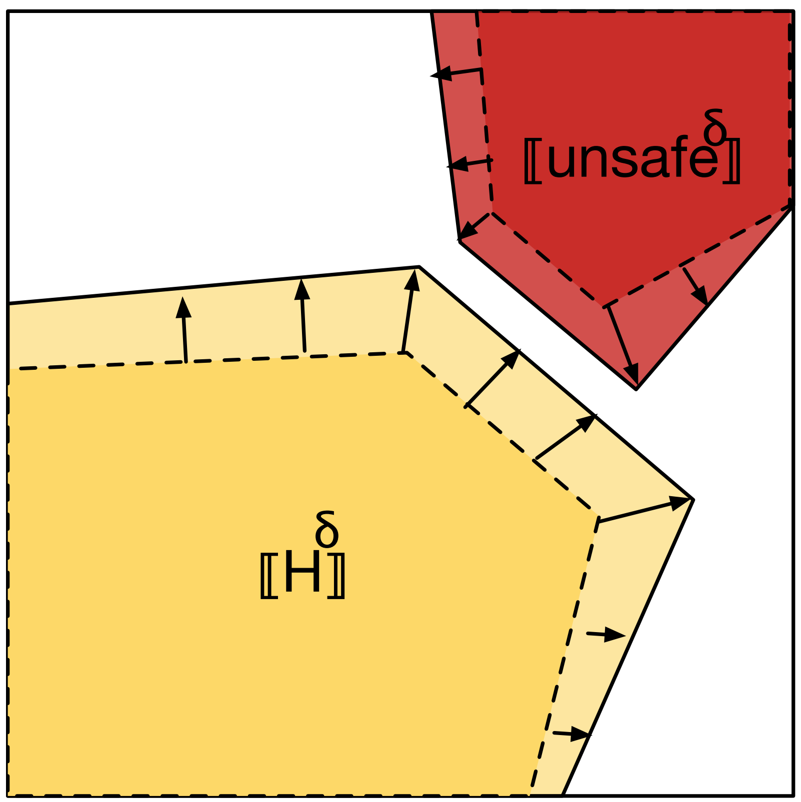
   A. Find a decidable fragment and solve it

   B. Use approximation

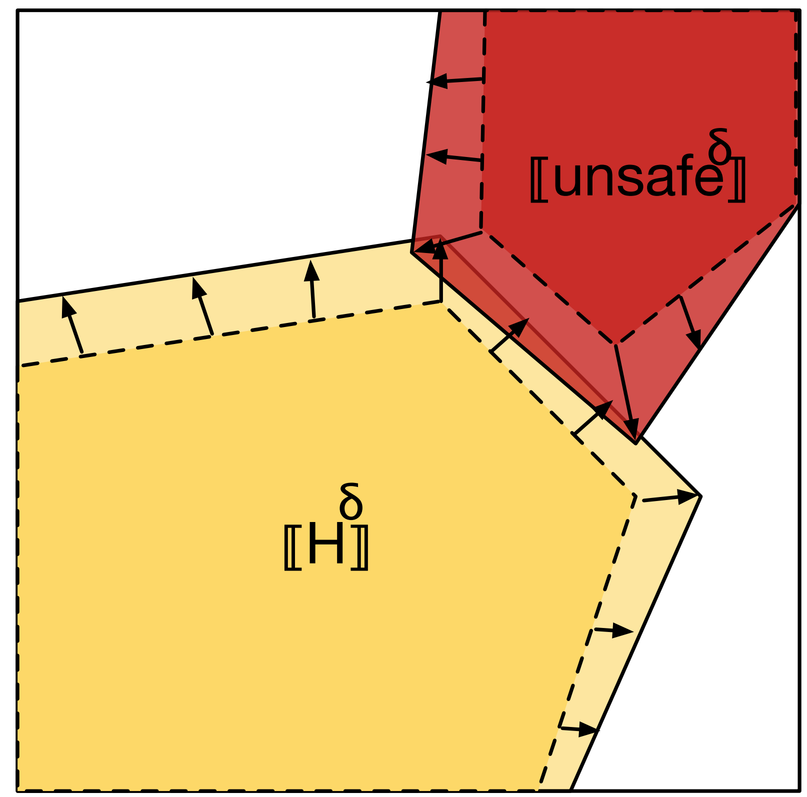
δ-Reachability Analysis of Hybrid Systems

Given $\delta \in \mathbb{Q}^+$, $[\![H]\!]^\delta$ and $[\![\text{unsafe}]\!]^\delta$ **over-approximate** $[\![H]\!]$ and $[\![\text{unsafe}]\!]$

δ-reachability problem asks for one of the following answers:



Unreachable
(Safe)

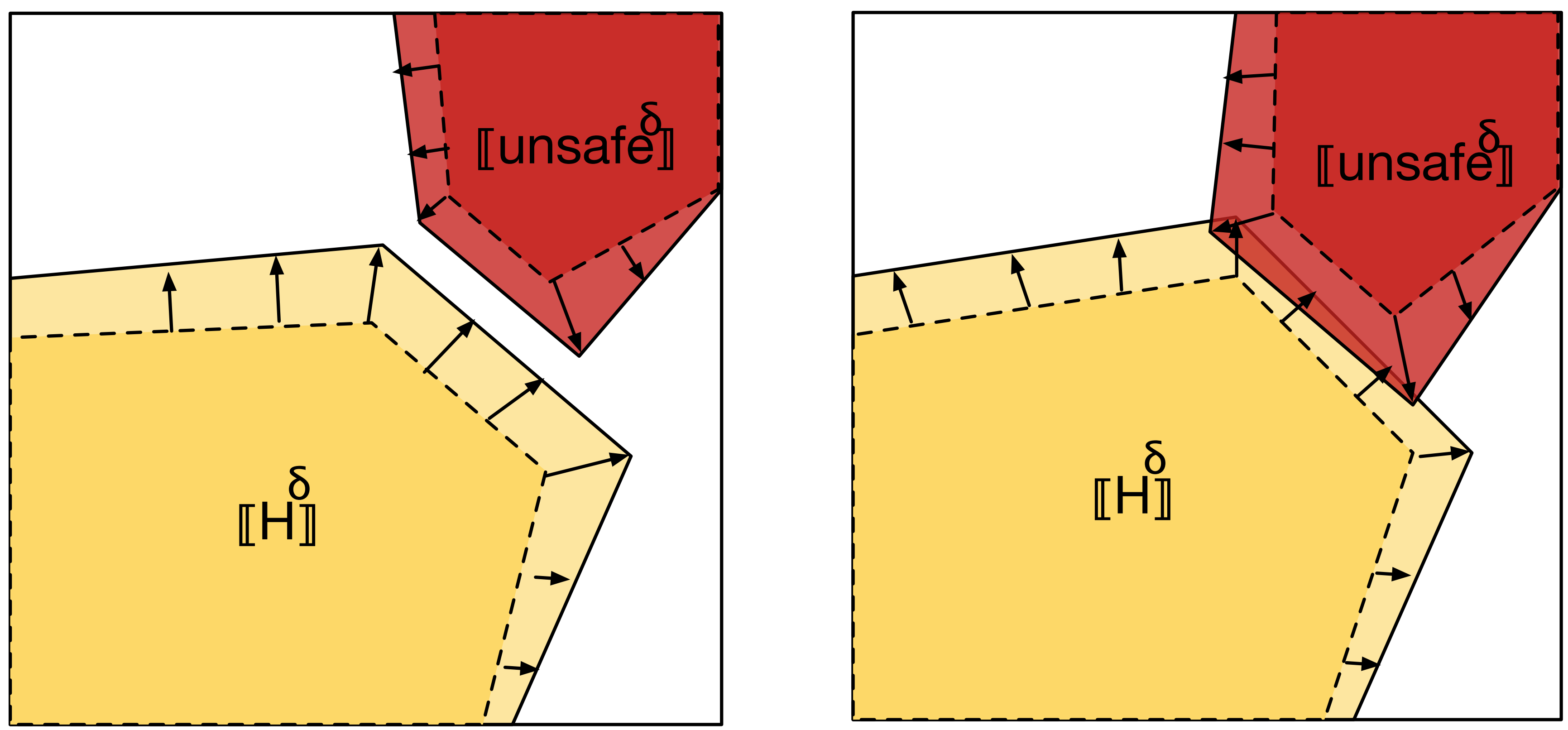


δ-reachable
(δ-Unsafe)

δ-Reachability Analysis of Hybrid Systems

Given $\delta \in \mathbb{Q}^+$, $[\![H]\!]^\delta$ and $[\![\text{unsafe}]\!]^\delta$ **over-approximate** $[\![H]\!]$ and $[\![\text{unsafe}]\!]$

δ-reachability problem asks for one of the following answers:



Unreachable
(Safe)

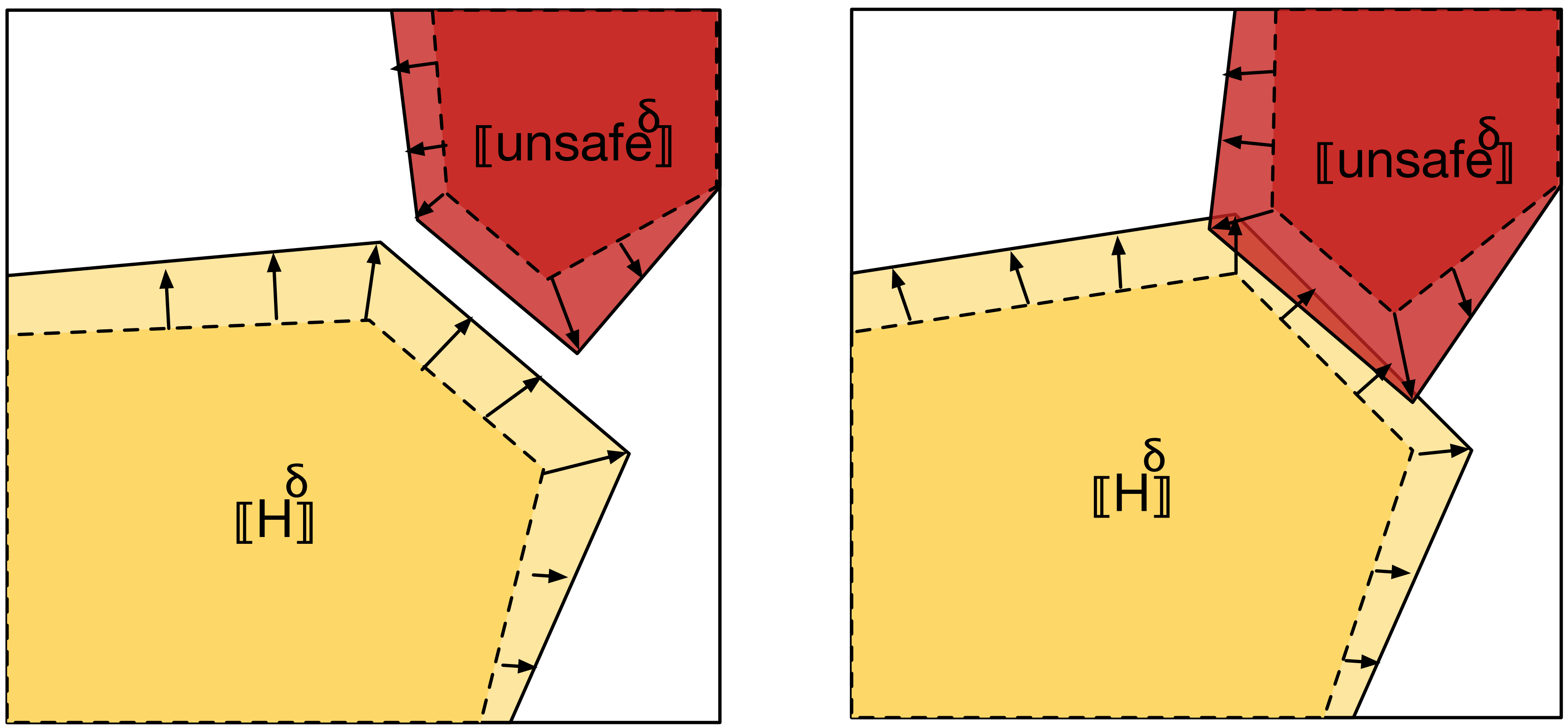
δ-reachable
(δ-Unsafe)

- **Decidable** for a wide range of **nonlinear** hybrid systems
    - polynomials, log, exp, trigonometric functions, …

δ-Reachability Analysis of Hybrid Systems

Given $\delta \in \mathbb{Q}^+$, $[\![H]\!]^\delta$ and $[\![\text{unsafe}]\!]^\delta$ **over-approximate** $[\![H]\!]$ and $[\![\text{unsafe}]\!]$

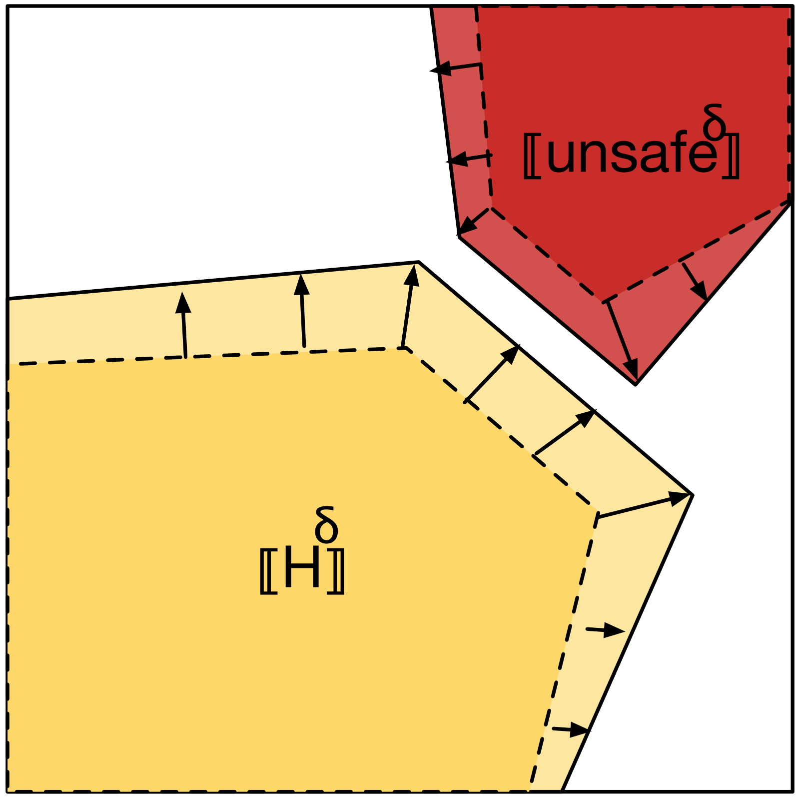
δ-reachability problem asks for one of the following answers:



Unreachable (Safe) | δ-reachable (δ-Unsafe)

- **Decidable** for a wide range of **nonlinear** hybrid systems
- **Reasonable** complexity bound (PSPACE-complete)

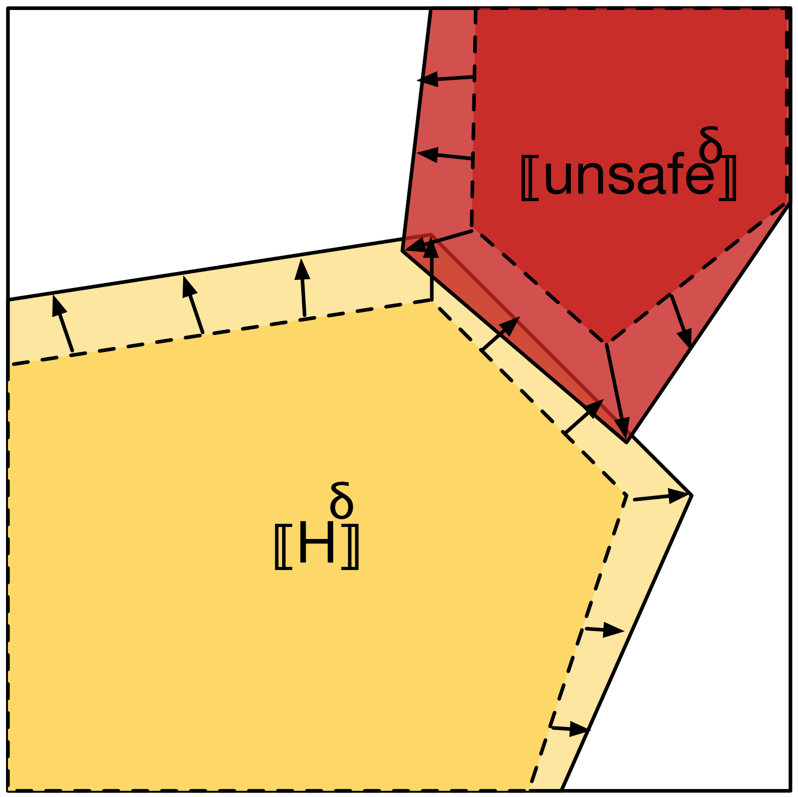
δ-Reachability Analysis of Hybrid Systems



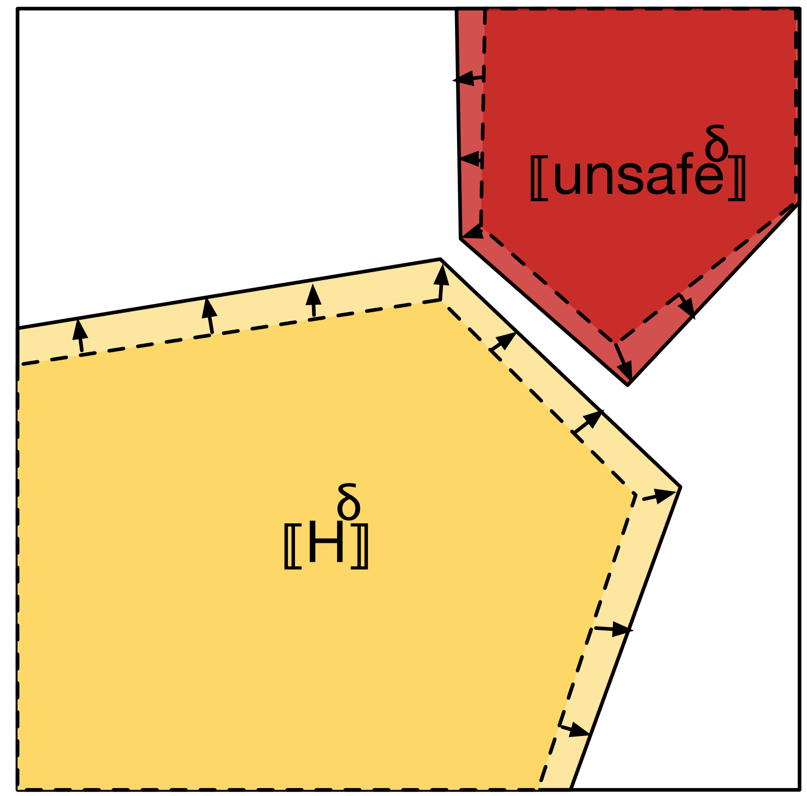
Unreachable
(Safe)

1. "Unreachable" answers is **sound**.

δ-Reachability Analysis of Hybrid Systems



δ-reachable
(δ-Unsafe)

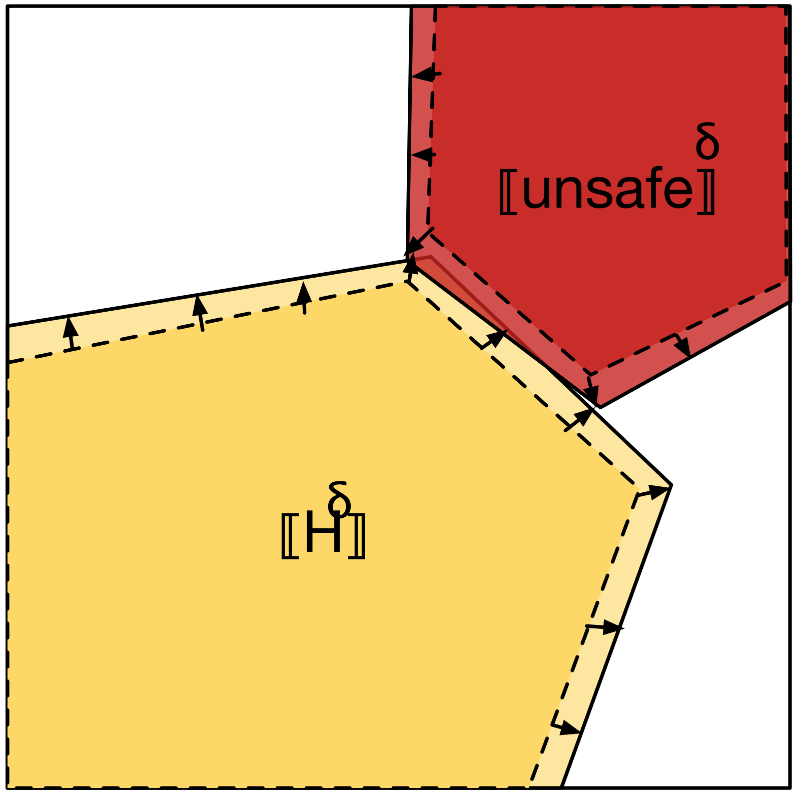


Unreachable with smaller δ
(Safe)

2. Analysis is parameterized with δ

If using a delta leads to a **infeasible** counterexample, you may try a **smaller delta** and possibly get rid of it.

δ-Reachability Analysis of Hybrid Systems



Unreachable
(Unsafe)

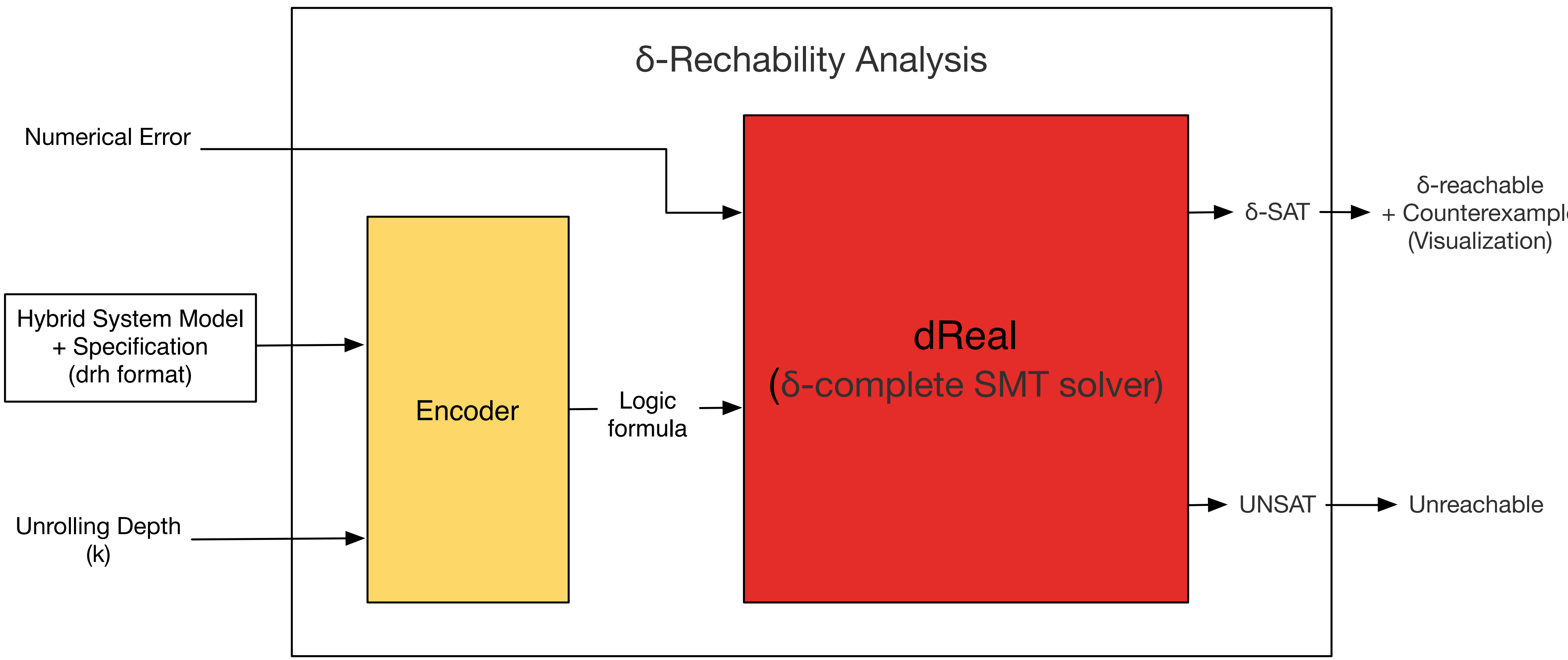
3. Robustness

If your system is **δ-reachable** under a reasonably small δ, then a small error can lead your system to an **unsafe** state

δ-Reachability Analysis of Hybrid Systems

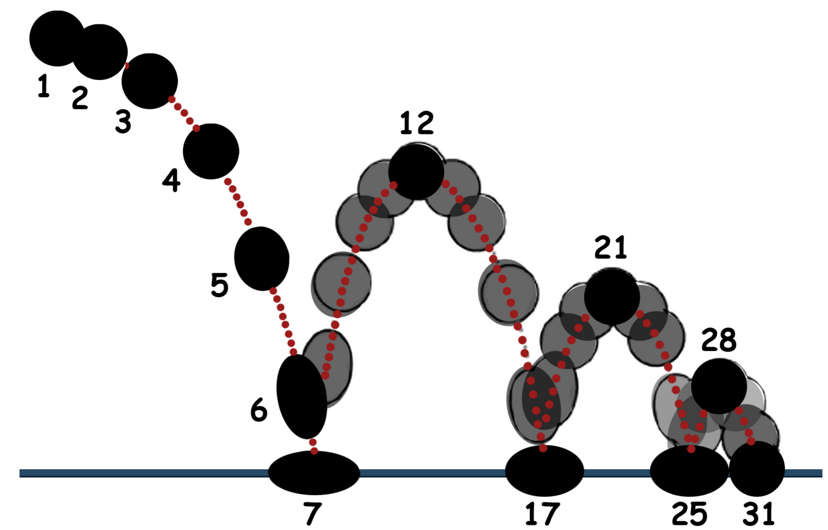
"δ-reachability analysis checks robustness which implies safety."

δ-Reachability Analysis of Hybrid Systems



Input Format (drh) for Hybrid System

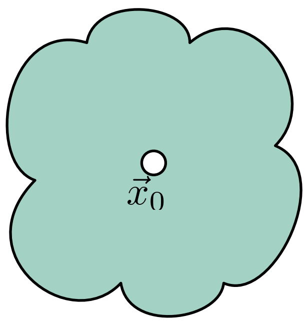
```
#define D 0.45
#define K 0.9
[0, 15] x;
[9.8] g;
[-18, 18] v;
[0, 3] time;

{    mode 1;
     invt: (v <= 0);
           (x >= 0);
     flow: d/dt[x] = v;
           d/dt[v] = -g - (D * v ^ 2);
     jump: (x = 0) ==> @2 (and (x' = x) (v' = - K * v)); }

{    mode 2;
     invt: (v >= 0);
           (x >= 0);
     flow: d/dt[x] = v;
           d/dt[v] = -g + (D * v ^ 2);
     jump: (v = 0) ==> @1 (and (x' = x) (v' = v)); }

init: @1 (and (x >= 5) (v = 0));
goal: @1 (and (x >= 0.45));
```



Inelastic bouncing ball with air resistance

Logical Encoding of Reachability Problem

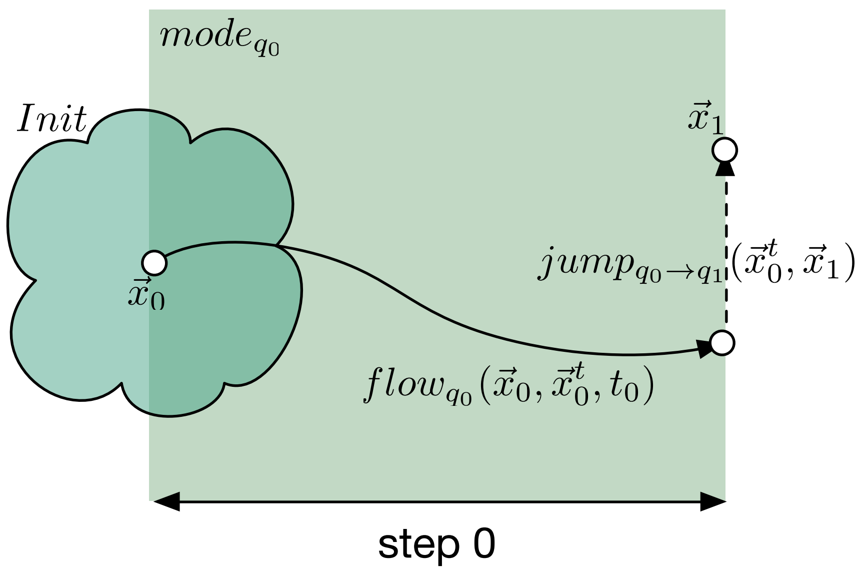
Can a system run into an **unsafe** region after making k steps?



$$Init(\vec{x}_0)$$

Logical Encoding of Reachability Problem

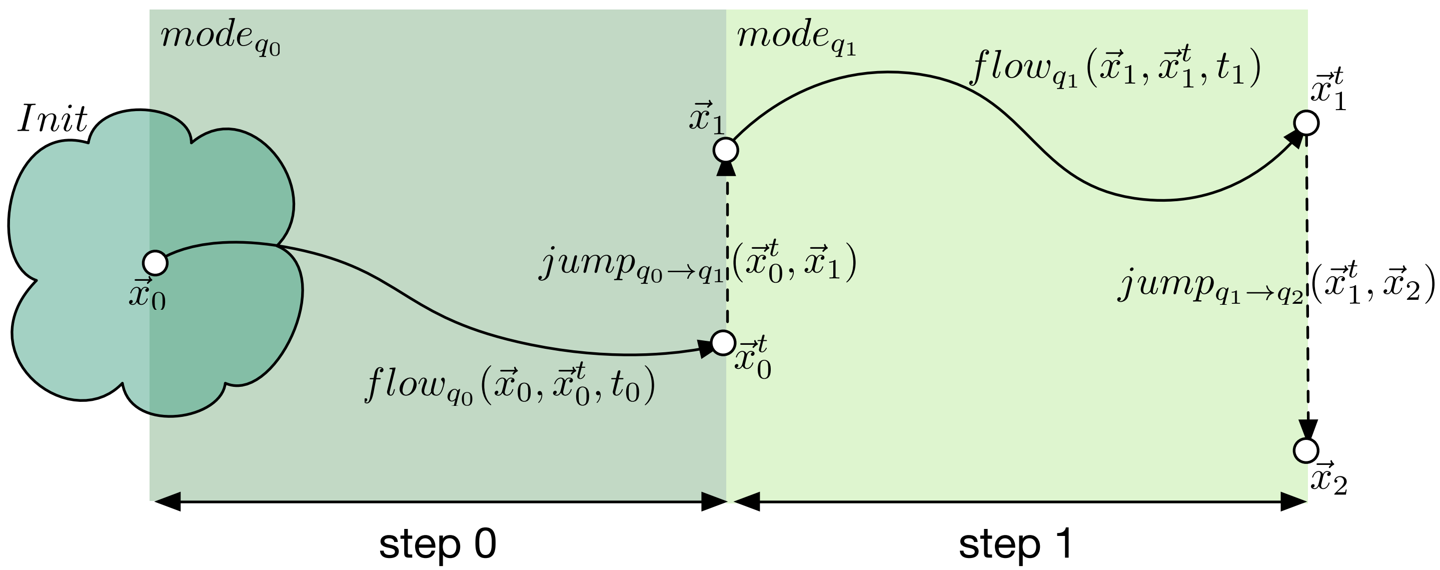
Can a system run into an **unsafe** region after making k steps?



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

Logical Encoding of Reachability Problem

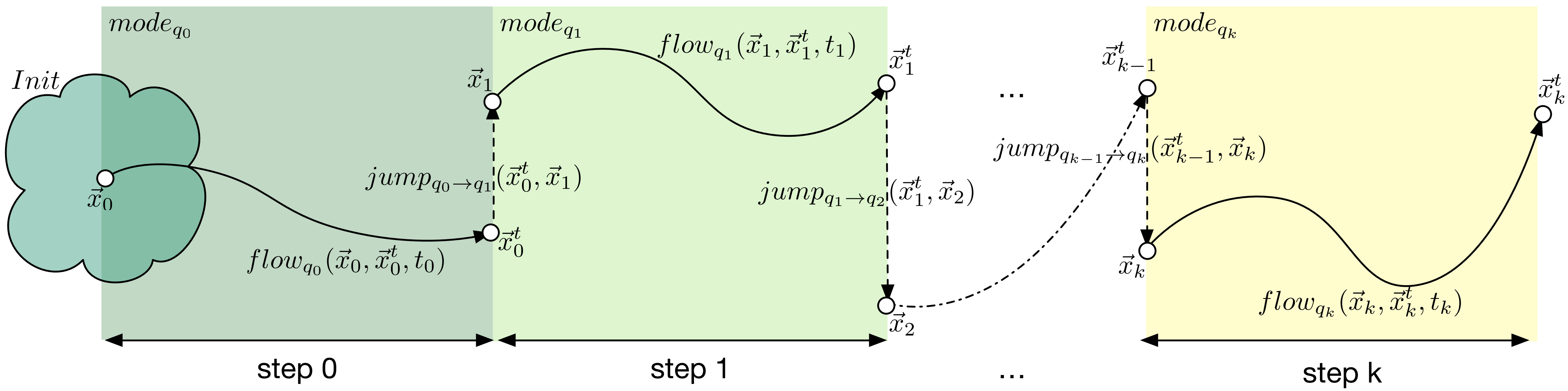
Can a system run into an **unsafe** region after making k steps?



$$Init(\vec{x}_0) \wedge flow_{q_0}(\vec{x}_0, \vec{x}_0^t, t_0) \wedge jump_{q_0 \to q_1}(\vec{x}_0^t, \vec{x}_1) \wedge$$
$$flow_{q_1}(\vec{x}_1, \vec{x}_1^t, t_1) \wedge jump_{q_1 \to q_2}(\vec{x}_1^t, \vec{x}_2)$$

Logical Encoding of Reachability Problem

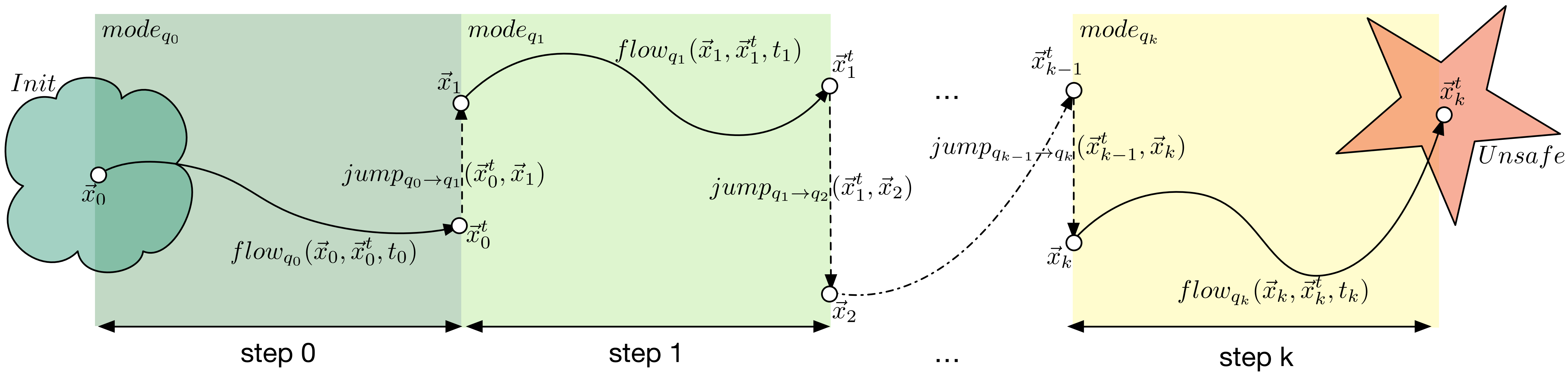
Can a system run into an **unsafe** region after making k steps?



$$
\begin{aligned}
Init(\vec{x}_0) \wedge flow_{q_0}(\vec{x}_0, \vec{x}^t_0, t_0) \wedge jump_{q_0 \to q_1}(\vec{x}^t_0, \vec{x}_1) \wedge \\
flow_{q_1}(\vec{x}_1, \vec{x}^t_1, t_1) \wedge jump_{q_1 \to q_2}(\vec{x}^t_1, \vec{x}_2) \wedge \\
\ldots \\
flow_{q_k}(\vec{x}_k, \vec{x}^t_k, t_k)
\end{aligned}
$$

Logical Encoding of Reachability Problem

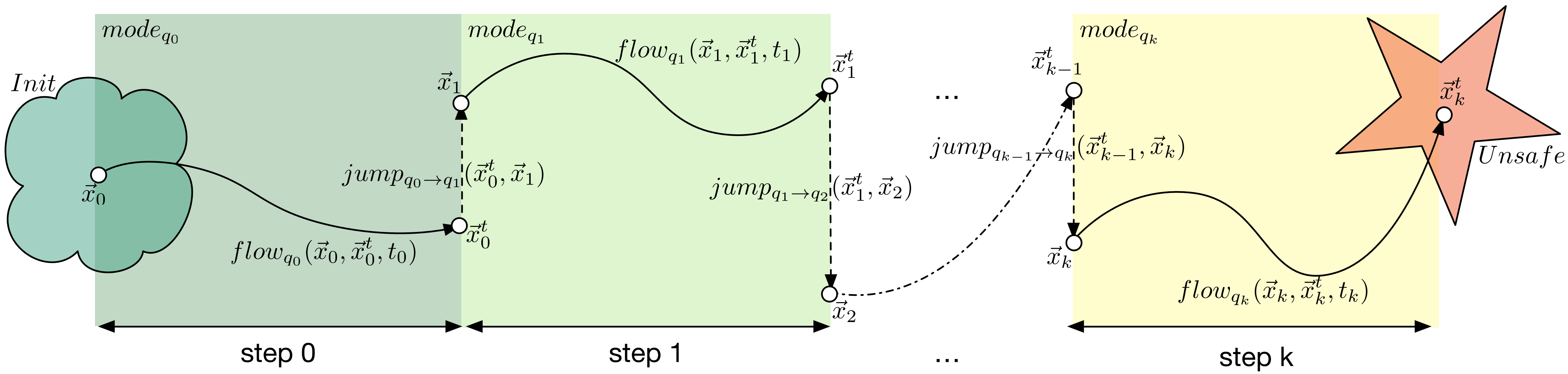
Can a system run into an **unsafe** region after making k steps?



$$
\begin{aligned}
Init(\vec{x}_0) \wedge flow_{q_0}(\vec{x}_0, \vec{x}^t_0, t_0) \wedge jump_{q_0 \to q_1}(\vec{x}^t_0, \vec{x}_1) \wedge \\
flow_{q_1}(\vec{x}_1, \vec{x}^t_1, t_1) \wedge jump_{q_1 \to q_2}(\vec{x}^t_1, \vec{x}_2) \wedge \\
\ldots \\
flow_{q_k}(\vec{x}_k, \vec{x}^t_k, t_k) \wedge Unsafe(\vec{x}^t_k)
\end{aligned}
$$

Logical Encoding of Reachability Problem

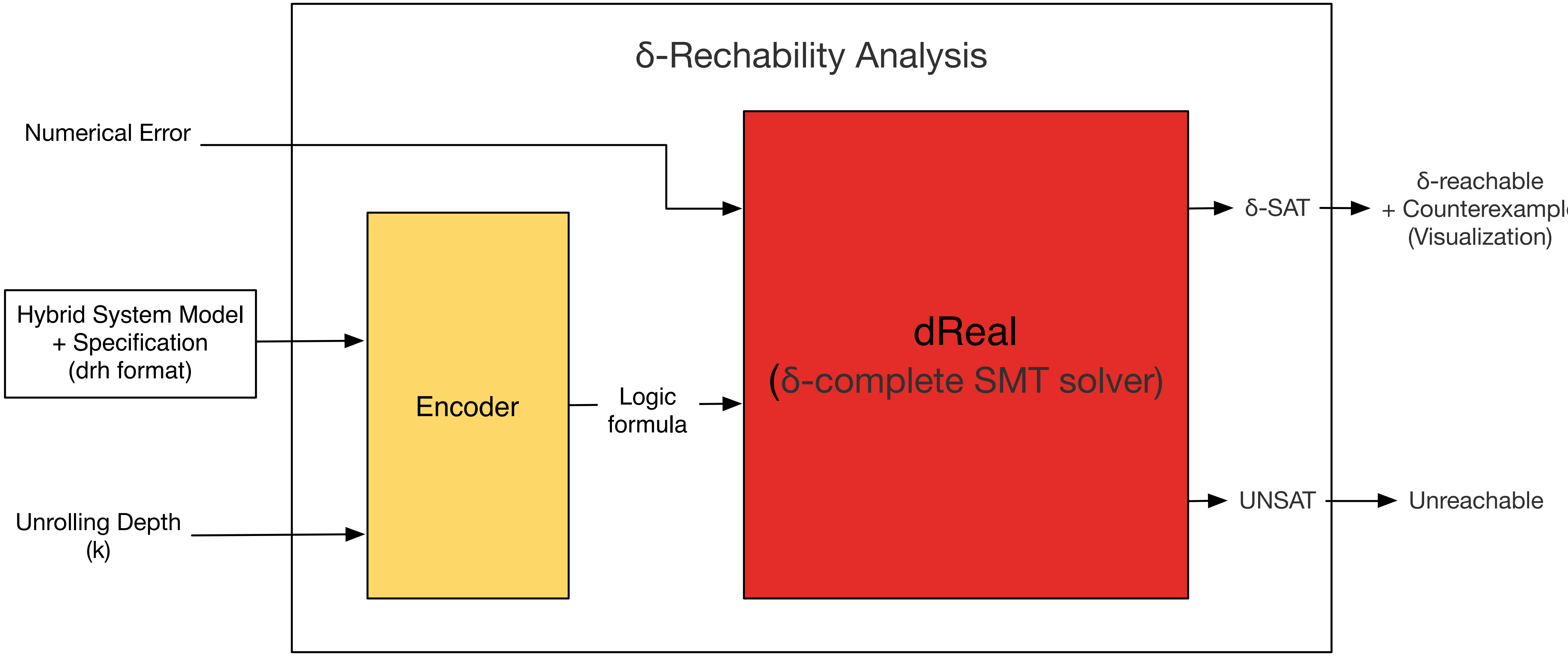
Can a system run into an **unsafe** region after making k steps?



