

SAT MOD ODEs

Sicun Gao / Soonho Kong / Edmund Clarke

2013/10/22, FMCAD'13

<http://dreal.cs.cmu.edu>

Decision Problems over Real Numbers

Given an arbitrary first-order φ over

$$\langle \mathbb{R}, \geq, \mathcal{F} \rangle$$

decide the truth value of φ .

Decision Problems over Real Numbers

Given an arbitrary first-order φ over

$$\langle \mathbb{R}, \geq, \mathcal{F} \rangle$$

decide the truth value of φ .

With a rich enough \mathcal{F} , we would be able to:

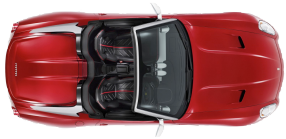
- solve many control-engineering problems
- verify and synthesize safety-critical embedded systems

High-speed Parking

High-speed Parking



High-speed Parking



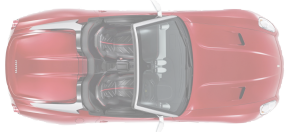
Speed up

High-speed Parking

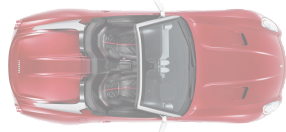


Speed up

High-speed Parking

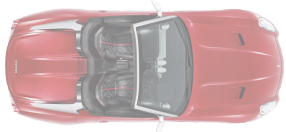


Speed up

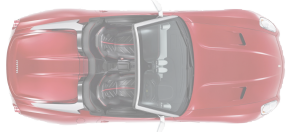


Turn

High-speed Parking

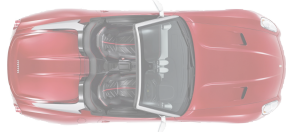


Speed up

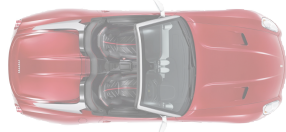


Turn

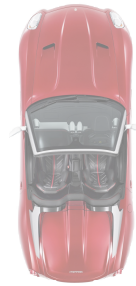
High-speed Parking



Speed up

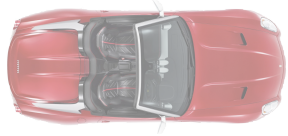


Turn

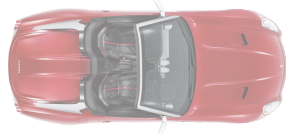


Drift

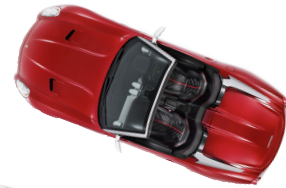
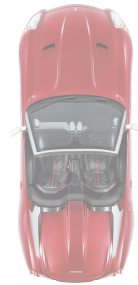
High-speed Parking



Speed up



Turn

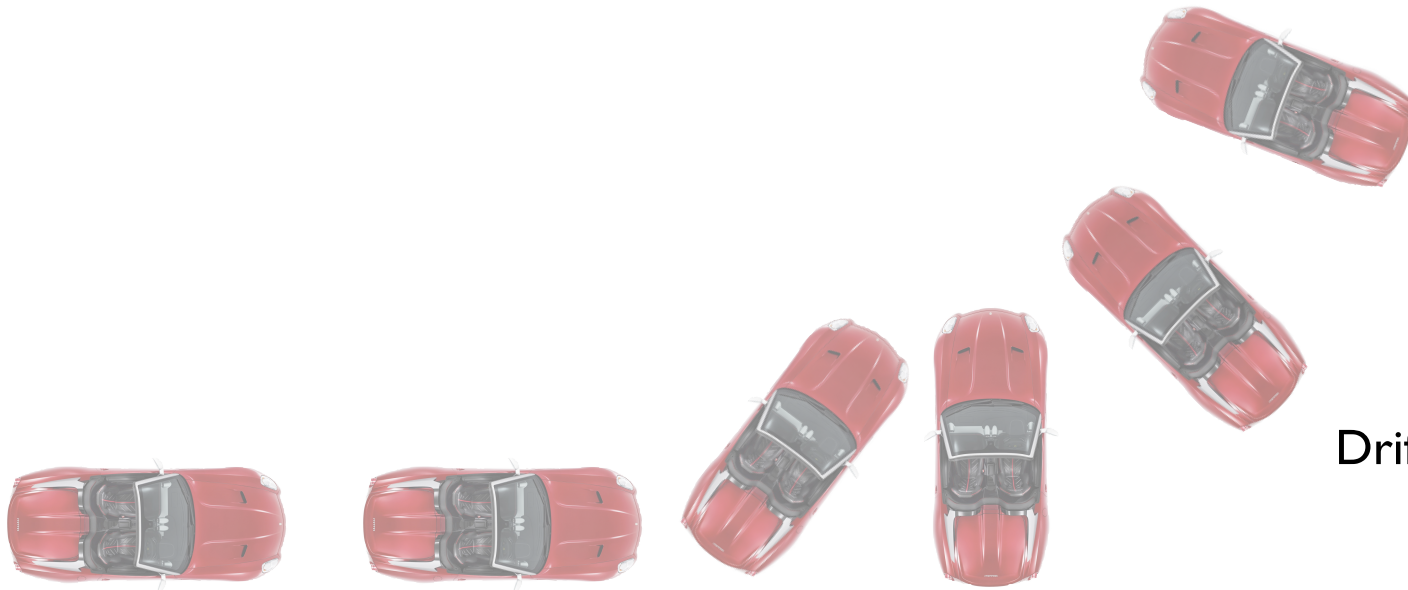


Drift

High-speed Parking



Parked

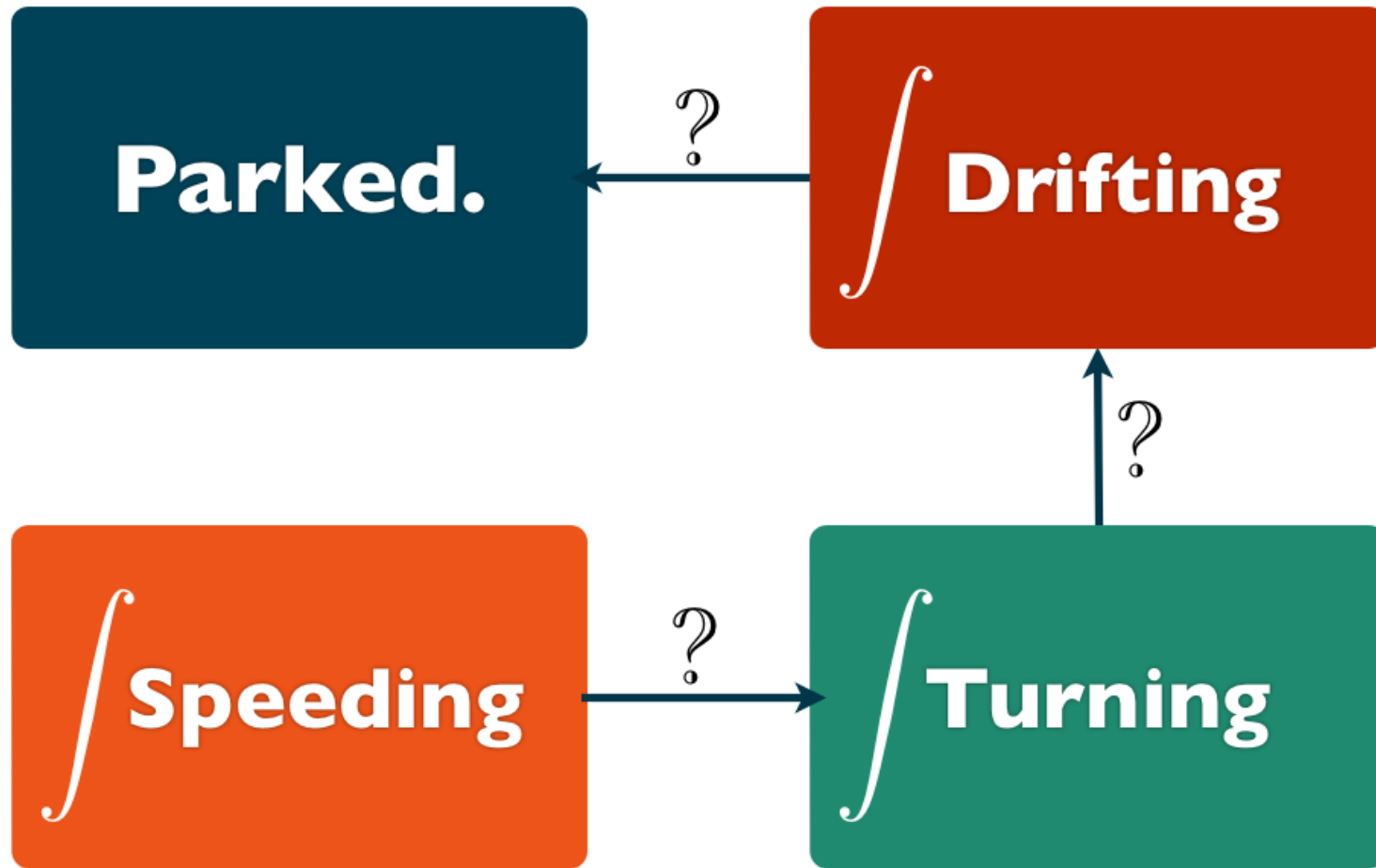


Speed up

Turn

Drift

High-speed Parking



Logic Encoding

We can do this if we can solve the following SMT formula in real-time:

$$\begin{aligned} & \text{speedup}(\vec{x}_0) \wedge \left(\vec{x}_1 = \vec{x}_0 + \int_0^{t_1} \text{speeding}(s) ds \right) \wedge \\ & \text{steer}(\vec{x}_1, \vec{x}_2) \wedge \left(\vec{x}_3 = \vec{x}_2 + \int_0^{t_2} \text{turning}(s) ds \right) \wedge \\ & \text{brake}(\vec{x}_3, \vec{x}_4) \wedge \left(\vec{x}_5 = \vec{x}_4 + \int_0^{t_3} \text{drifting}(s) ds \right) \wedge \text{parked}(\vec{x}_5) \end{aligned}$$

Isn't this problem too hard?

Difficulty

Suppose \mathcal{F} is $\{+, \times\}$.

$$\mathbb{R} \stackrel{?}{\models} \exists a \forall b \exists c (ax^2 + bx + c > 0)$$

- Decidable [Tarski 1948].
- Double-exponential lower-bound. Extensive research on practical solvers.

Difficulty

Suppose \mathcal{F} further contains **sine**:

$$\mathbb{R} \stackrel{?}{\models} \exists x, y, z (\sin^2(\pi x) + \sin^2(\pi y) + \sin^2(\pi z) = 0 \wedge x^3 + y^3 = z^3)$$

- Σ_1 case already undecidable.
- Partial algorithms are of extremely high complexity.
- Engineers would rather be left alone.

The key is to change the decision problem.

The Delta-Decision Problem (one version)

Given φ and $\delta \in \mathbb{Q}^+$, return one of the following:

- φ is false.
- A weakening of the original formula, $\varphi^{-\delta}$, is true.

We now define what $\varphi^{-\delta}$ is.

