HybridSAL: Tool for Analyzing Hybrid Systems

Using Relational Abstraction

Background: Few automated tools for verifying systems with mixed discrete and continuous dynamics, and none are compositional

Accomplishment: We have developed two new techniques for analyzing open components based on

- certificate-based techniques for generating assume-guarantee pairs
- relational abstractions

HybridSAL Supports Relational Abstraction

Progress: The HybridSAL tool can construct relational abstractions

HybridSAL Abstractor		HybridSAL		HSAL RelAbstractor
Constructs qualitative abstractions of HybridSAL models	$\Leftarrow$	Formal language for describing systems with hybrid dynamics	$\Rightarrow$	Constructs relational abstractions of HybridSAL models
Finite state	Old		New	Infinite state

HSAL Relational Abstractor

The use case:

1. User creates a HybridSAL model of the system/component of interest
   - Using a text editor
   - From Vanderbilt's CyPhy environment

   Model resides in filename.hsal
2. User adds properties of interest to the model

   Properties also go inside filename.hsal
3. HSal RelAbstractor automatically constructs filename.sal
4. Sal model checkers can be used to verify filename.sal

   ```
   sal-inf-bmc -i -d 5 filename property
   ```

These steps can be seen in the demo

HSAL Relational Abstractor

Is developed compositionally

Independently usable components of HSAL Relational Abstractor:

- hsal2hxml: A parser for HybridSAL, creates HSAL model in XML
- hxml2hsal: Pretty printer for HSAL XML
- hsal2hasal: HSal relational abstractor, from .hsal, or .hxml to .hasal
  The original model and its abstraction are both stored in .hasal file
  hsal2hxml can parse .hasal file
  hxml2hsal can also pretty print .haxml file
- hasal2sal: Extract the abstract SAL model from .hasal file

Key Idea: Enriched components, .hasal file stores components, properties, and abstractions

Relational Abstraction: Concept

Consider a dynamical system $(X, \rightarrow)$ where

X :variables defining state space of the system

$\rightarrow$:binary relation over state space defining system dynamics

We do not care if

- the system is discrete- or continuous- or hybrid-time, or
- the system has a discrete, continuous, or hybrid state space

For discrete-time systems, $\rightarrow$ is the one-step transition relation

For continuous-time systems, $\rightarrow = \cup_{t \geq 0} \xrightarrow{t}$ where $\xrightarrow{t}$ is the transition relation corresponding to an elapse of t time units

Relational Abstraction: Concept

Relational abstraction of a dynamical system $(X, \rightarrow)$ is another dynamical system $(X, \rightarrow)$ such that

$$\text{TransitiveClosure}(\rightarrow) \;\subseteq\; \rightarrow$$

Relational Abstraction: An over-approximation of the transitive closure of the transition relation

Benefit:

Eliminates need for iterative fixpoint computation

Useful for proving safety properties, and establishing conservative safety bounds

Relational Abstraction: Example

For the continuous-time continuous-space dynamical system:

$$\begin{aligned}\frac{dx}{dt} &= -x + y \\ \frac{dy}{dt} &= -x - y\end{aligned}$$

we have the following continuous-space discrete-time relational abstraction:

$$(x, y) \rightarrow (x', y') \quad := \quad \texttt{max}(|x|, |y|) \geq \texttt{max}(|x'|, |y'|)$$

If initially $x \in [0, 3]$, $y \in [-2, 2]$, then in any future time, x, y will remain in the range $[-3, 3]$

Relational Abstraction: Challenge

Is it possible to compute relational abstractions?

We do not want to abstract discrete-time transition relations, because model checkers (and static analyzers) can handle them (compute fixpoint)

Is it possible to compute relational abstractions of continuous-time dynamics?

Computing Relational Abstractions

We have an algorithm for computing relational abstractions of linear systems

Dynamics	Relational Abstraction
$\dot{x} = 1, \dot{y} = 1$	$x' - x = y' - y$
$\dot{x} = 2, \dot{y} = 3$	$(x' - x)/2 = (y' - y)/3$
$\dot{\vec{x}} = A\vec{x}$	$(0 \leq p' \leq p) \vee (0 \geq p' \geq p)$, where $p = \vec{c}^T\vec{x}$, $\vec{c}$ eigenvector of A^T corr. to negative eigenvalue
$\dot{\vec{x}} = A\vec{x} + \vec{b}$	$\ldots$

Why are such simple dynamics important?