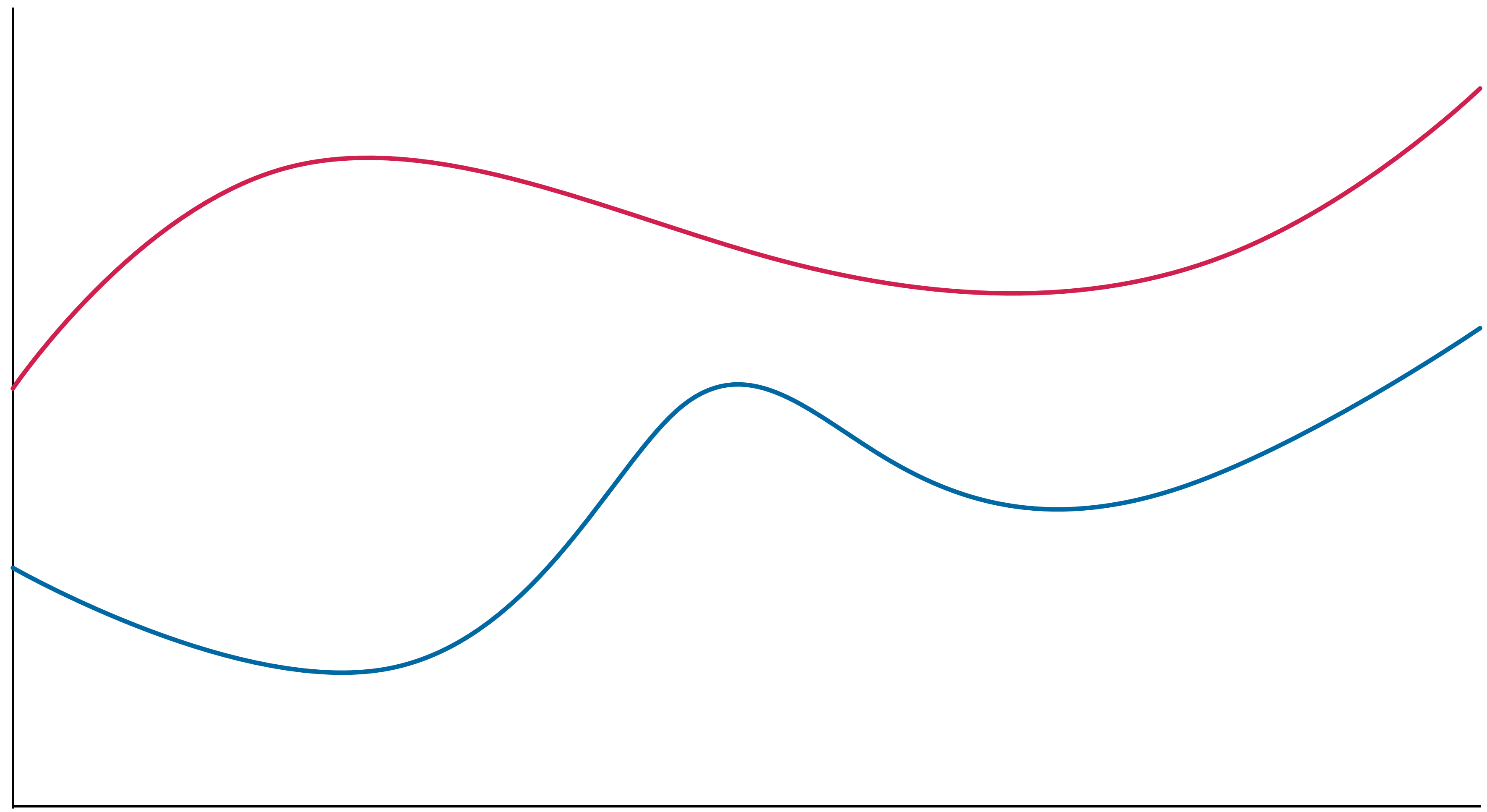
$$\exists \vec{x}_0, \vec{x}_1, \ldots, \vec{x}_k \exists \vec{x}^t_0, \vec{x}^t_1, \ldots, \vec{x}^t_k \exists t_0, t_1, \ldots, t_k$$

$$Init(\vec{x}_0) \wedge flow_{q_0}(\vec{x}_0, \vec{x}^t_0, t_0) \wedge jump_{q_0\to q_1}(\vec{x}^t_0, \vec{x}_1) \wedge$$
$$flow_{q_1}(\vec{x}_1, \vec{x}^t_1, t_1) \wedge jump_{q_1\to q_2}(\vec{x}^t_1, \vec{x}_2) \wedge$$
$$\ldots$$
$$flow_{q_k}(\vec{x}_k, \vec{x}^t_k, t_k) \wedge Unsafe(\vec{x}^t_k)$$

δ-Reachability Analysis of Hybrid Systems



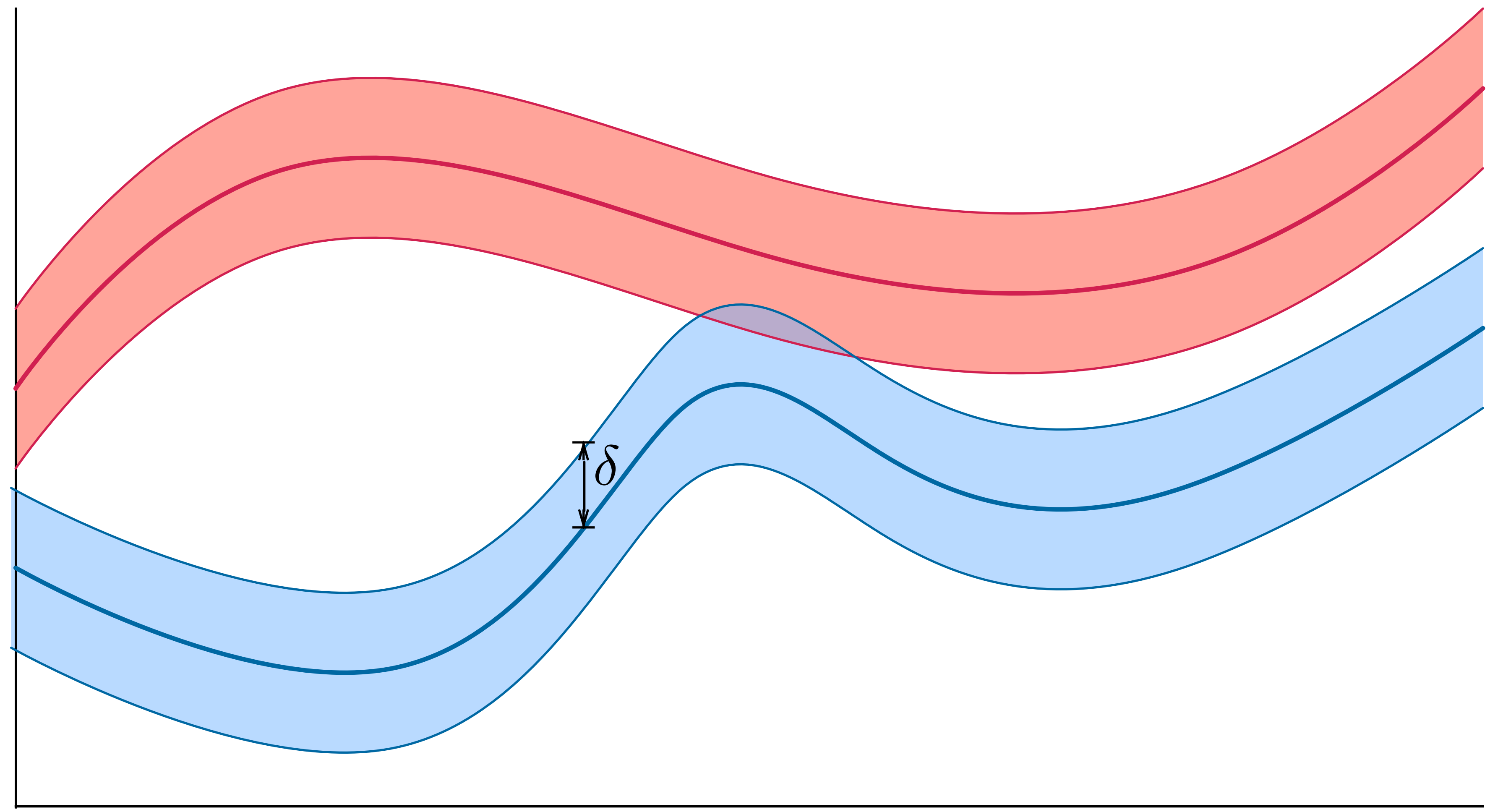
Decision Problem



Standard Form

$$\phi := \exists^{\mathbf{I}} \mathbf{x} \bigvee_i \bigwedge_j f_{i,j}(\mathbf{x}) = 0$$

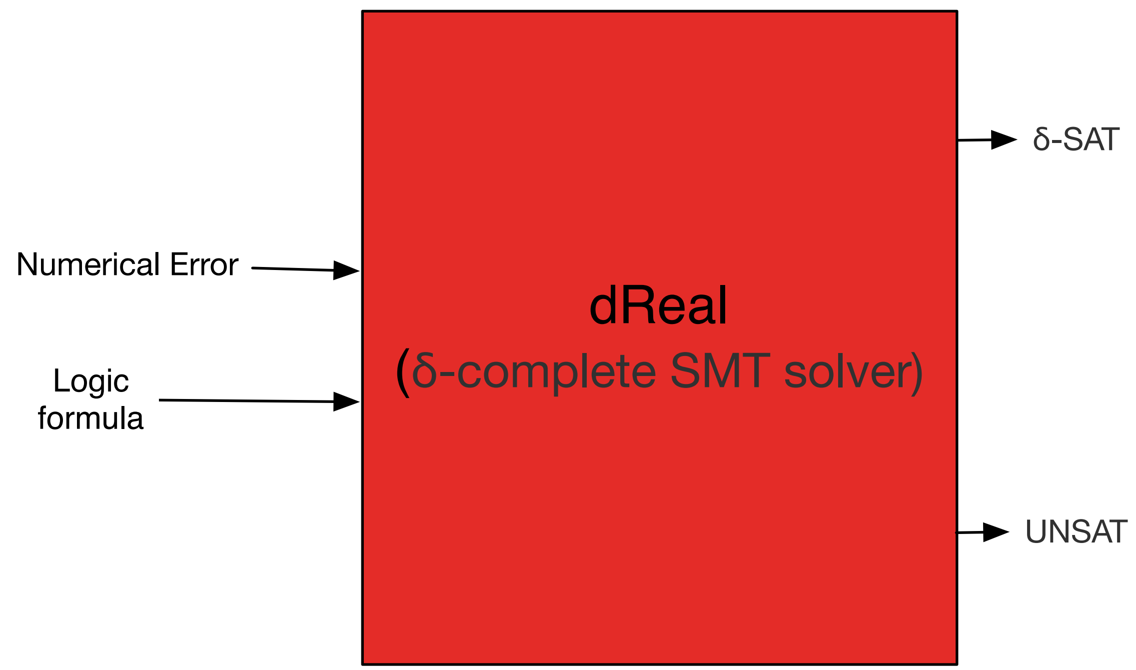
δ-Decision Problem



δ-Weakening of φ

$$\phi^{\delta} := \exists^{\mathbf{I}}\mathbf{x} \bigvee_i \bigwedge_j |f_{i,j}(\mathbf{x})| \leq \delta$$

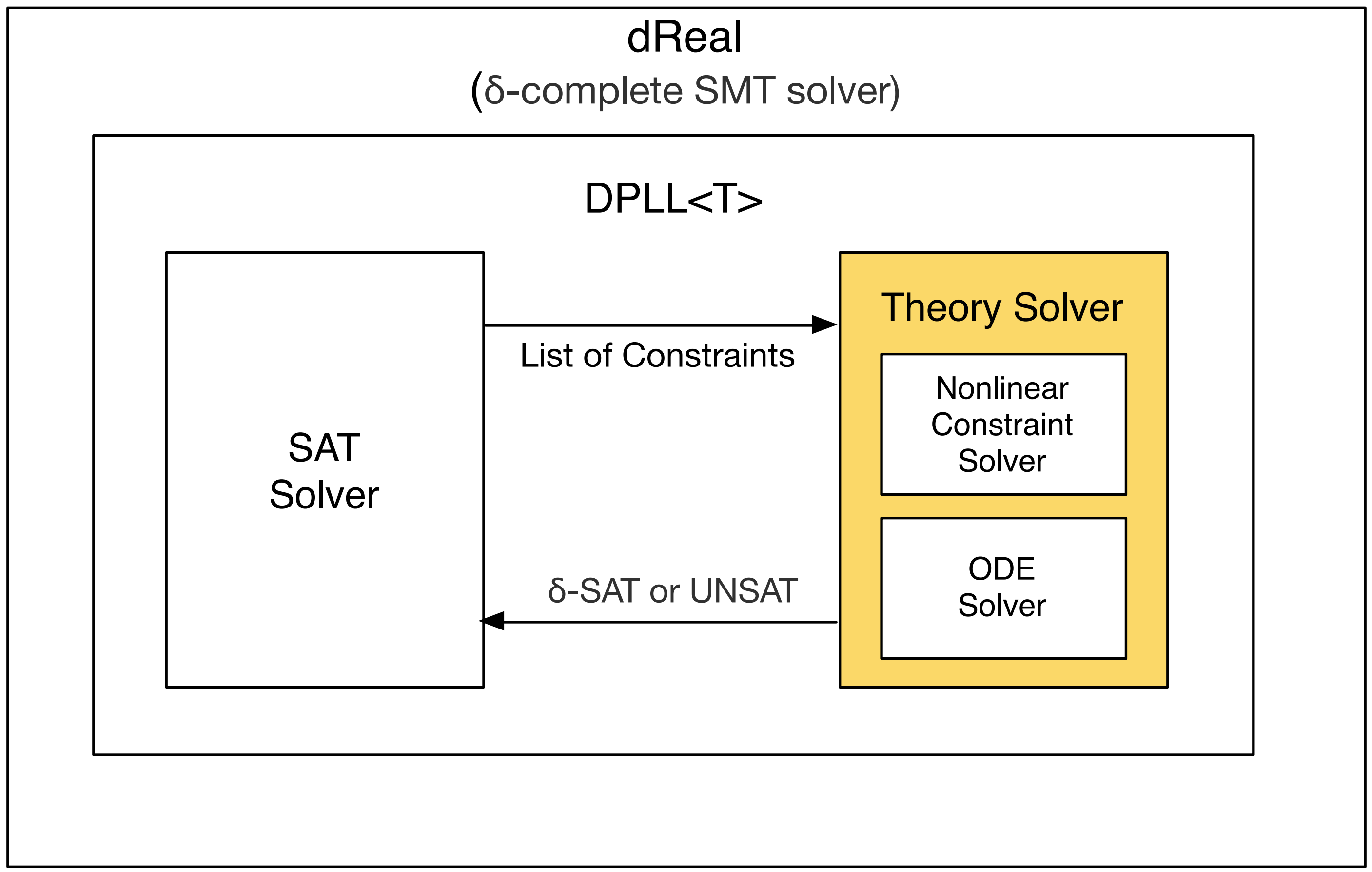
Solving Logic Formula



DPLL<T> Framework

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

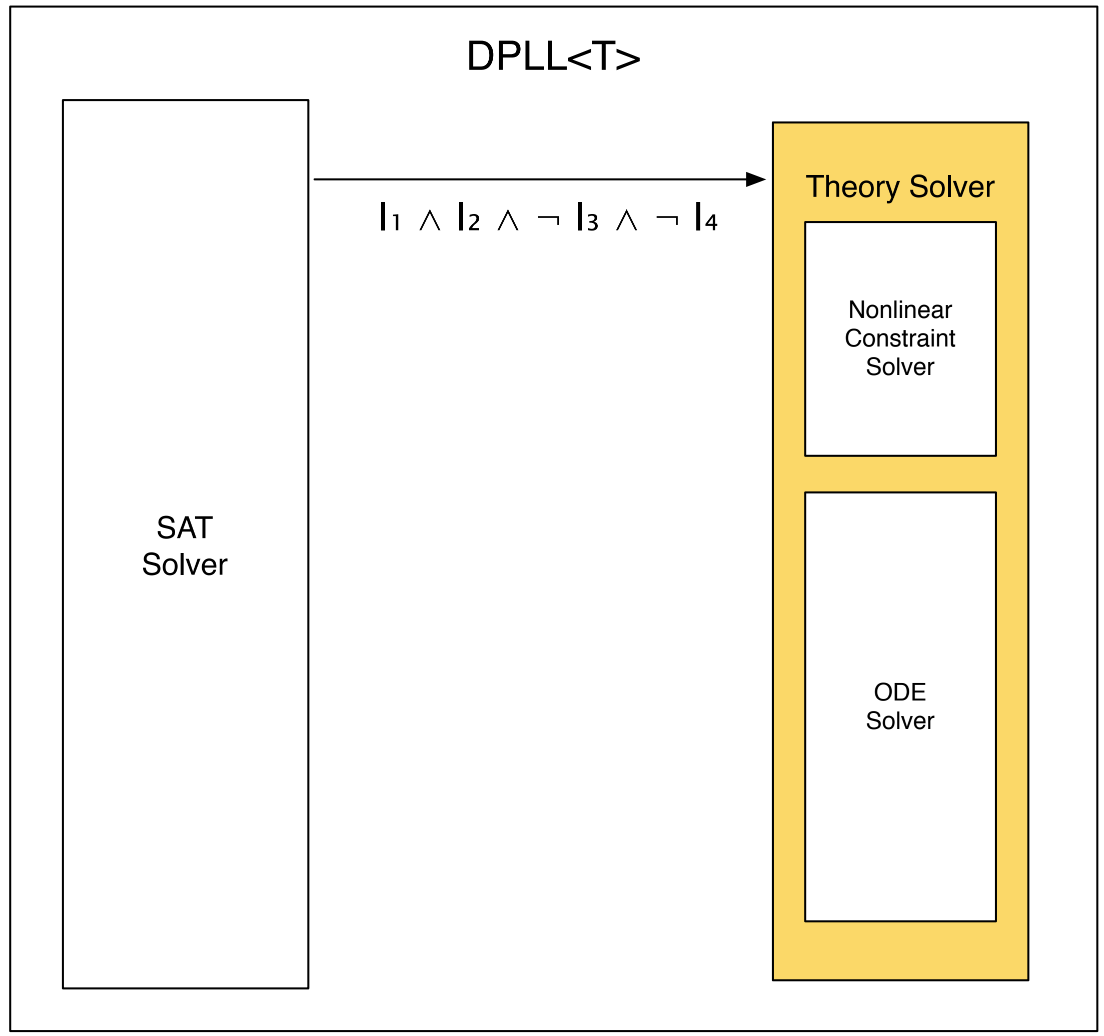
Solving Logic Formula



DPLL<T> Framework

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

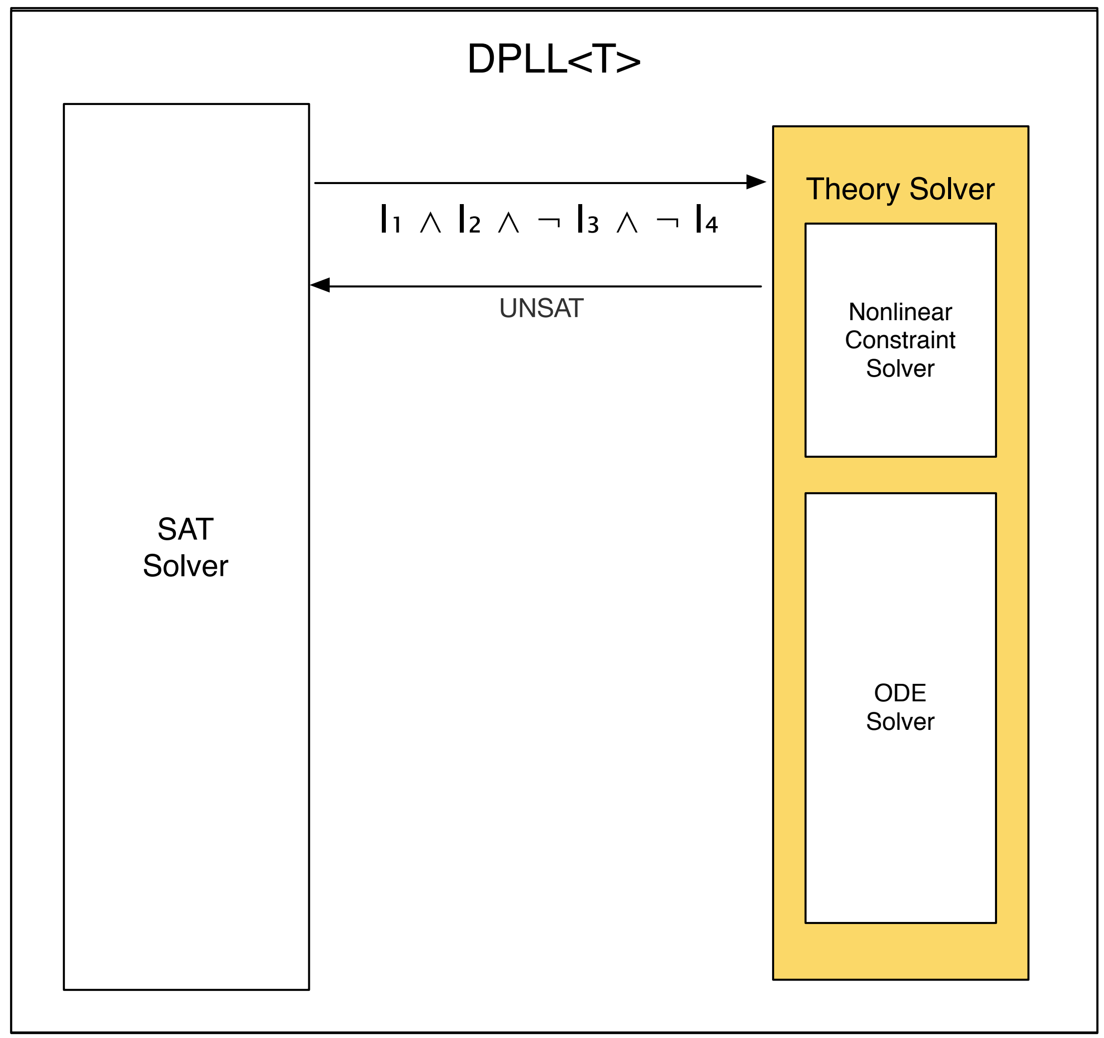
Solving Logic Formula



DPLL<T> Framework

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

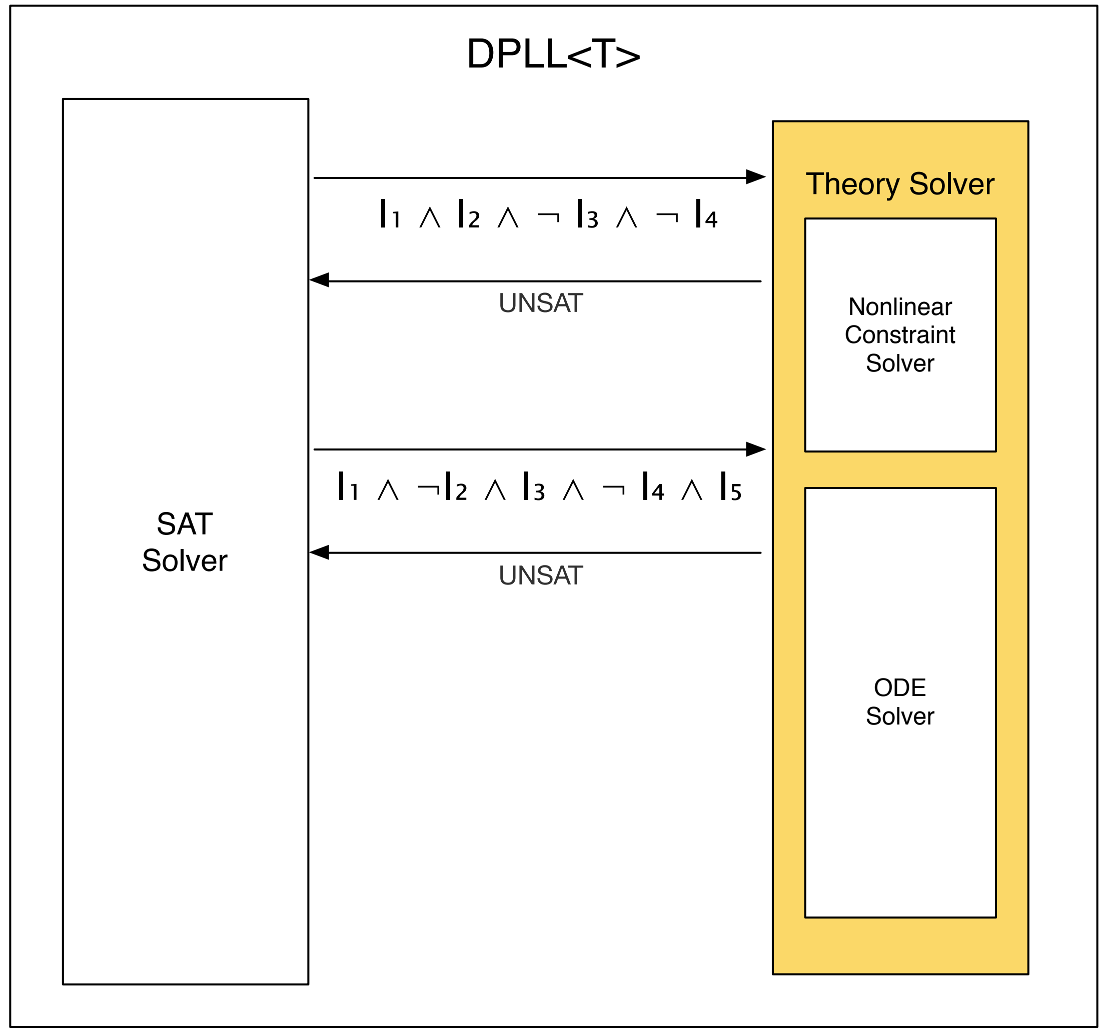
Solving Logic Formula



DPLL<T> Framework

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

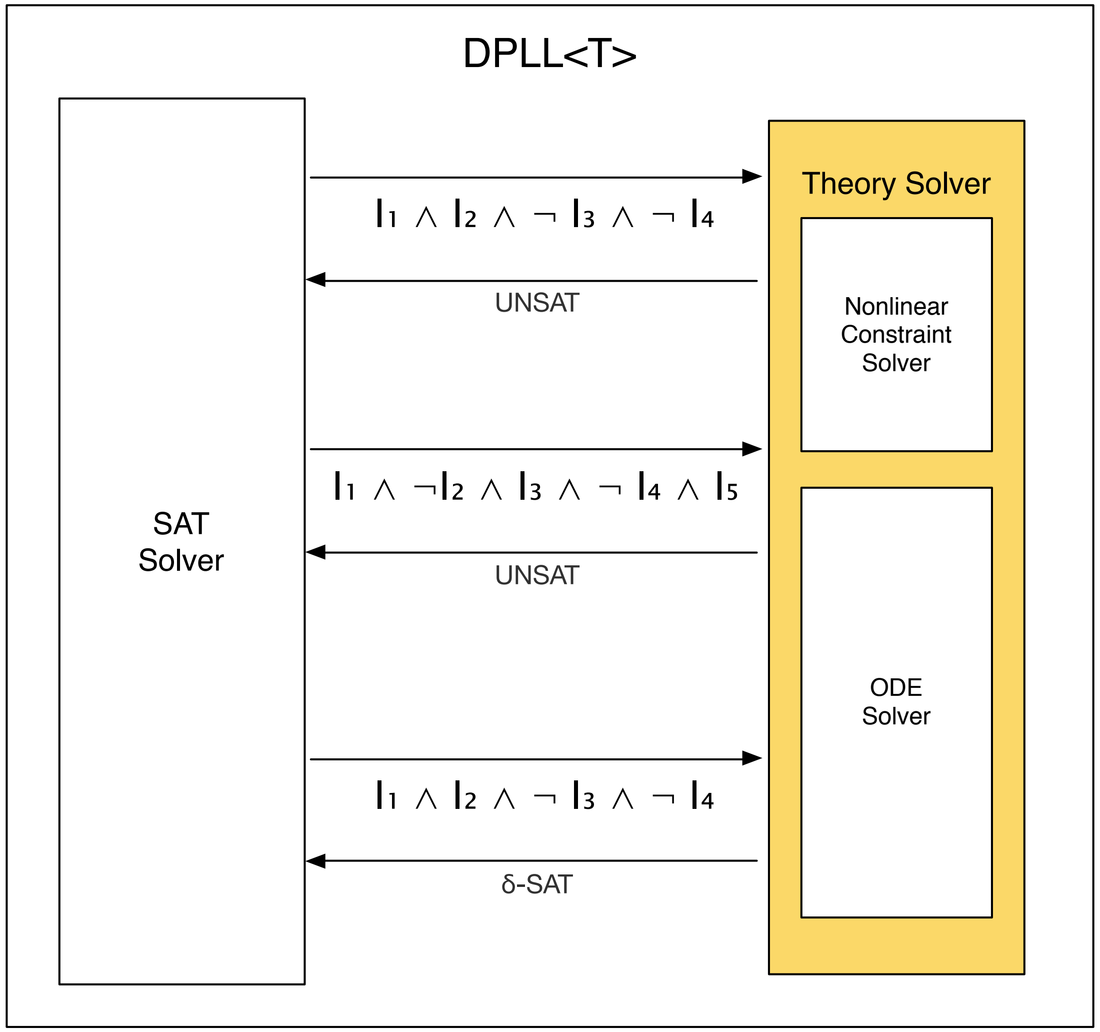
Solving Logic Formula



DPLL<T> Framework

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

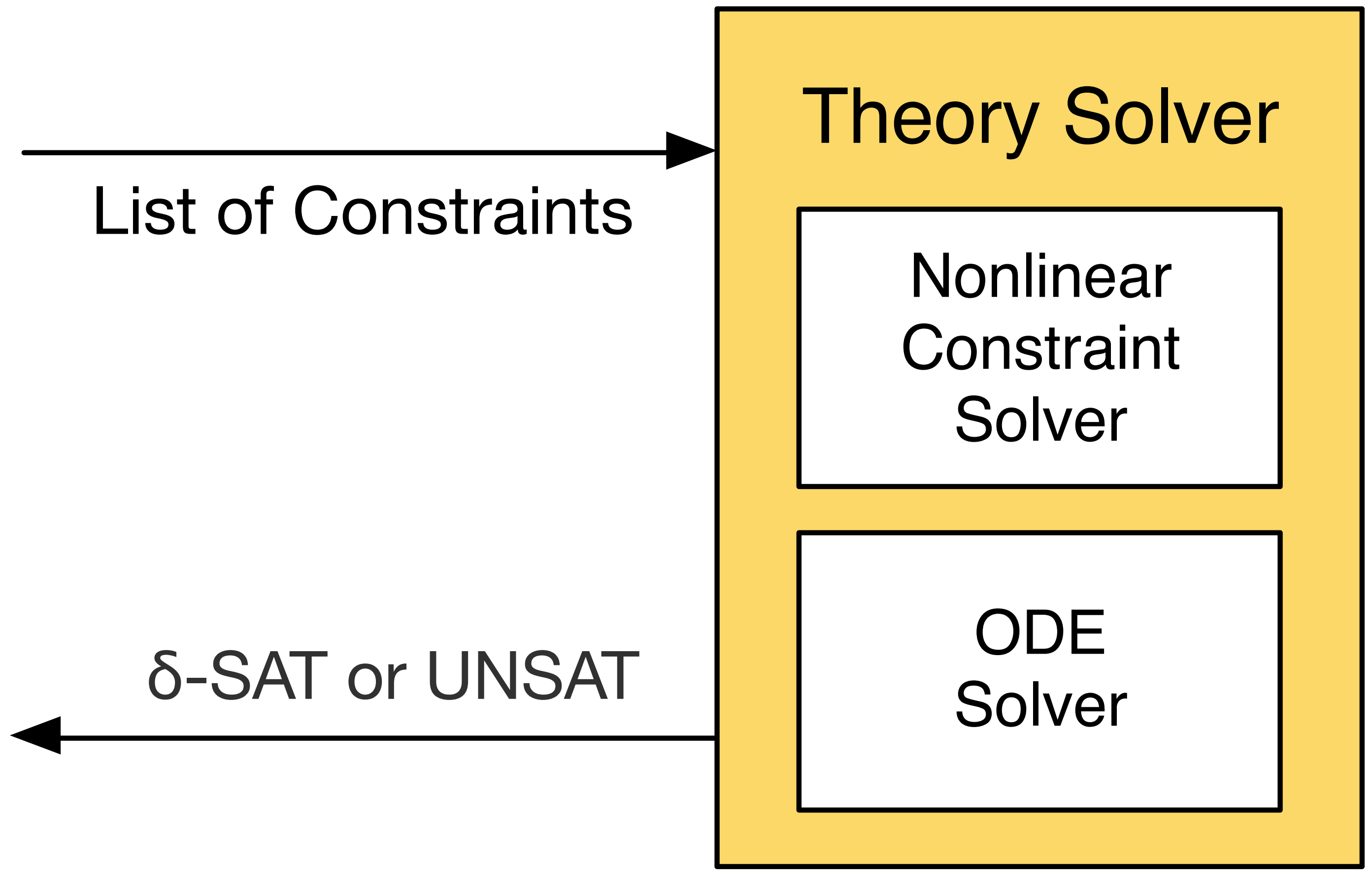
Solving Logic Formula



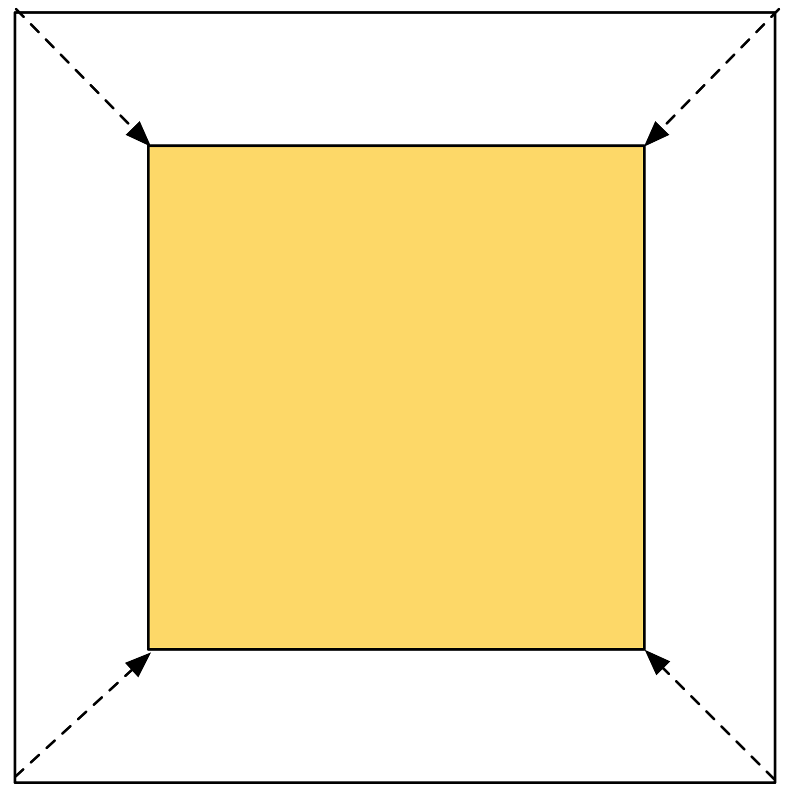
DPLL<T> Framework

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

Main Algorithm of Theory Solver



Main Algorithm of Theory Solver: ICP(Interval Constraint Propagation)