δ -Variants

Any bounded $\mathcal{L}_{\mathcal{F}}$ -sentence φ can be written in the form

$$Q_1^{[u_1, v_1]} x_1 \cdots Q_n^{[u_n, v_n]} x_n \bigwedge (\bigvee t(\vec{x}) > 0 \vee \bigvee t(\vec{x}) \geq 0)$$

Definition (δ -weakening)

Let $\delta \in \mathbb{Q}^+ \cup \{0\}$. The **δ -weakening** $\varphi^{-\delta}$ of φ is

$$Q_1^{[u_1, v_1]} x_1 \cdots Q_n^{[u_n, v_n]} x_n \bigwedge (\bigvee t(\vec{x}) > -\delta \vee \bigvee t(\vec{x}) \geq -\delta)$$

δ -Decisions

Let $\delta \in \mathbb{Q}^+$ be arbitrary.

Definition (δ -decisions)

Decide, for any given bounded φ , whether

- φ is false, or
- $\varphi^{-\delta}$ is true.

When the two cases overlap, either answer can be returned.

δ -Decidability

Let \mathcal{F} be an arbitrary collection of Type 2 computable functions.

Theorem [Gao et al. LICS'12]

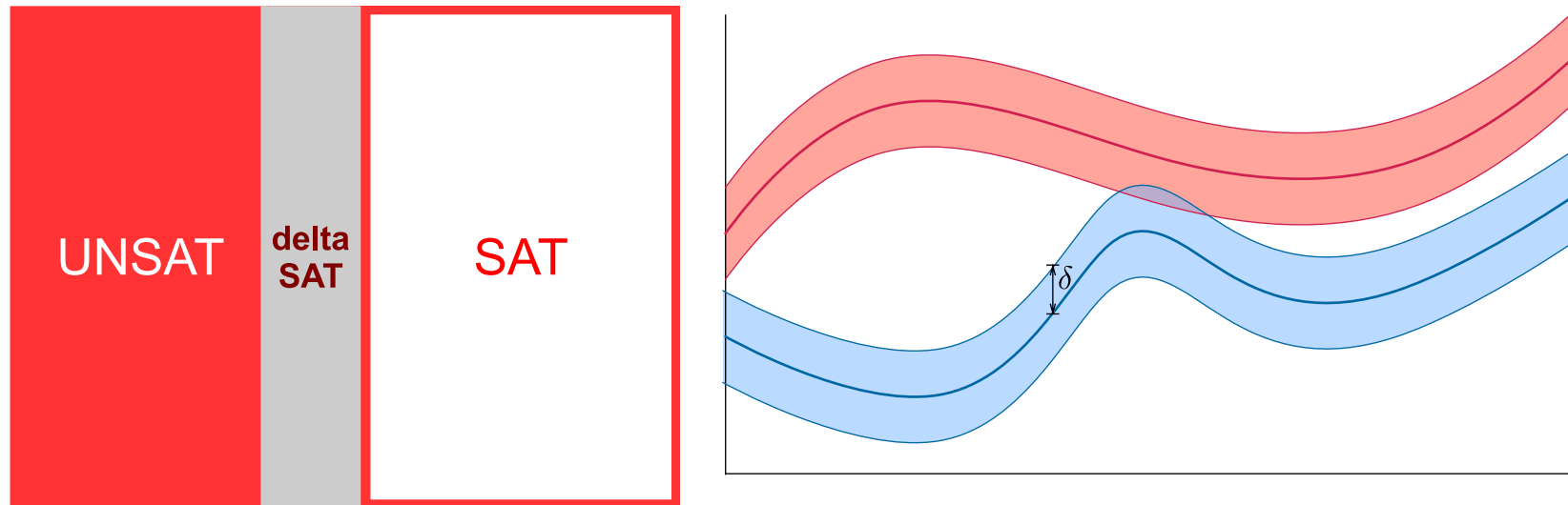
The δ -decision problem over $\mathbb{R}_{\mathcal{F}}$ is decidable.

Type 2 computable functions:

- Polynomials
- exp, sine, ...
- L-continuous ODEs
- PDEs, ...

δ -Decisions

There is a grey area that a δ -complete algorithm can be wrong about.



δ is good

A system S is **safe** if some formula φ is false.

- $\exists x_0 \exists t \exists x_t (\text{Reach}(x_0, t, x_t) \wedge \text{Unsafe}(x_t))$

Now the interpretation of δ -decisions is:

- False: S is **safe** (within bounds, for BMC).
- δ -True: S is **unsafe**, or **some δ -perturbation would make it unsafe**. You shouldn't rely on it anyway.

Complexity

Theorem

- $\mathcal{F} = \{+, \times, \exp, \sin, \dots\} : \Sigma_k^P$ -complete.
- $\mathcal{F} = \{\text{ODEs with PTIME deriv.}\} : \text{PSPACE -complete.}$

These are extremely low compared to the original ones.

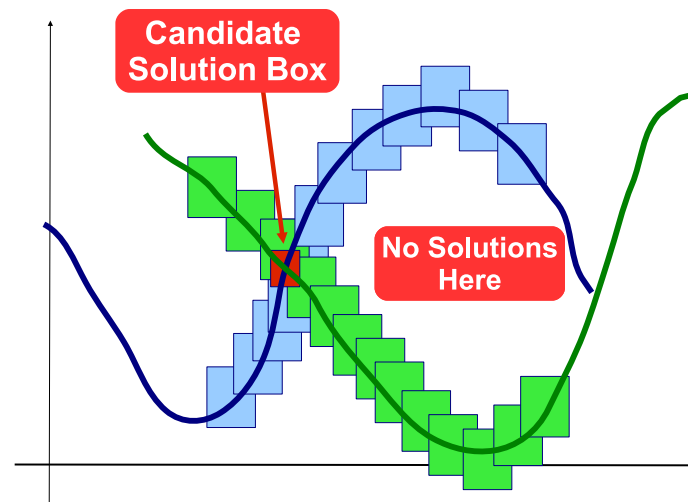
**In theory, it may be possible to solve
some. In practice?**

Formal Analysis of Numerical Algorithms

- We say an algorithm is **δ -complete** if it solves δ -decision problem.
- Many numerically-driven procedures satisfy δ -completeness after **formal analysis** [Gao et al, IJCAR'12].

Interval Constraint Propagation

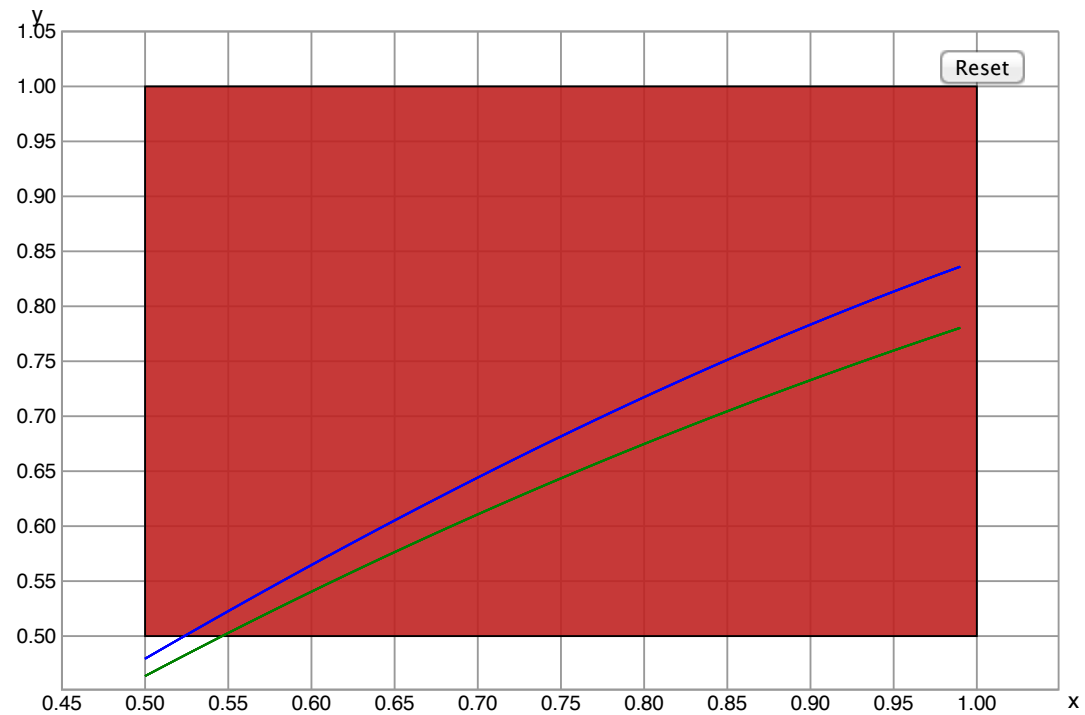
- Contract big initial interval boxes to small ones that cover solutions.
- If some constraints are satisfiable, then the interval relaxations always have overlapping boxes.



Interval Constraint Propagation

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

Begin x dim : x y dim : y Next



δ -Completeness of ICP

We gave conditions for a pruning operator to be **well-defined**, formalizing practical implementation strategies used in ICP.

Theorem [Gao et al. IJCAR'12]

DPLL(ICP) is δ -complete **iff** its pruning operators are well-defined.

We now go into the details of ODE solving.

Handling Differential Equations

An ODE system

$$\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}, t)$$

when put in Picard–Lindelöf form:

$$\vec{x}_t = \vec{x}_0 + \int_0^t f(\vec{x}, s) ds$$

is seen as a constraint between \vec{x}_0 , \vec{x}_t , and t .