Timed automata, Multirate automata, linear hybrid systems

Computing Relational Abstractions

For linear systems, we can use plenty of linear algebra to automatically generate relational abstractions

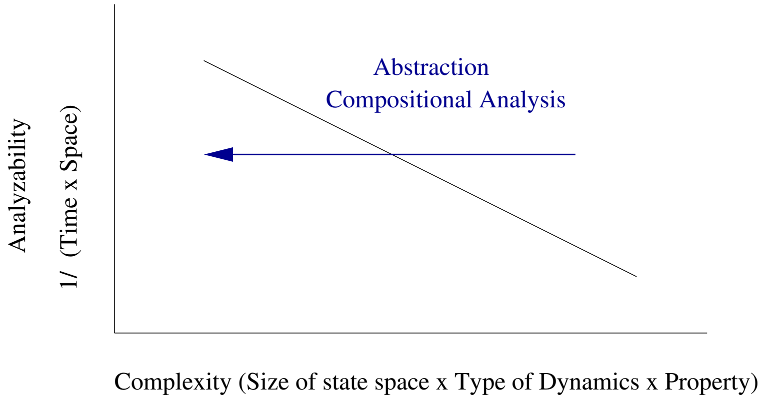
More generally, we can use the certificate-based approach to generate relational abstractions using constraint solving

By fixing a form for the relational abstraction, we can find the abstraction by solving an $\exists\forall$ formula

The algorithm for creating relational abstractions of linear systems can be viewed as a special case of this generic method, where the $\exists\forall$ problems are being solved using linear algebra tricks.

Relational Abstraction: Summary

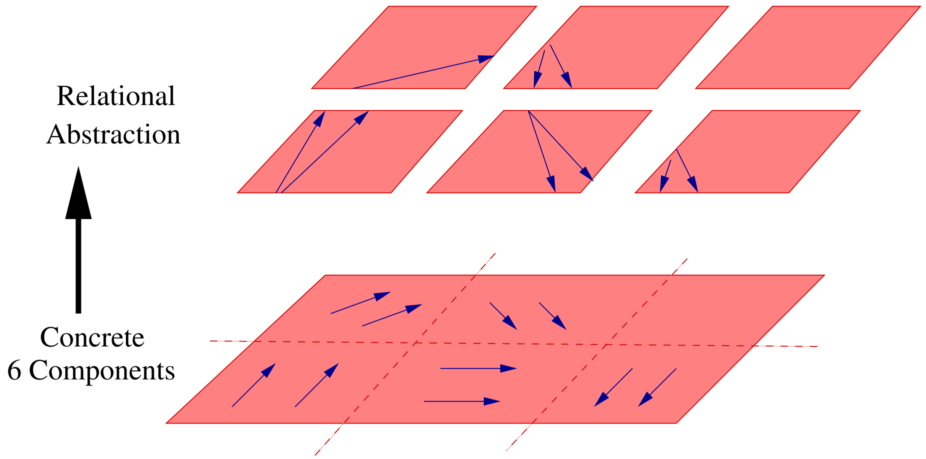
Benefit: Enables analyzability of complex systems



Feature: Compositional analysis: Abstracts open components with hybrid dynamics

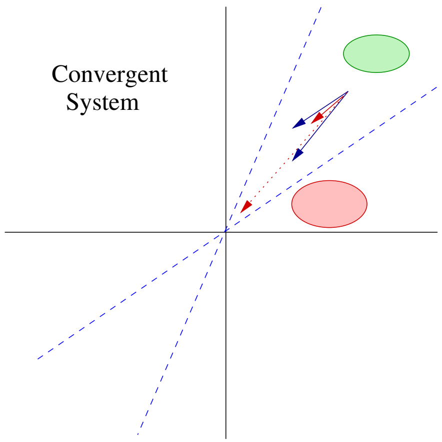
Feature: Compatible with other abstraction and model checking techniques