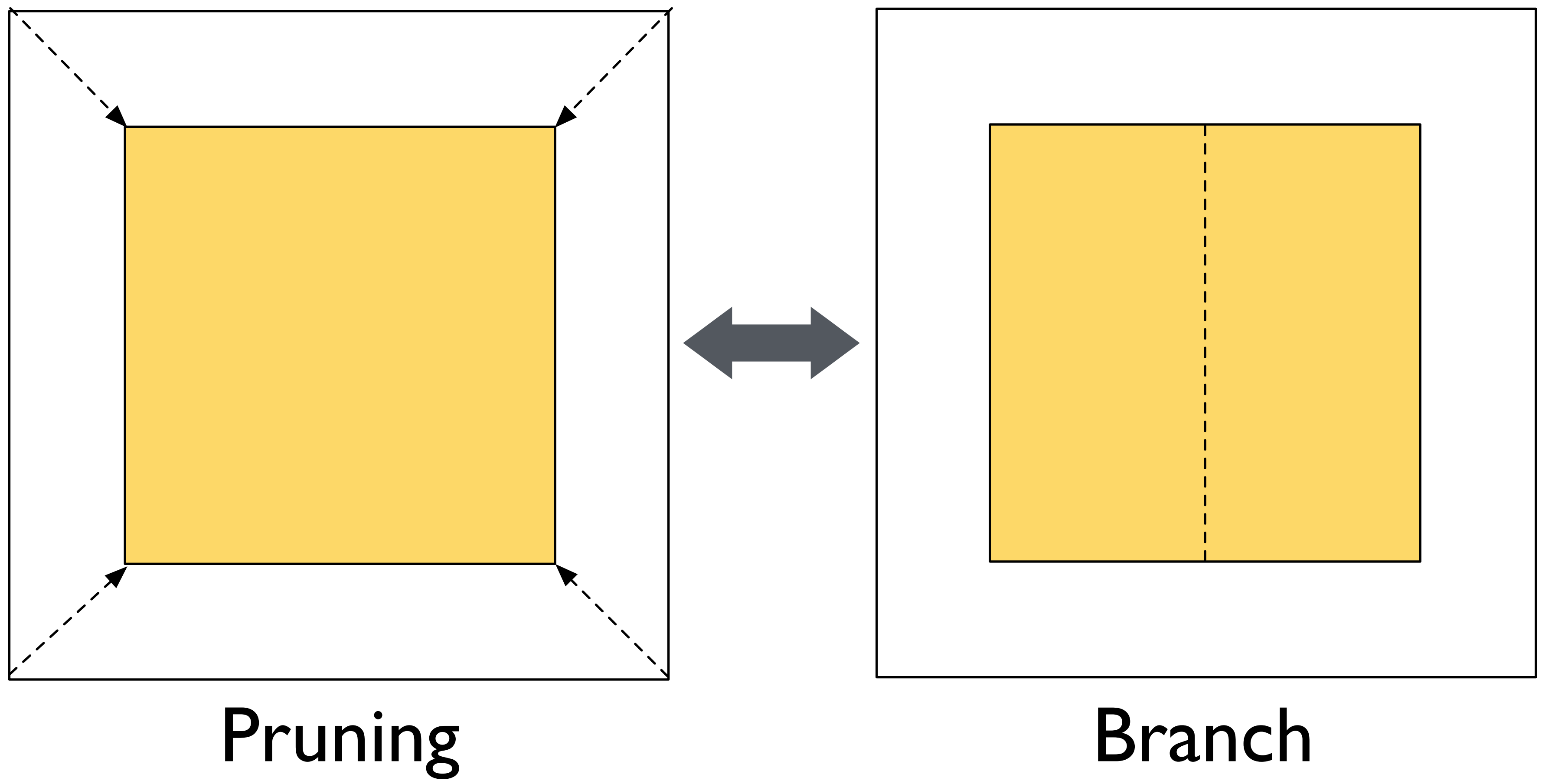
Pruning

Monotone $B_1 \subseteq B_2 \implies f(B_1) \subseteq f(B_2)$

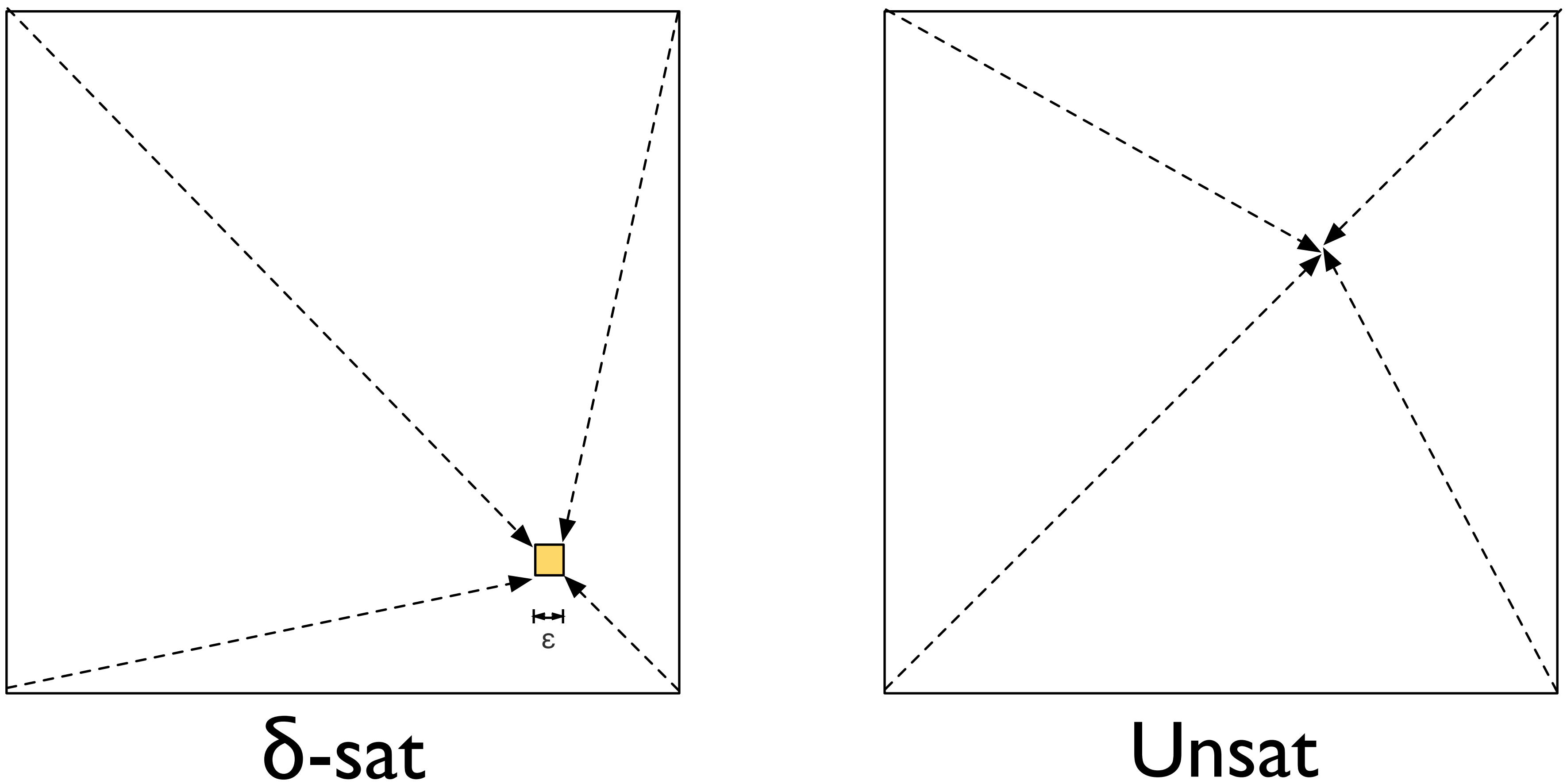
Reductive $f(B) \subseteq B$

Solution-Preserving $x \in B \wedge x \in Sol(f) \implies x \in f(B)$

Main Algorithm of Theory Solver: ICP(Interval Constraint Propagation)

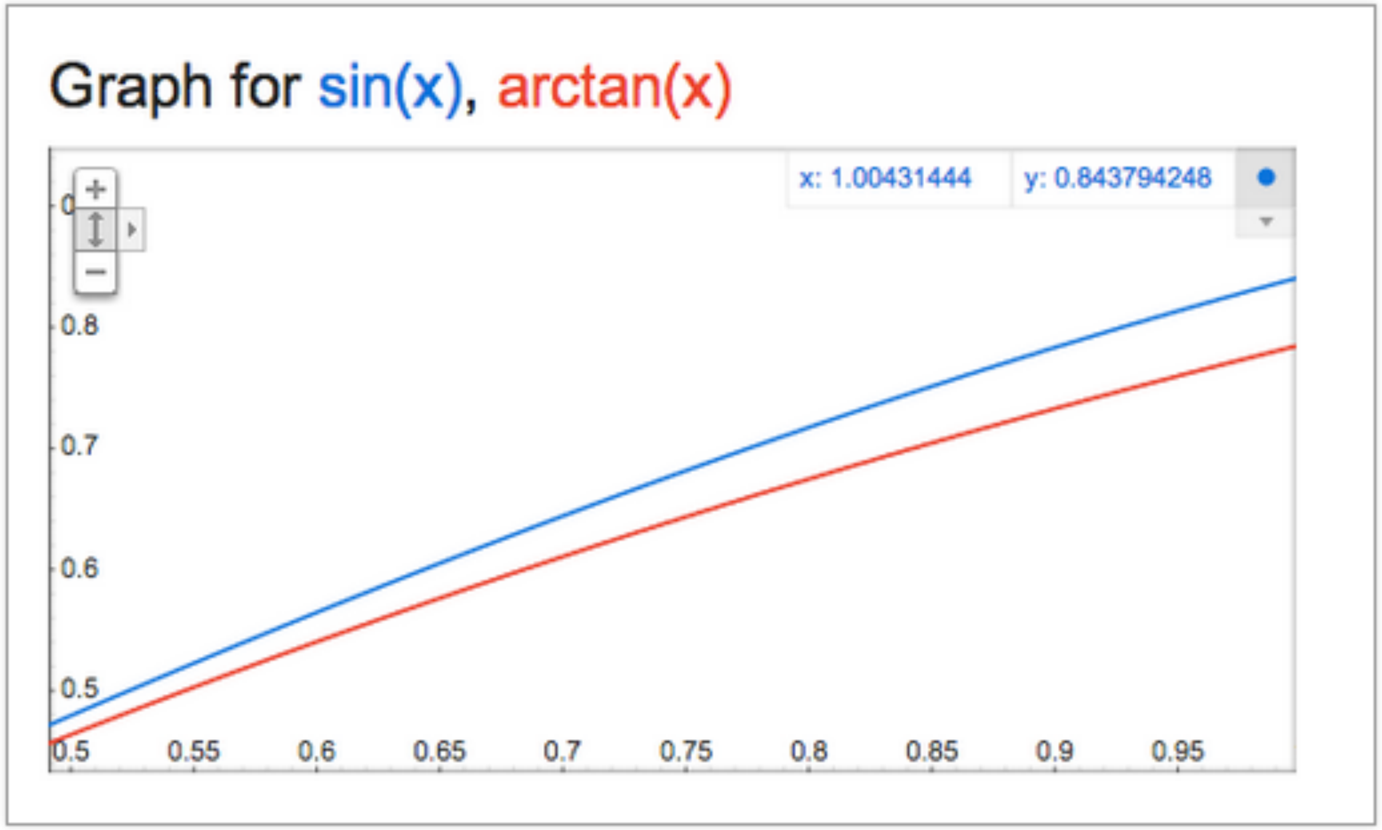


Two Termination Conditions of ICP



Example of Pruning Operations

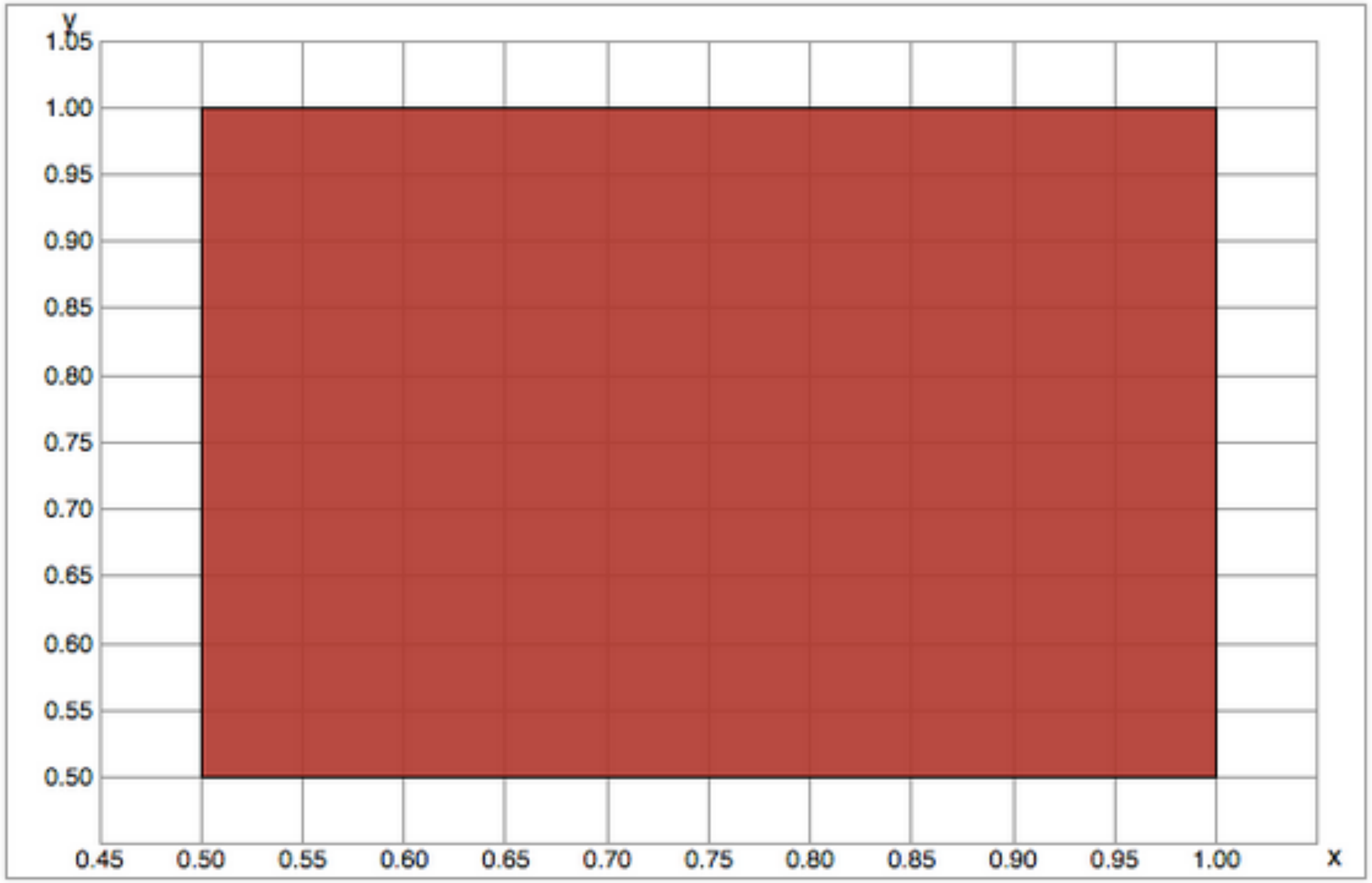
$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



ANSWER: **UNSAT**

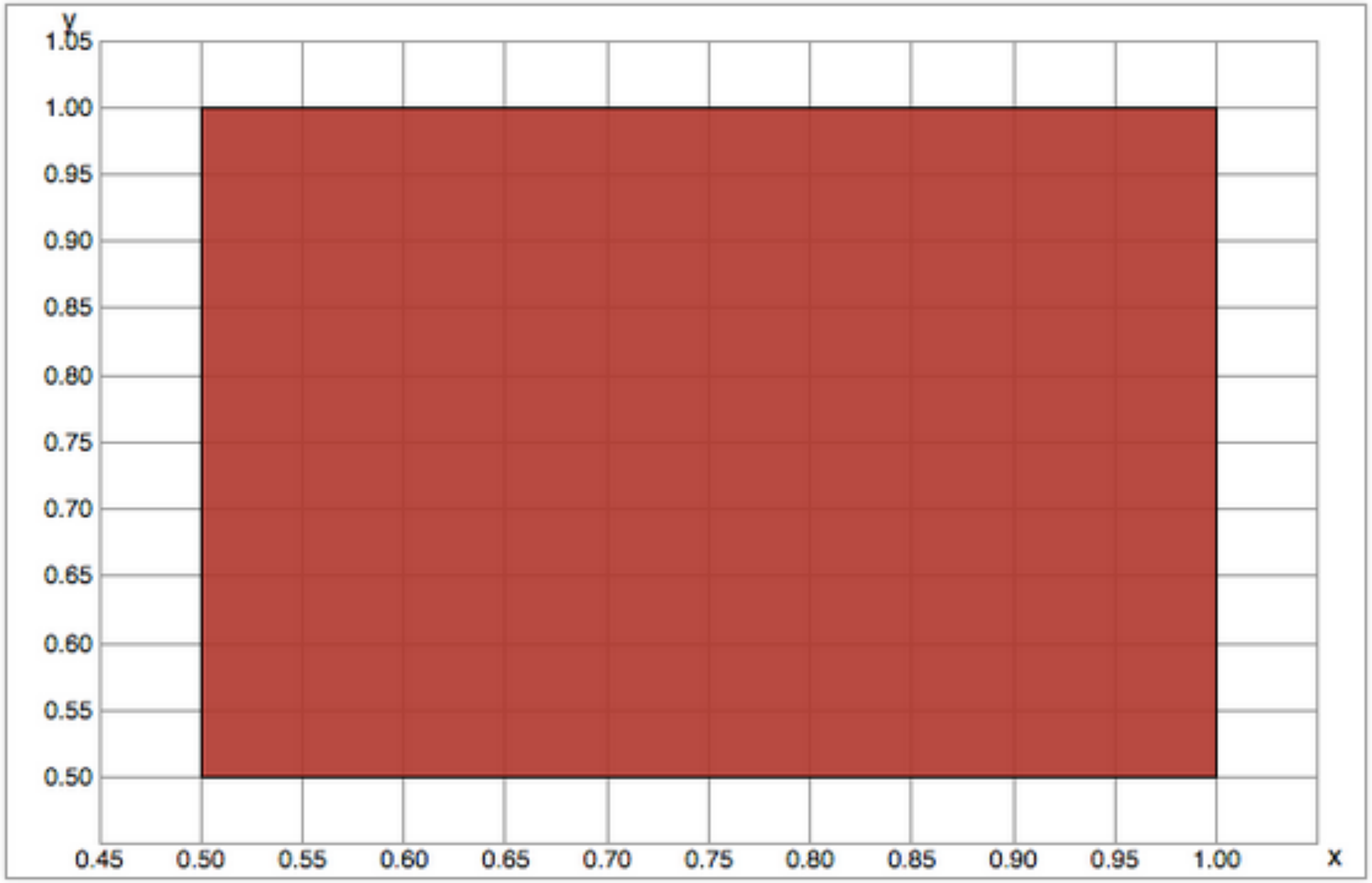
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \text{atan}(x)$$



Example of Pruning Operations

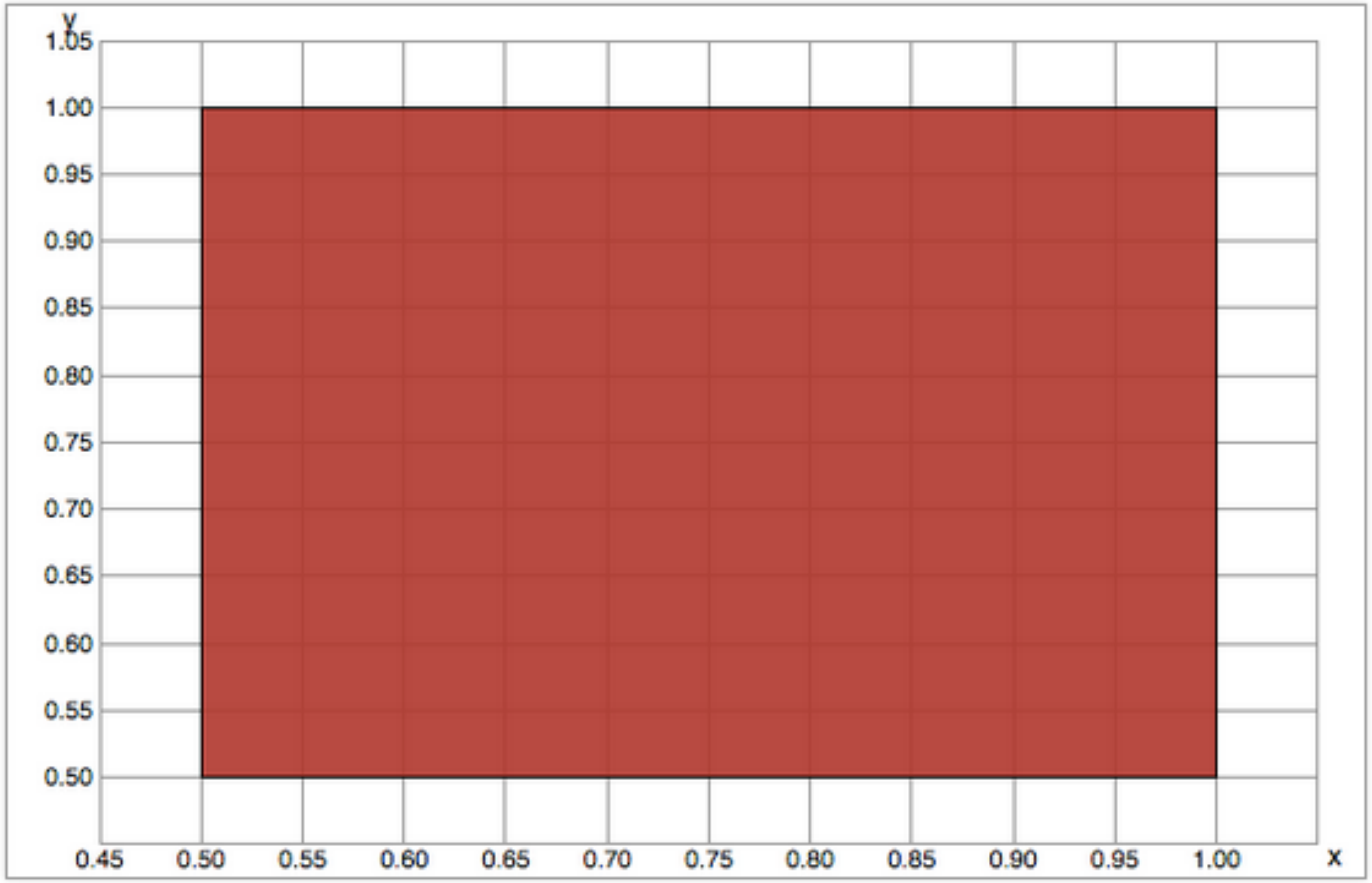
$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



$$\begin{aligned} x' &= x \cap \sin^{-1}(y) \\ &= [0.5, 1.0] \cap \sin^{-1}([0.5, 1.0]) \\ &= [0.5, 1.0] \cap [0.524, 1.570] \\ &= [0.524, 1.0] \end{aligned}$$

Example of Pruning Operations

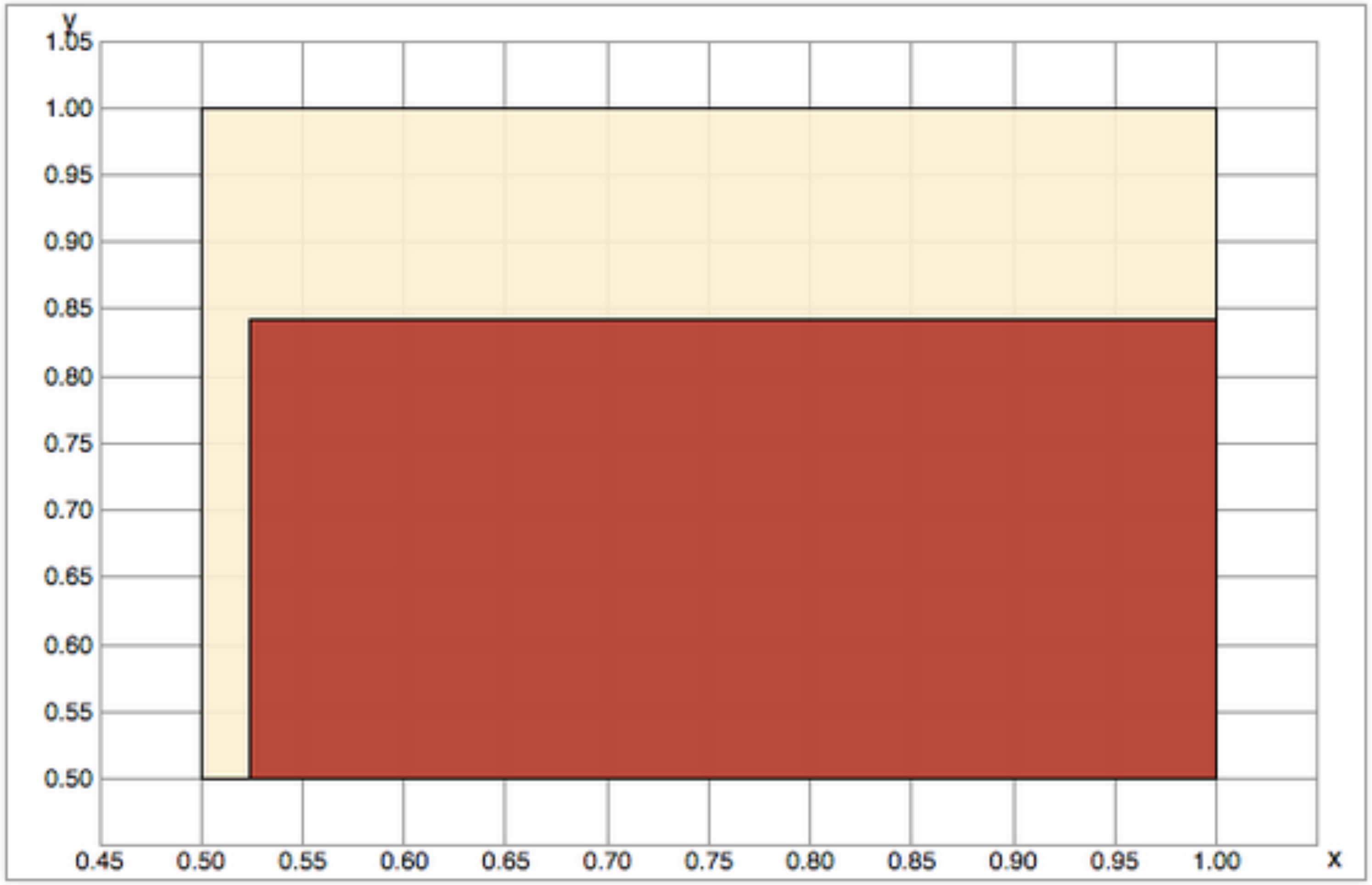
$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



$$\begin{aligned} y' &= y \cap \sin(x) \\ &= [0.5, 1.0] \cap \sin([0.524, 1.0]) \\ &= [0.5, 1.0] \cap [0.5, 0.841] \\ &= [0.5, 0.841] \end{aligned}$$

Example of Pruning Operations

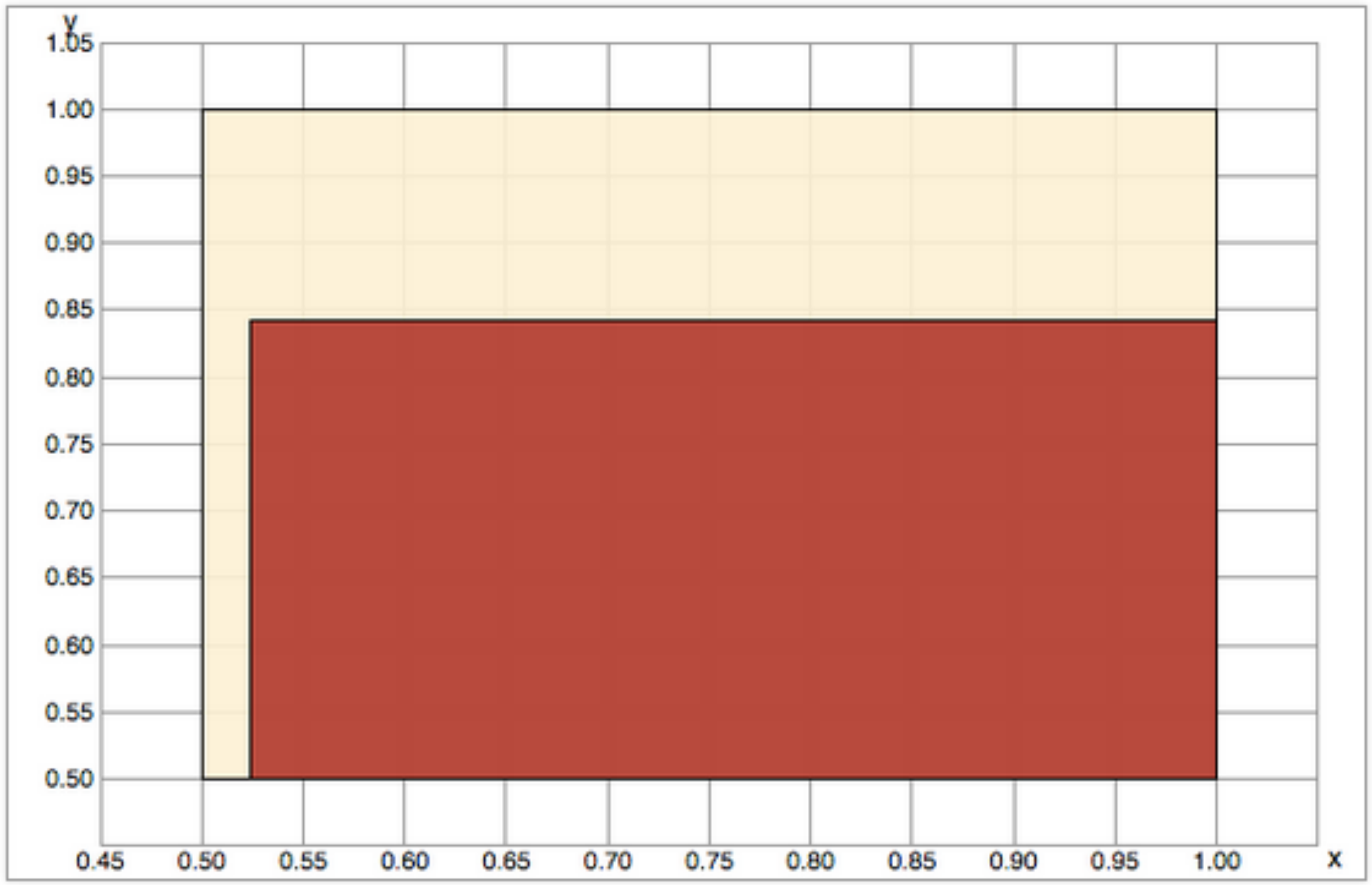
$$\exists x, y \in [0.5, 1.0] : {\color{red} y = \sin(x)} \wedge y = \text{atan}(x)$$



$$x : [0.524, 1.0], y : [0.5, 0.841]$$

Example of Pruning Operations

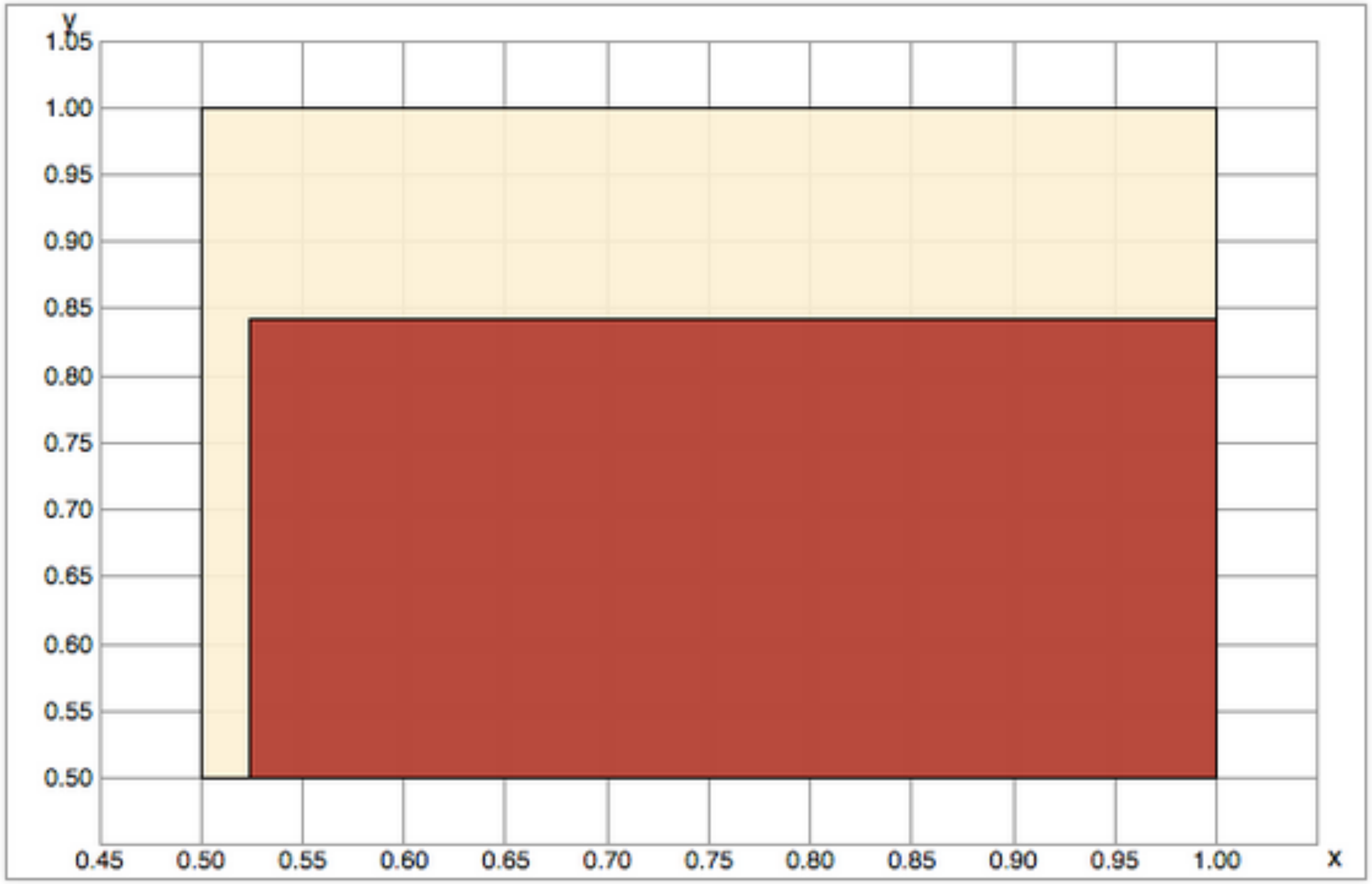
$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



$$\begin{aligned} x' &= x \cap \mathrm{atan}^{-1}(y) \\ &= [0.524, 1] \cap \mathrm{atan}^{-1}([0.5, 0.841]) \\ &= [0.524, 1] \cap [0.546, 1.117] \\ &= [0.546, 1.0] \end{aligned}$$

Example of Pruning Operations

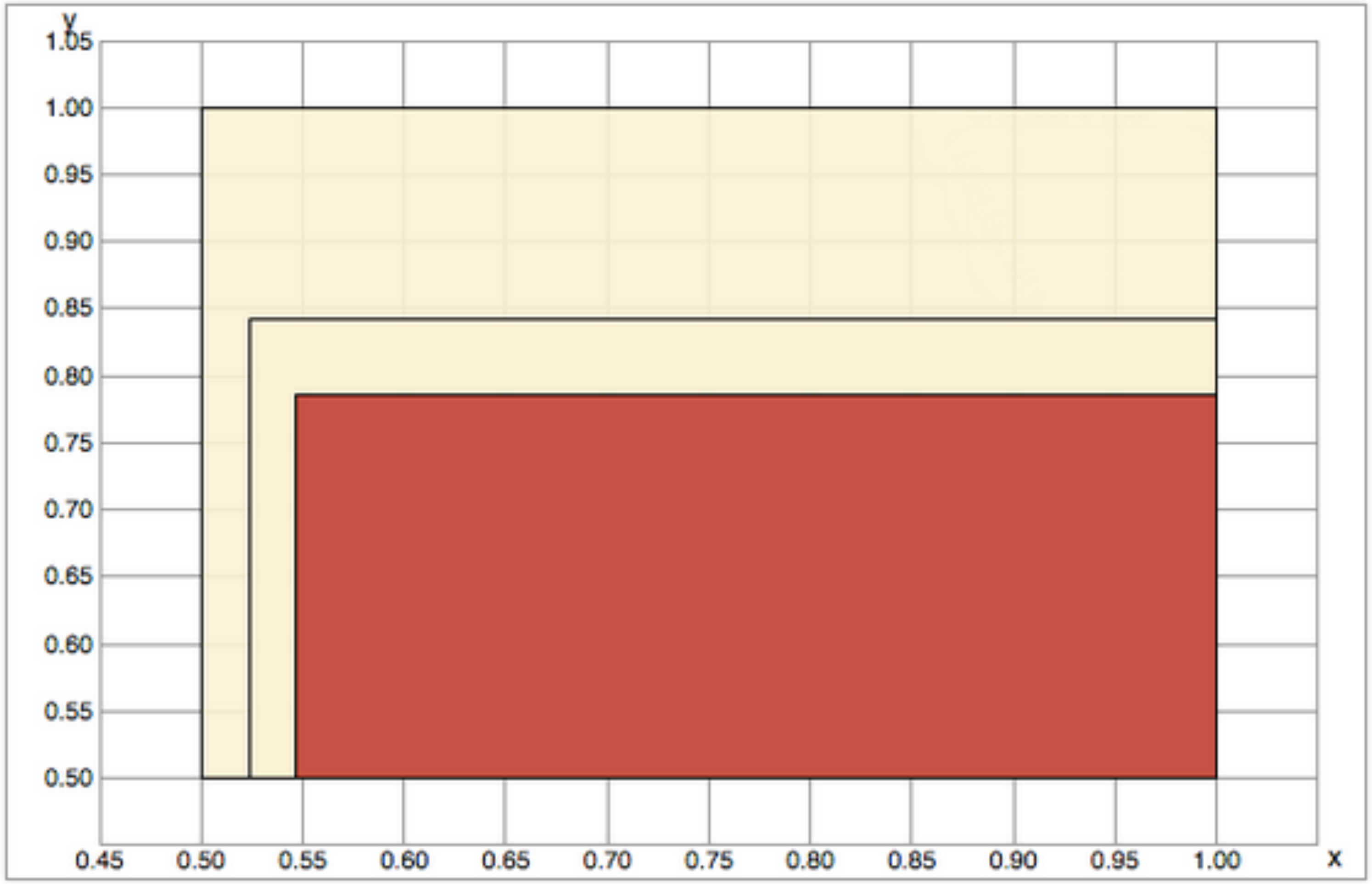
$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \text{atan}(x)$$



$$\begin{aligned} y' &= y \cap \text{atan}(x) \\ &= [0.5, 0.841] \cap \text{atan}([0.546, 1.0]) \\ &= [0.5, 0.841] \cap [0.5, 1.0] \\ &= [0.5, 0.785] \end{aligned}$$