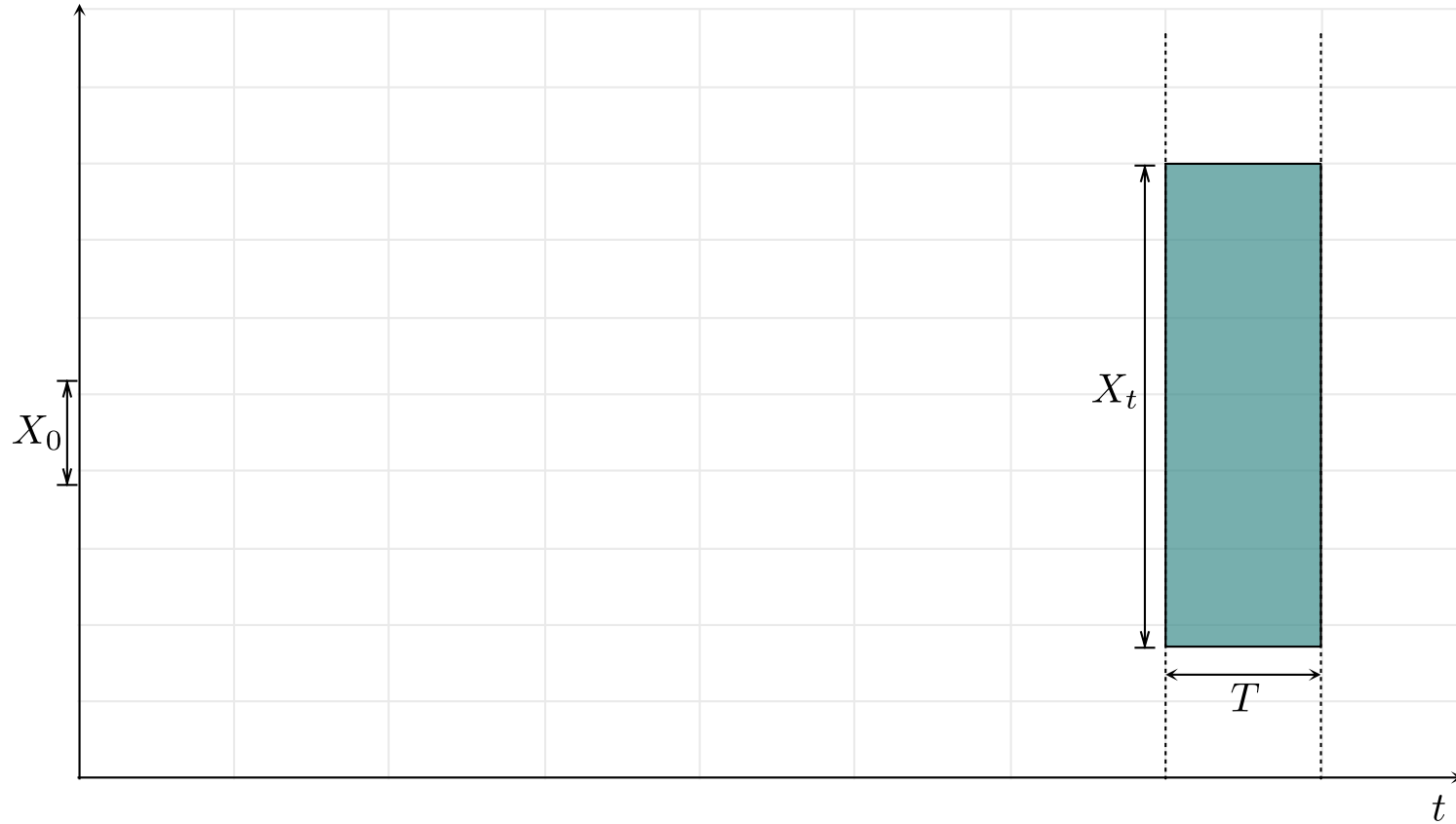
ODE Pruning

Starting with big intervals for

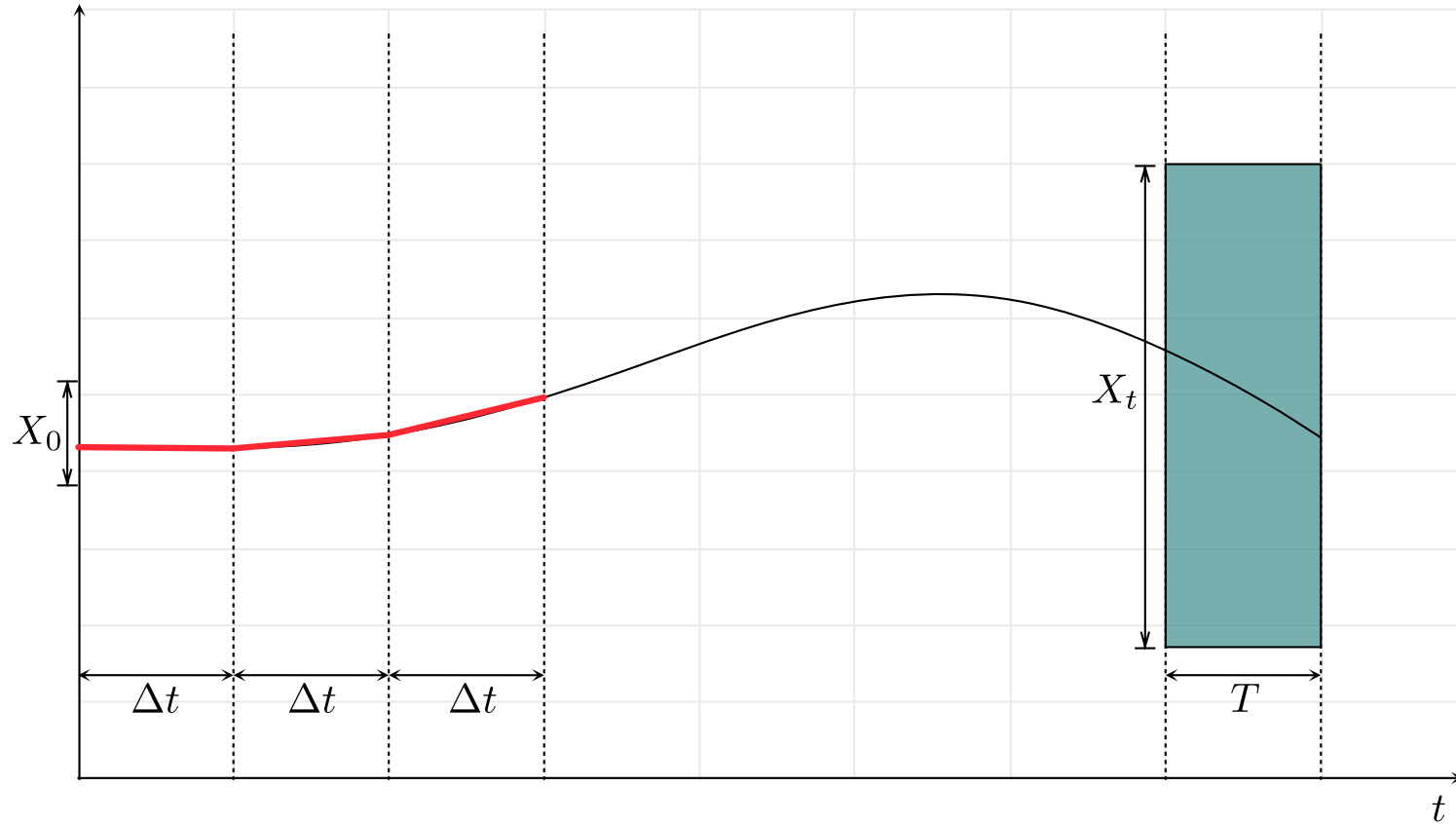
$$\vec{x}_t, \vec{x}_0, t$$

use the ODE constraints to find smaller intervals for them.

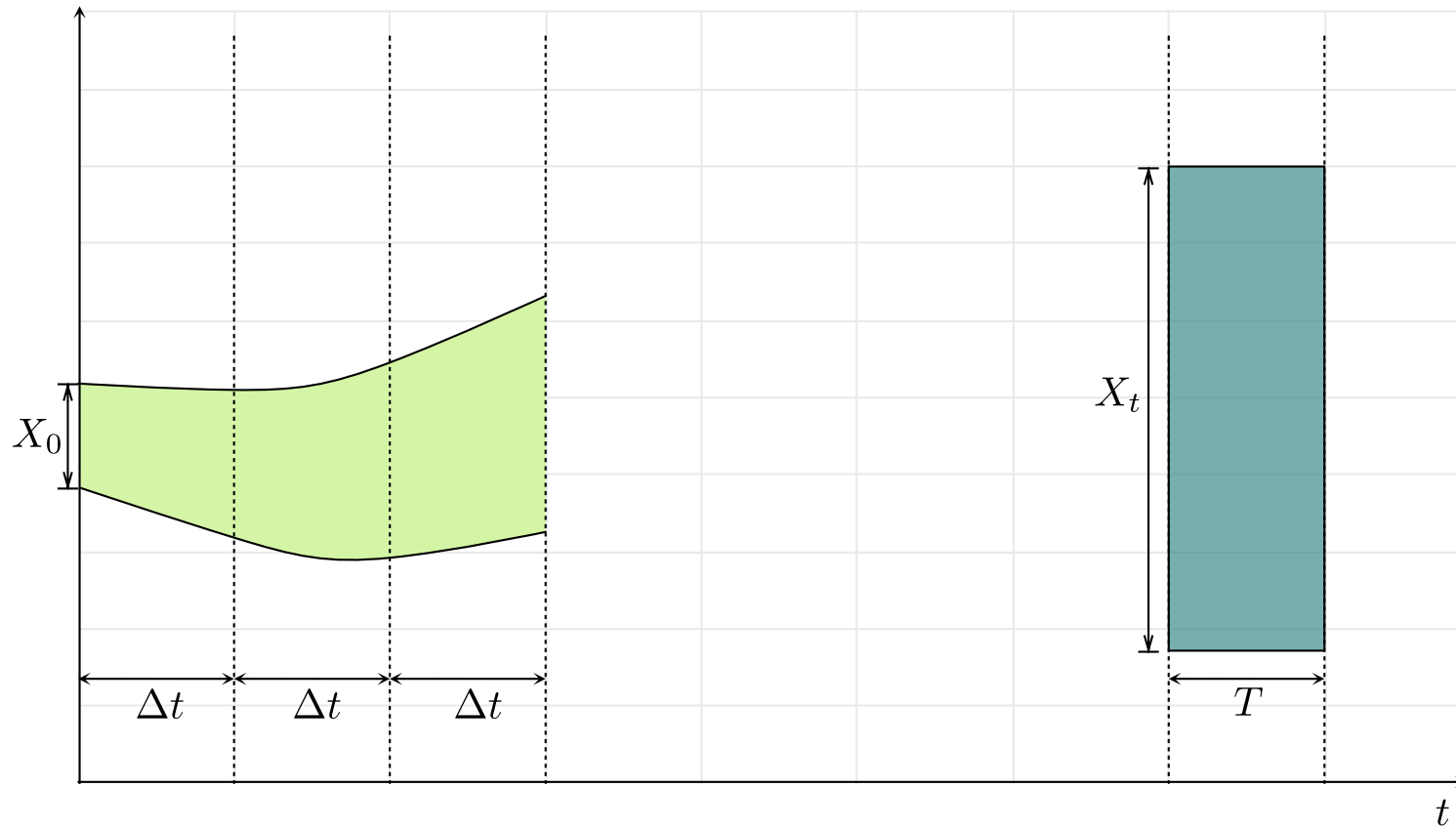
Forward Pruning (on X_t)



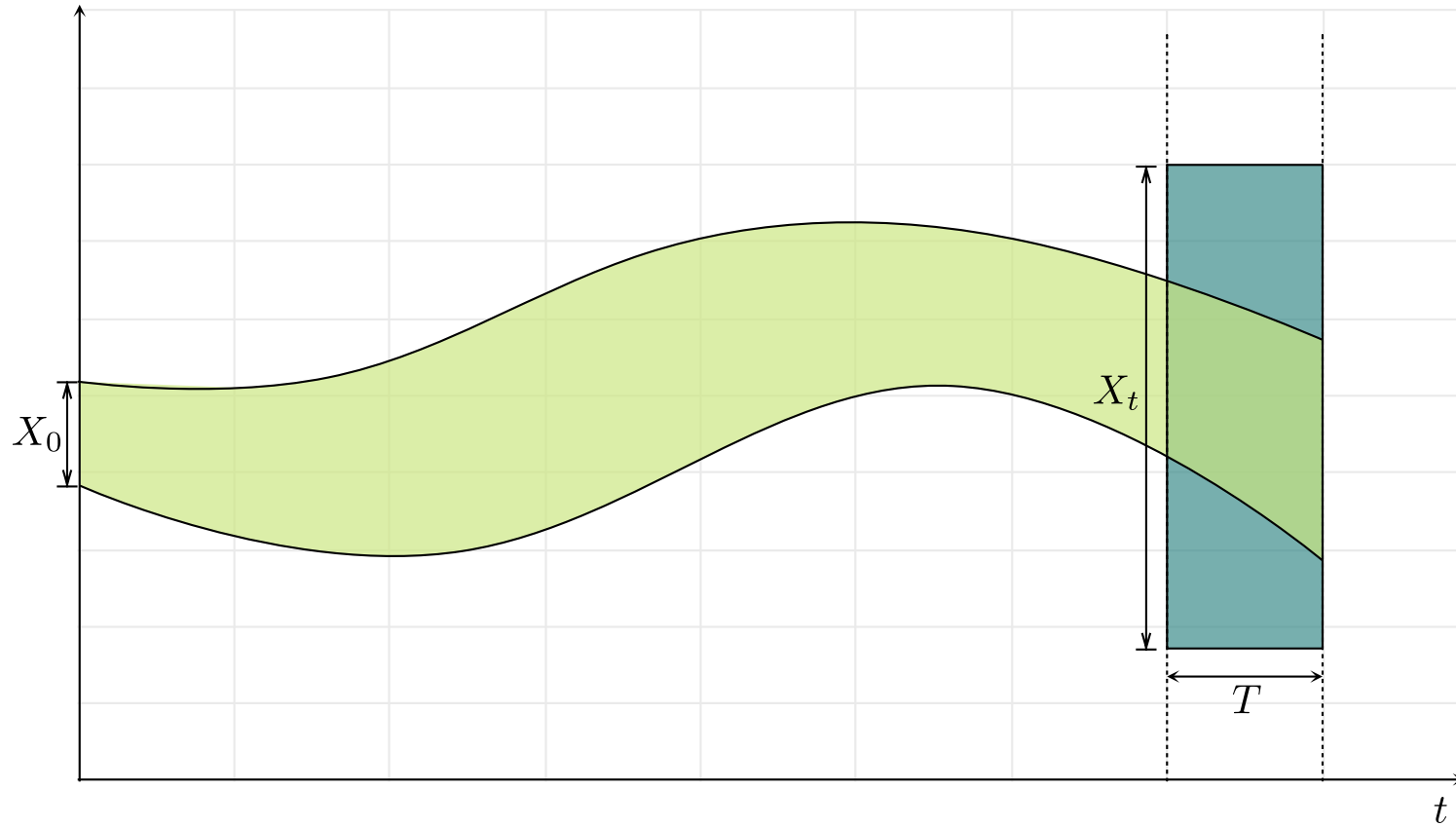
Forward Pruning (on X_t)