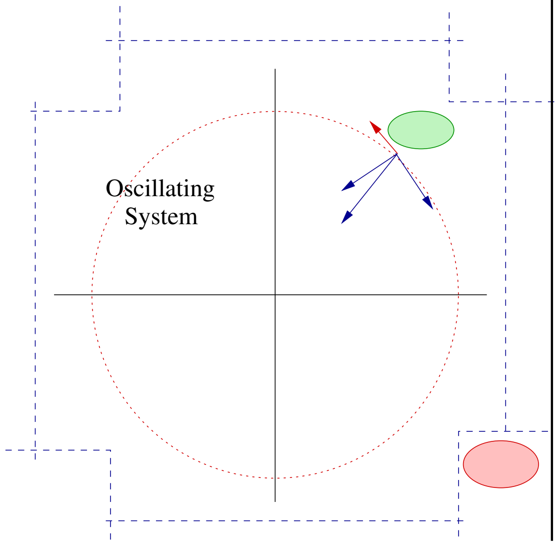
Novelty: Abstracts the transition relation, not the state space

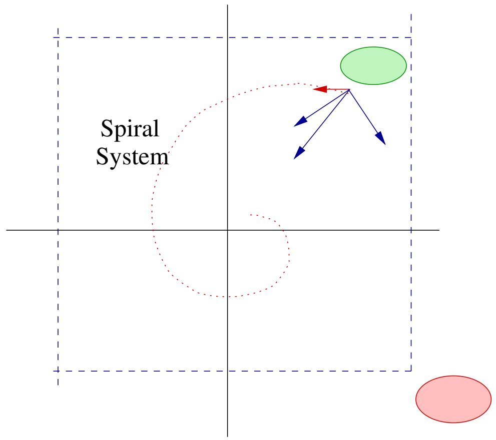


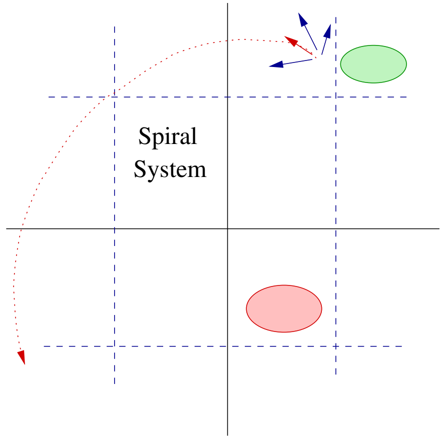
Scope: Applies to all dynamical systems. Effective relational abstractions can be computed for several classes.