Example of Pruning Operations

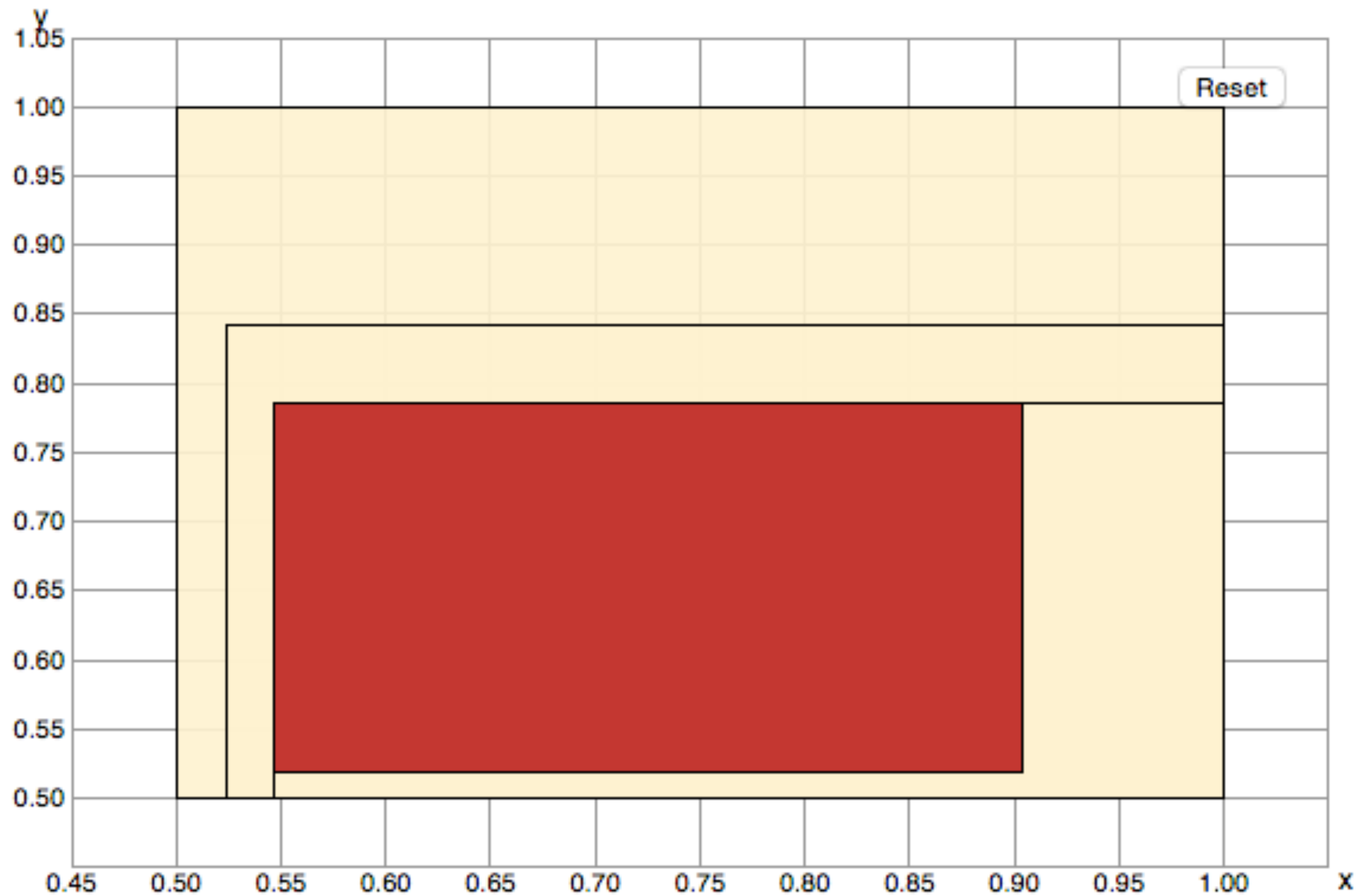
$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



$$x : [0.524, 1.0], y : [0.5, 0.785]$$

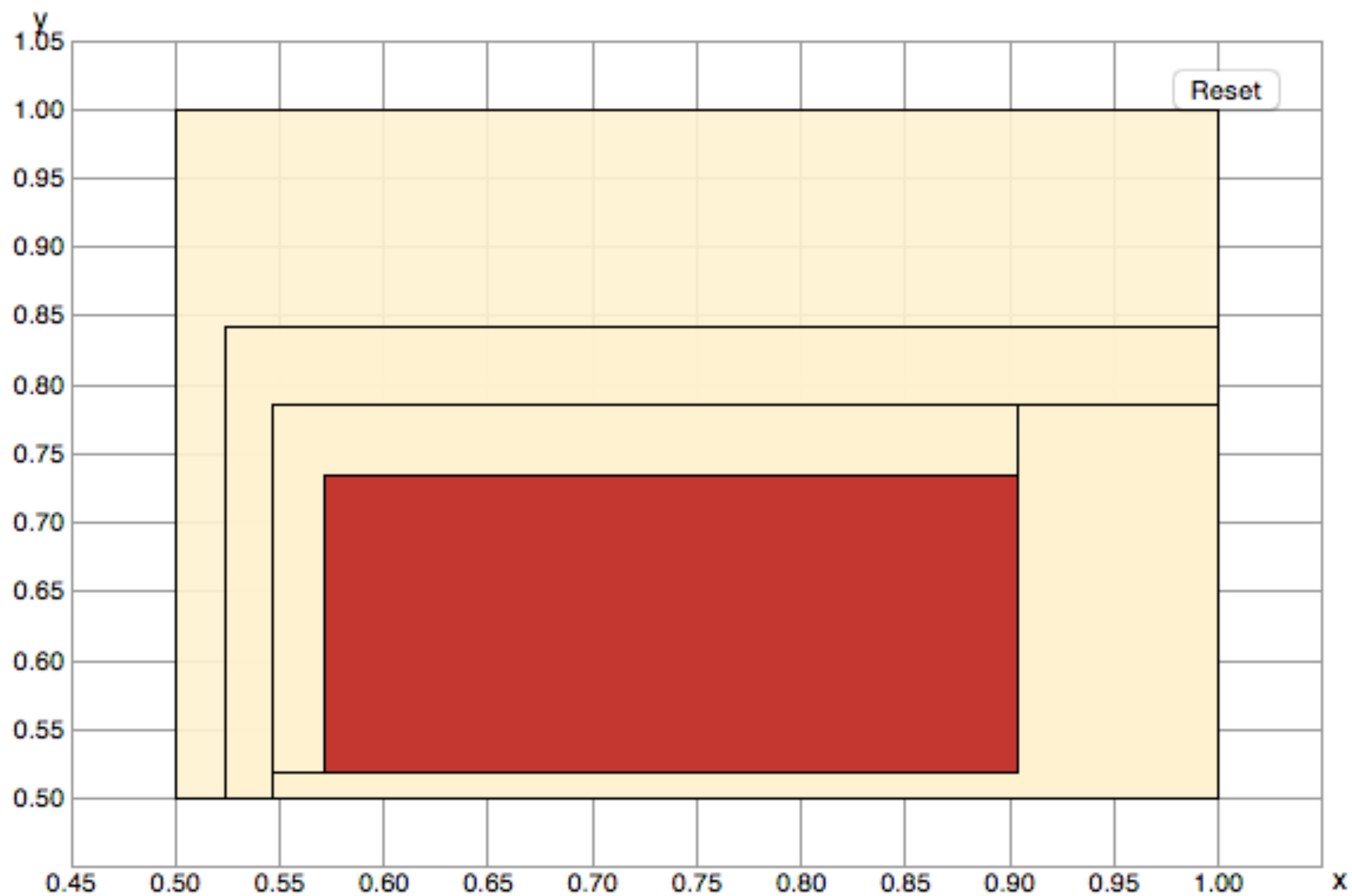
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \operatorname{atan}(x)$$



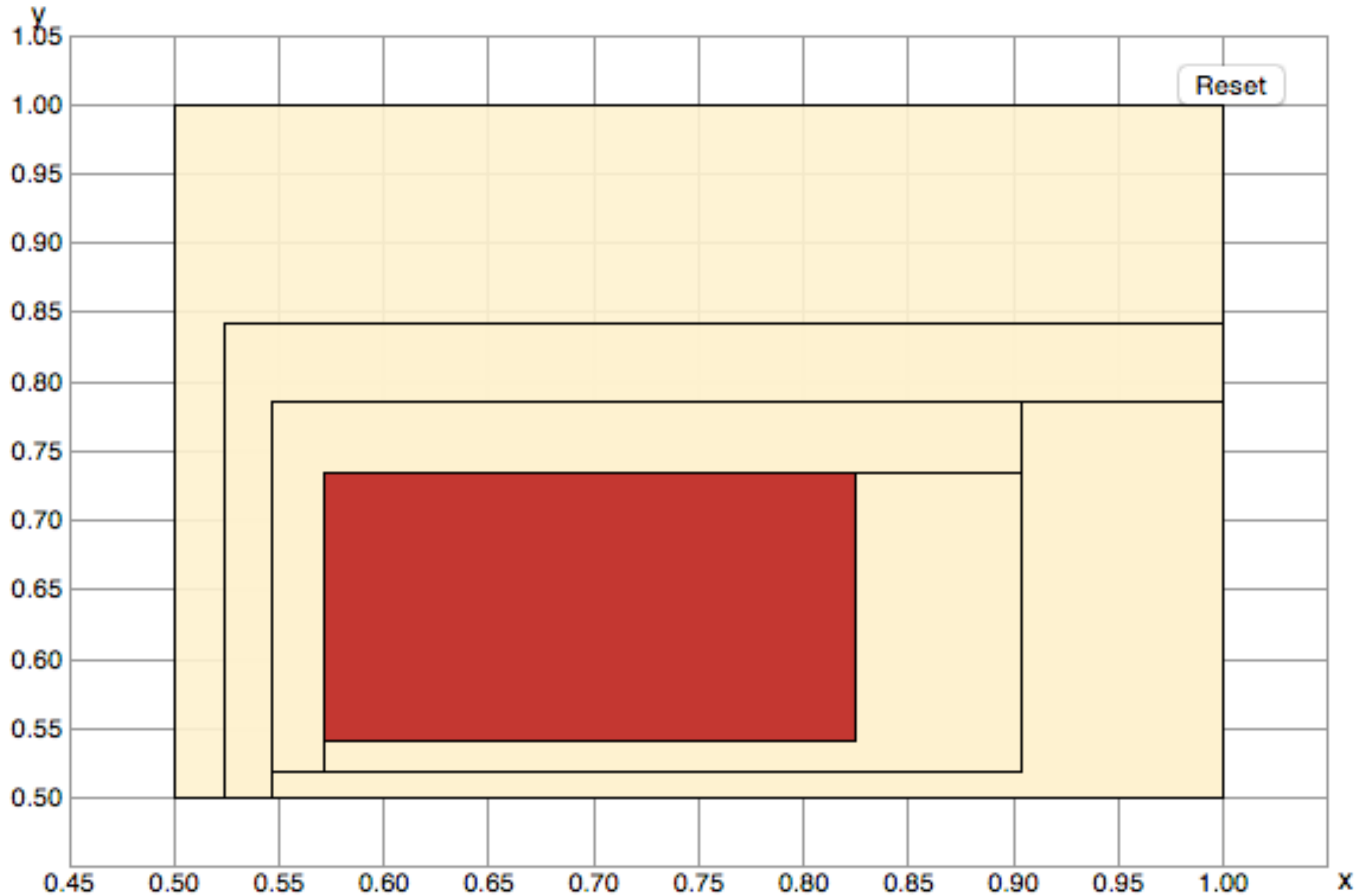
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



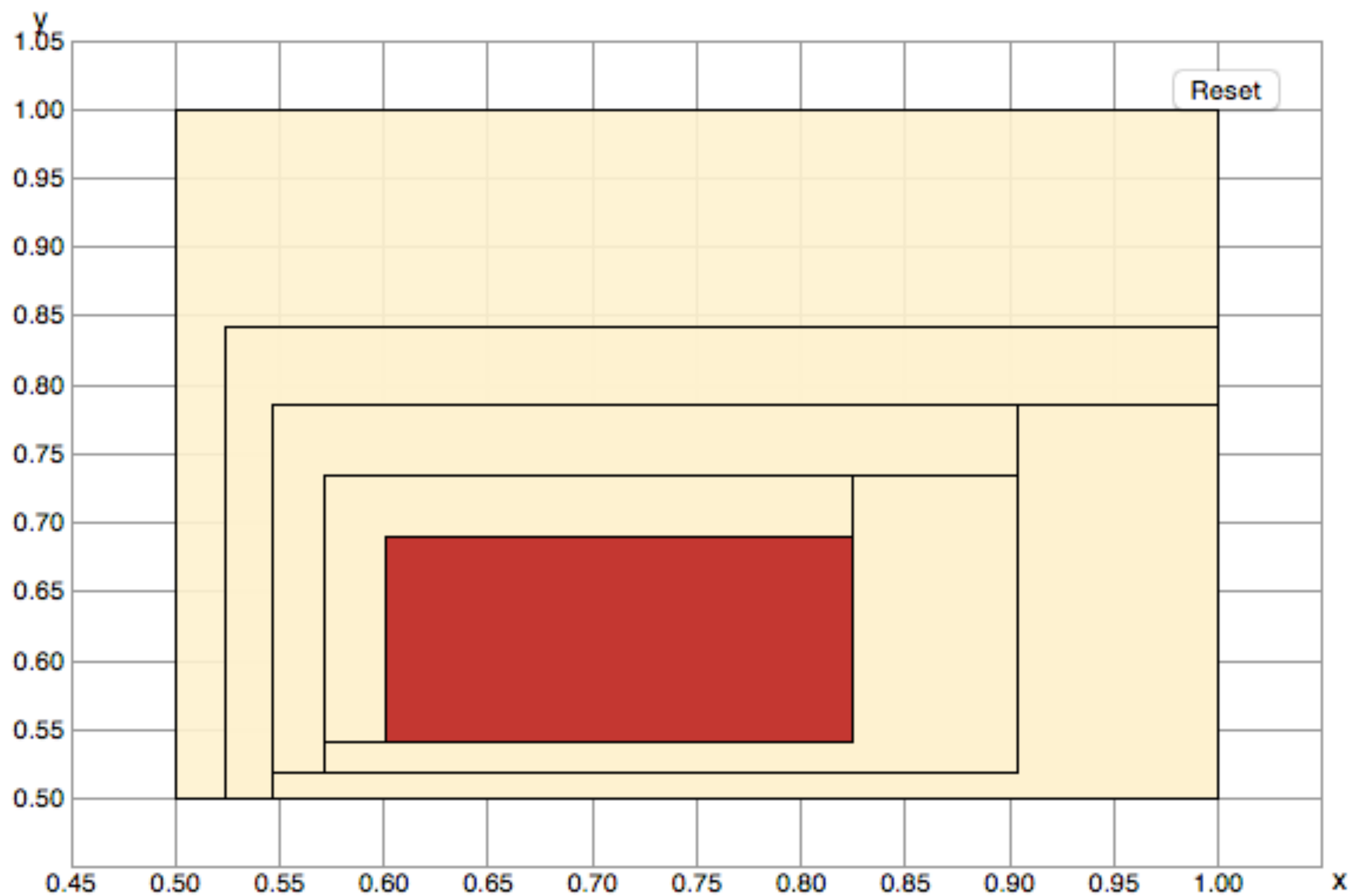
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



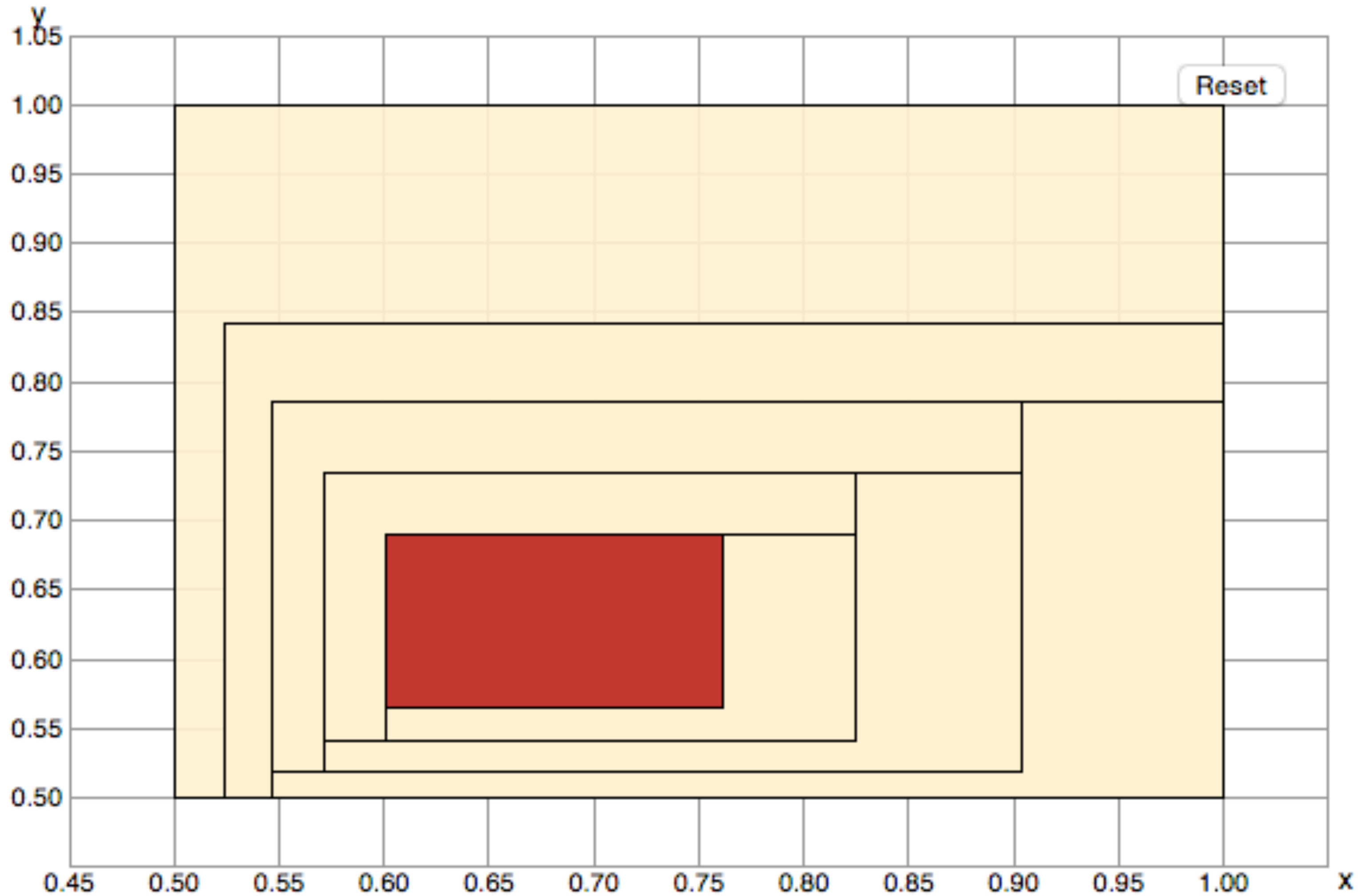
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \operatorname{atan}(x)$$



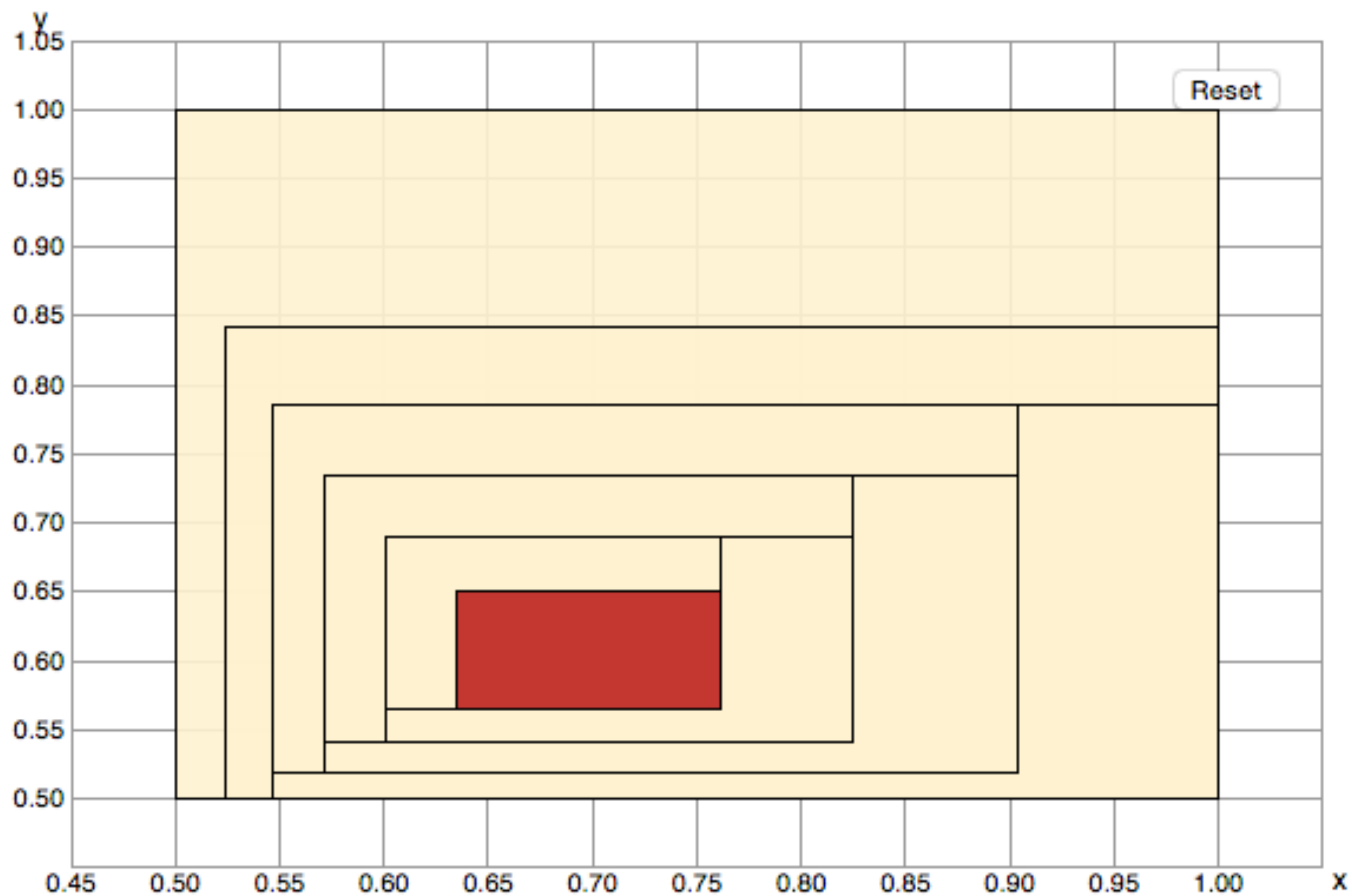
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



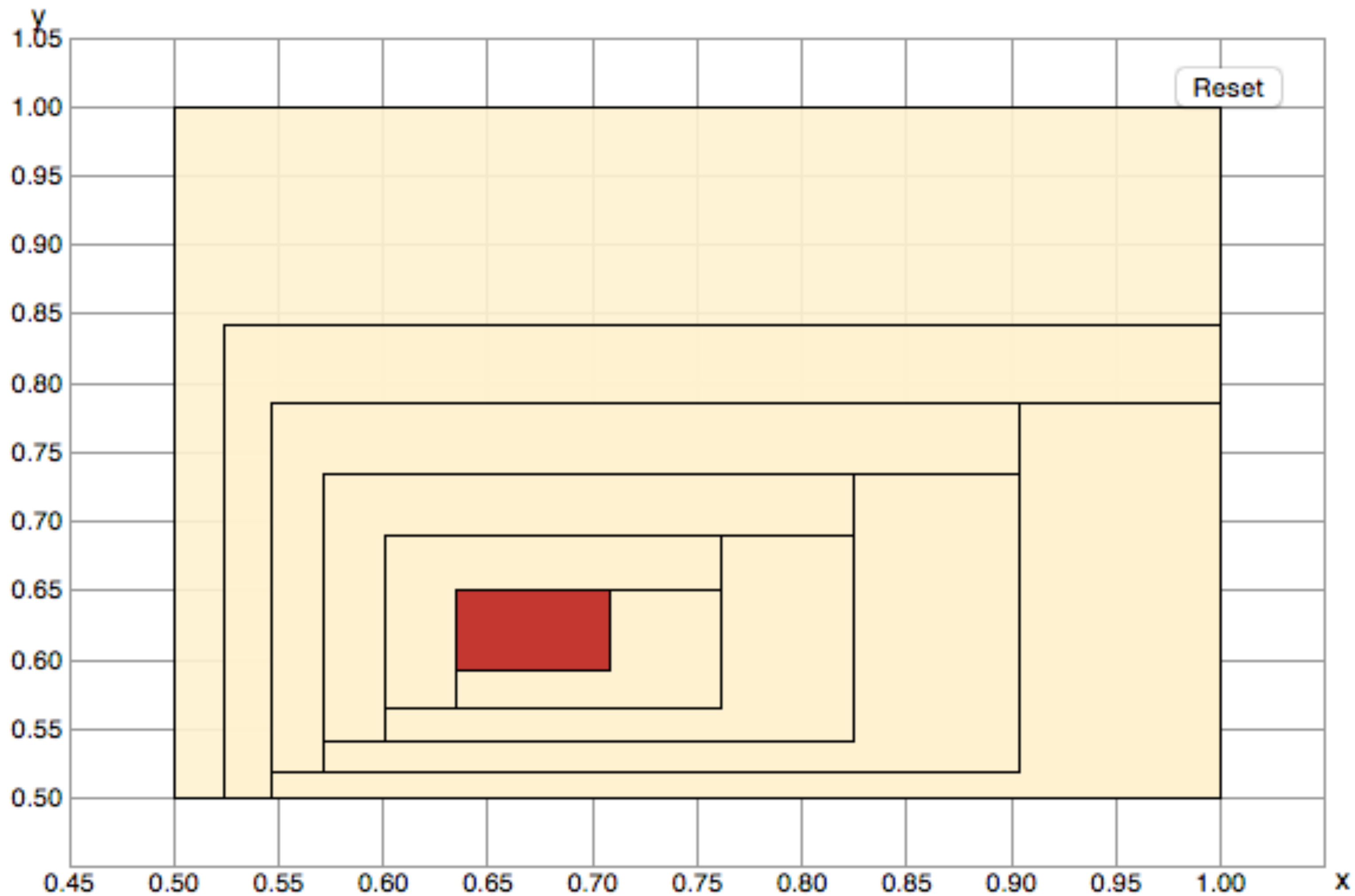
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \operatorname{atan}(x)$$



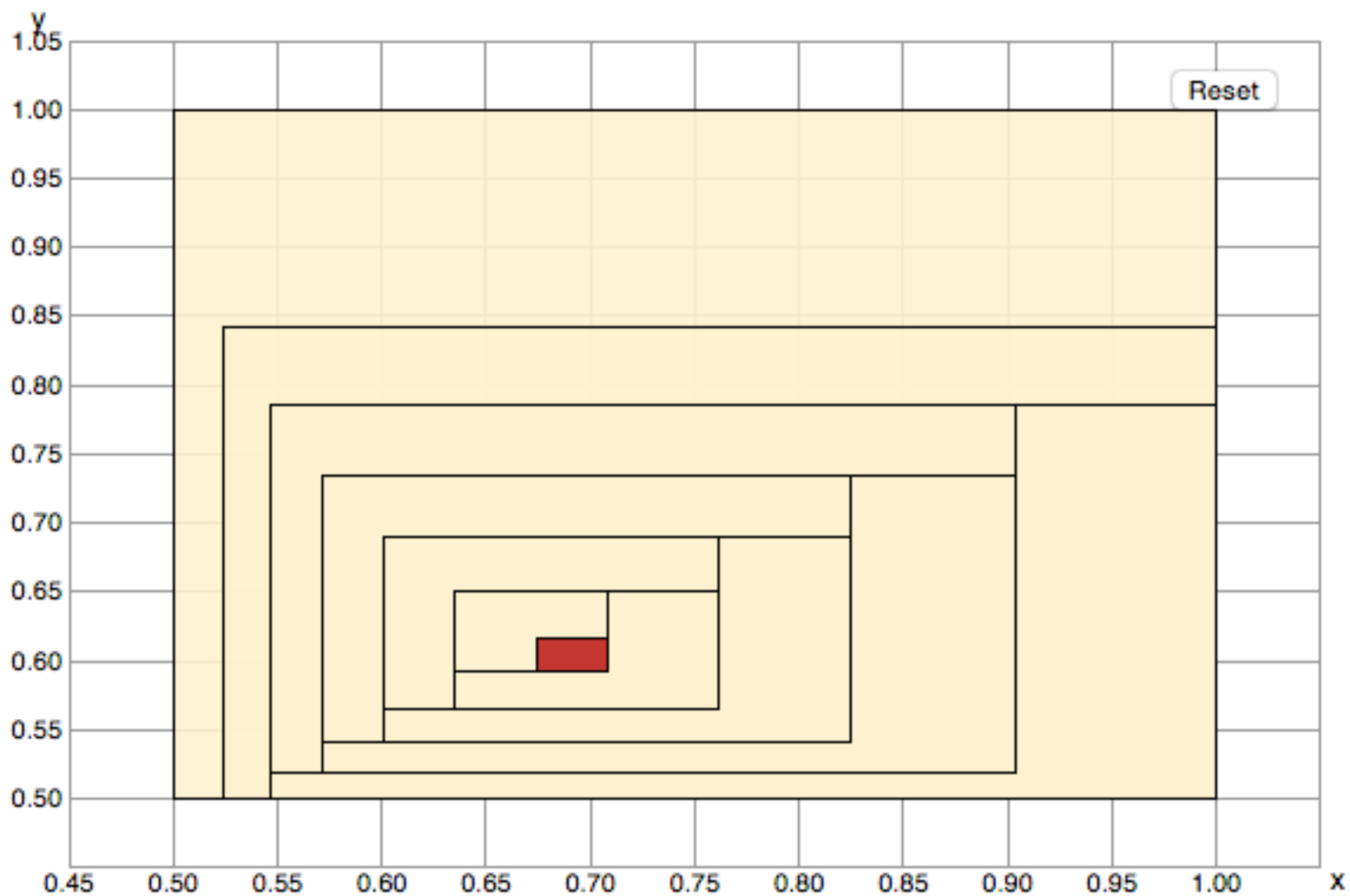
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



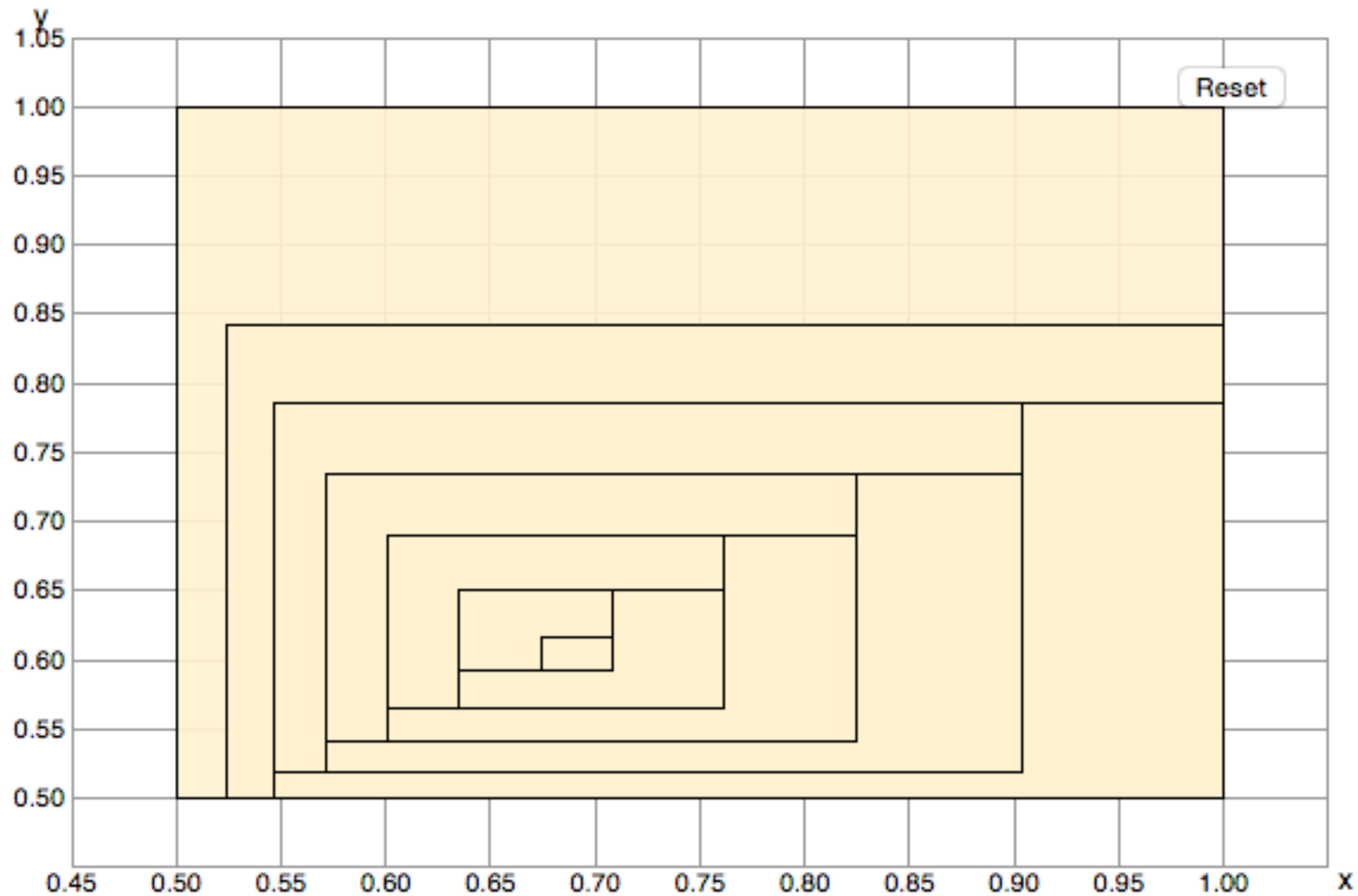
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \land y = \operatorname{atan}(x)$$