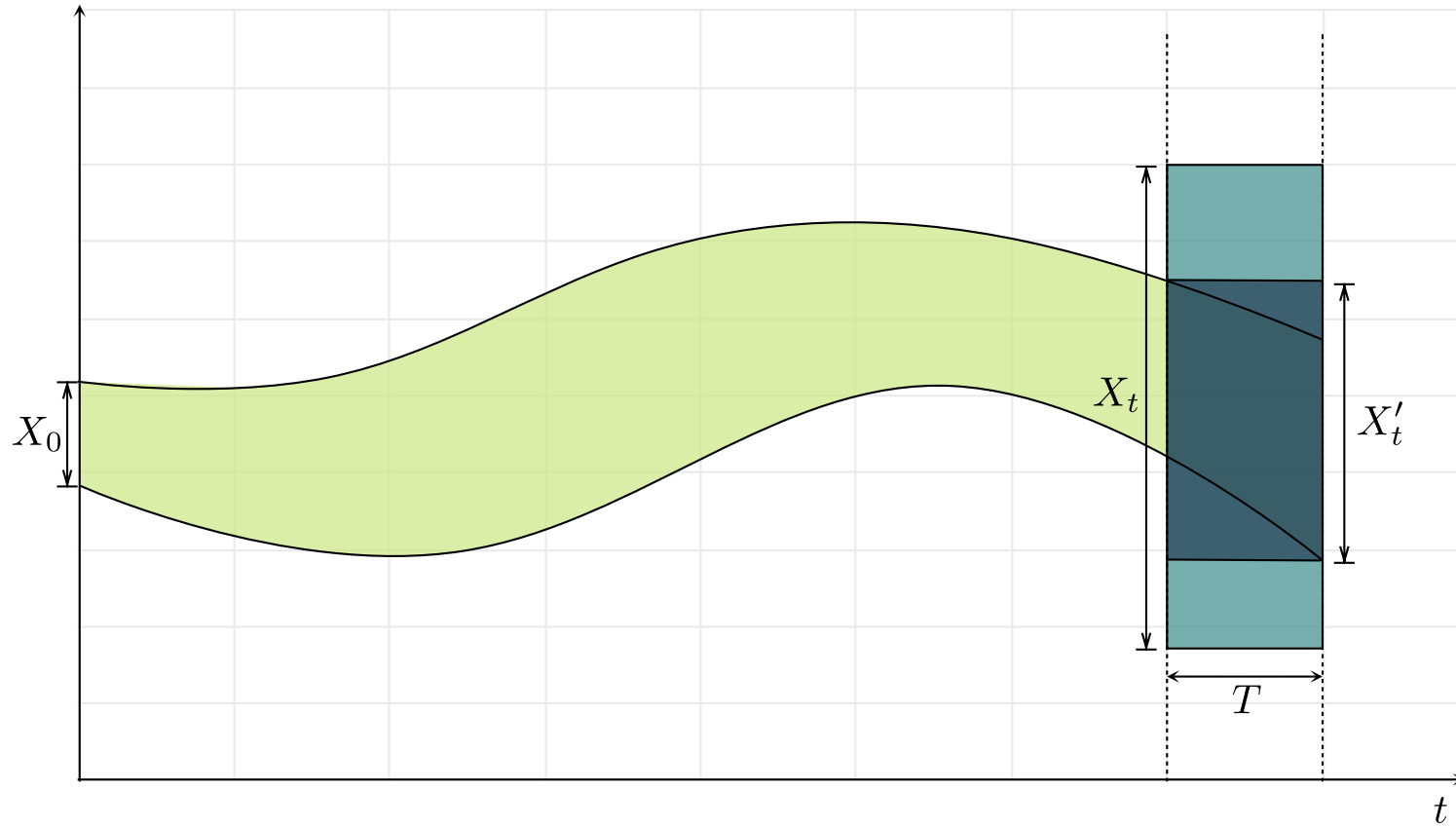
Forward Pruning (on X_t)



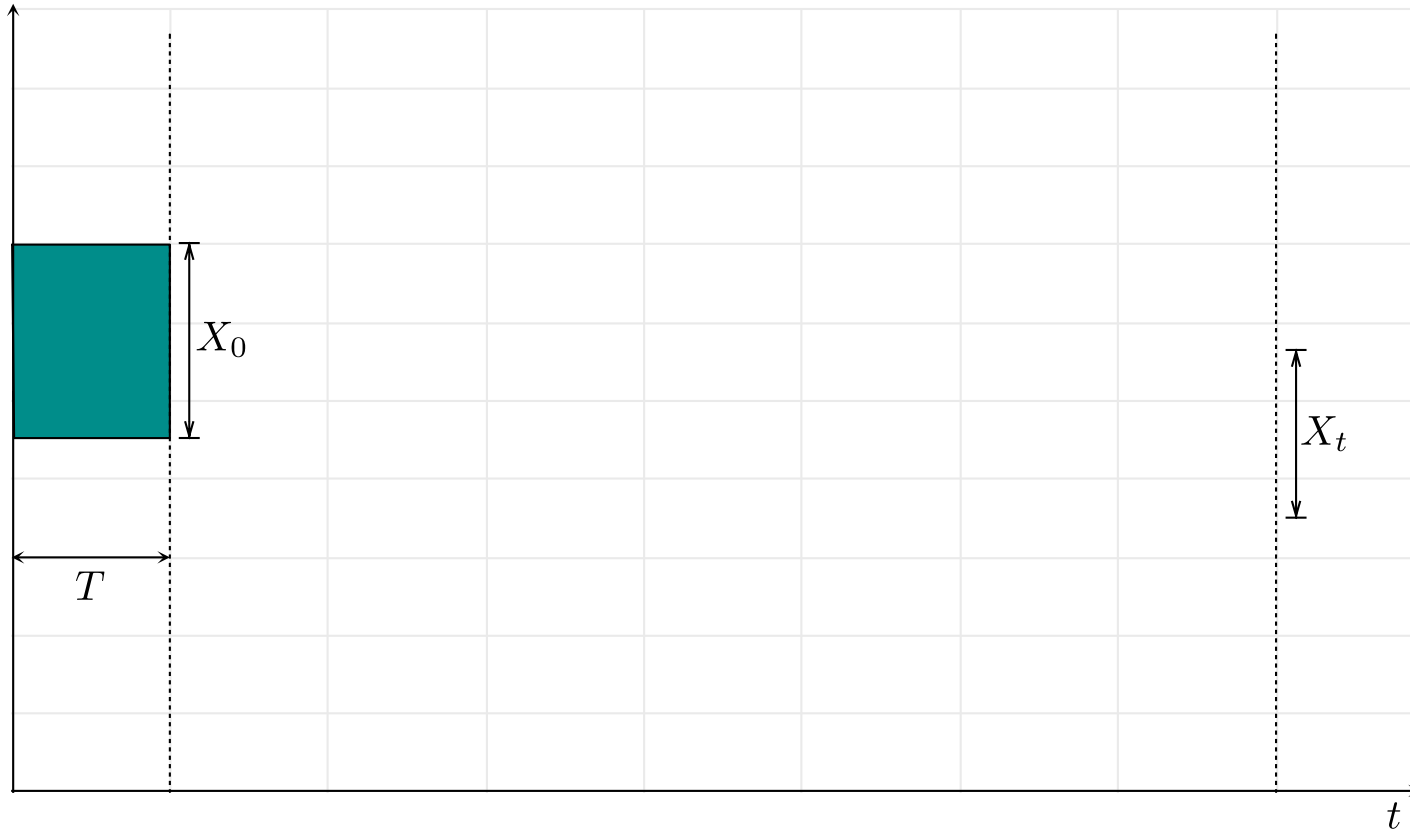
Forward Pruning (on X_t)



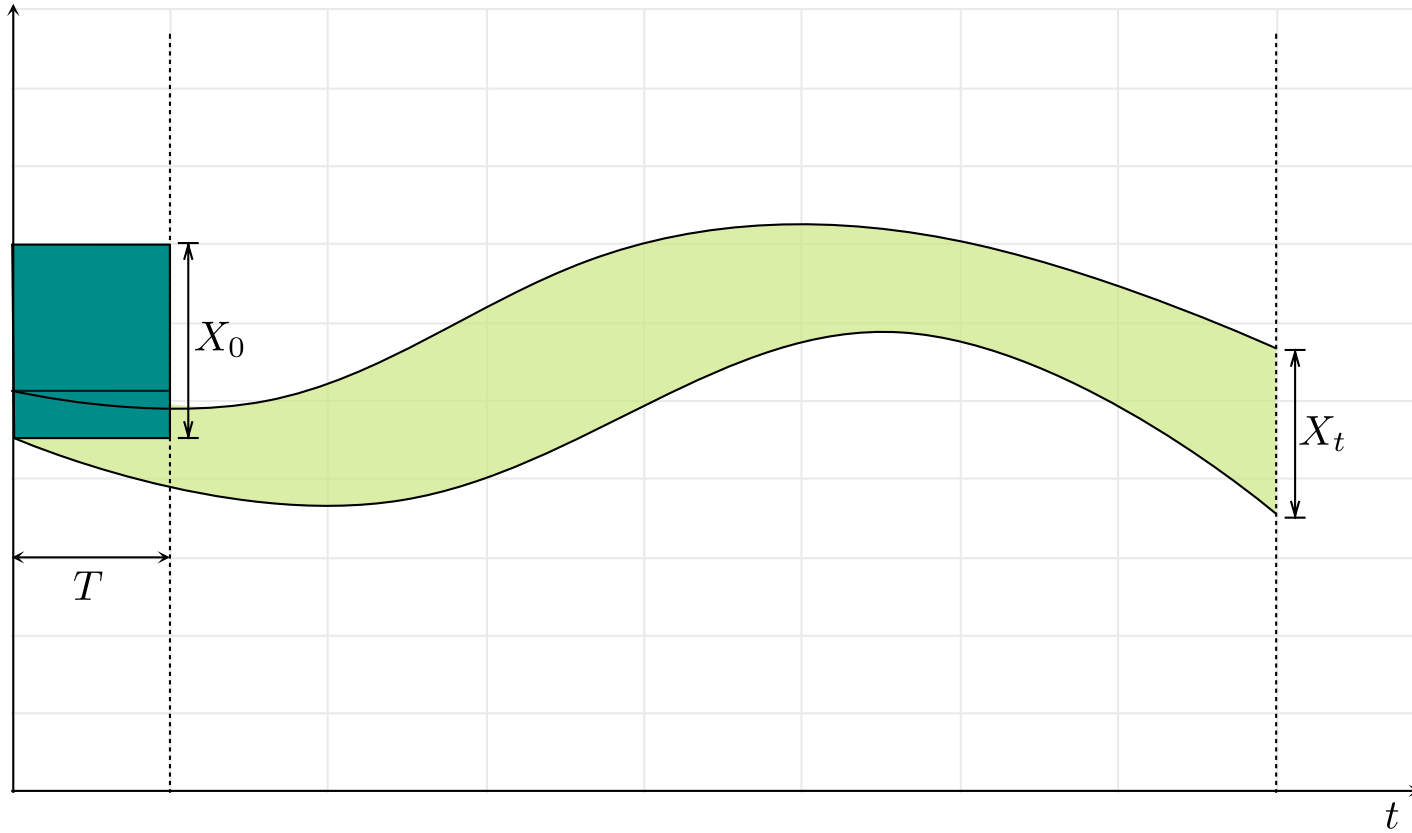
Forward Pruning (on X_t)