Relational Abstraction: Examples







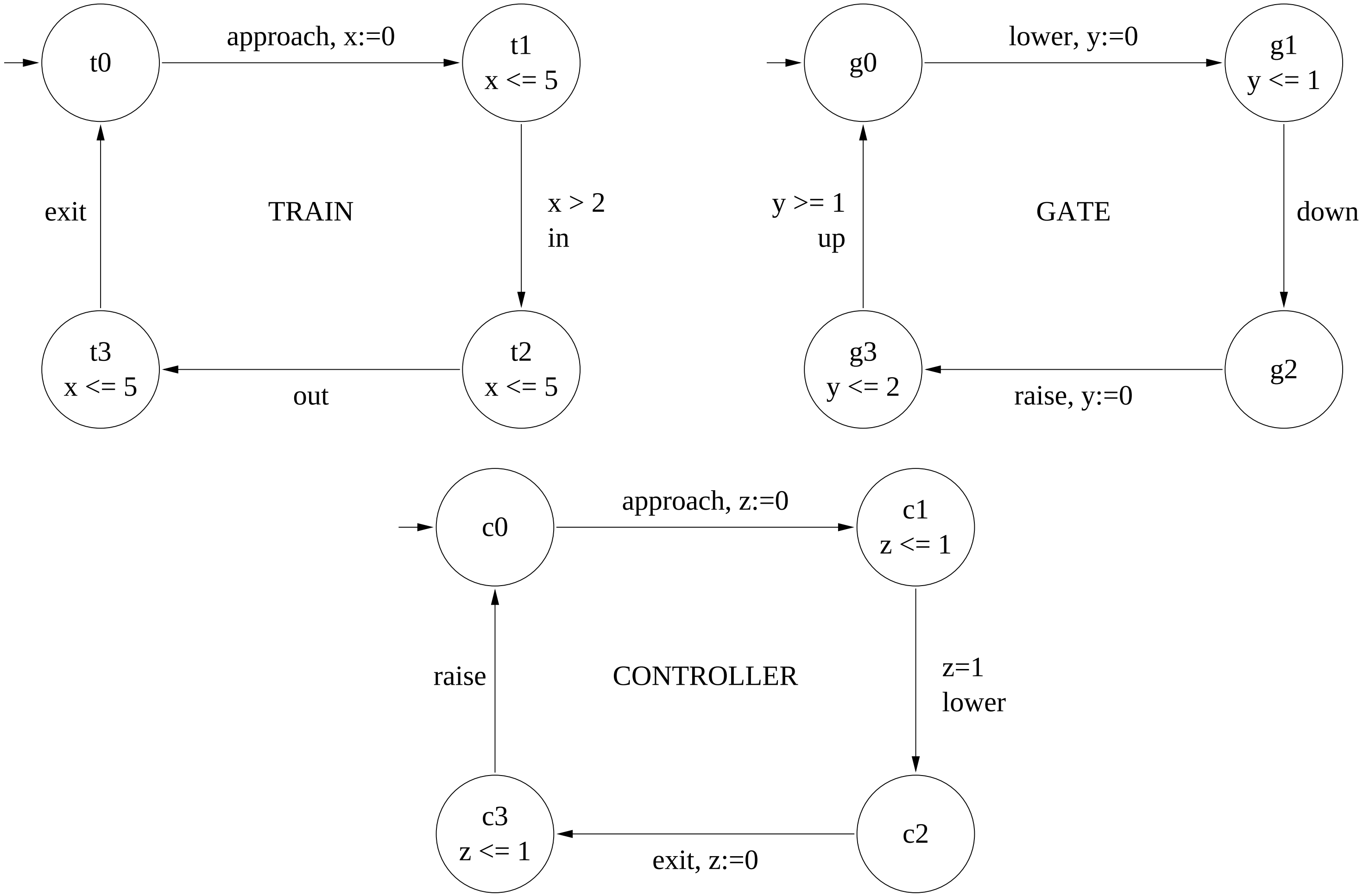


Class	$\frac{d\vec{x}}{dt}$	RelAbs
Timed System	$\dot{x} = 1$, $\dot{y} = 1$	$x' - x = y' - y$
Multirate System	$\dot{x} = 2$, $\dot{y} = 3$	$\frac{x'-x}{2} = \frac{y'-y}{3}$
Linear Hybrid System	$\dot{\vec{x}} = A\vec{x}$	$(0 \leq p' \leq p$
...	...	...

On Hybrid System benchmarks, verification time reduces from 10 hours to a few minutes (100x improvement).