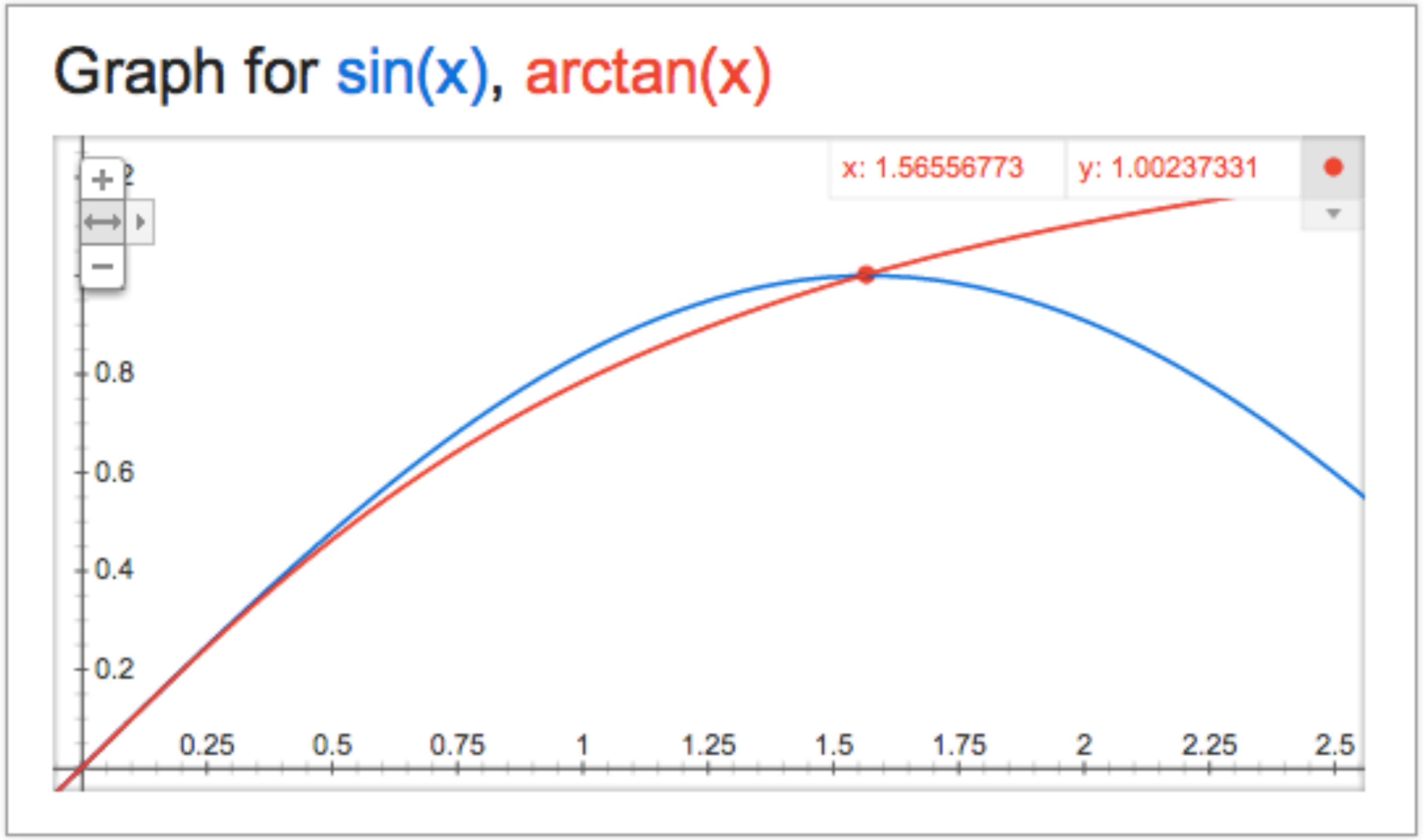
Example of Pruning Operations

$$\exists x, y \in [0.5, 1.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



Unsat

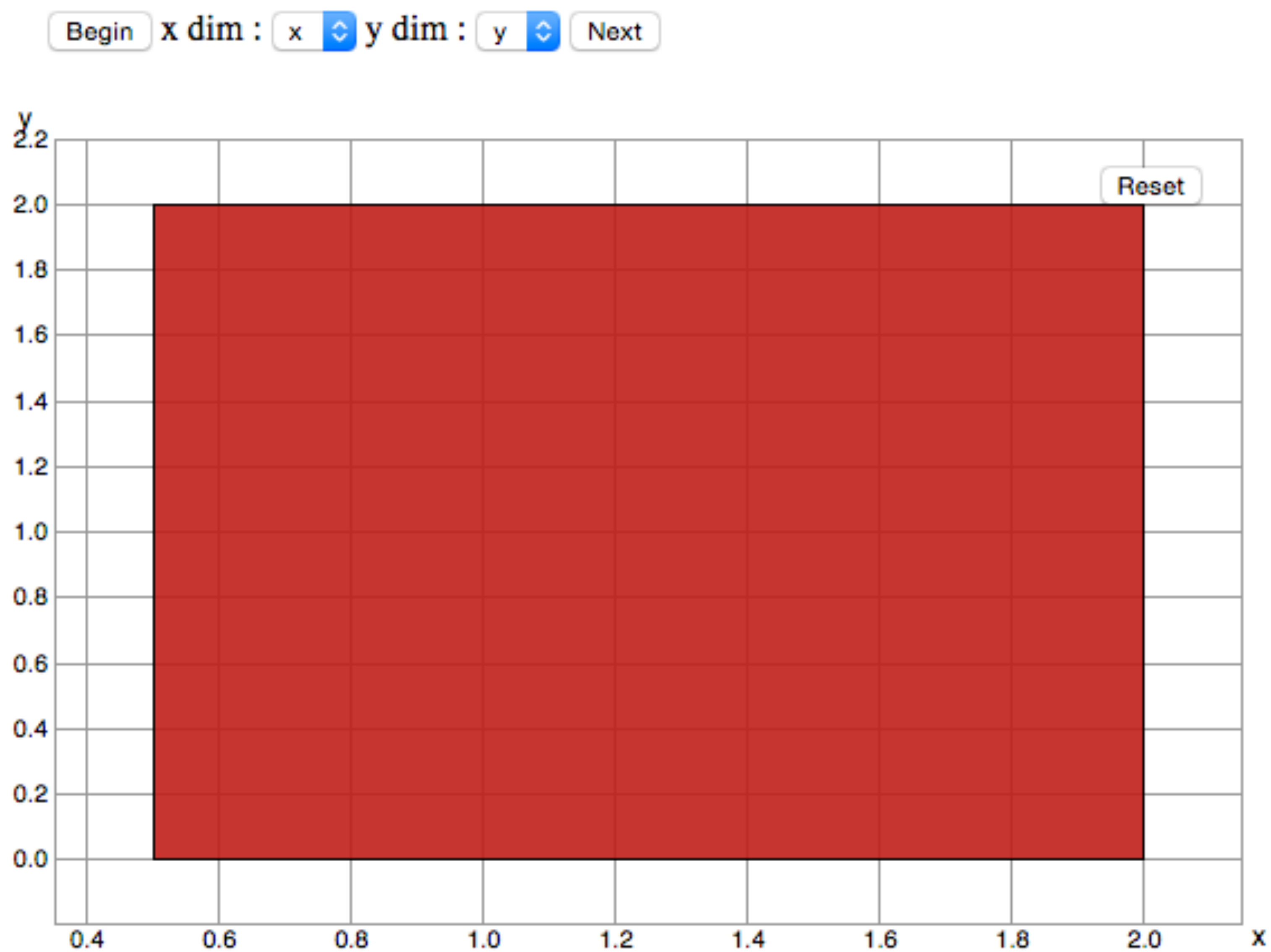
Example of ICP



ANSWER: **SAT**

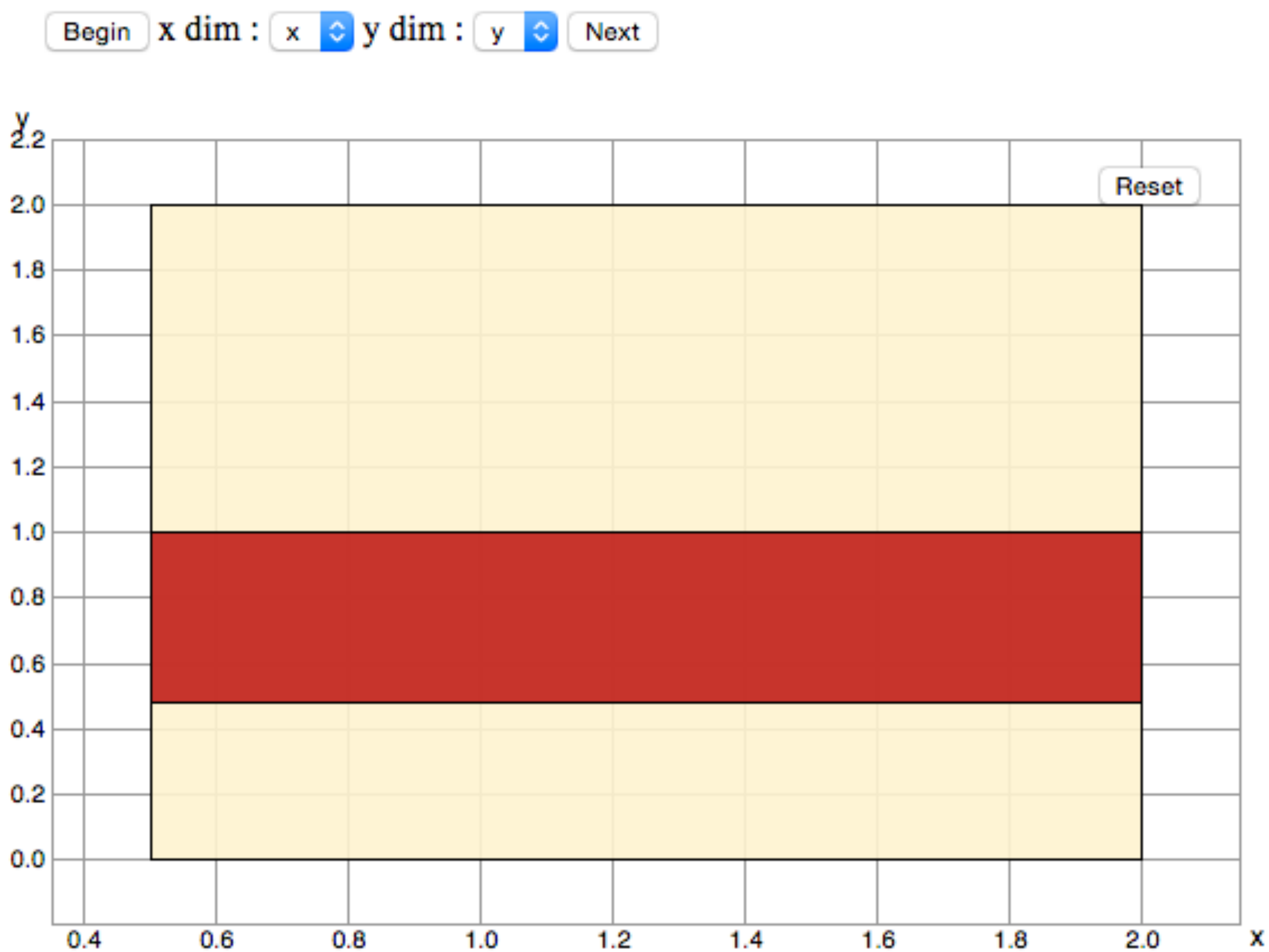
Example of ICP

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



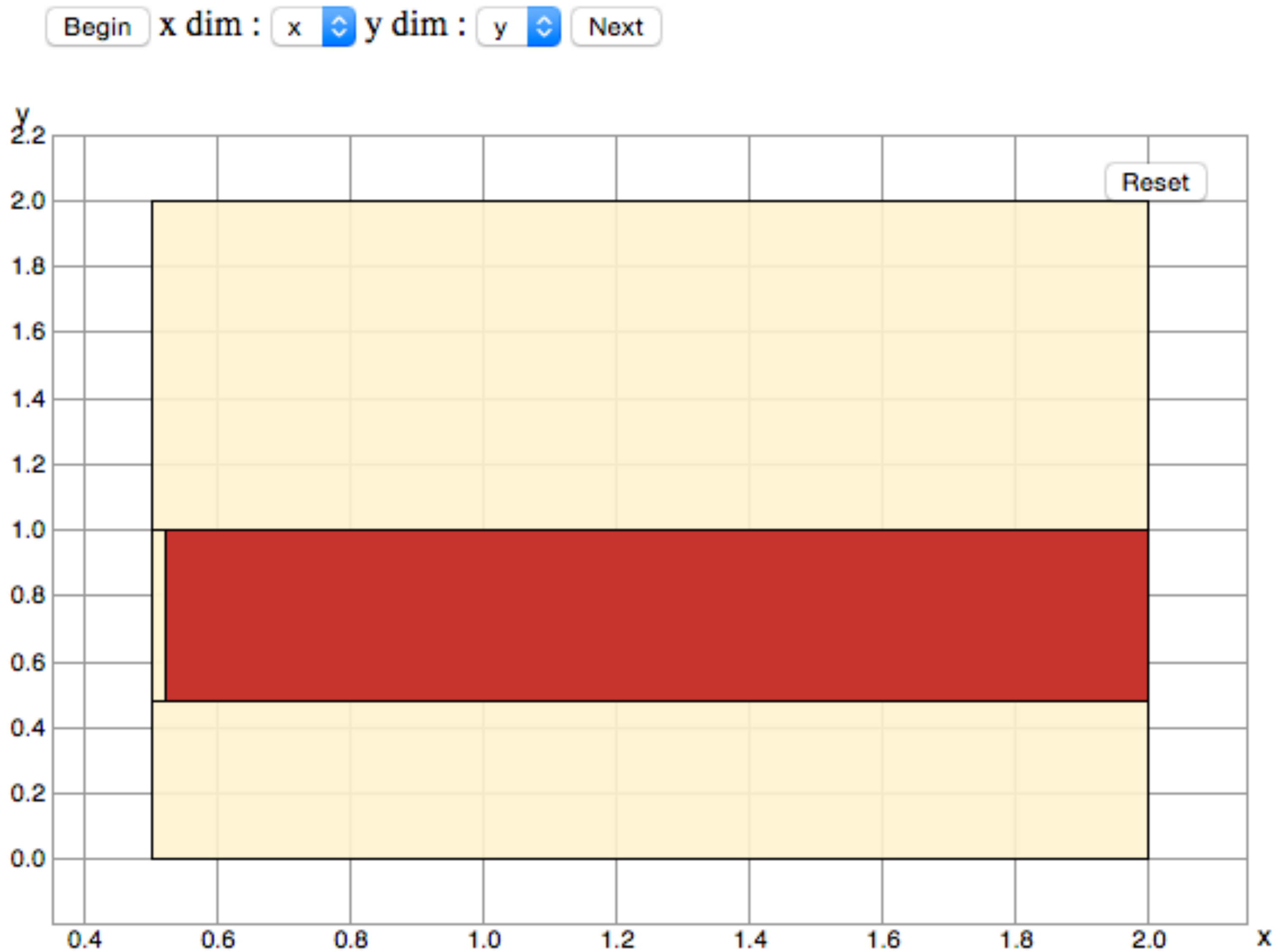
Example of ICP

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



Example of ICP

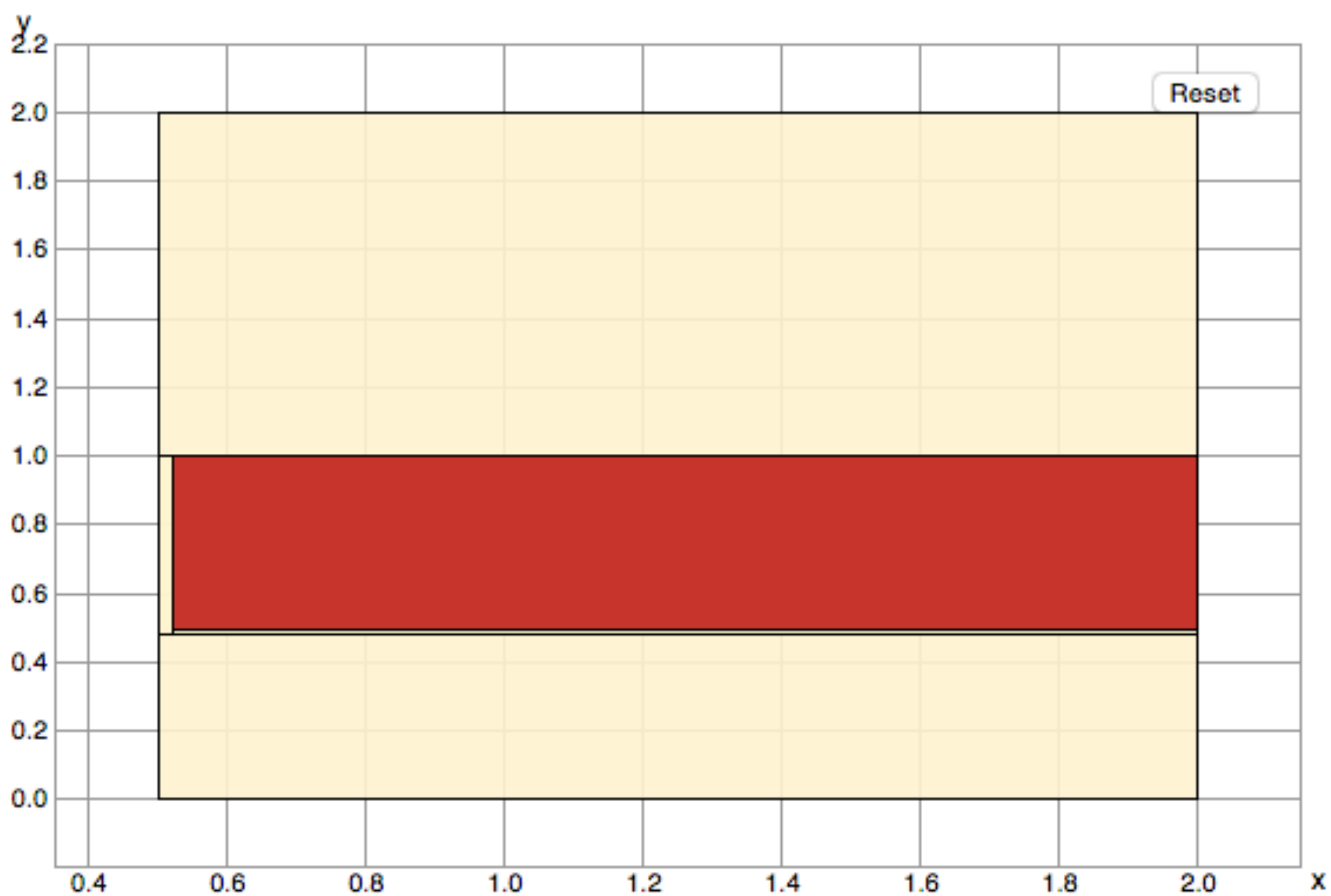
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$



Example of ICP

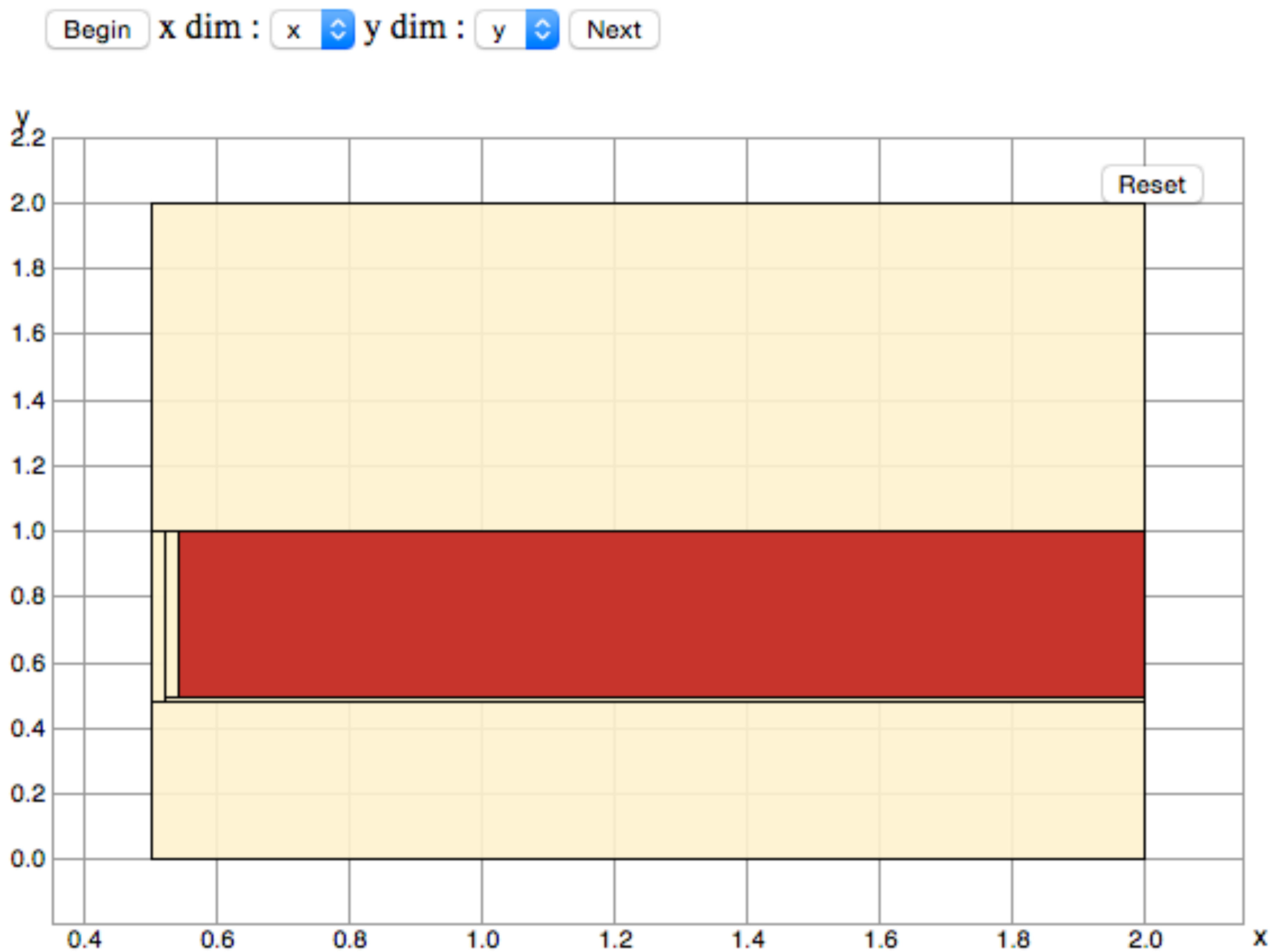
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \mathrm{atan}(x)$$