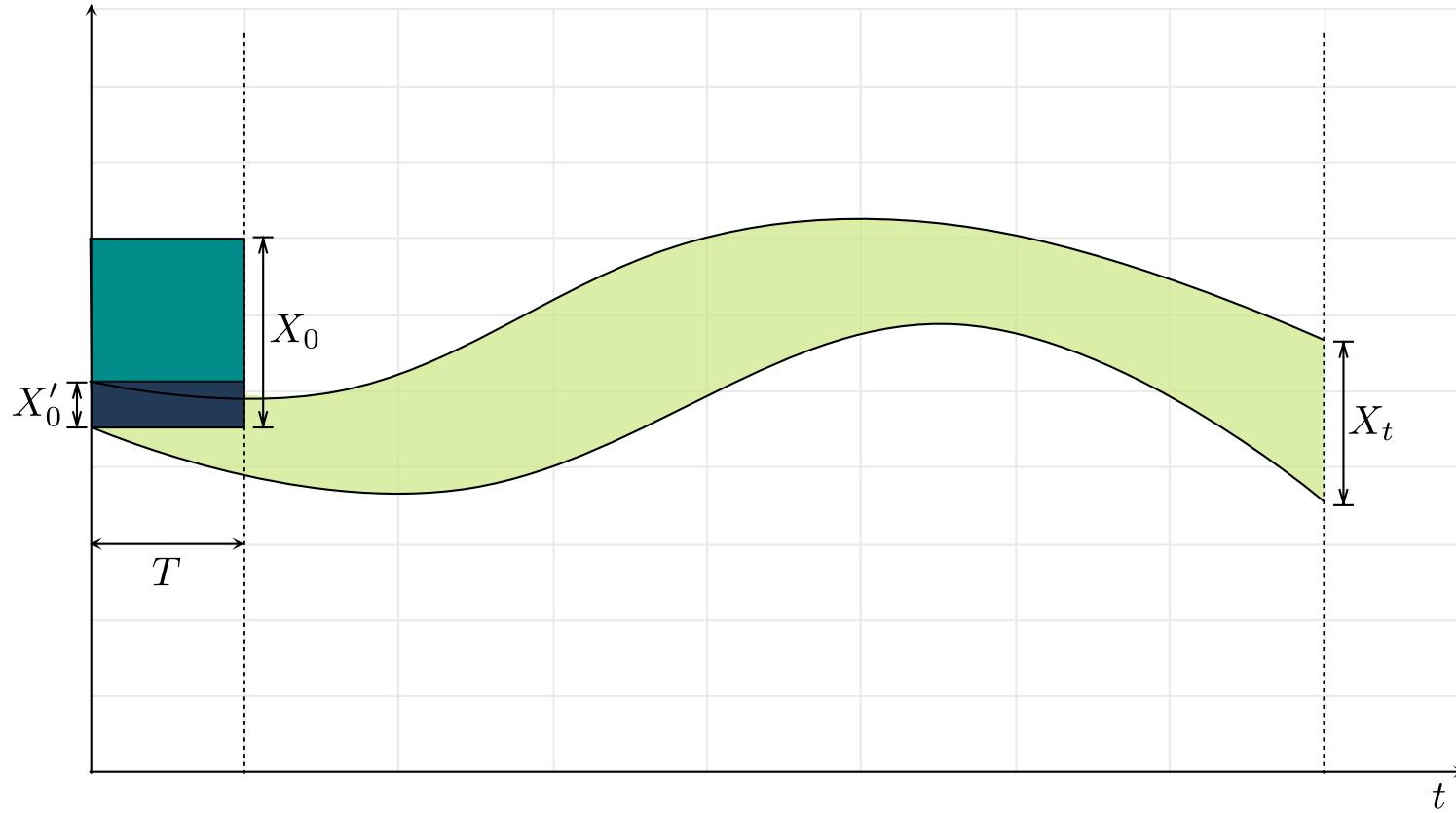
Backward Pruning (on X_0)



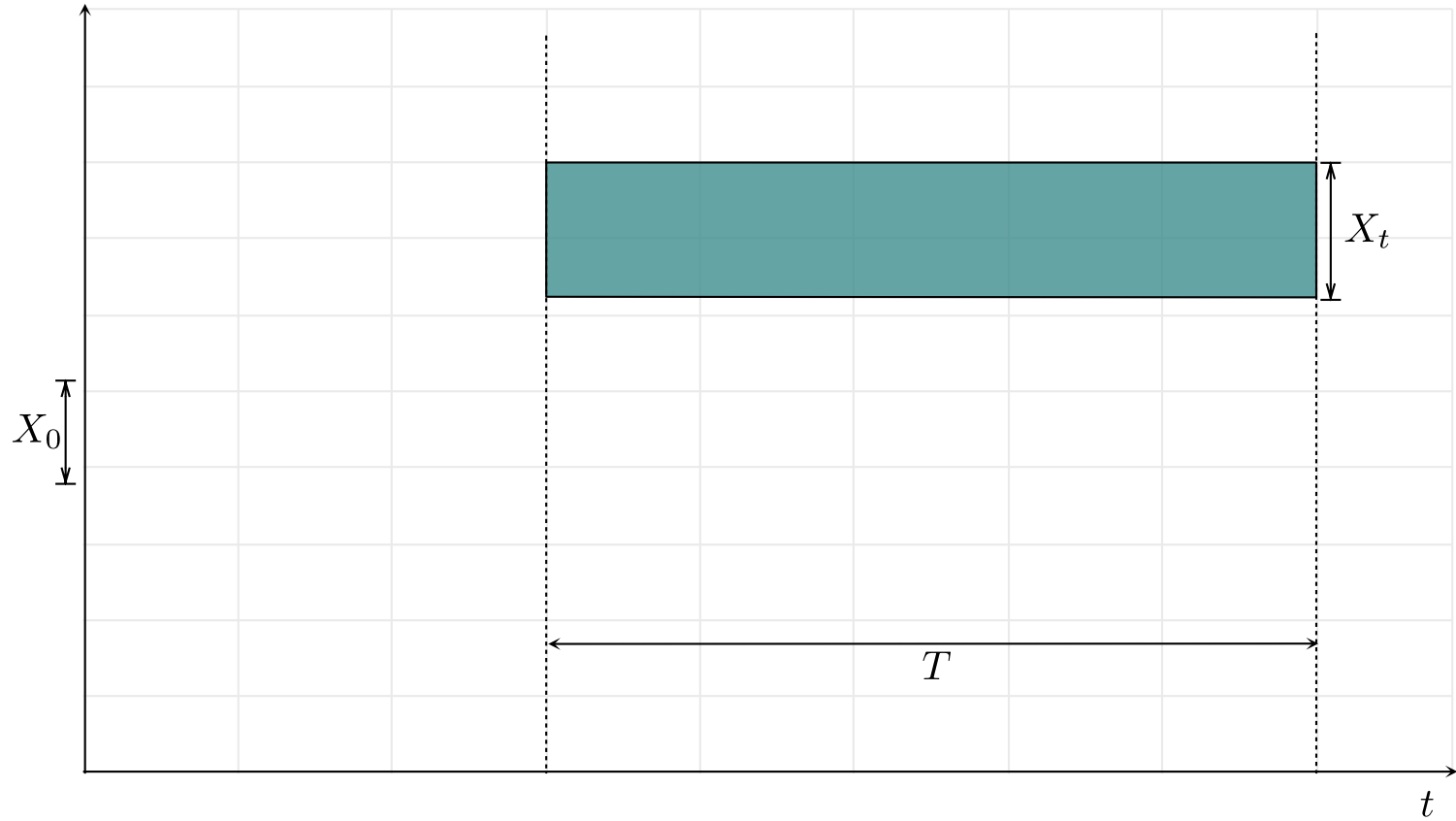
Backward Pruning (on X_0)



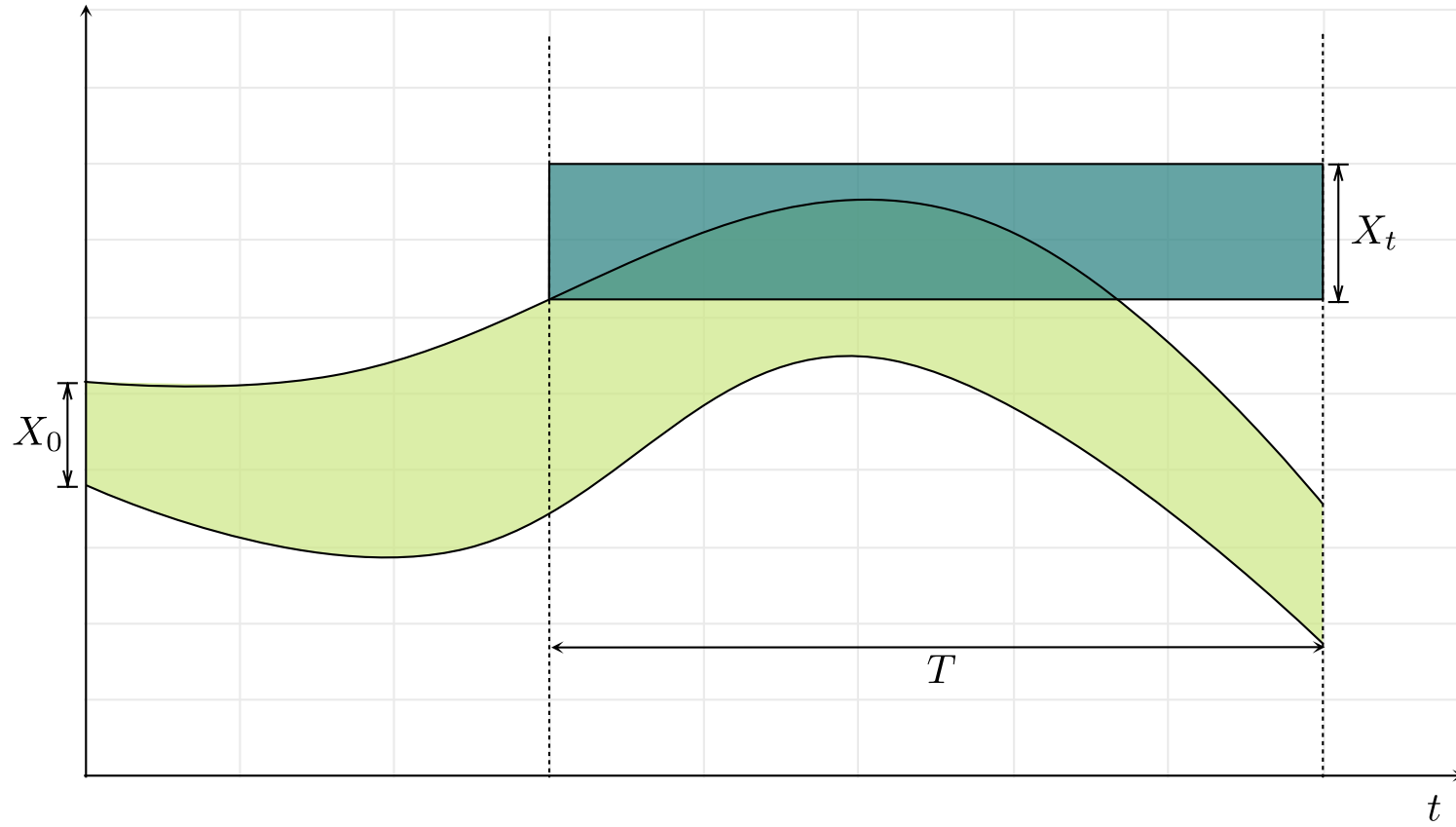
Backward Pruning (on X_0)