Begin x dim : x y dim : y Next



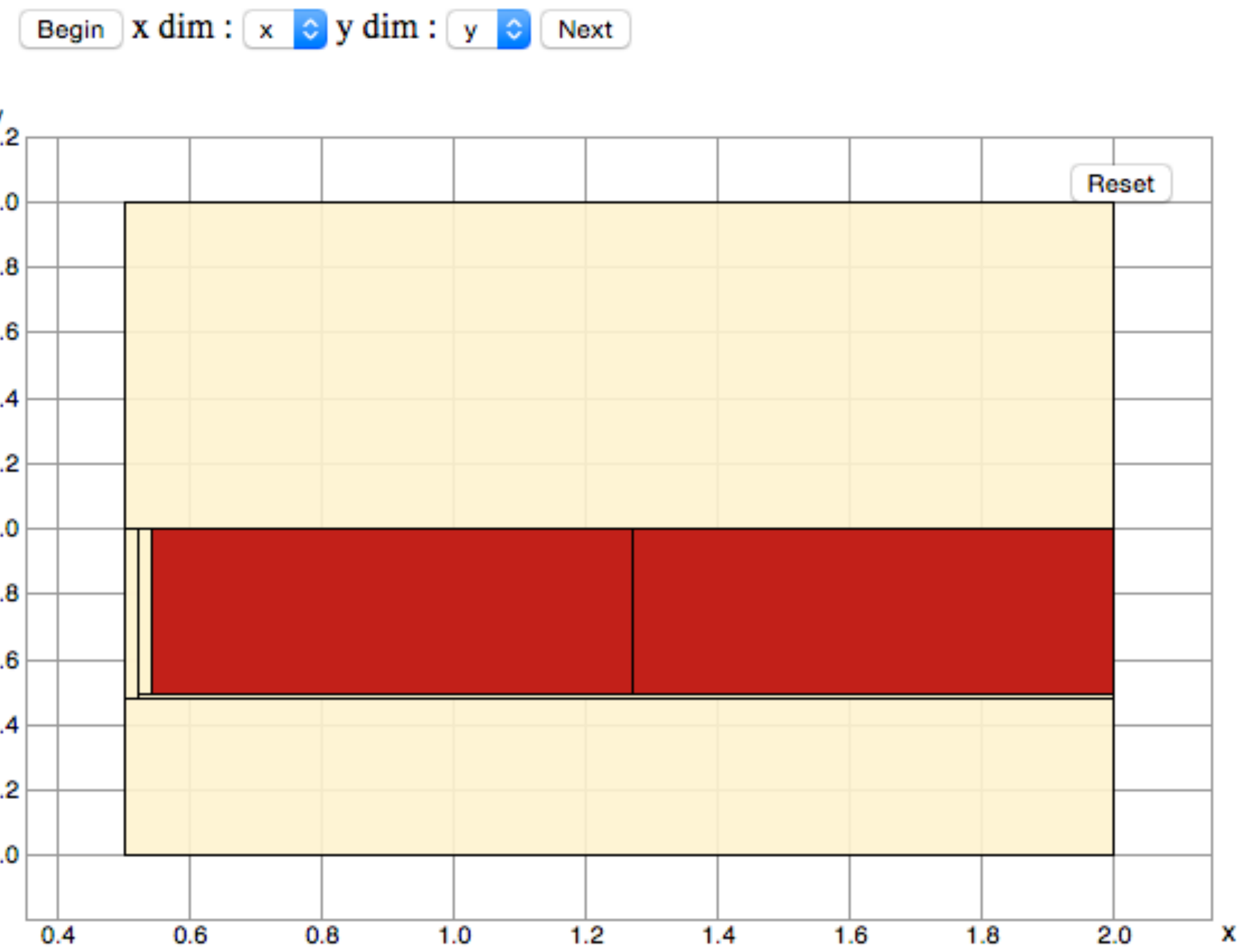
Example of ICP

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



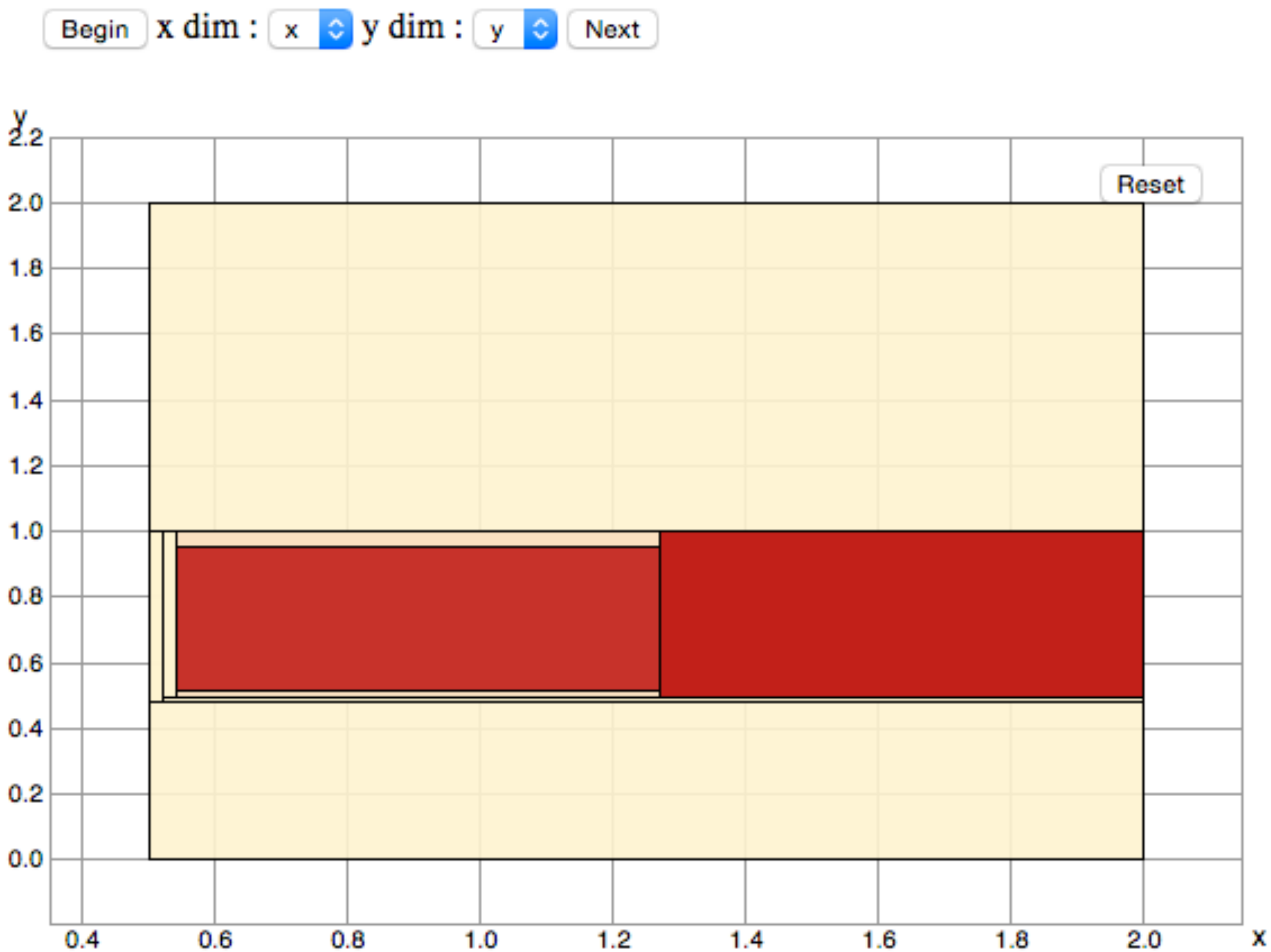
Example of ICP

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



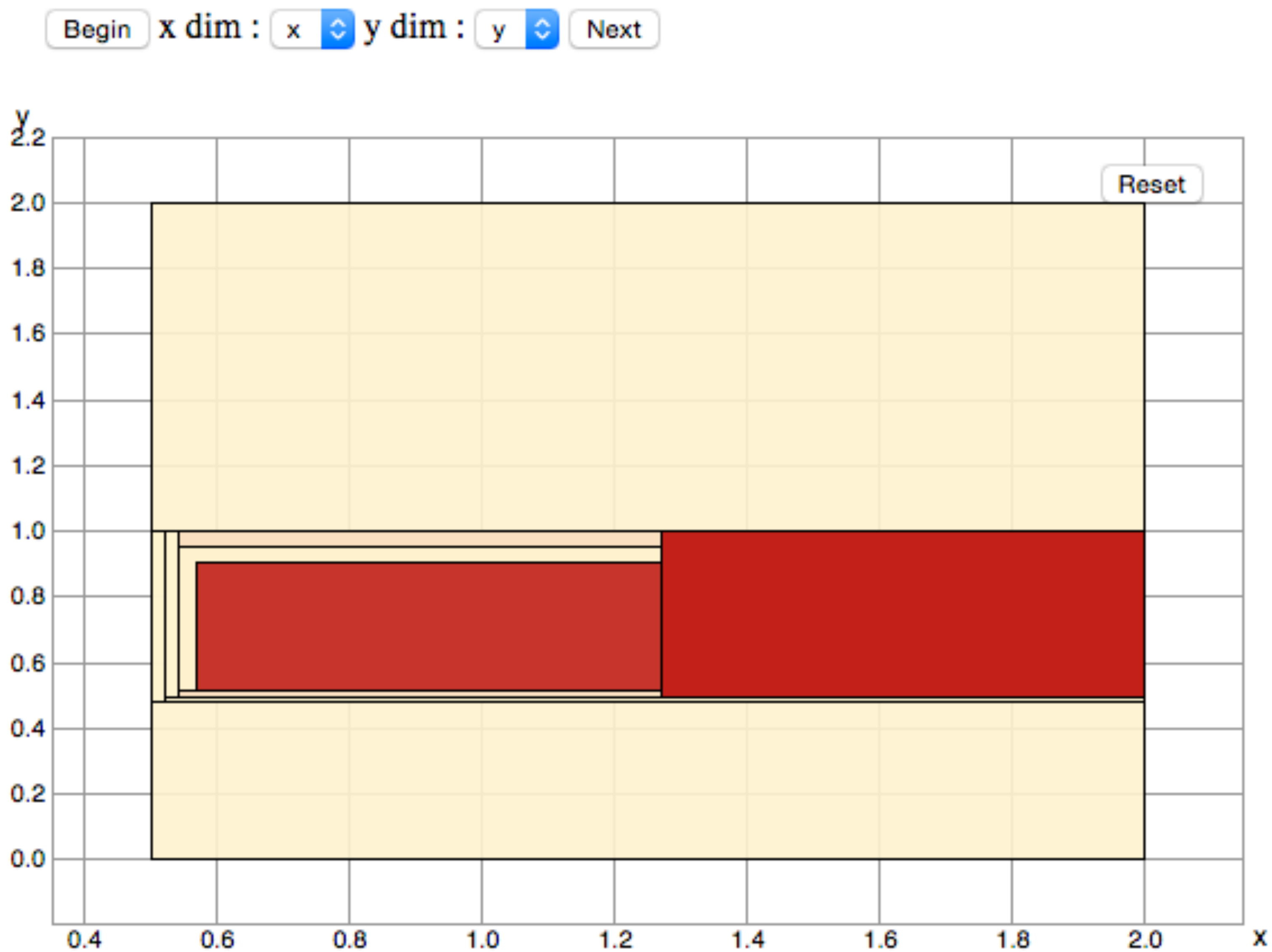
Example of ICP

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



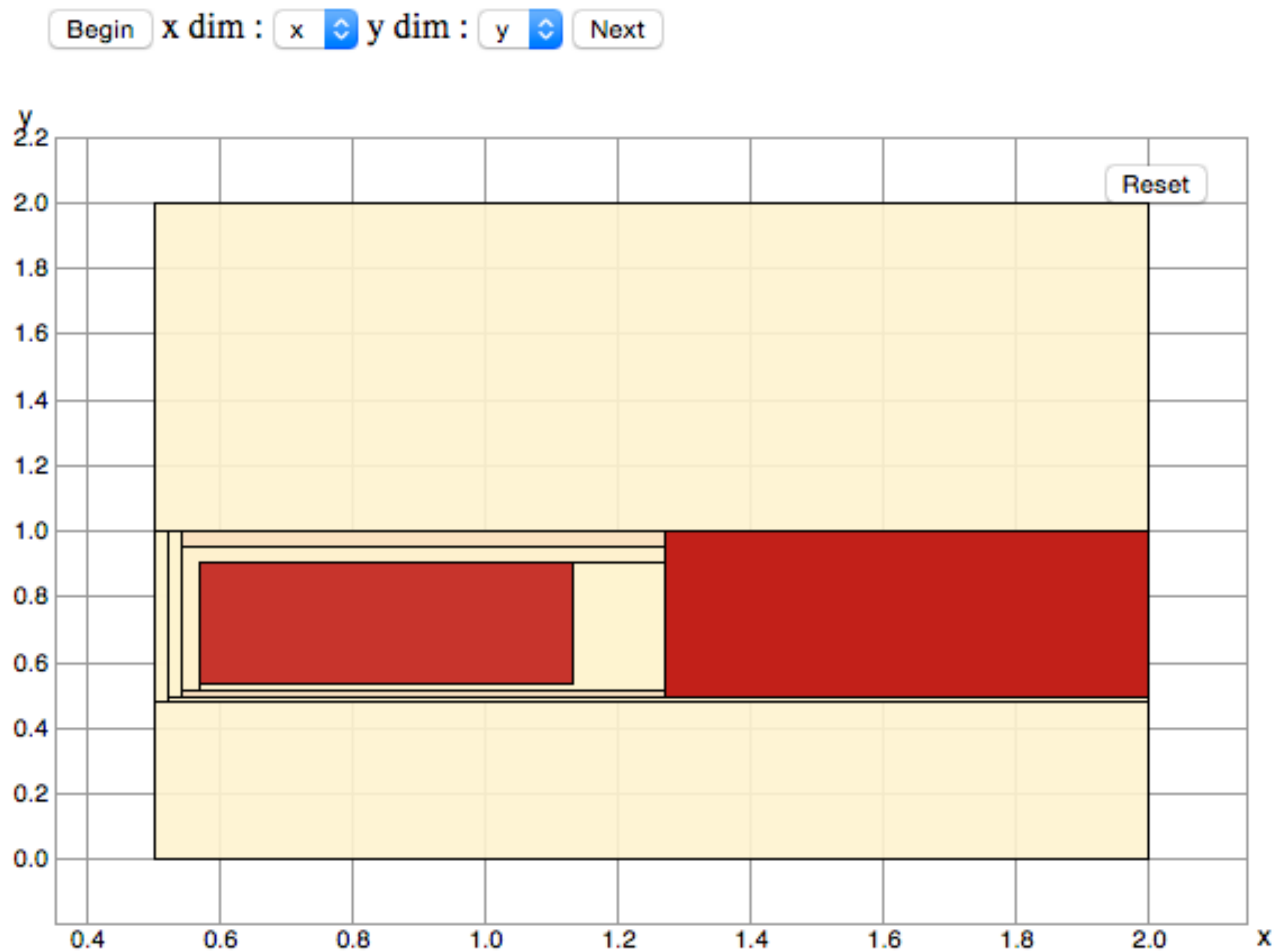
Example of ICP

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



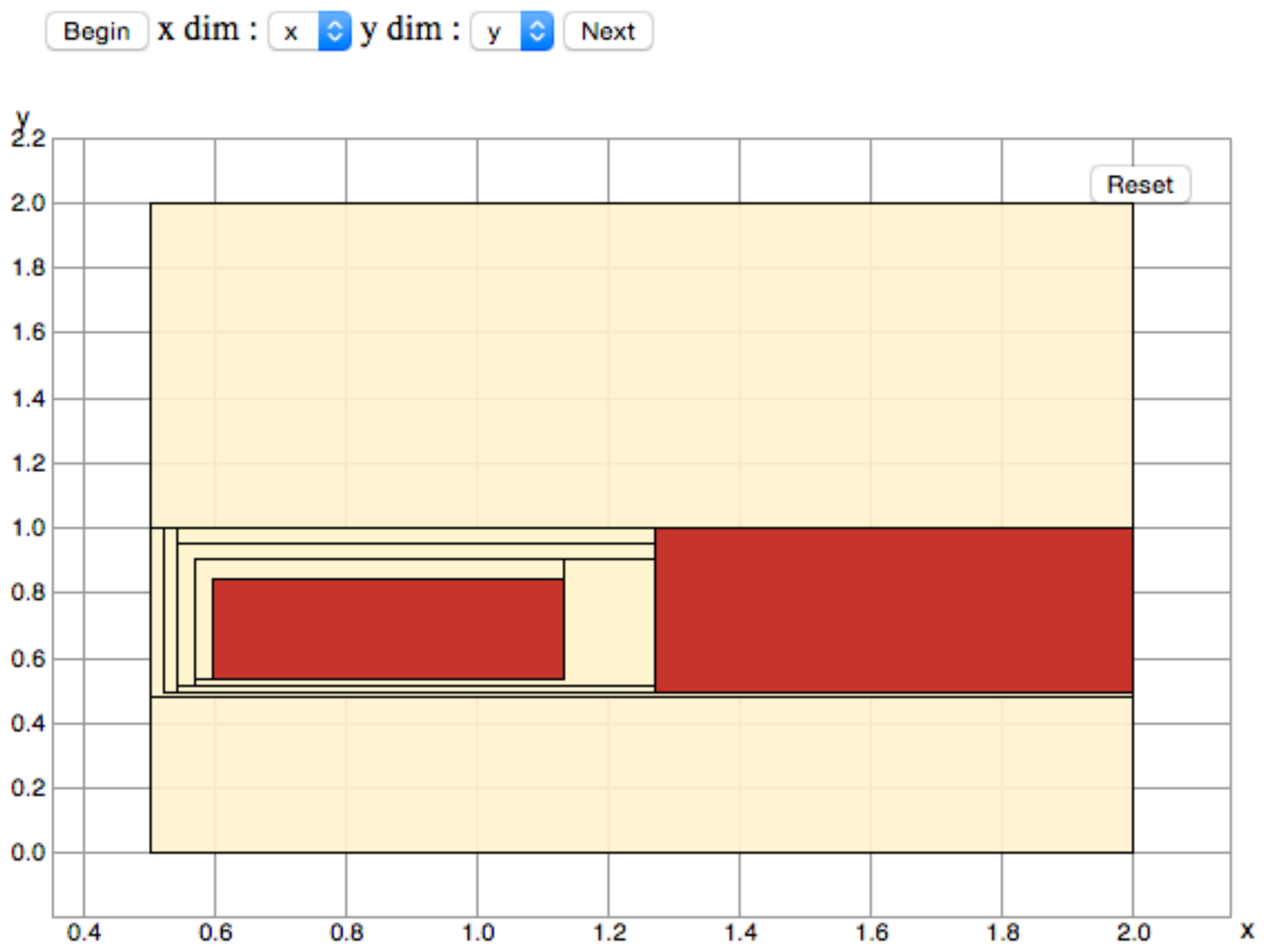
Example of ICP

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



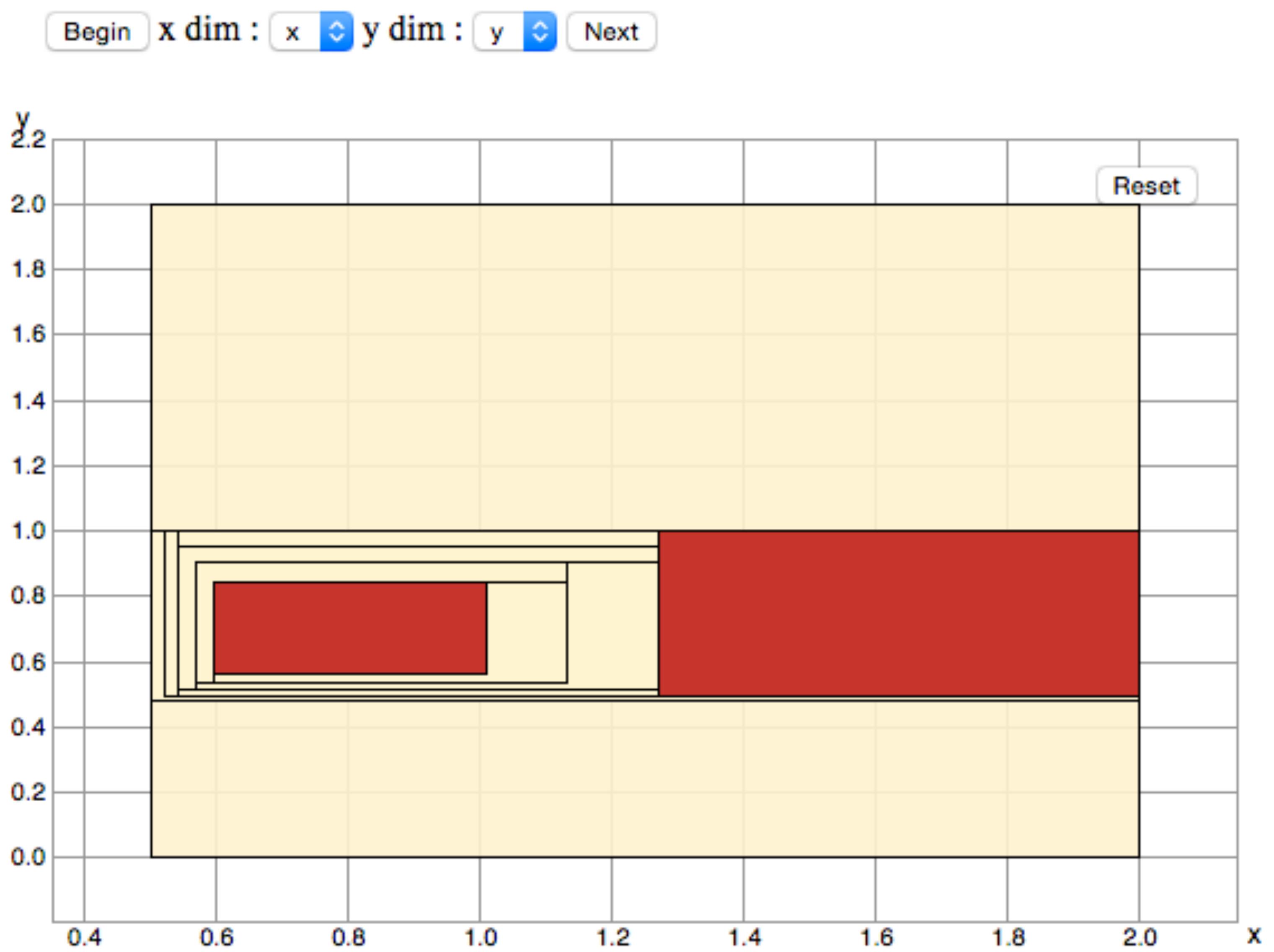
Example of ICP

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



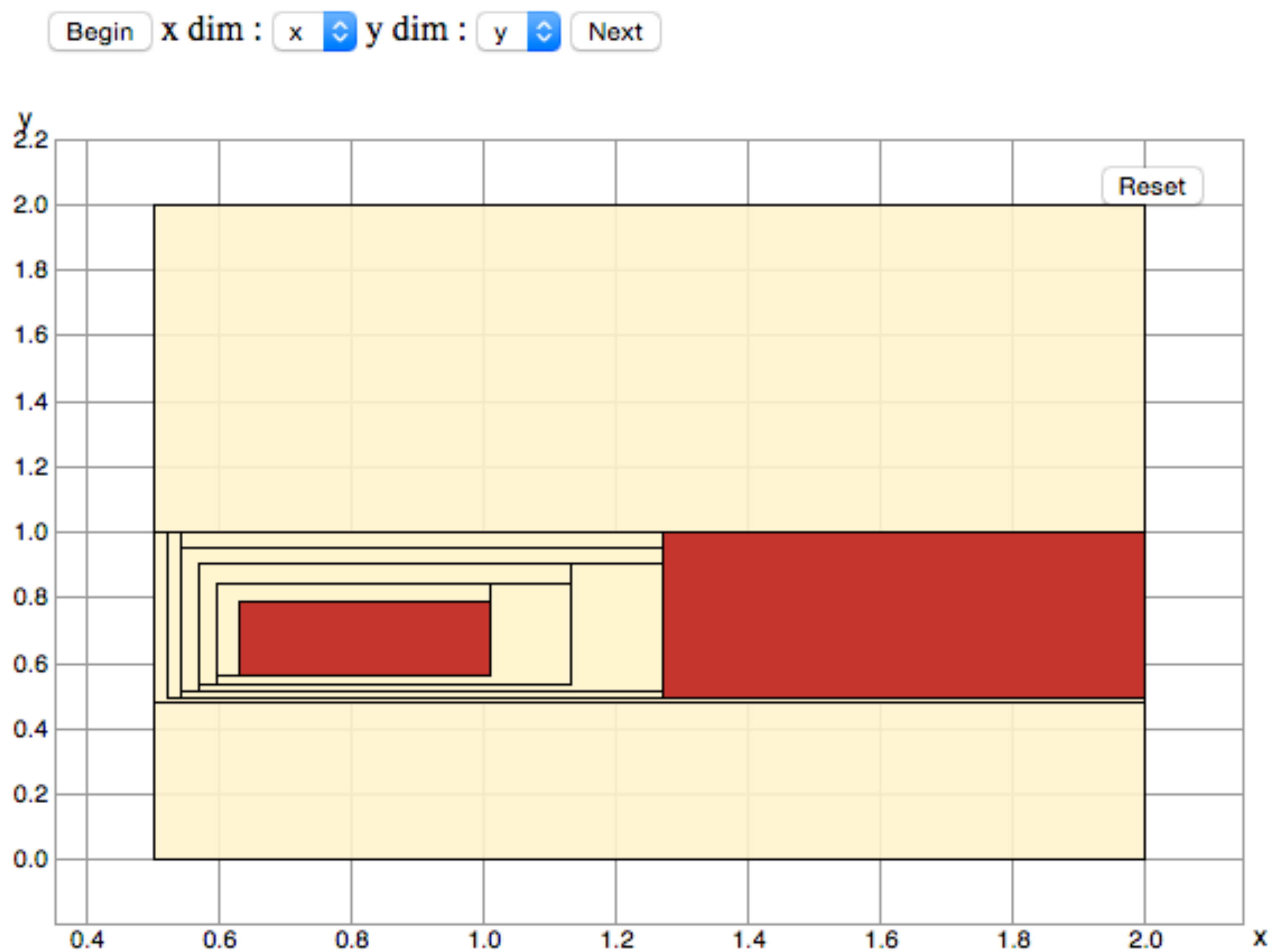
Example of ICP

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



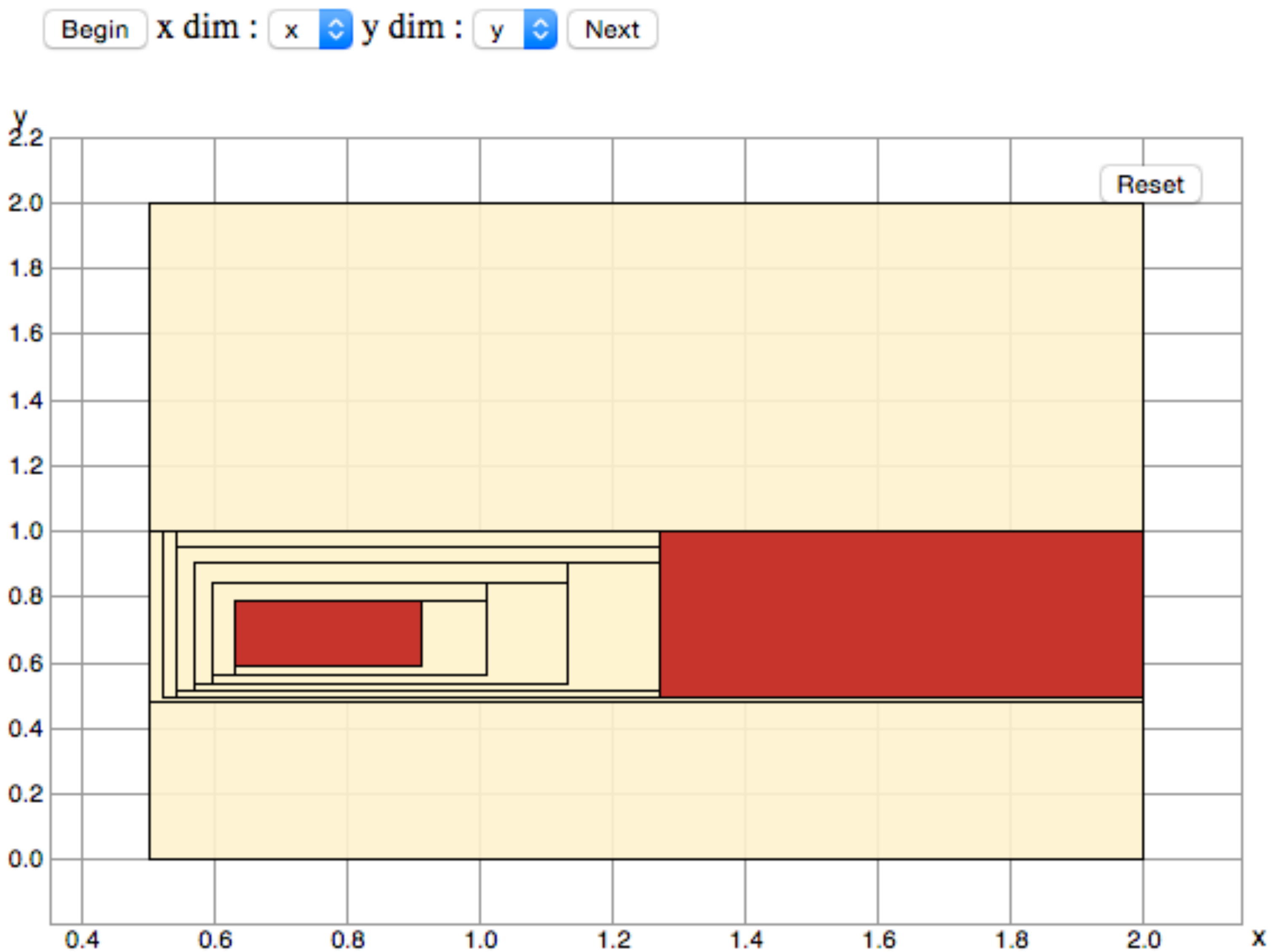
Example of ICP

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



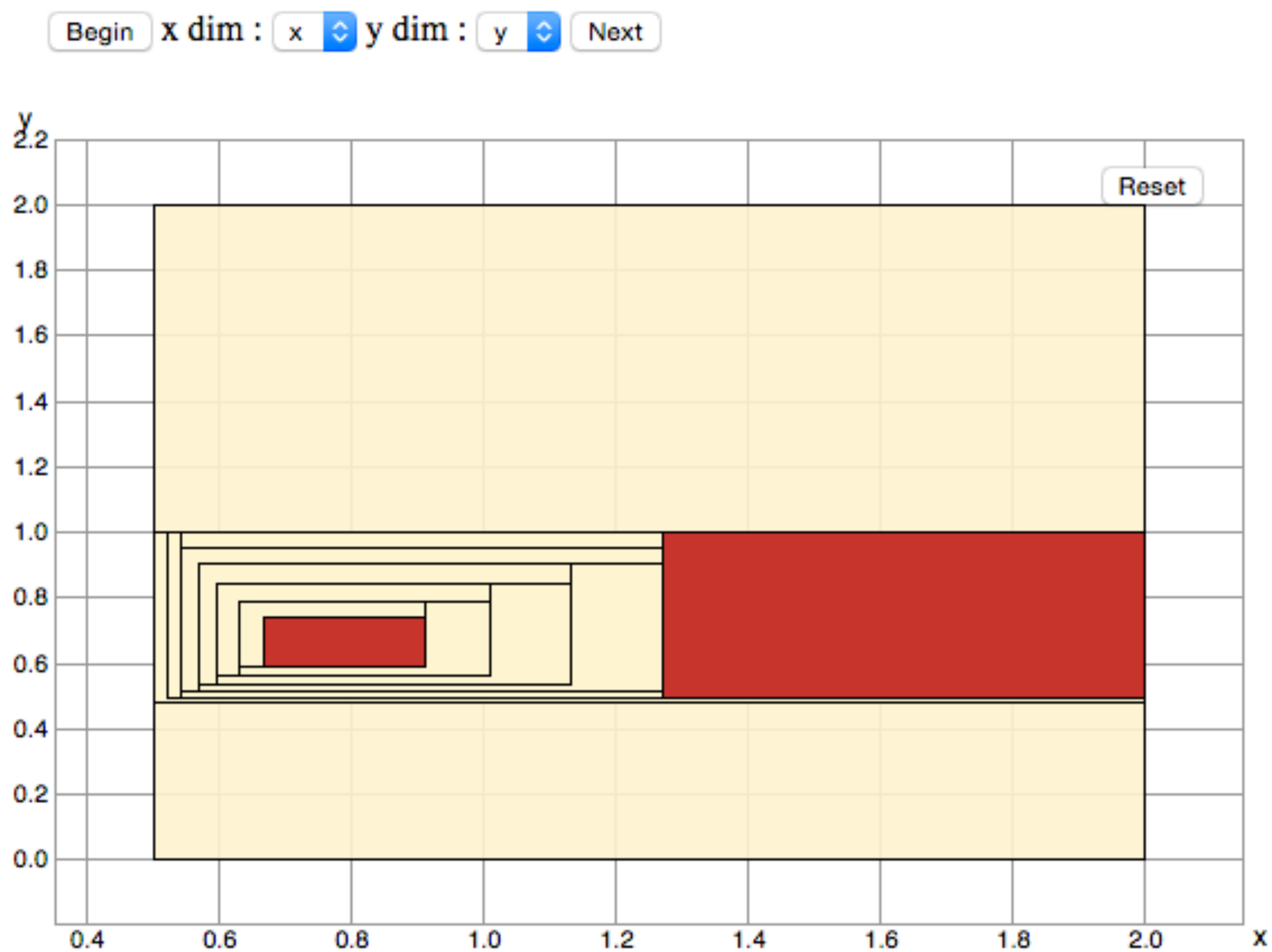
Example of ICP

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



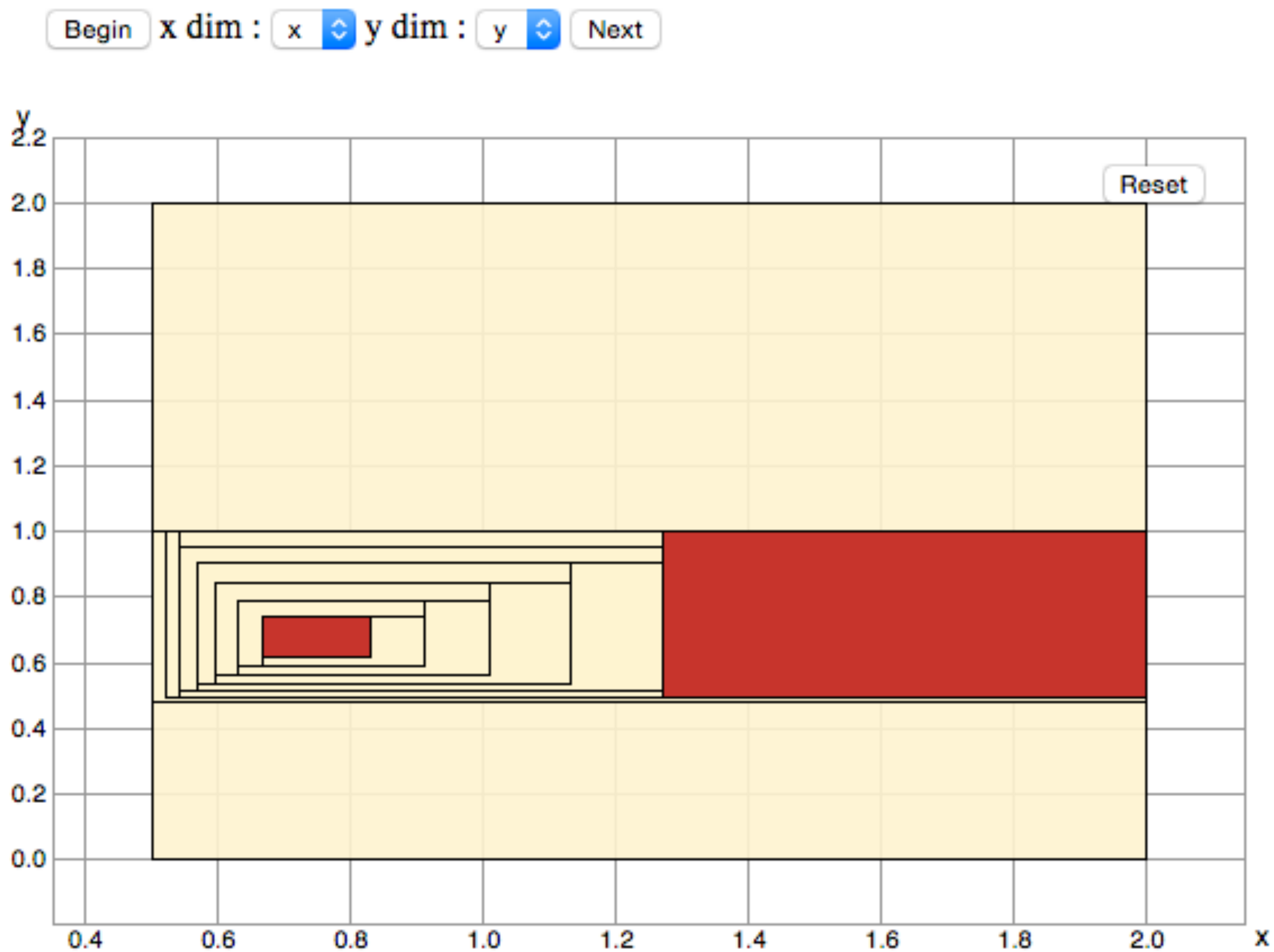
Example of ICP

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



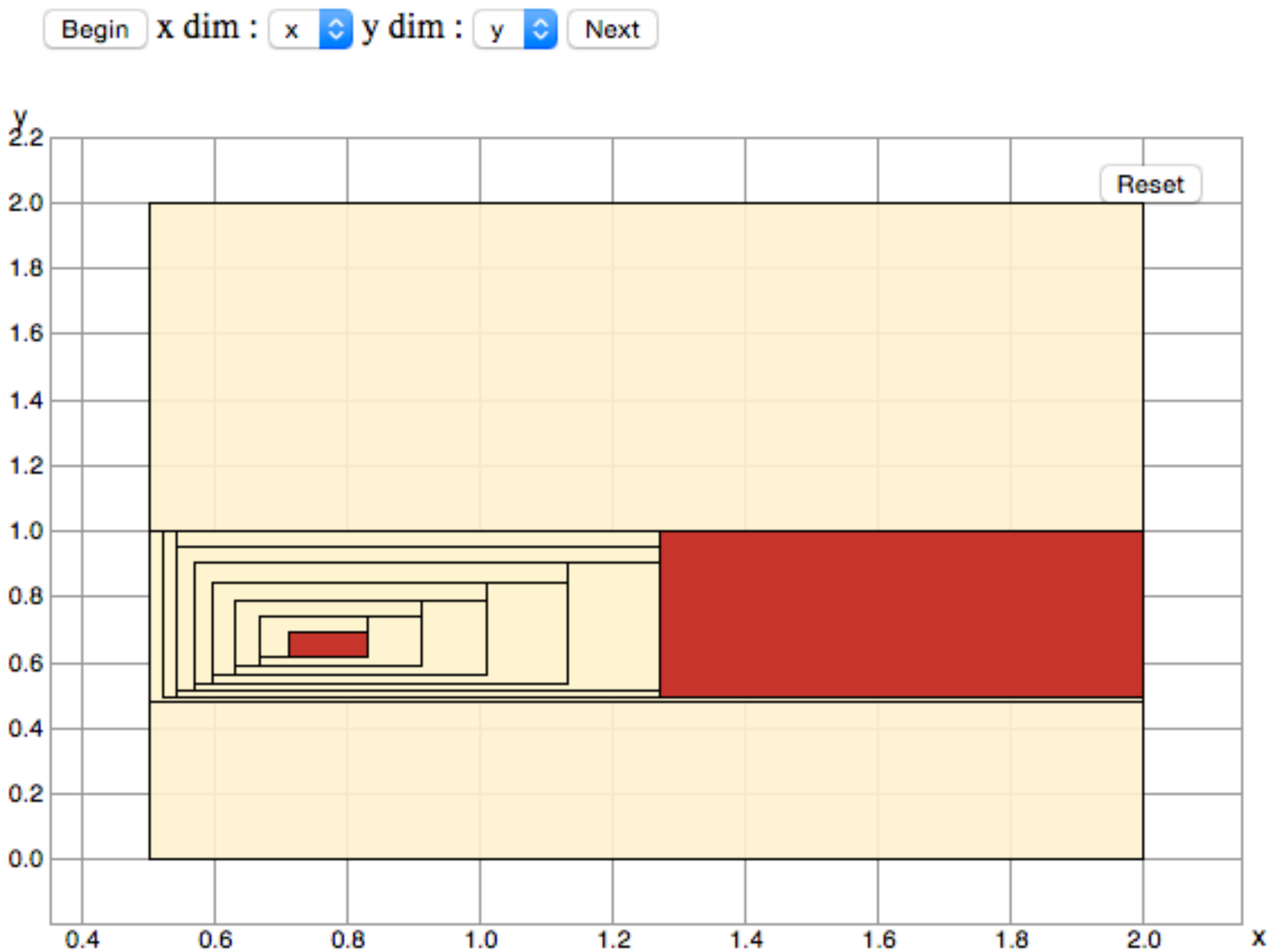
Example of ICP

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



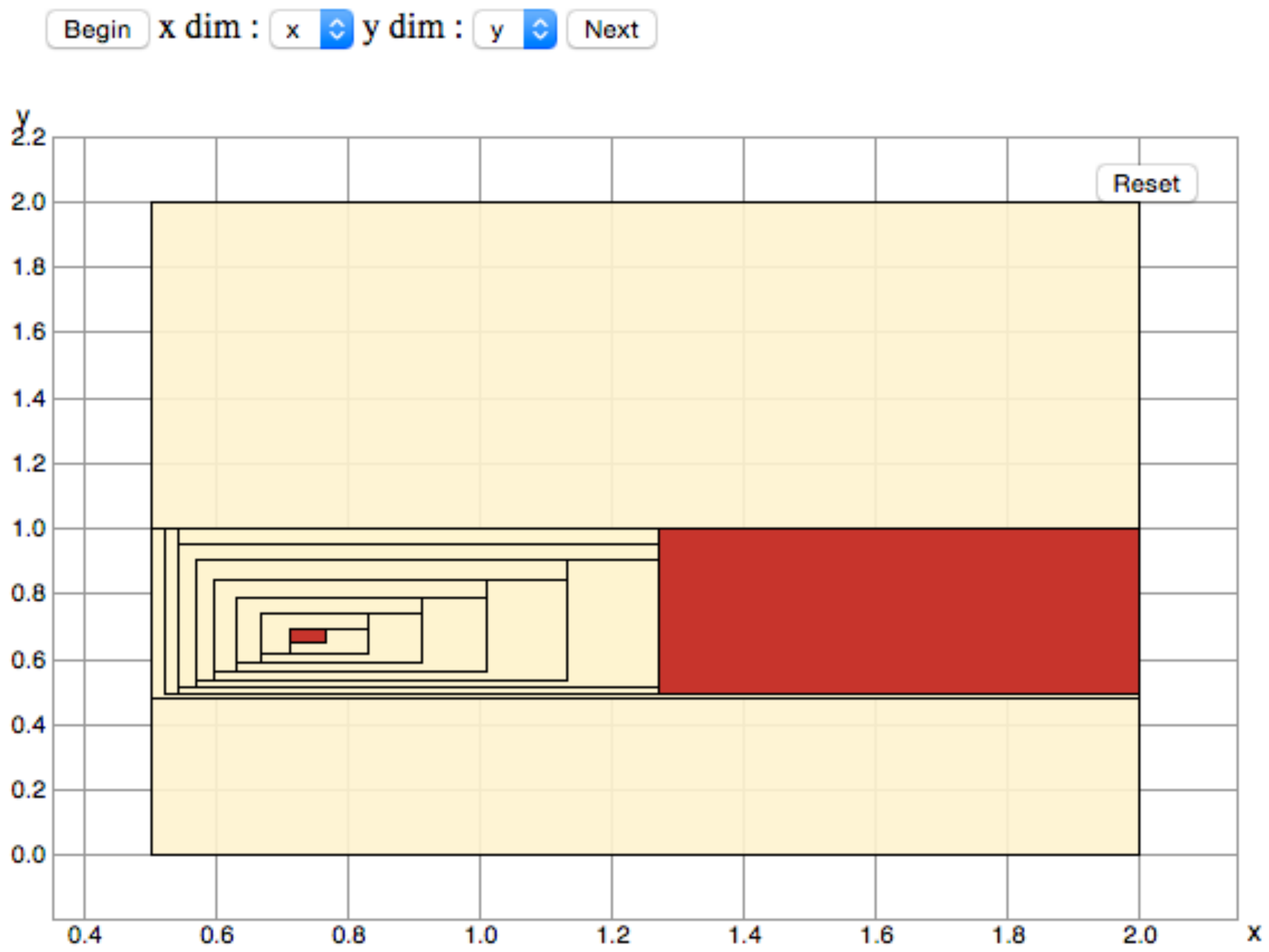
Example of ICP

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



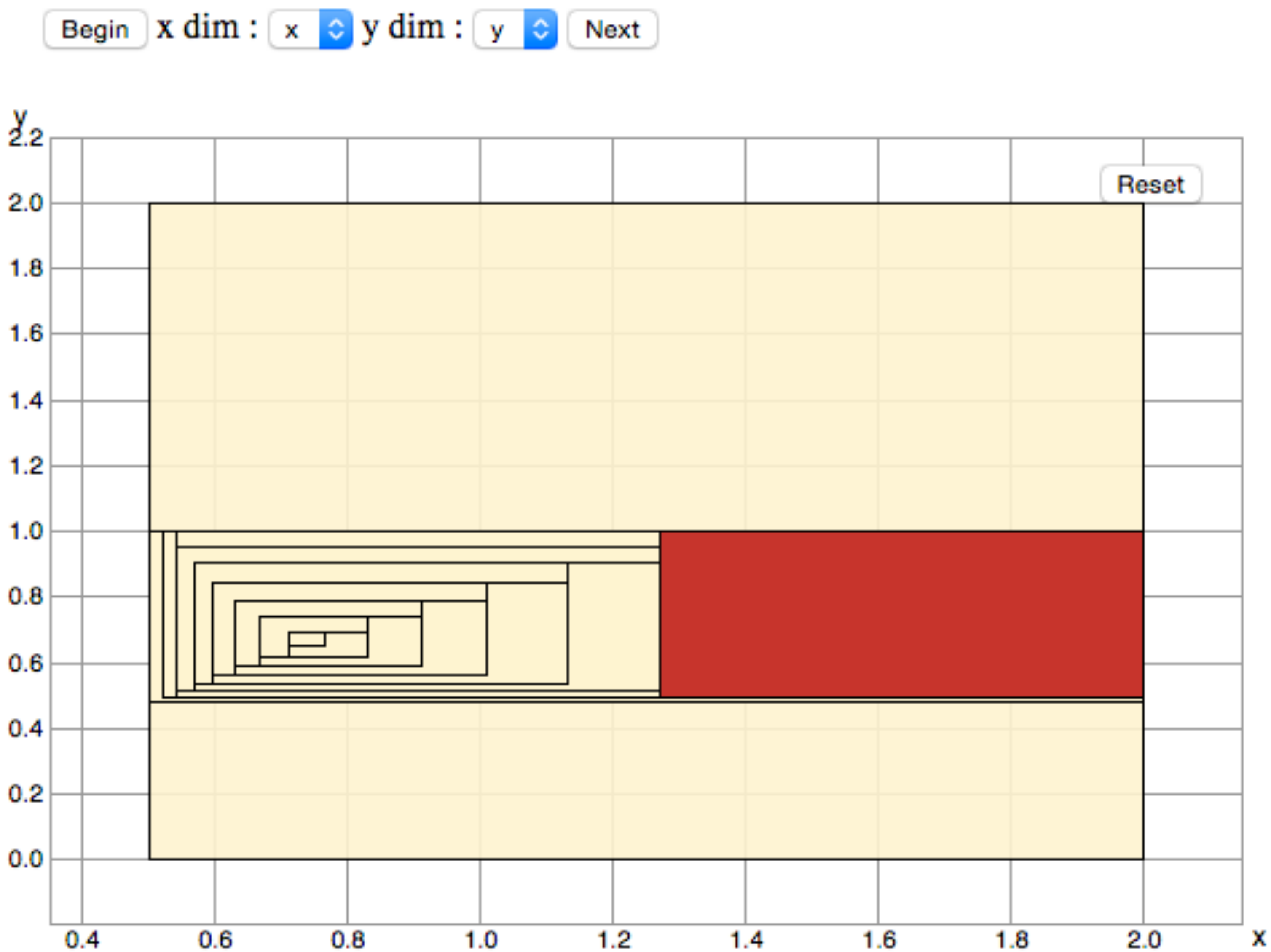
Example of ICP

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



Example of ICP

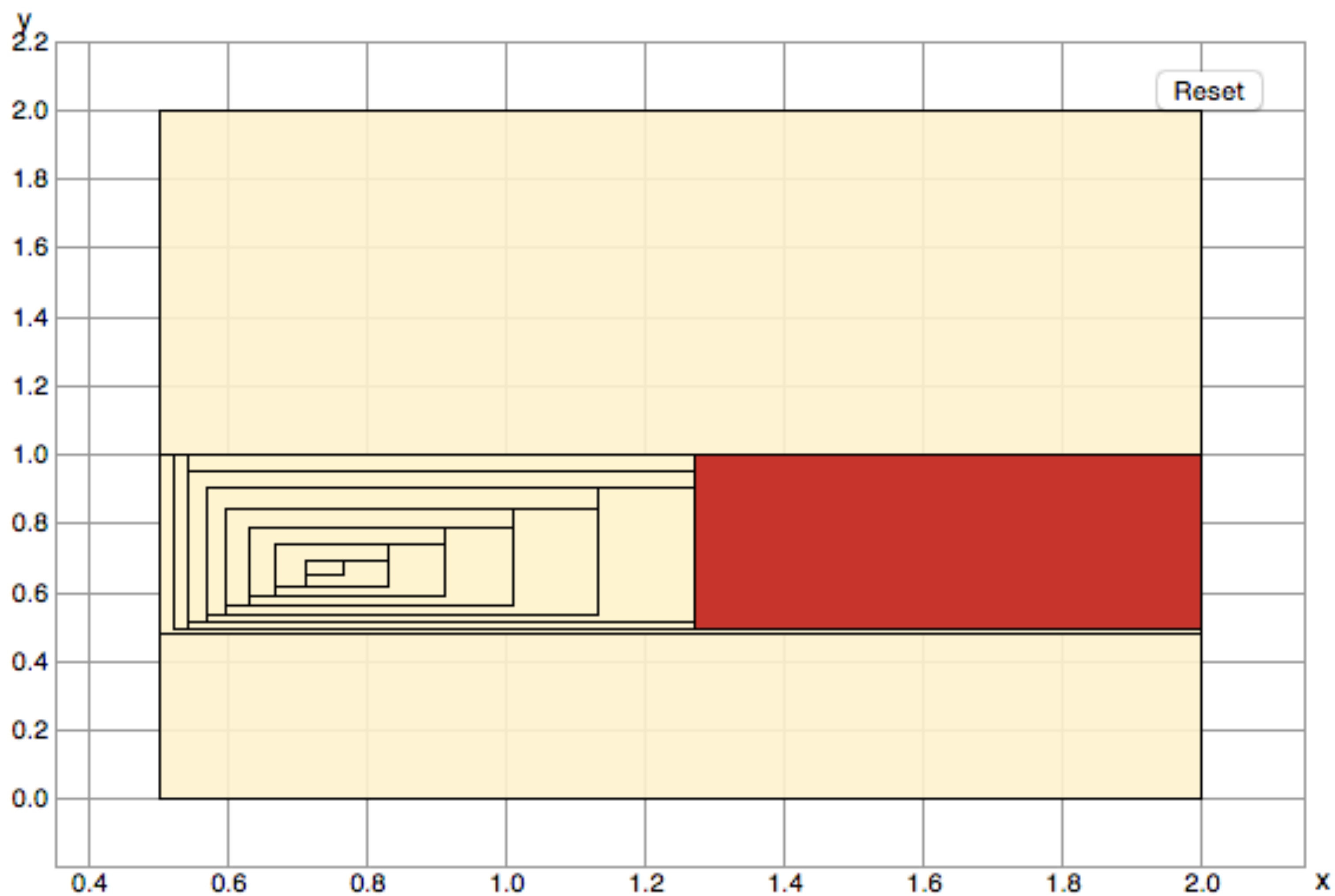
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \operatorname{atan}(x)$$



Example of ICP

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

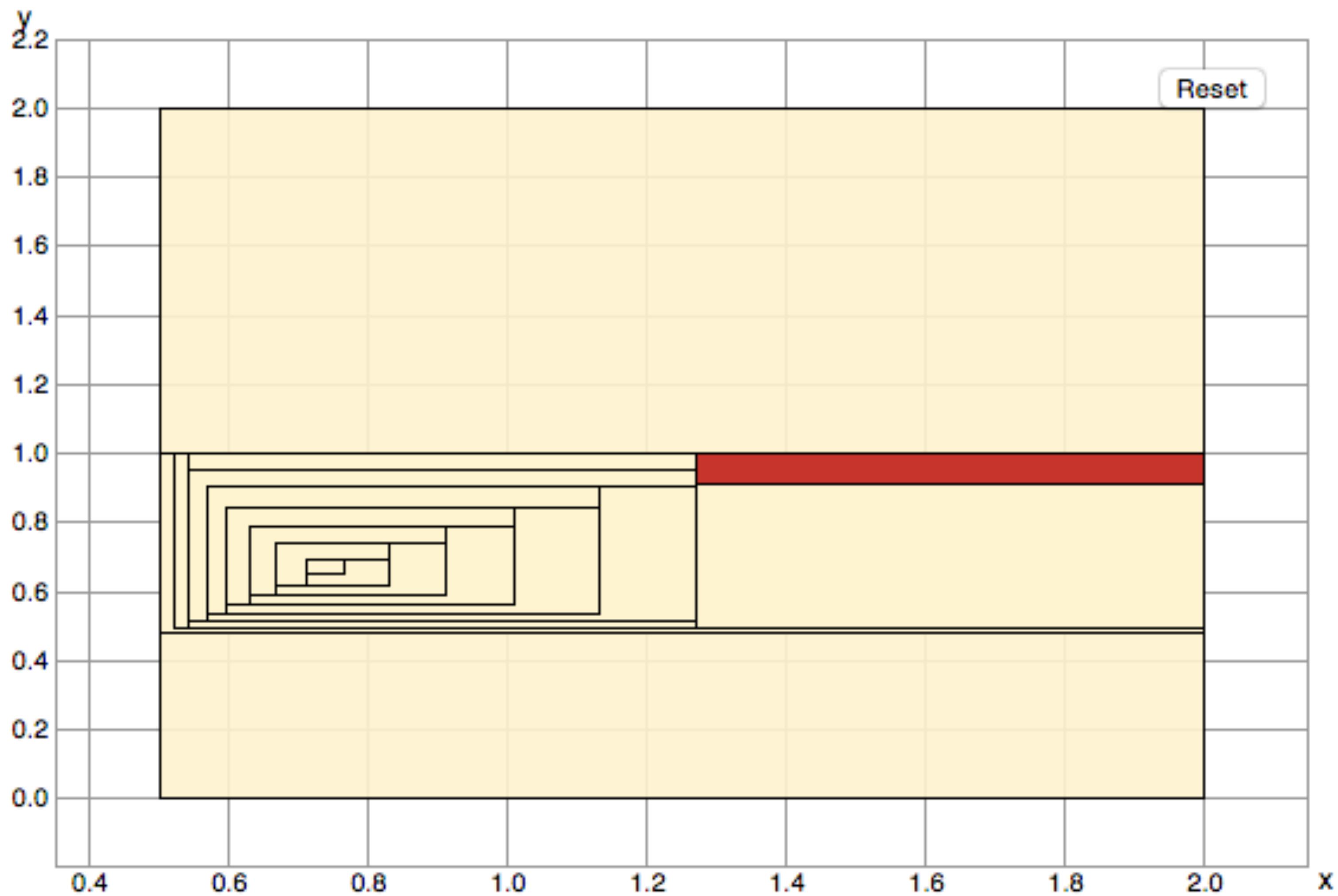
Begin x dim : x y dim : y Next



Example of ICP

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

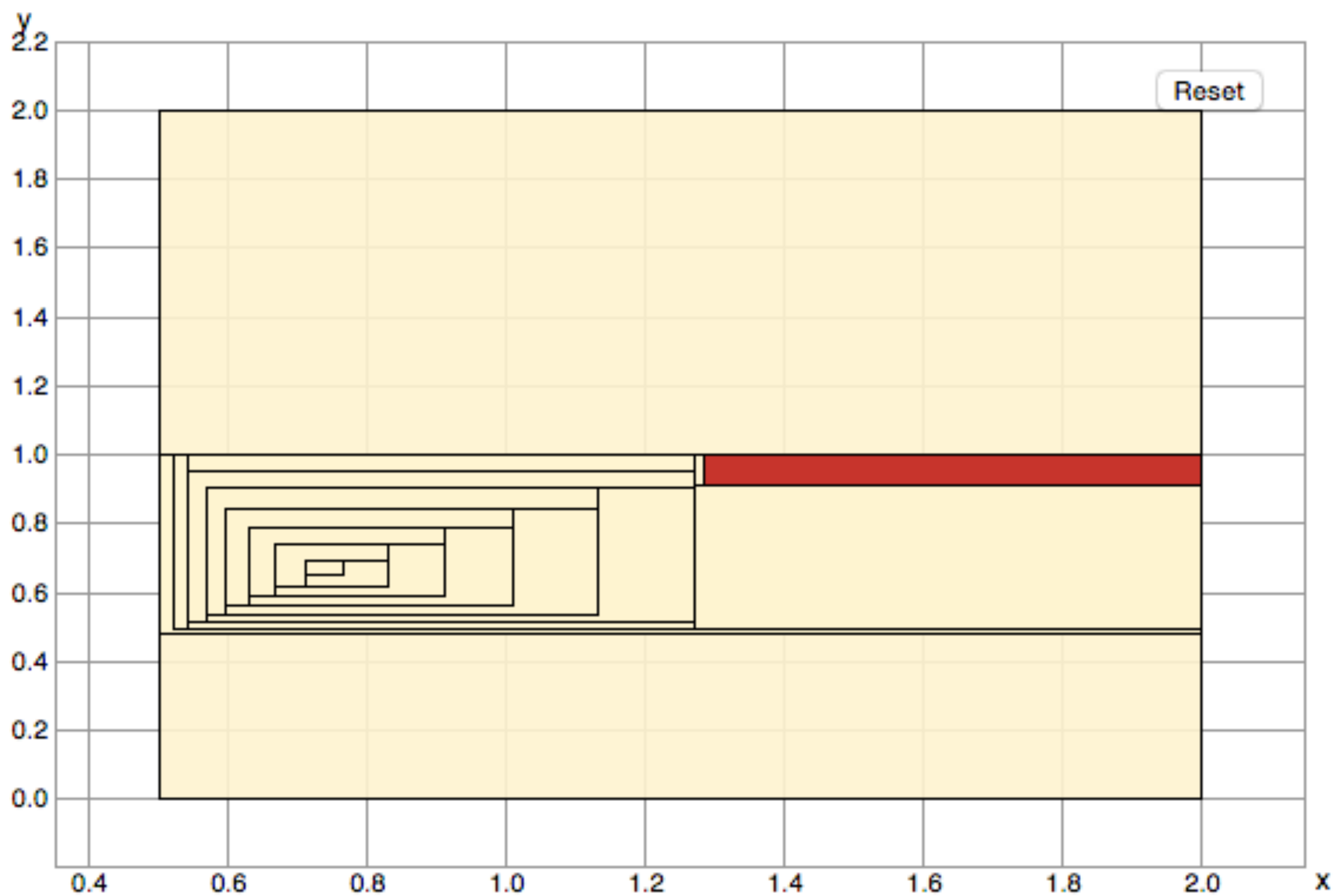
Begin x dim : x y dim : y Next



Example of ICP

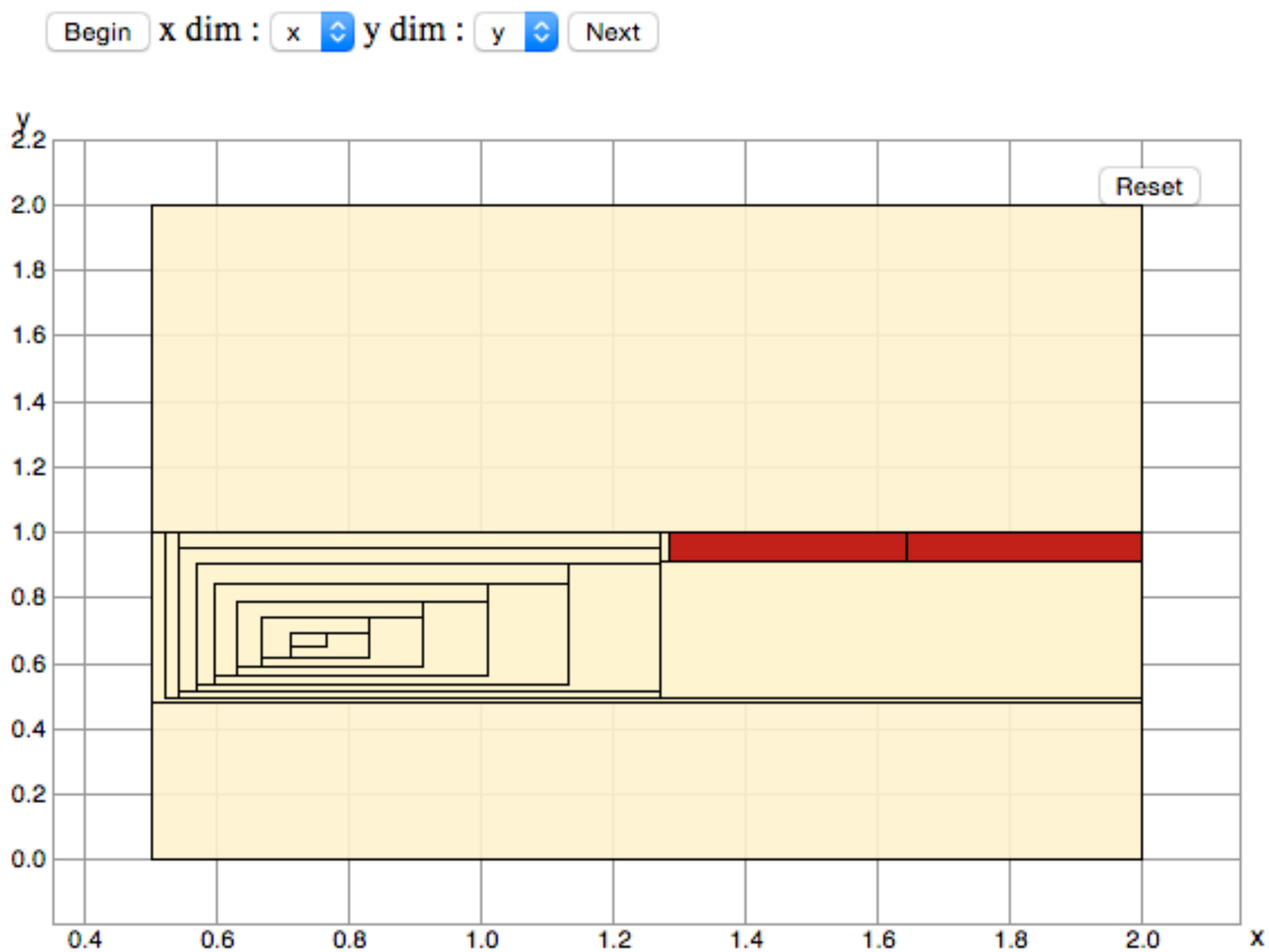
$$\exists x \in [0.5, 2.0], y \in [0.0, 2.0] : y = \sin(x) \wedge y = \text{atan}(x)$$