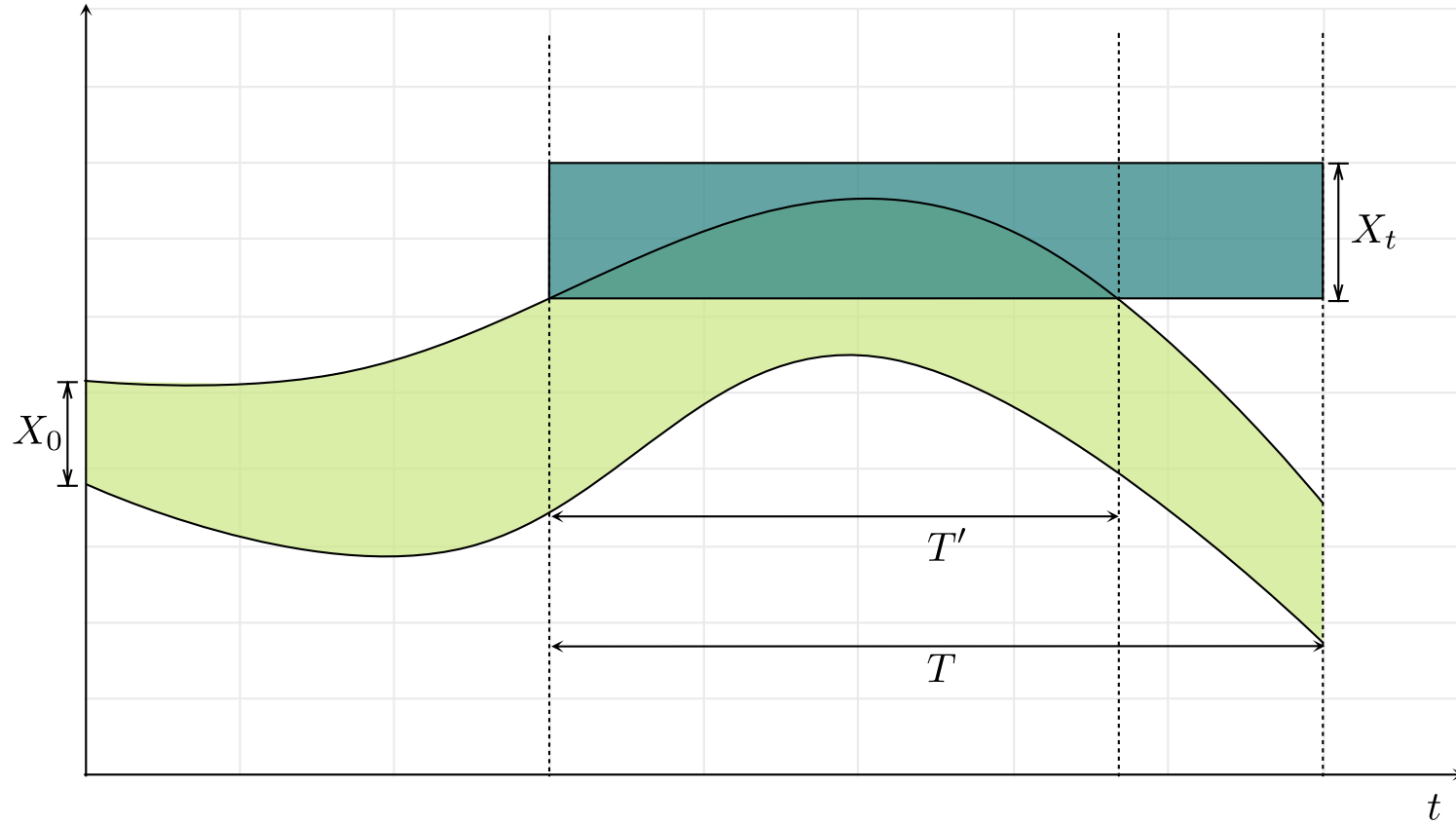
Time Pruning (on T)



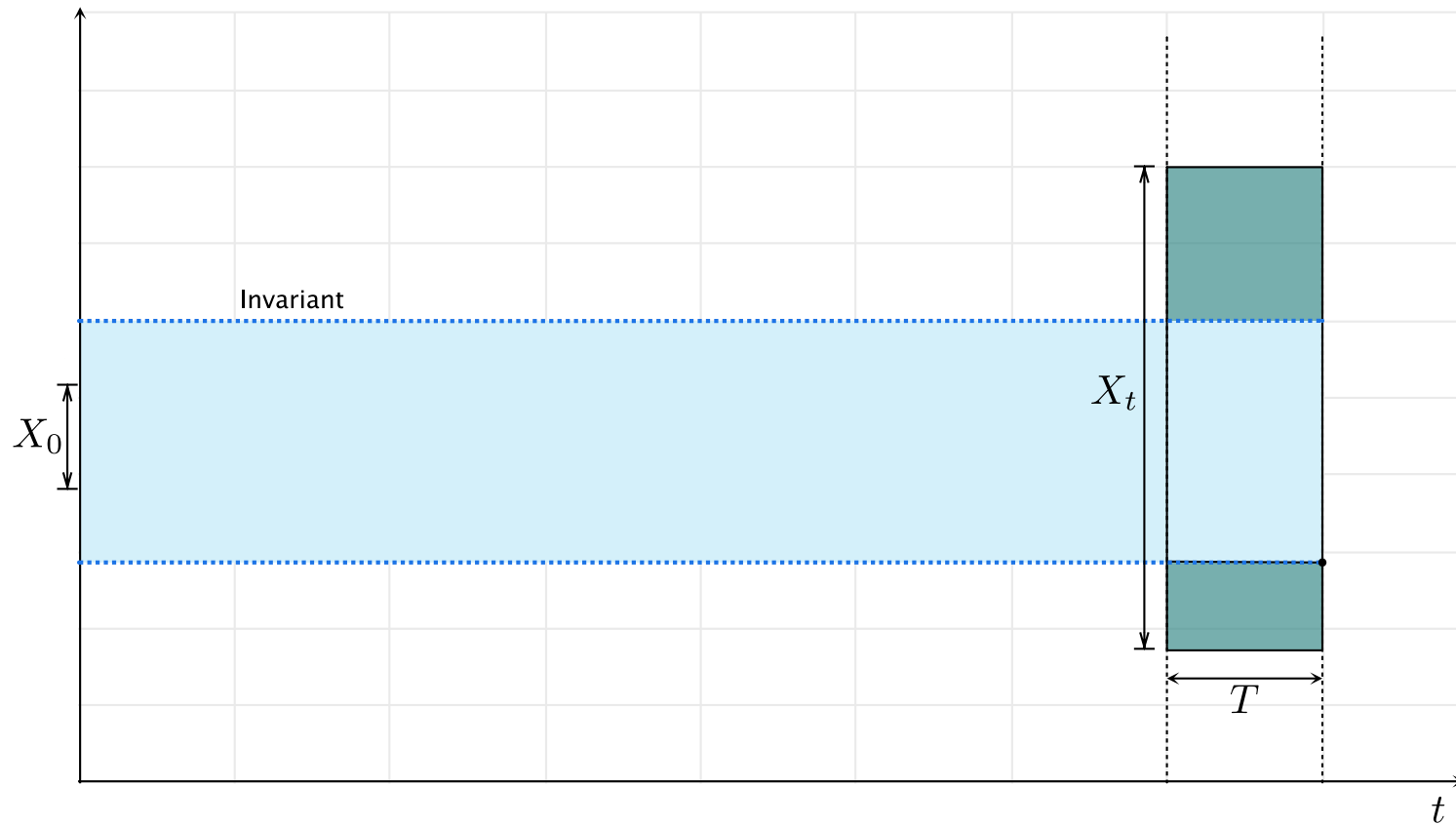
Time Pruning (on T)