Begin x dim : x y dim : y Next



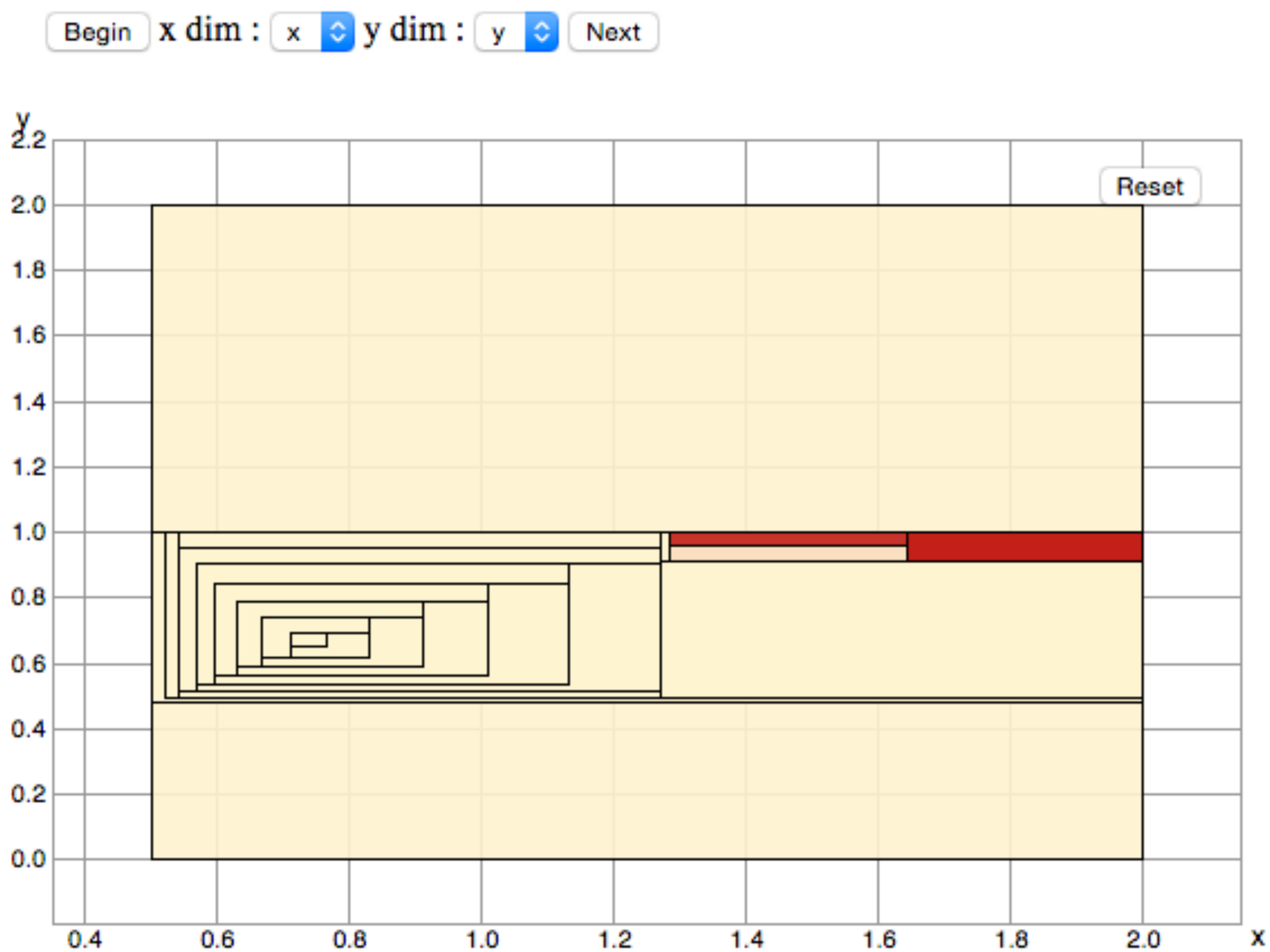
Example of ICP

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



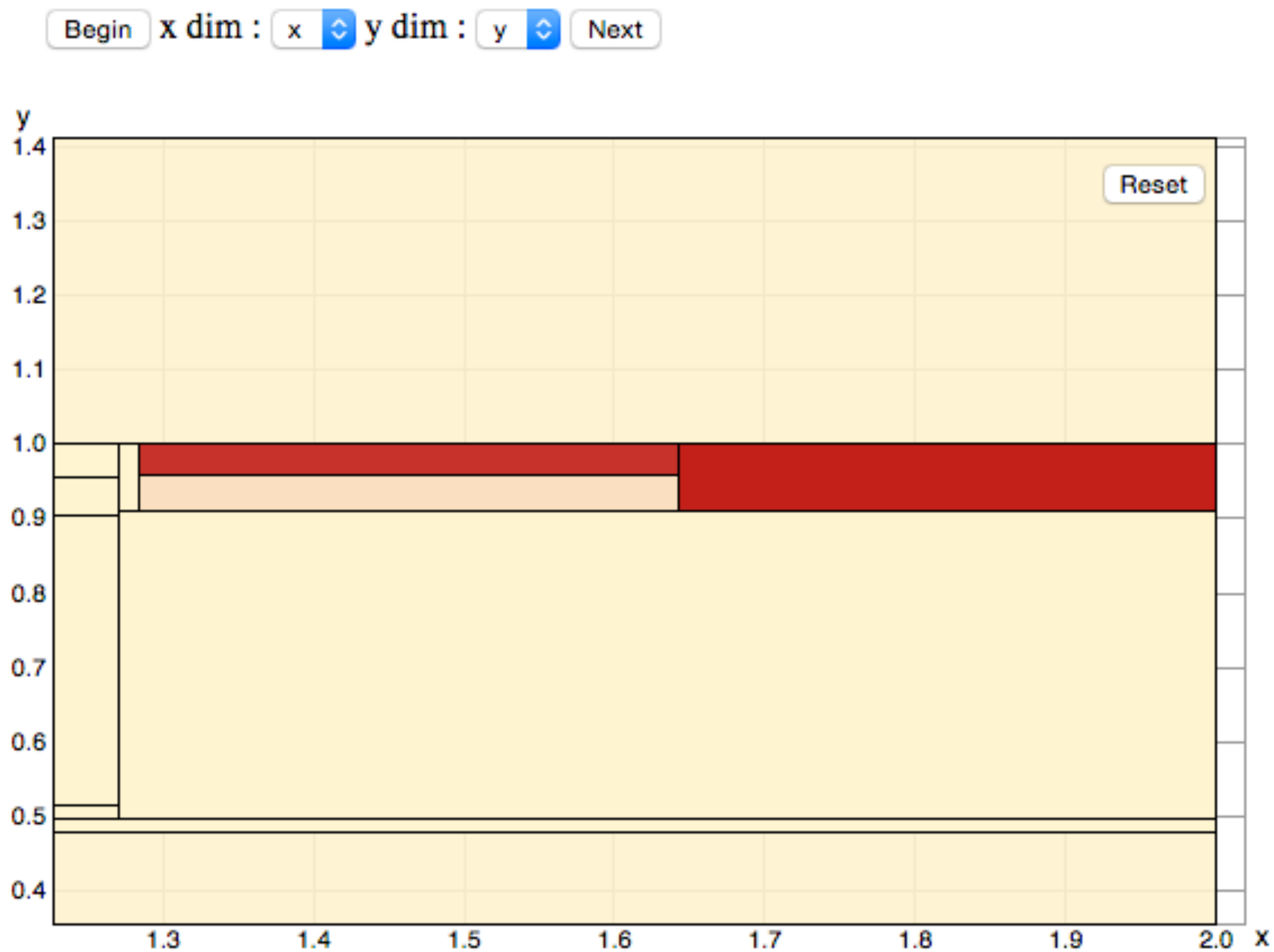
Example of ICP

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



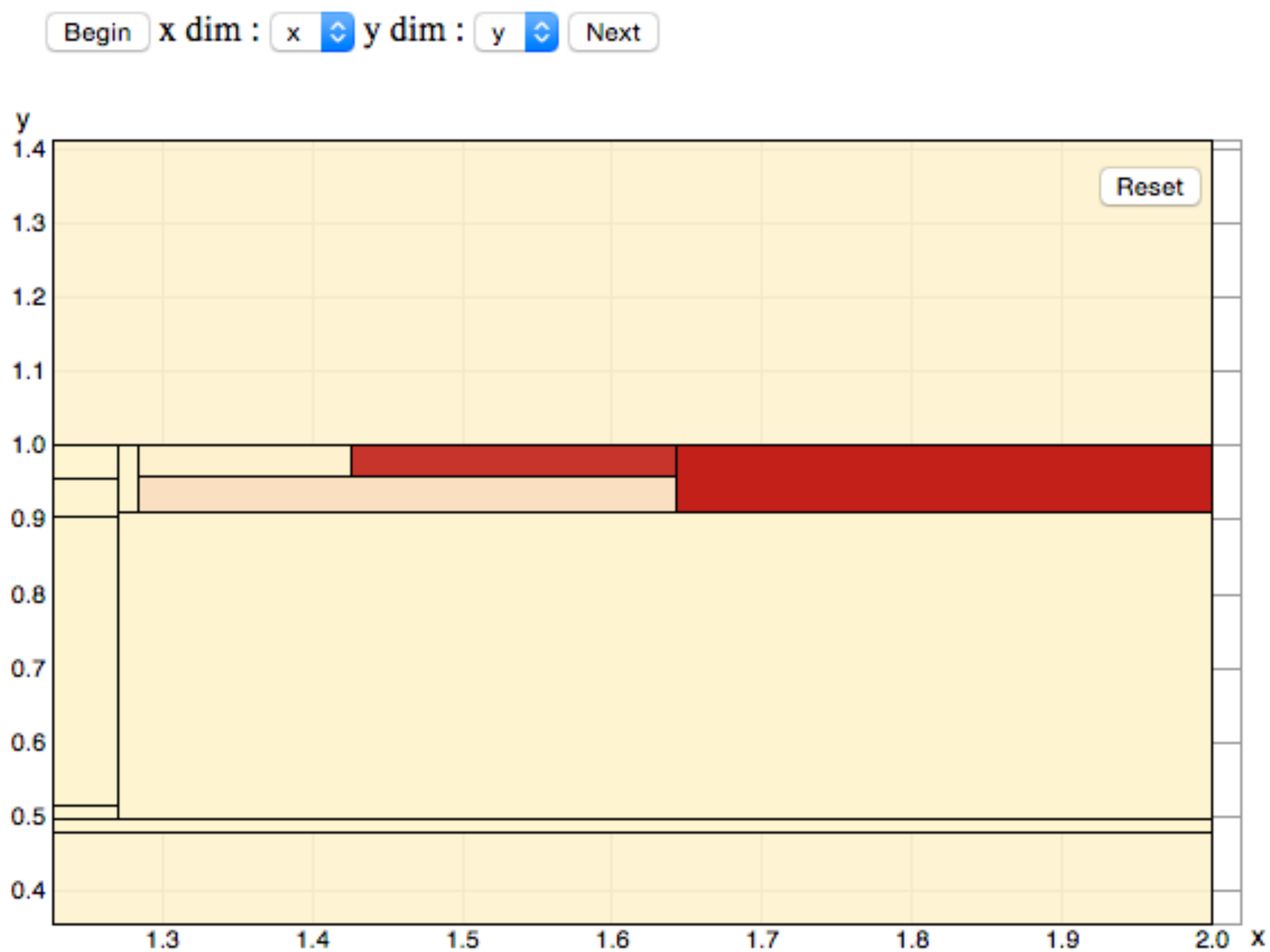
Example of ICP

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



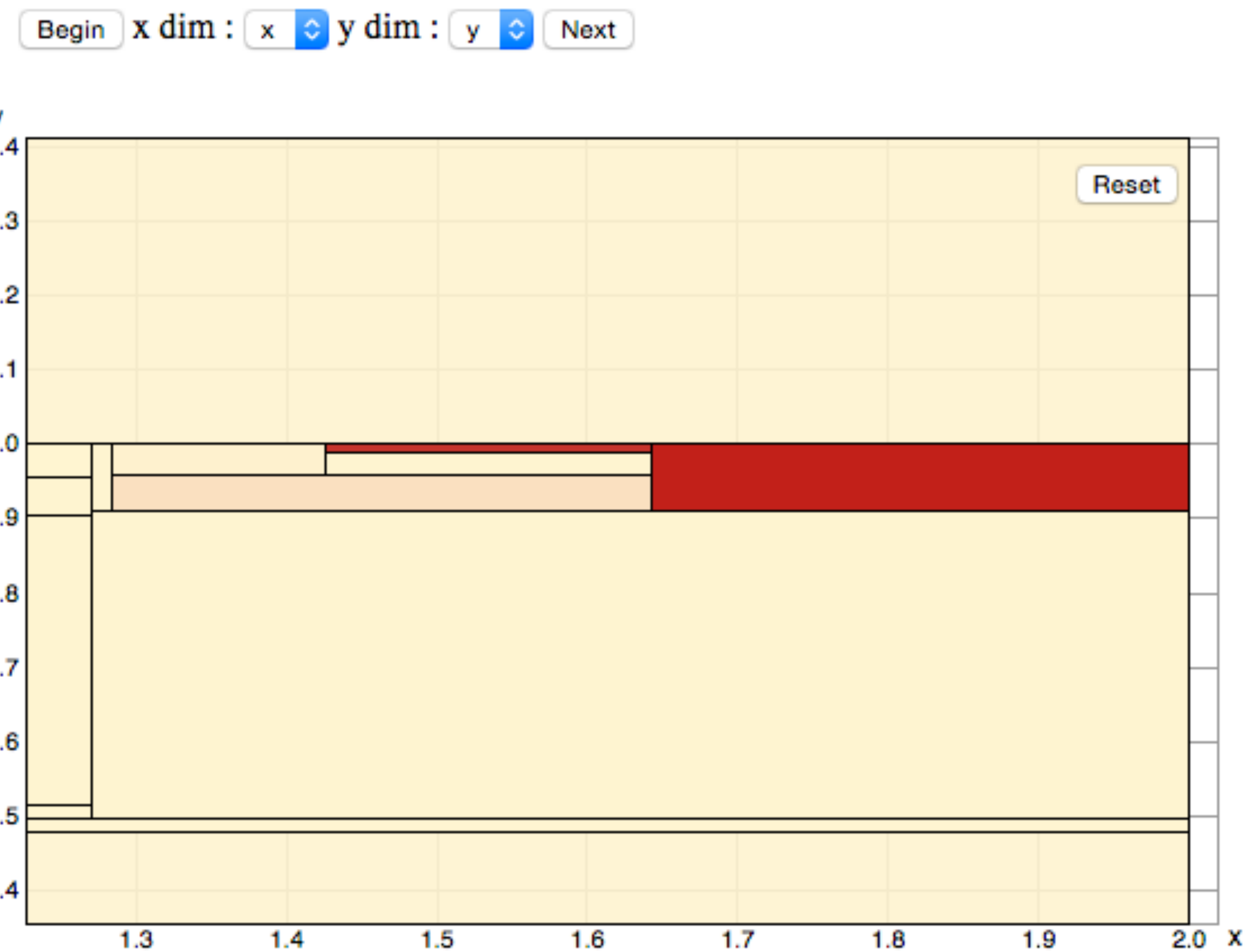
Example of ICP

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



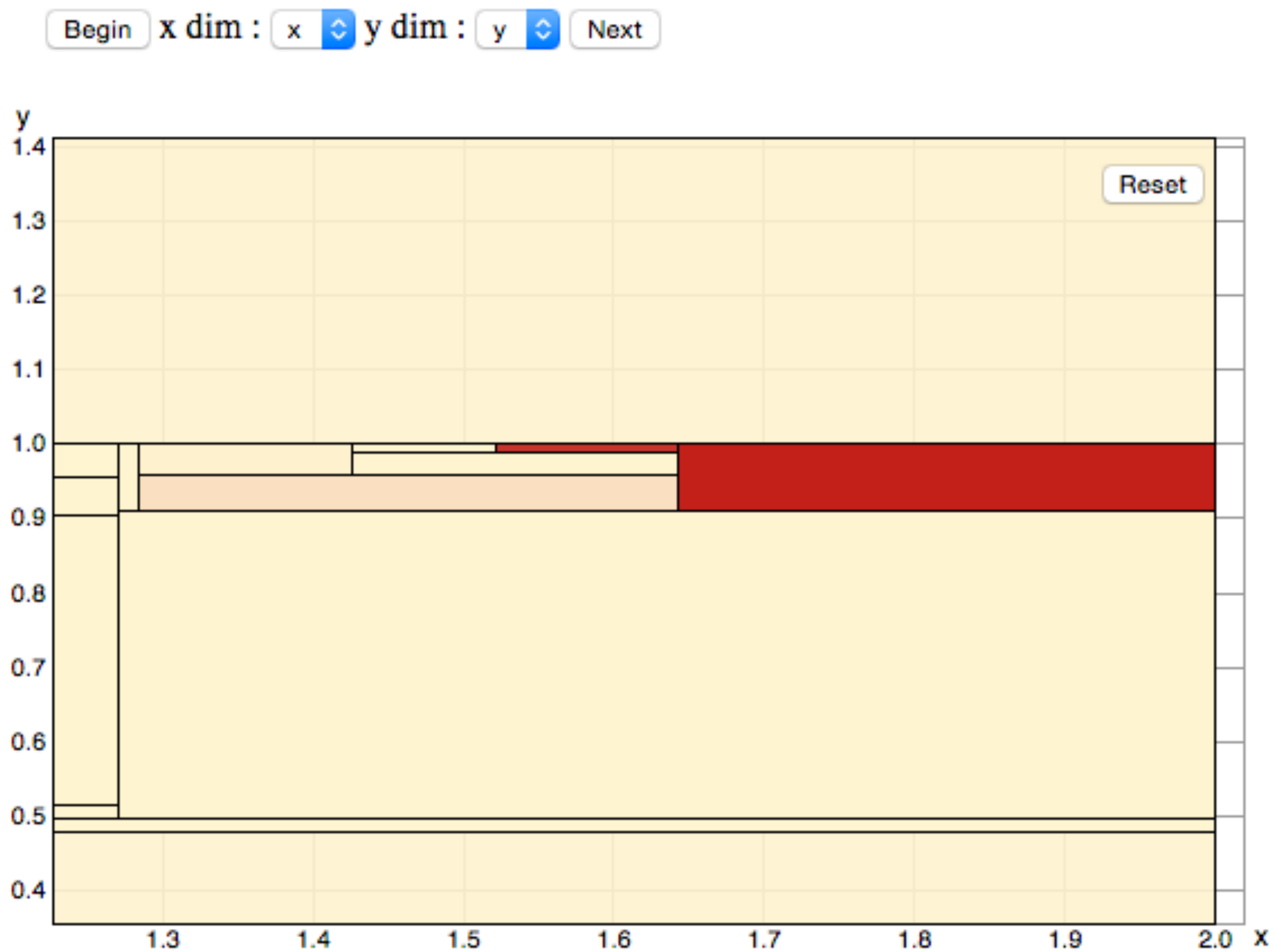
Example of ICP

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



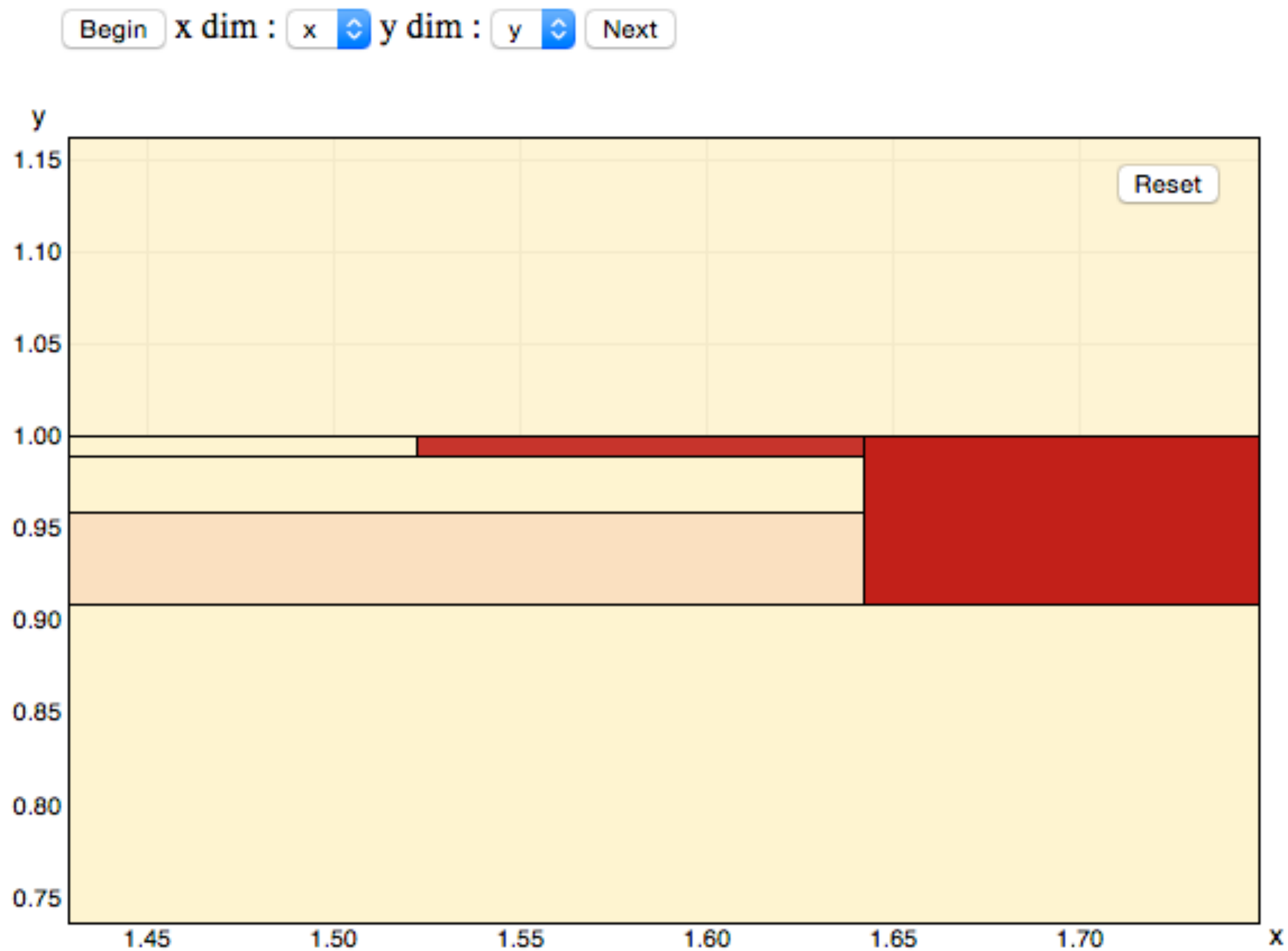
Example of ICP

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



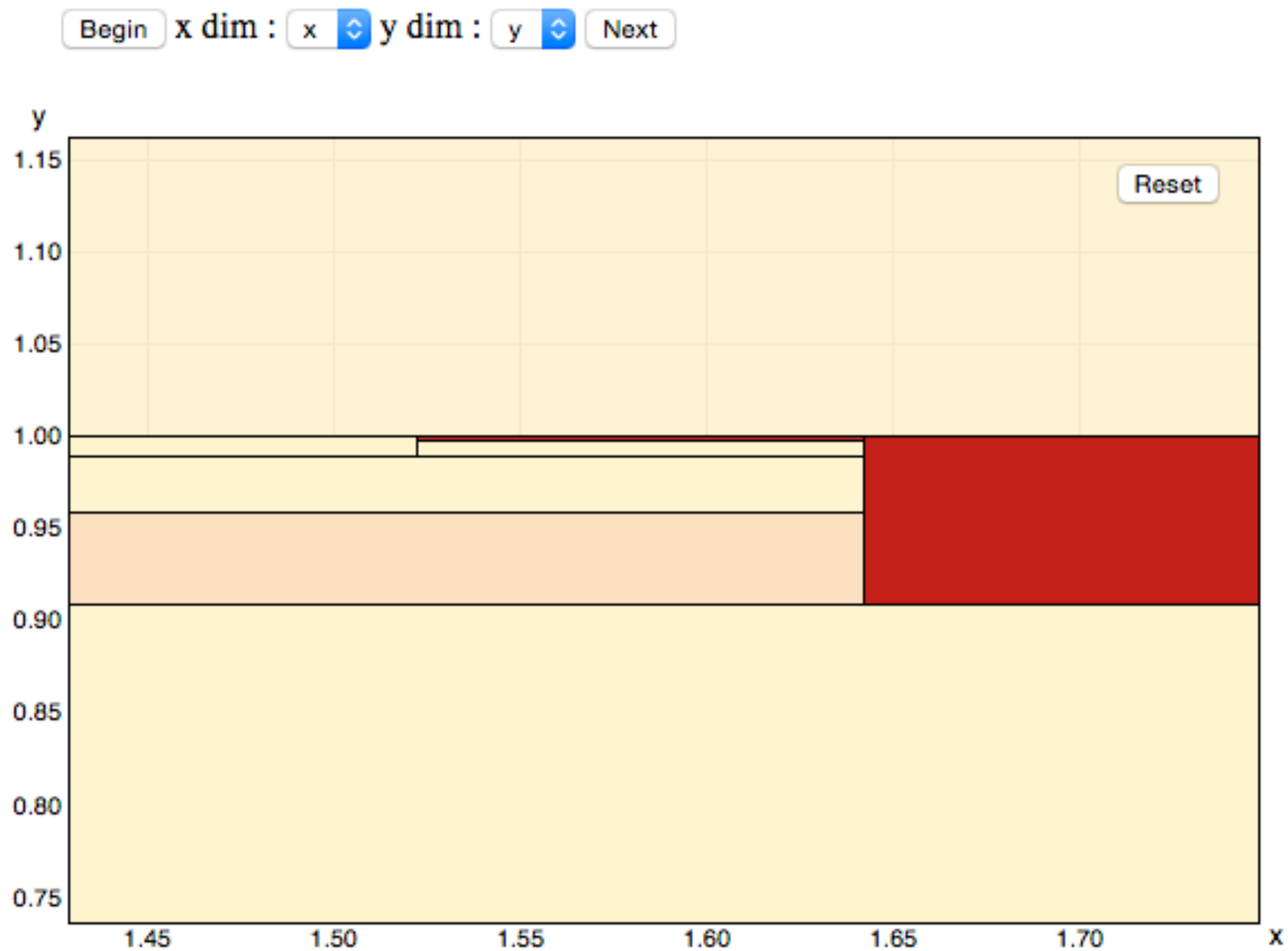
Example of ICP

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



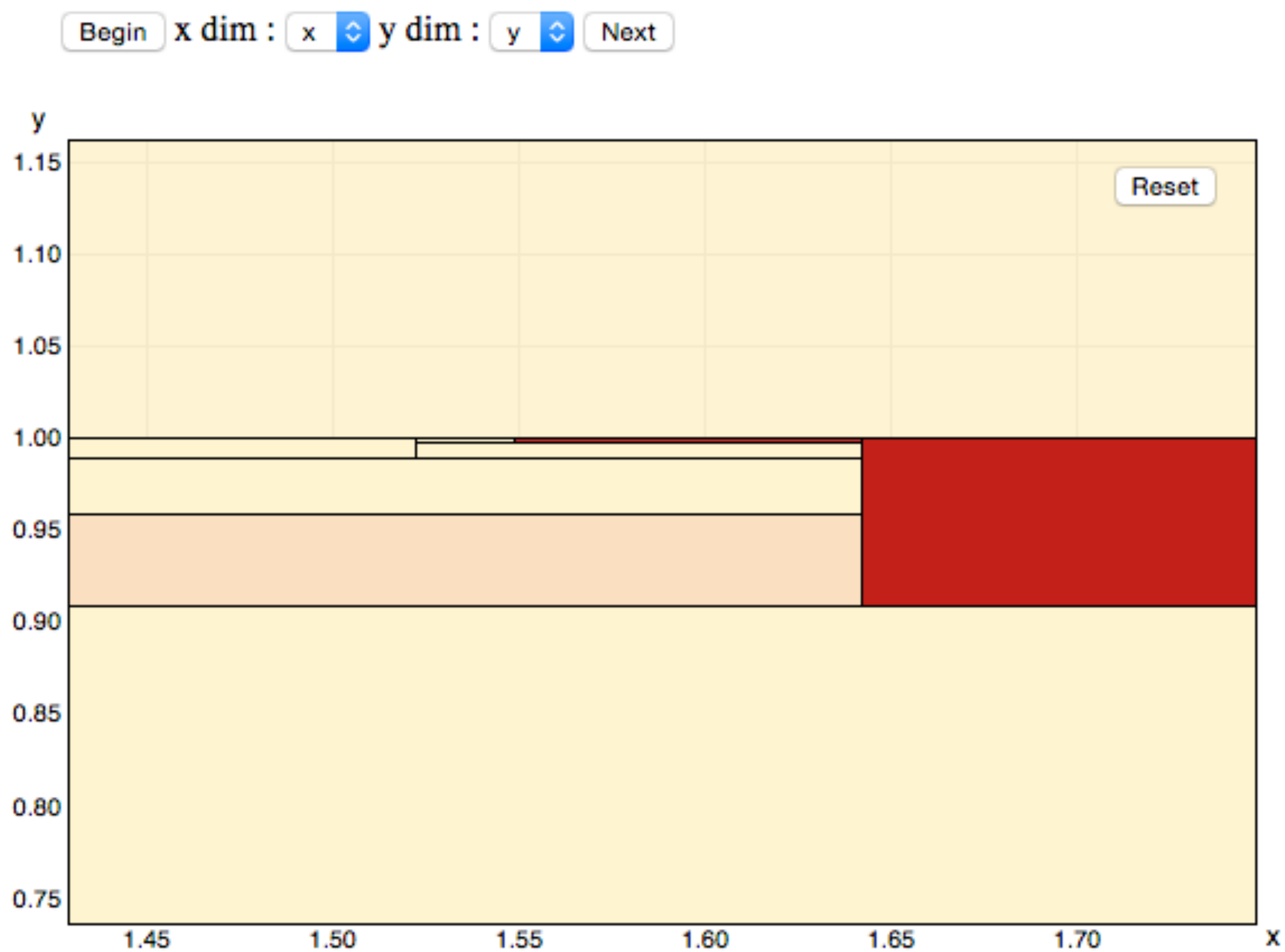
Example of ICP

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



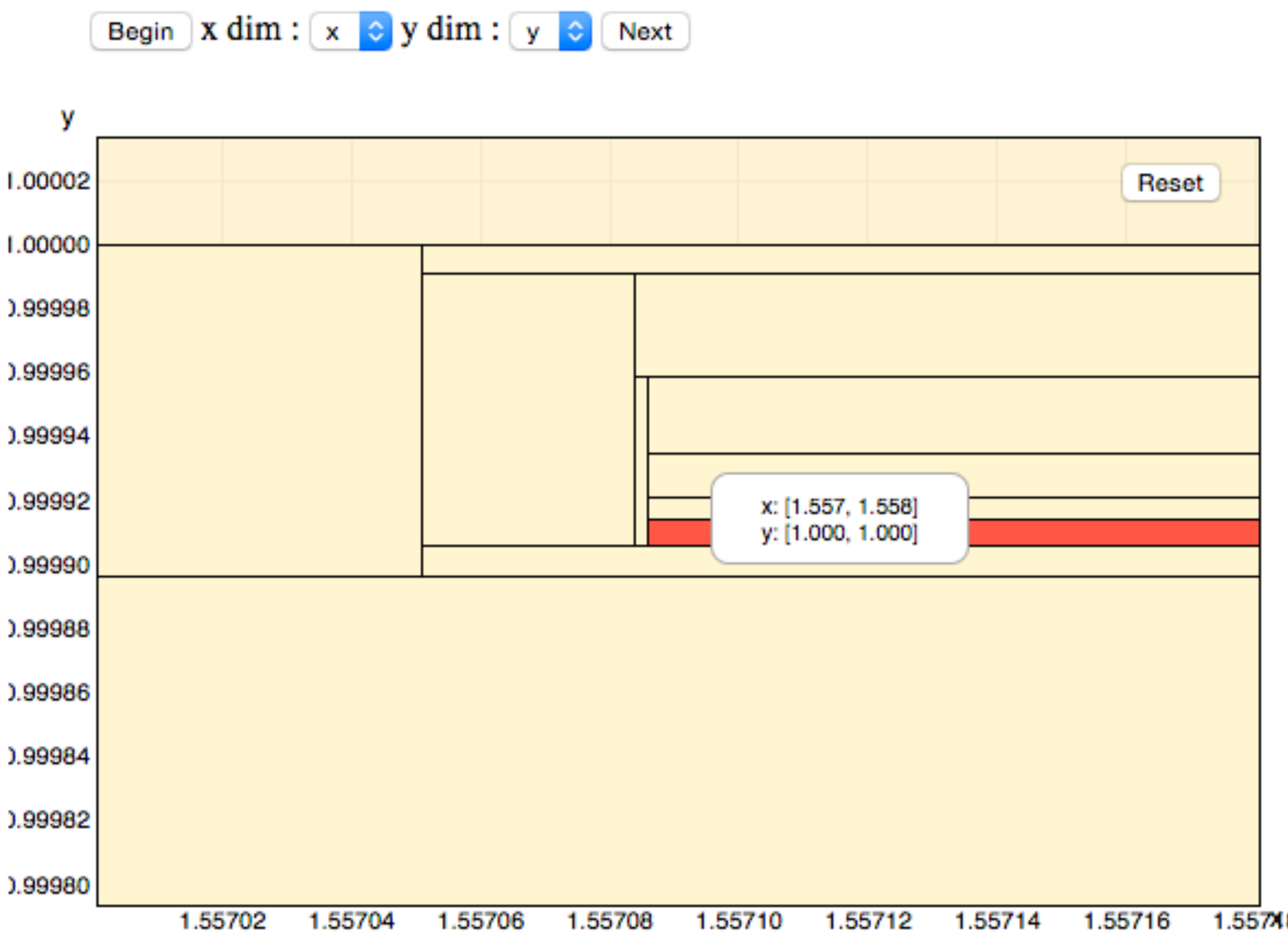
Example of ICP

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



Example of ICP

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



$$\epsilon = 0.001$$

ANSWER: **δ-SAT**

Main Algorithm of ICP

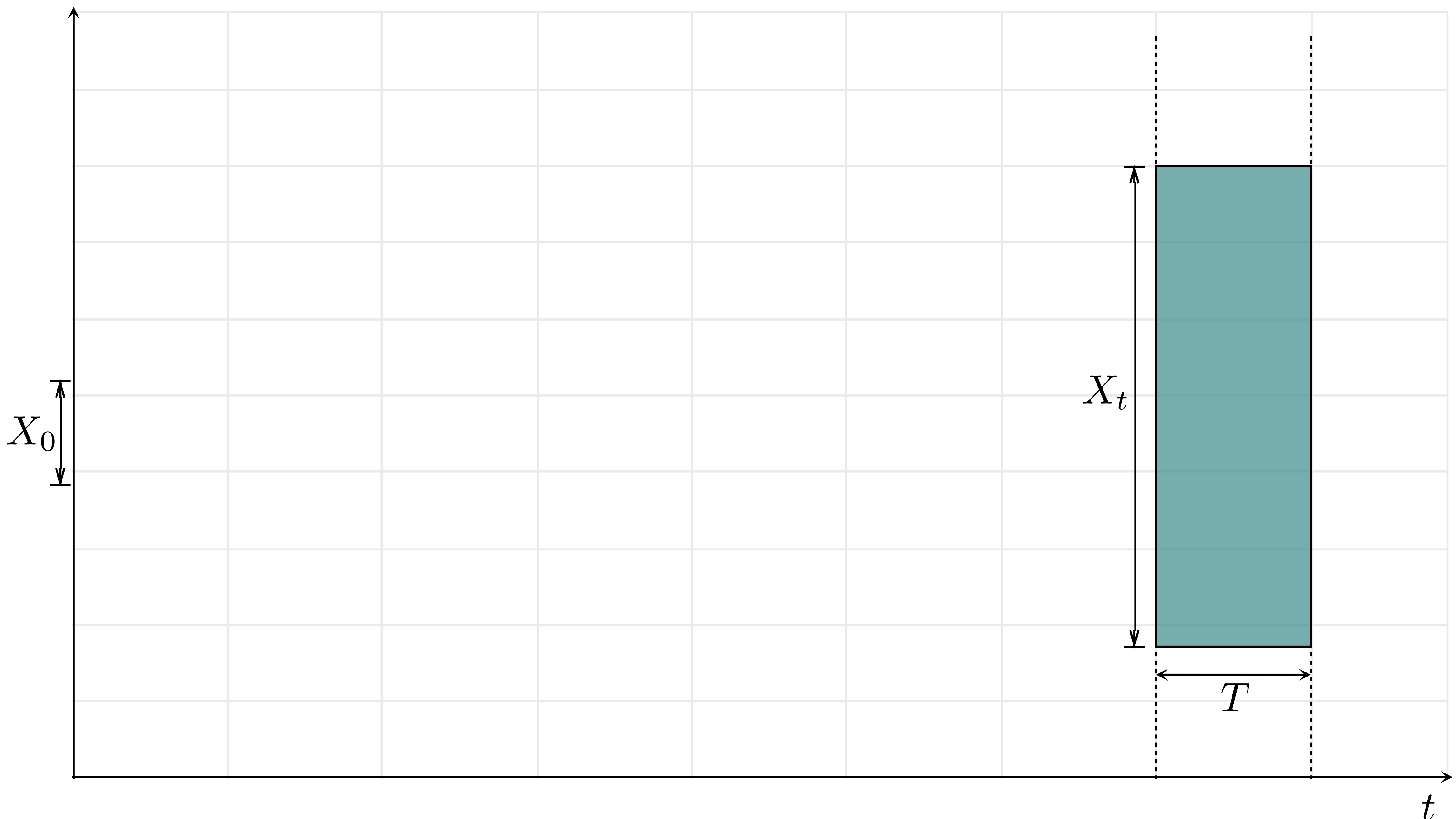
Algorithm 1: Theory Solving in DPLL$\langle$ICP$\rangle$