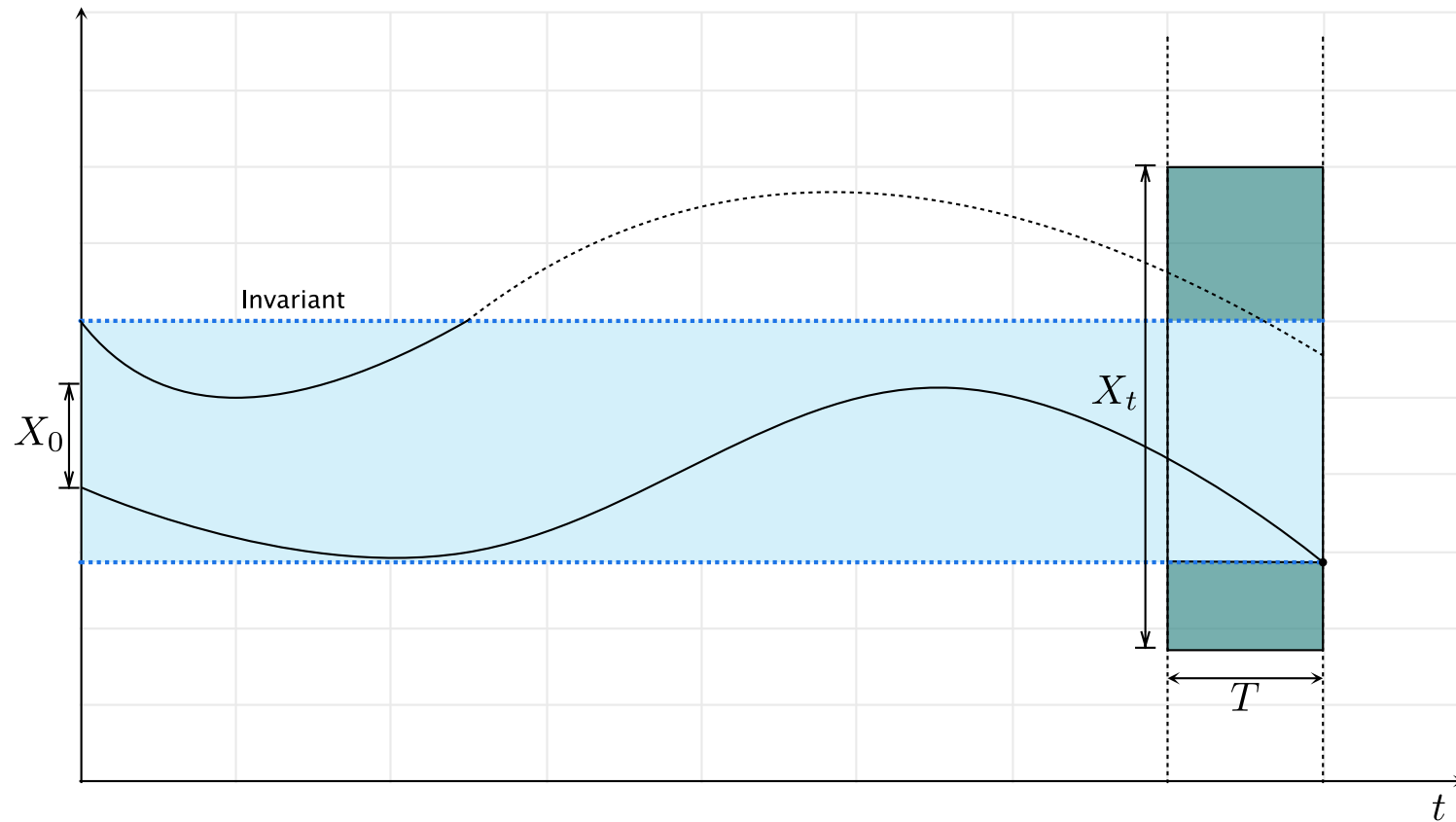
Time Pruning (on T)



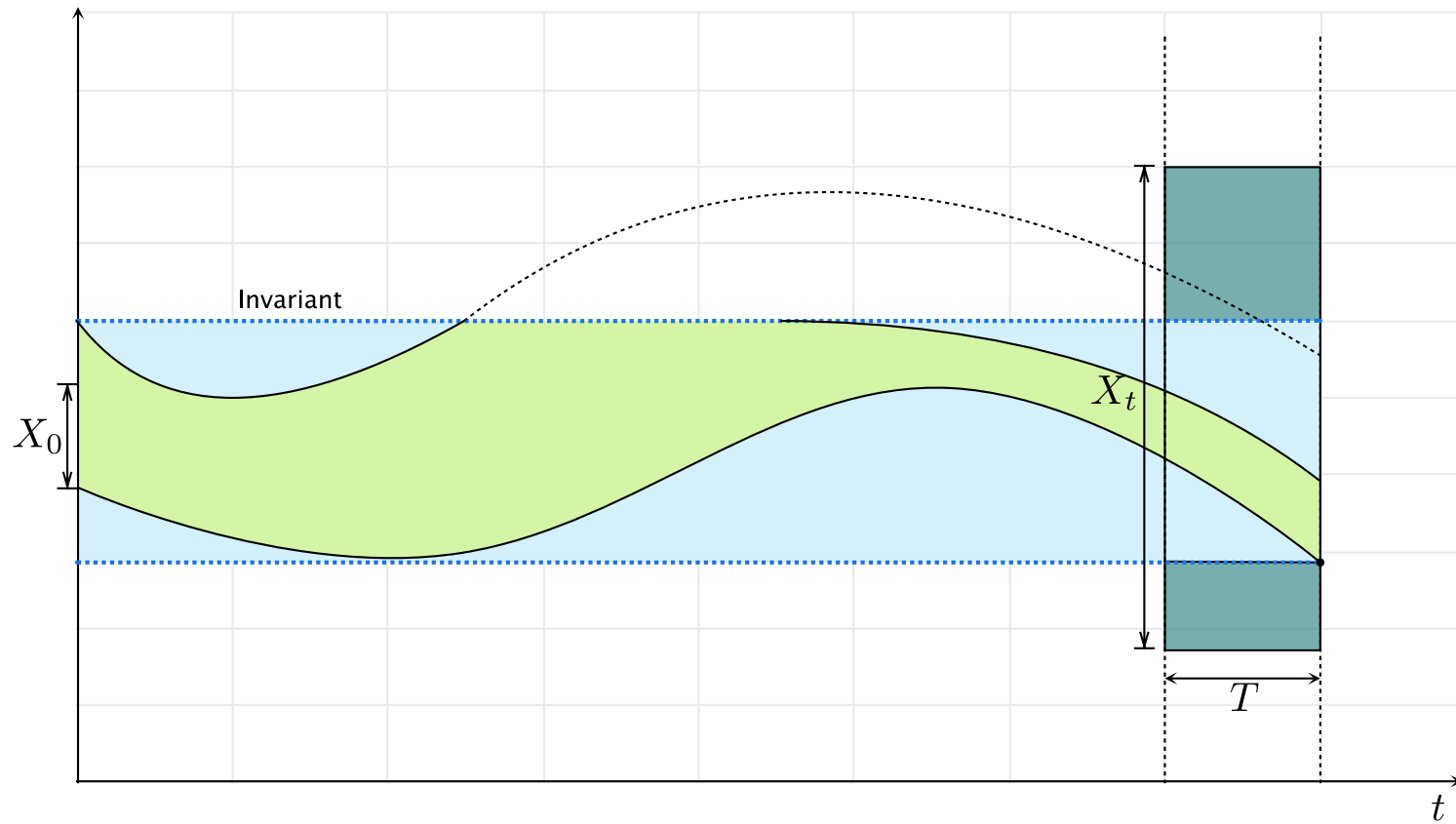
Pruning with Invariant



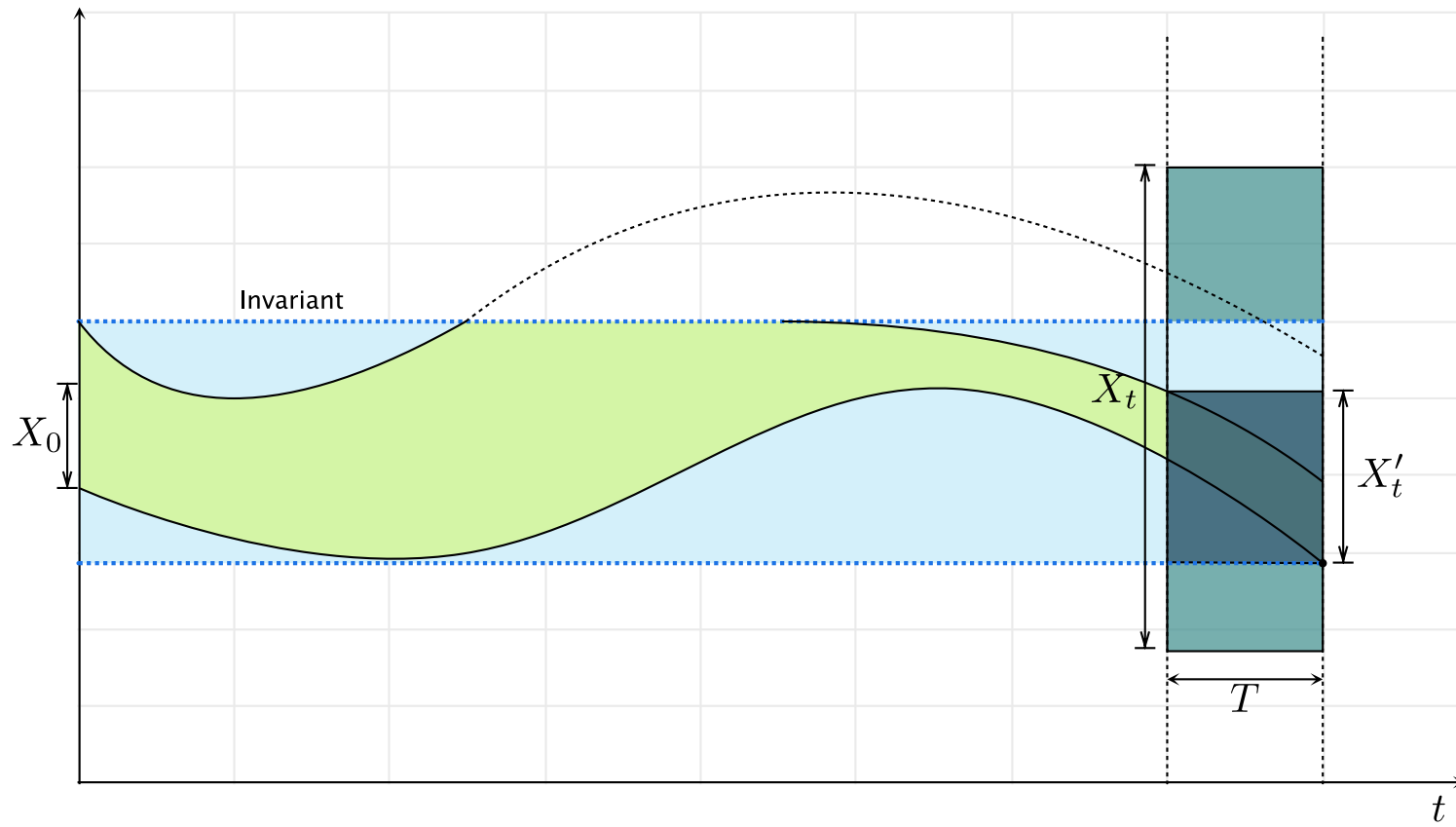
Pruning with Invariant



Pruning with Invariant

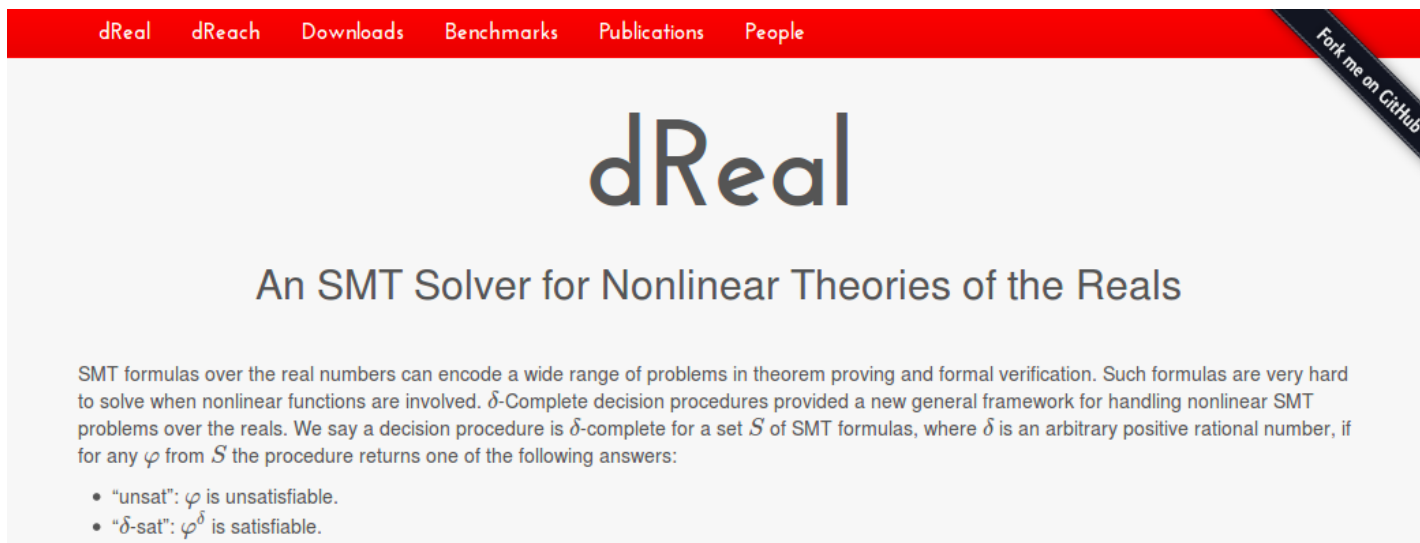


Pruning with Invariant



Tool

- Open-source at <http://dreal.cs.cmu.edu>
- Nonlinear ODEs, and of course, polynomials, transcendental functions, etc.
- Formulas with hundreds of nonlinear ODEs have been solved.