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;

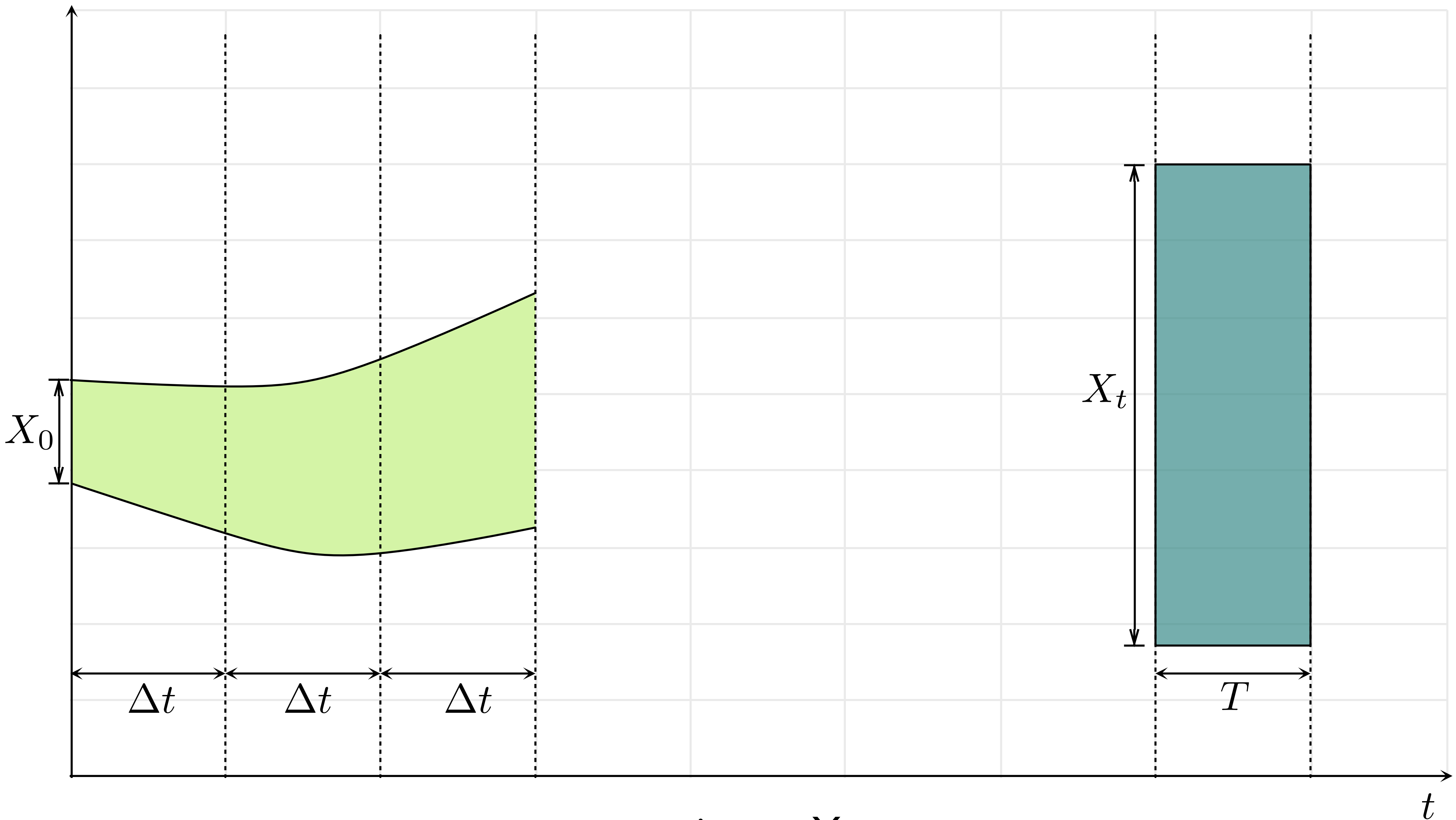
Pruning using ODEs
(Forward)



pruning on Xt

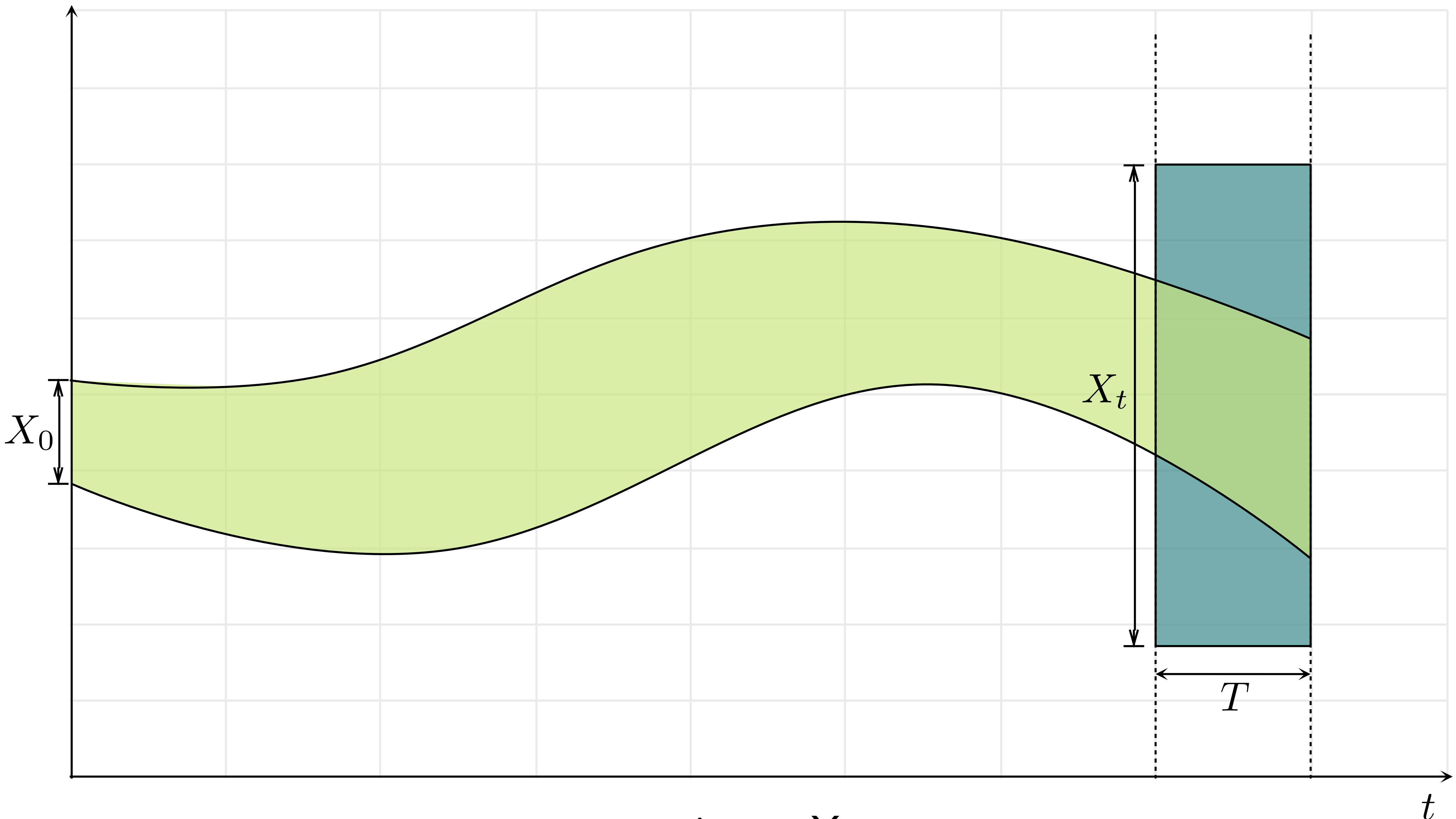
How can we have smaller X'_t?

Pruning using ODEs (Forward)



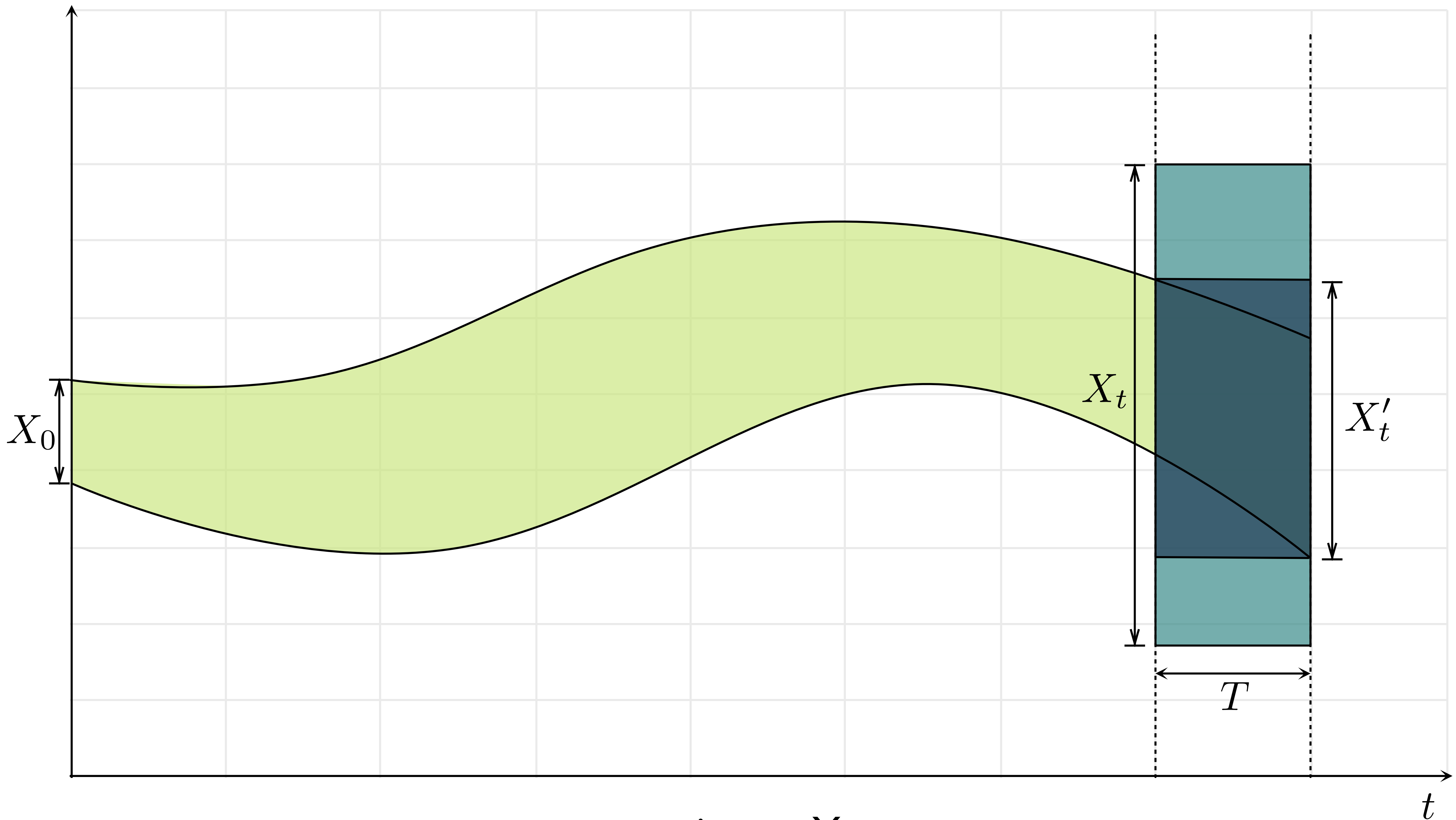
pruning on Xt

Pruning using ODEs
(Forward)



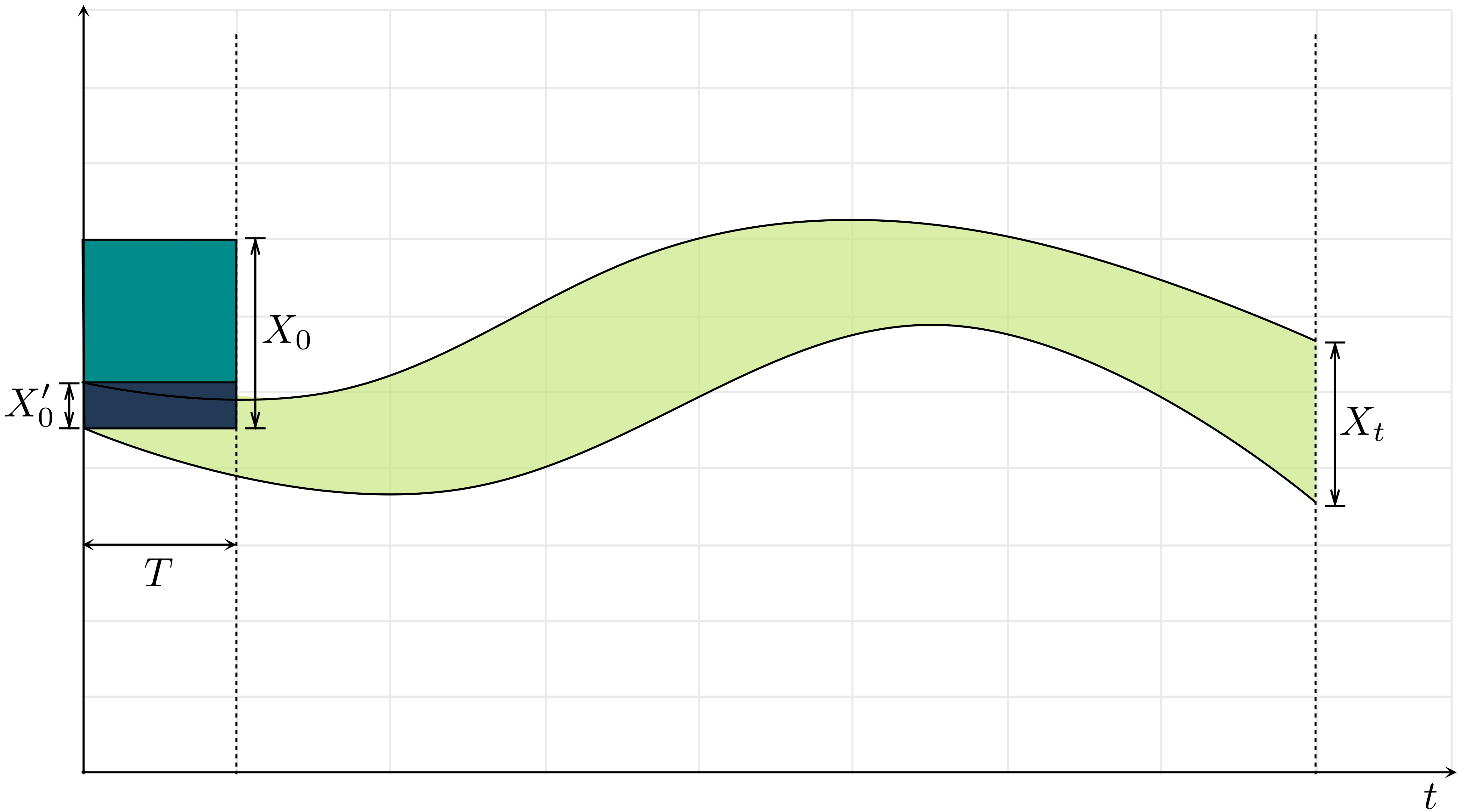
pruning on Xt

Pruning using ODEs
(Forward)



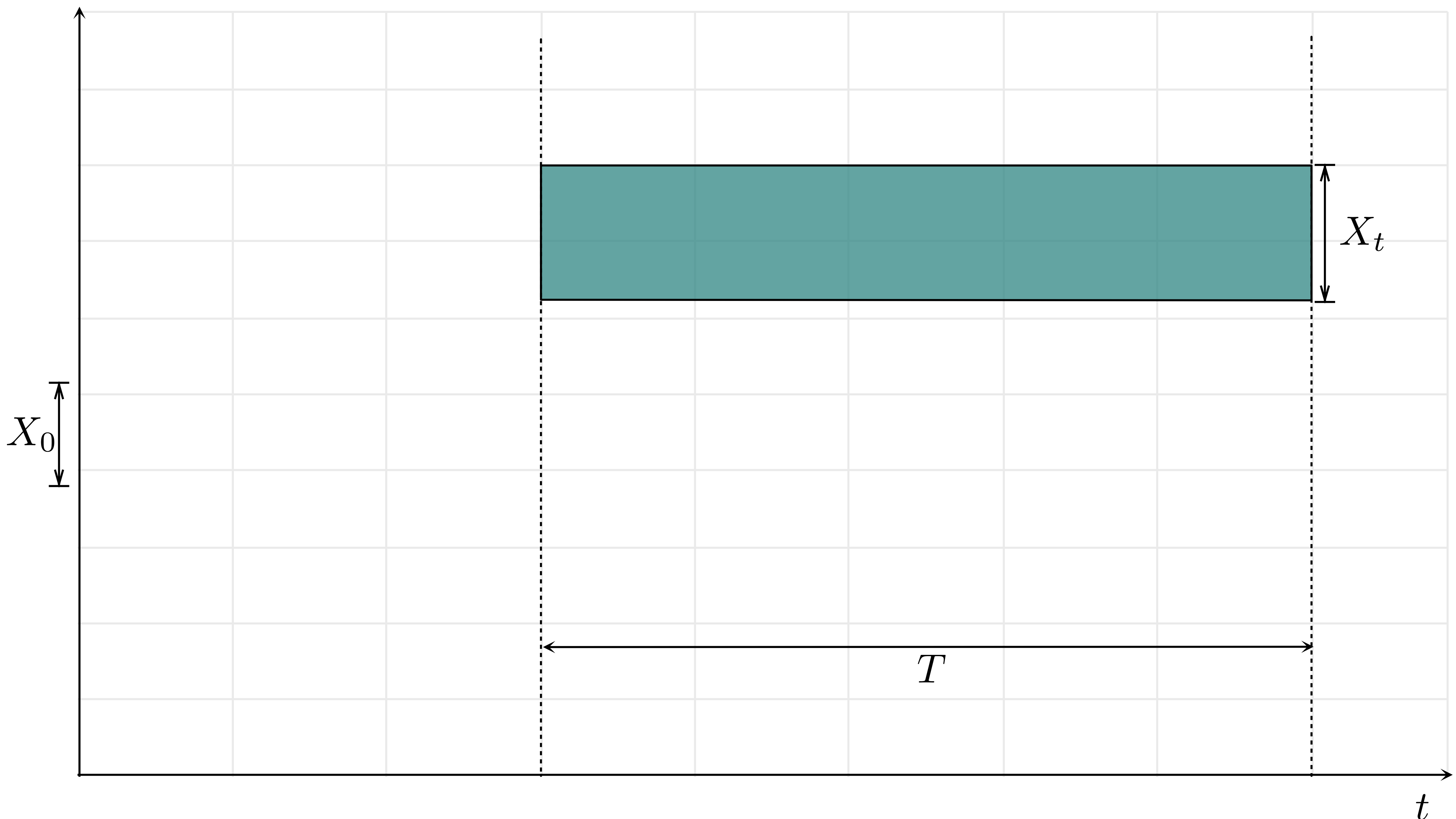
pruning on Xt

Pruning using ODEs
(Backward)



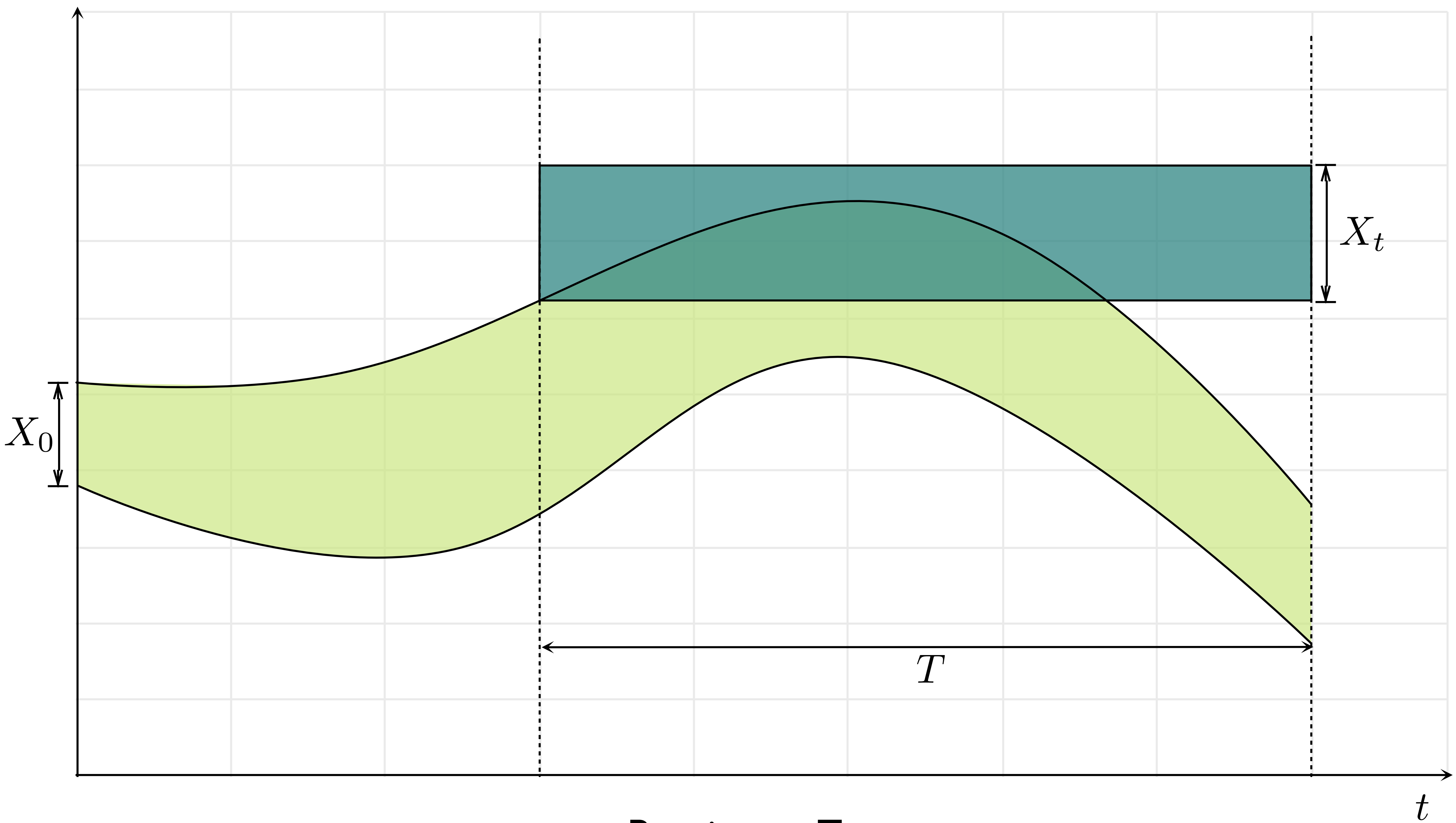
Pruning on X0

Pruning using ODEs (on Time)



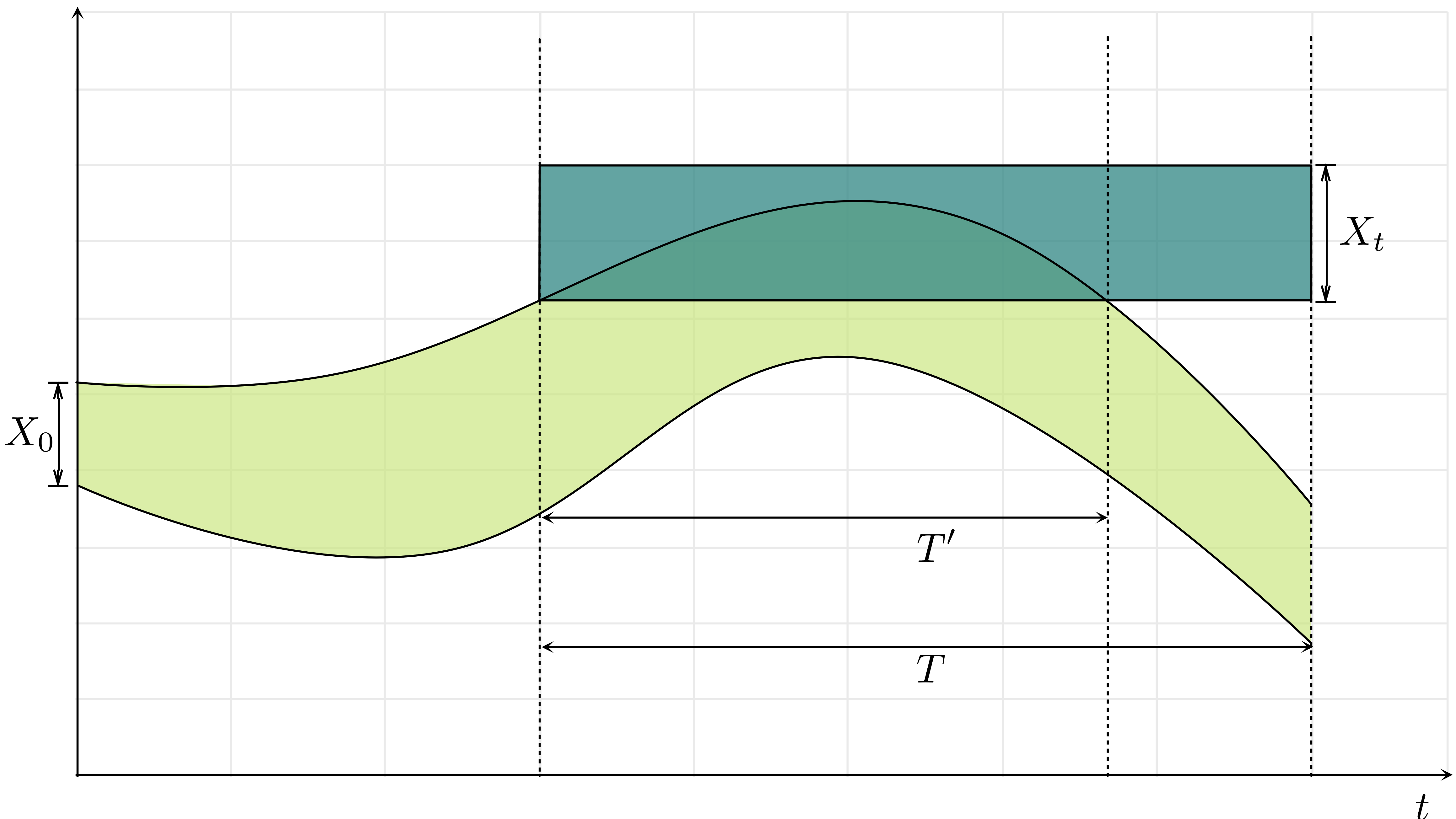
Pruning on T

Pruning using ODEs
(on Time)



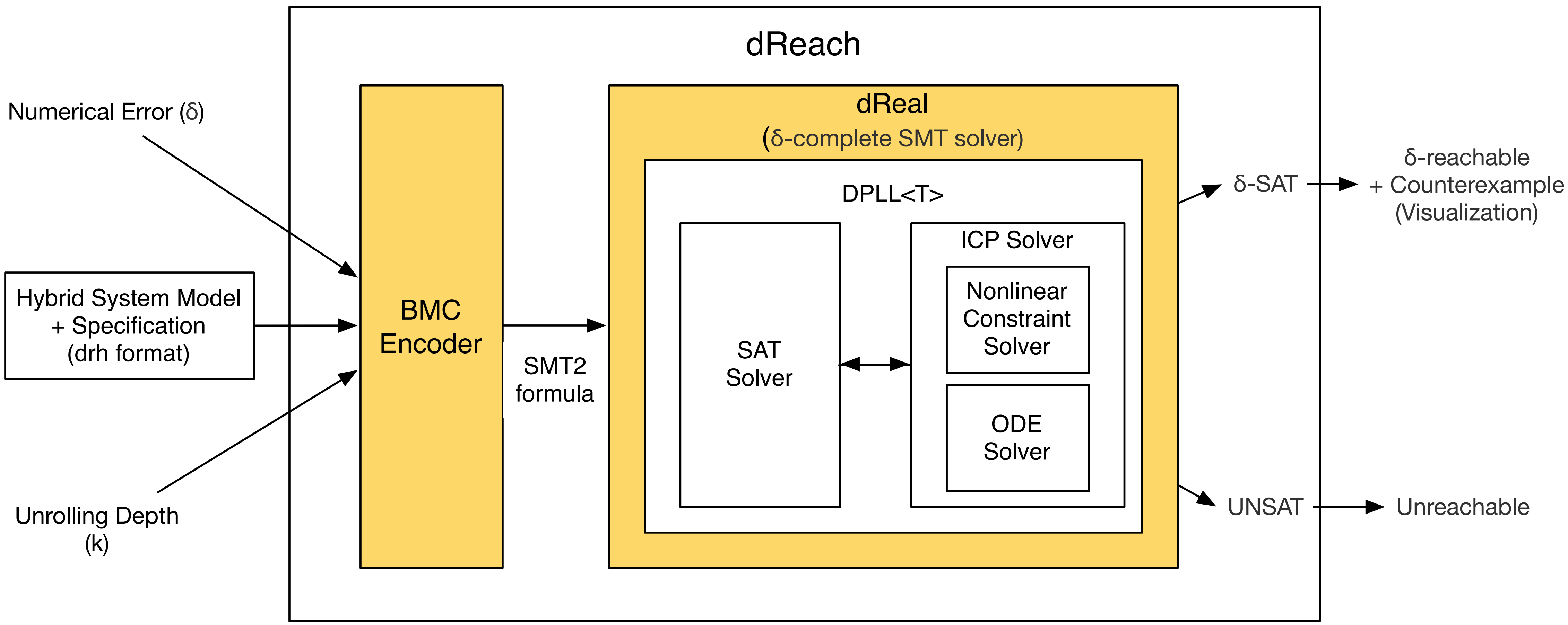
Pruning on T

Pruning using ODEs
(on Time)



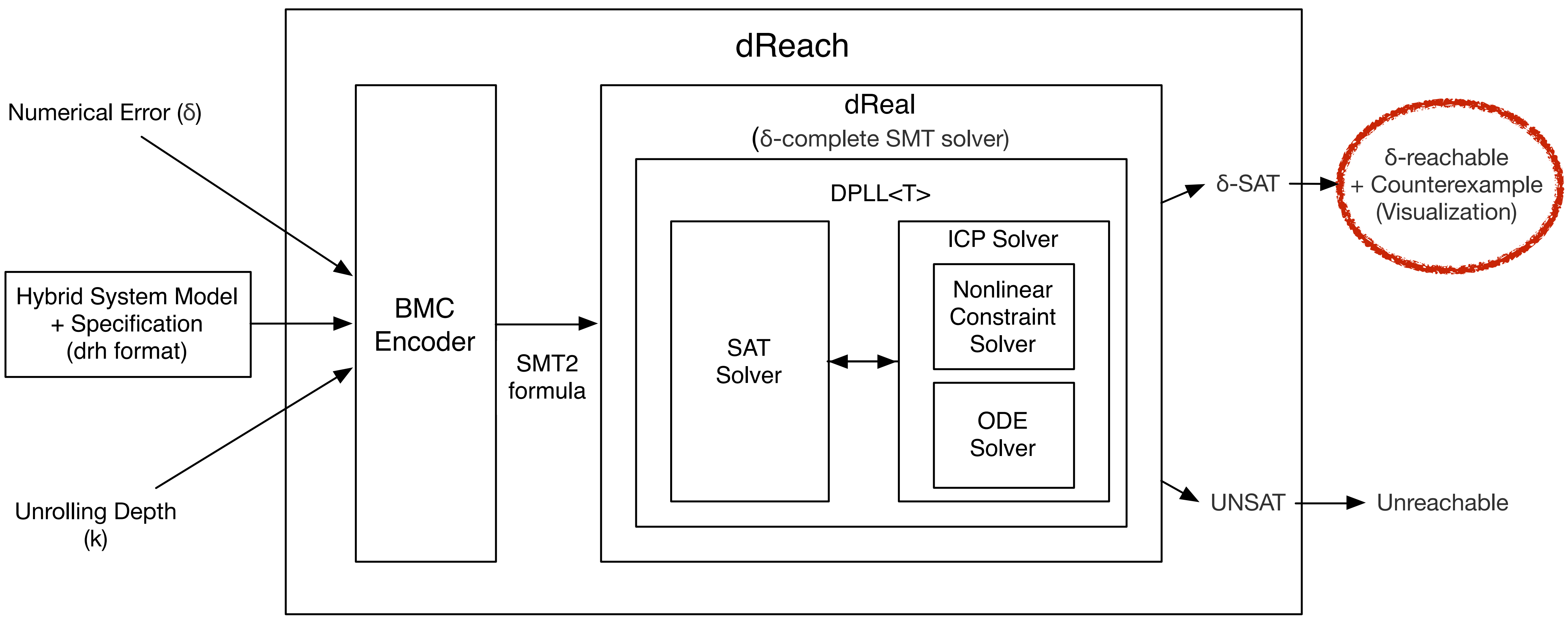
Pruning on T

dReach: δ-Reachability Analysis of Hybrid Systems

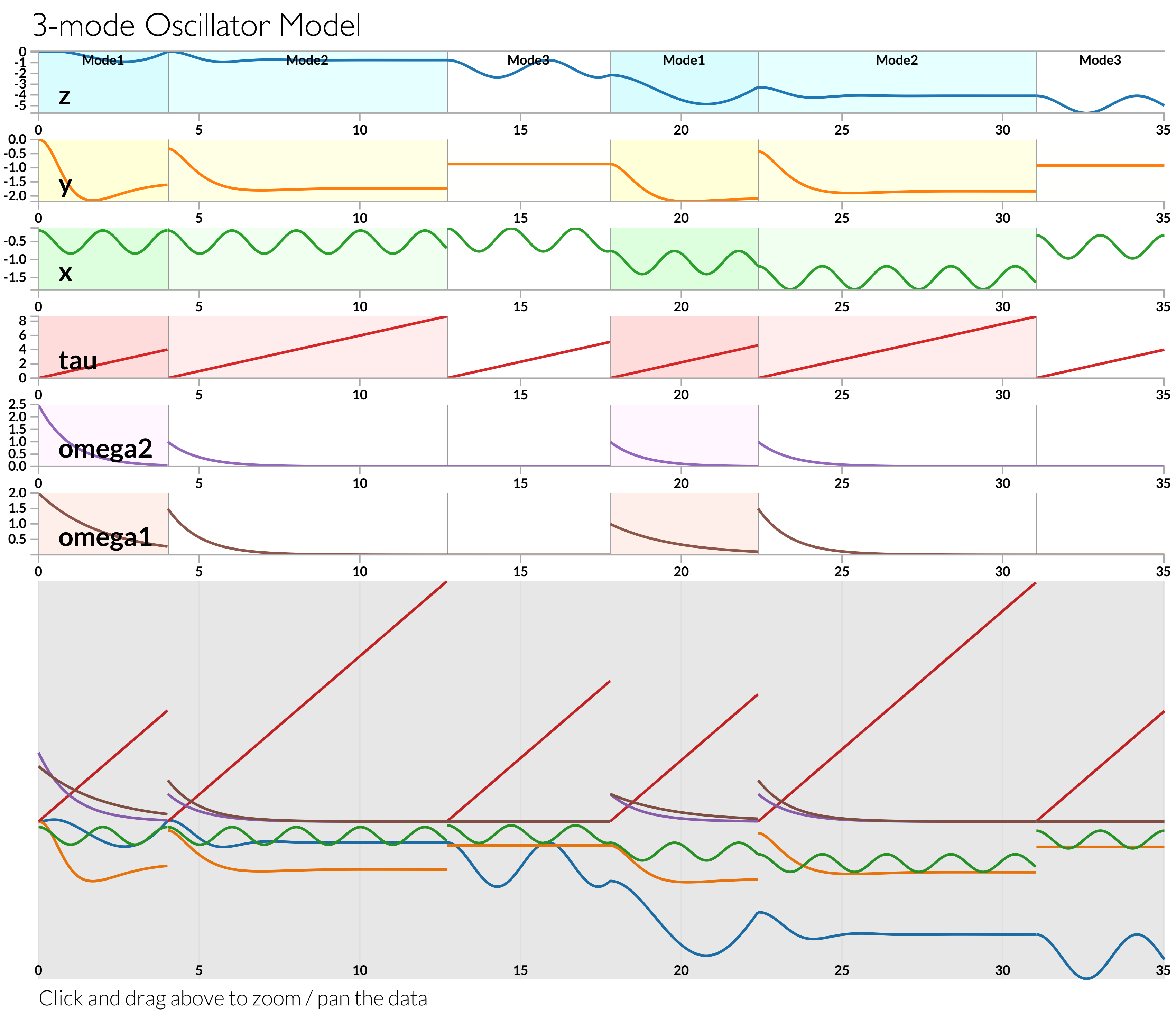


- Open Source (GPL3), available at https://dreal.github.io
- Support polynomials, transcendental functions and nonlinear ODEs
- Formulas with 100+ ODEs have been solved.

Visualization of Counterexample



Visualization of Counterexample



Applications

* Cardiac Cells, Prostate Cancer (CMU, GIT, TU Vienna)

* Prostate Cancer (CMU, UPITT)

* Power-train Control (Toyota Research Lab)

* Microfluidic Chip Designs (Waterloo)

* Analog Circuits (City University London)

* Quadcopter Control, Autonomous Driving (CMU)

* FDA-accepted non-linear hybrid physiological model for diabetes, (UPENN)

Tools based on dReal/dReach

* ProbReach: Probabilistic reachability analysis of hybrid systems (Univ. of Newcastle)
* BioPSy: Parameter set synthesis on biological models (Univ. of Newcastle)
* SReach: Bounded model checker for stochastic hybrid systems (CMU)
* Osmosis: Semantic importance sampling for statistical model checking (CMU SEI)
* Sigma: Probabilistic programming language (MIT)

Conclusion

* δ-reachability analysis checks robustness of hybrid systems which implies safety.
* Decidable (PSPACE-Complete)
* It uses dReal, a δ-complete SMT solver, which supports nonlinear functions and nonlinear ODEs
* Based on DPLL⟨ICP⟩ framework
* Scalable with our experiments
* Open-source: available at http://dreal.github.io

Future Work

* Scalability
  * Parallelization
  * Learning from failures during ICP
  * Smart Backtracking in ICP (Non-chronological BackJumping)
* Expressivity
  * Support Exist-Forall formulas (for optimization problems)

Thank you

Any Questions?