The screenshot shows the homepage of the dReal website. At the top, there is a red navigation bar with links for "dReal", "dReach", "Downloads", "Benchmarks", "Publications", and "People". On the right side of the navigation bar, there is a black ribbon with the text "Fork me on GitHub". The main content area has a light gray background. In the center, the word "dReal" is written in a large, bold, black font. Below it, the subtitle "An SMT Solver for Nonlinear Theories of the Reals" is displayed in a smaller, black font. Further down, there is a paragraph of text explaining the capabilities of the solver, followed by a bulleted list of two possible answers: "unsat" and " δ -sat".

dReal dReach Downloads Benchmarks Publications People

dReal

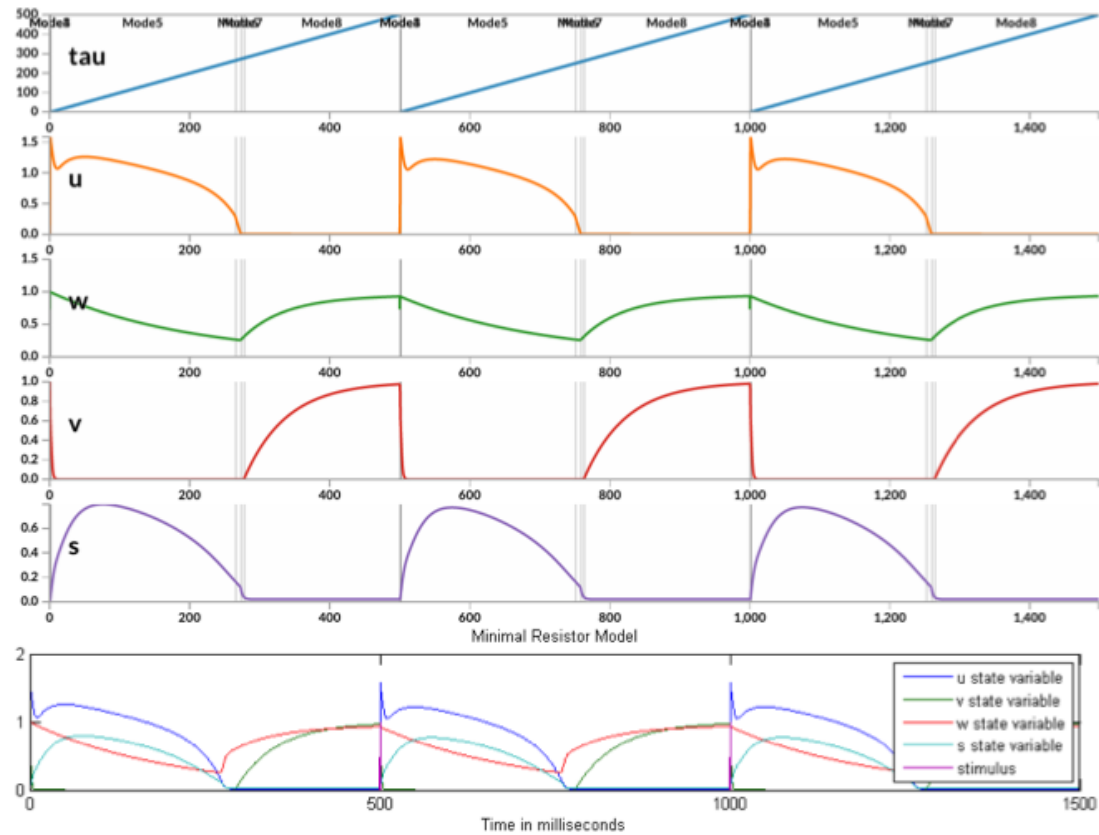
An SMT Solver for Nonlinear Theories of the Reals

SMT formulas over the real numbers can encode a wide range of problems in theorem proving and formal verification. Such formulas are very hard to solve when nonlinear functions are involved. δ -Complete decision procedures provided a new general framework for handling nonlinear SMT problems over the reals. We say a decision procedure is δ -complete for a set S of SMT formulas, where δ is an arbitrary positive rational number, if for any φ from S the procedure returns one of the following answers:

- "unsat": φ is unsatisfiable.
- " δ -sat": φ^δ is satisfiable.

Experiments

$$\begin{aligned} \frac{du}{dt} &= e + (u - \theta_v)(u_u - u)vgfi \\ &\quad + wsg_{si} - g_{s0}(u) \\ \frac{ds}{dt} &= \frac{g_{s2}}{(1 + e^{-2k(u-us)})} - g_{s2}s \\ \frac{dv}{dt} &= -g_v^+ \cdot v \\ \frac{dw}{dt} &= -g_w^+ \cdot w \end{aligned}$$



Experiments

```
(set-logic QF_NRA_ODE)
(declare-fun w_0_1_t () Real)
(declare-fun w_0_1_0 () Real)
(declare-fun w_1_2_t () Real)
(declare-fun w_1_2_0 () Real)
(declare-fun w_2_3_t () Real)
(declare-fun w_2_3_0 () Real)
(declare-fun w_3_4_t () Real)
(declare-fun w_3_4_0 () Real)
(declare-fun w_4_5_t () Real)
(declare-fun w_4_5_0 () Real)
(declare-fun w_5_6_t () Real)
(declare-fun w_5_6_0 () Real)
(declare-fun w_6_7_t () Real)
(declare-fun w_6_7_0 () Real)
(declare-fun w_7_8_t () Real)
(declare-fun w_7_8_0 () Real)
(declare-fun w_8_1_t () Real)
(declare-fun w_8_1_0 () Real)
(declare-fun w_9_2_t () Real)
(declare-fun w_9_2_0 () Real)
(declare-fun w_10_3_t () Real)
(declare-fun w_10_3_0 () Real)
```

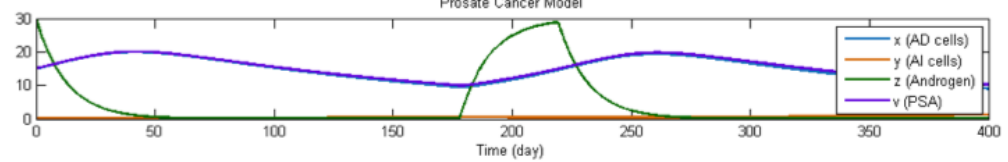
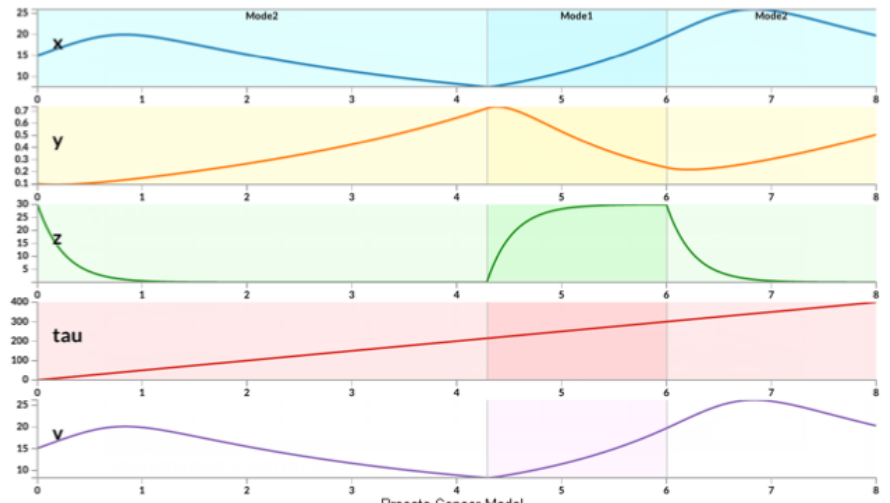
Experiments

$$\frac{dx}{dt} = \left(\alpha_x \left(k_1 + (1 - k_1) \frac{z}{z + k_2} \right) - \beta_x \left((1 - k_3) \frac{z}{z + k_4} + k_3 \right) - m_1 \left(1 - \frac{z}{z_0} \right) \right) x + c_1 x$$

$$\frac{dy}{dt} = m_1 \left(1 - \frac{z}{z_0} \right) x + \left(\alpha_y \left(1 - d \frac{z}{z_0} \right) - \beta_y \right) y + c_2 y$$

$$\frac{dz}{dt} = \frac{-z}{\tau} + c_3 z$$

$$\frac{dv}{dt} = \left(\alpha_x \left(k_1 + (1 - k_1) \frac{z}{z + k_2} \right) - \beta_x \left(k_3 + (1 - k_3) \frac{z}{z + k_4} \right) \right) - m_1 \left(1 - \frac{z}{z_0} \right) x + c_1 x + m_1 \left(1 - \frac{z}{z_0} \right) x + \left(\alpha_y \left(1 - d \frac{z}{z_0} \right) - \beta_y \right) y + c_2 y$$



Experiments

P	#M	#D	#O	#V	delta	R	Time(s)	Trace
AF	4	3	20	44	0.001	S	43.10	90K
AF	8	7	40	88	0.001	S	698.86	20M
AF	8	23	120	246	0.001	S	4528.13	59M
AF	8	31	160	352	0.001	S	8485.99	78M
AF	8	47	240	528	0.001	S	15740.41	117M
AF	8	55	280	616	0.001	S	19989.59	137M
CT	2	2	15	36	0.005	S	345.84	3.1M
CT	2	2	15	36	0.002	S	362.84	3.1M
EO	3	2	18	42	0.01	S	52.93	998K
EO	3	2	18	42	0.001	S	57.67	847K
EO	3	11	72	168	0.01	U	7.75	—
BB	2	10	22	66	0.01	S	0.25	123K
BB	2	20	42	126	0.01	S	0.57	171K
BB	2	20	42	126	0.001	S	2.21	168K
BB	2	40	82	246	0.01	U	0.27	—
BB	2	40	82	246	0.001	U	0.26	—
D1	3	2	9	24	0.1	S	30.84	72K
DU	3	2	6	16	0.1	U	0.04	—

TABLE I: Experimental results. #M = Number of modes in the hybrid system, #D = Unrolling depth, #O = Number of ODEs in the unrolled formula, #V = Number of variables in the unrolled formula, R = Bounded Model Checking Result (delta-SAT/UNSAT) Time = CPU time (s), Trace = Size of the ODE trajectory, AF = Atrial Filbrillation Model, CT = Cancer Treatment Model, EO = Electronic Oscillator Model, BB = Bouncing Ball with Drag Model, D1,DU = Decay Model.

Conclusion



Conclusion



This is not much harder than SAT solving.