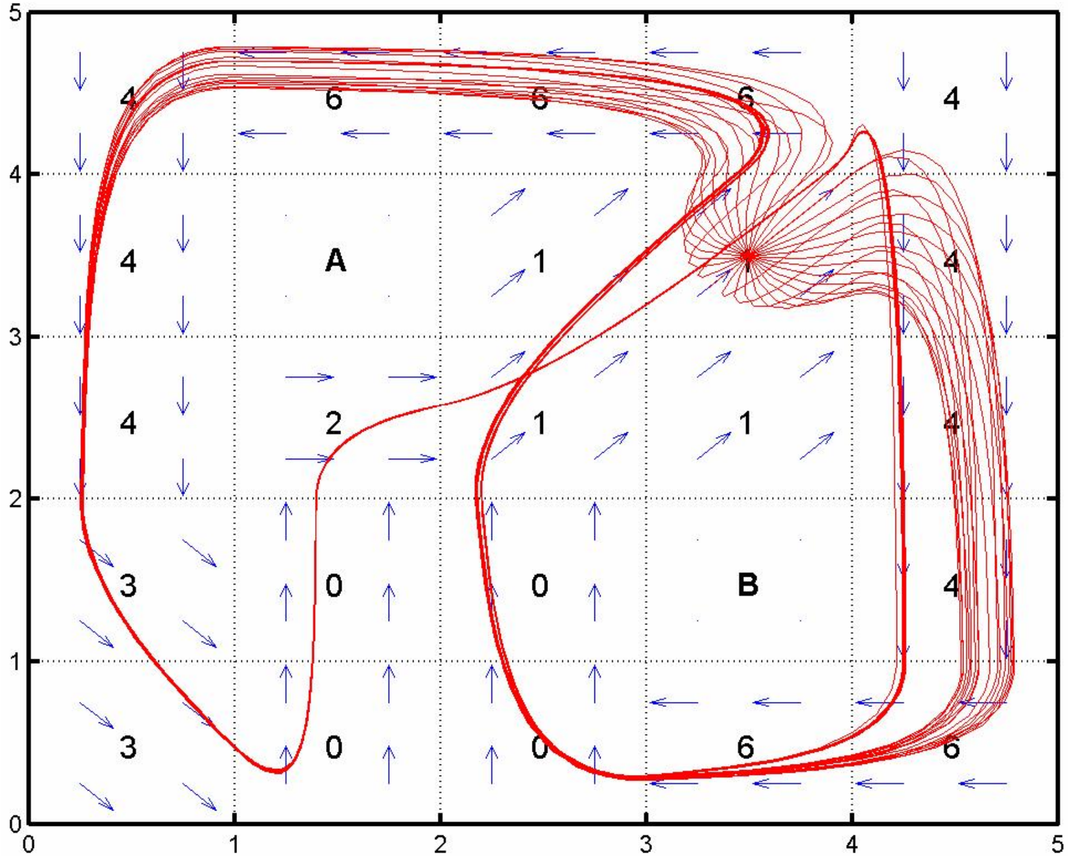
Demo: TGC Example

Consider a train-gate-controller system: Is it safe? From [Dutertre and Sorea, 2004]



Demo: Navigation Example

Consider a robot moving in a 2d space.



It should reach A, while avoiding B.

Dynamics:

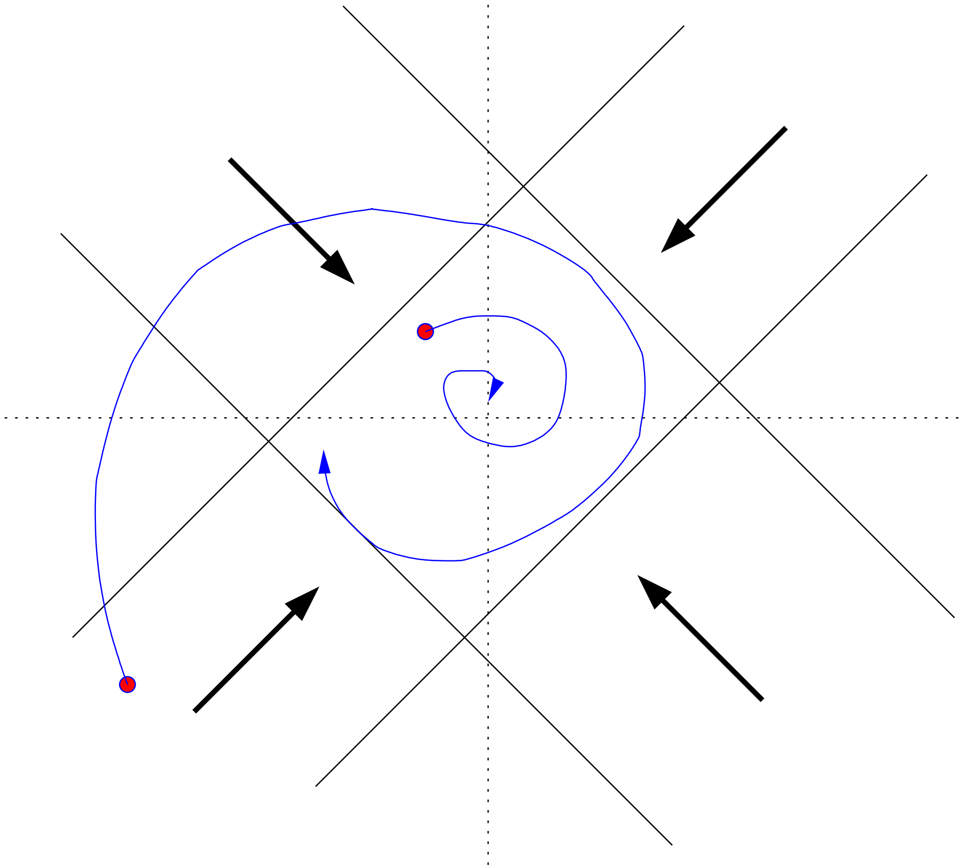
$$\begin{aligned} \dot{\vec{x}} &= \vec{v} \\ \dot{\vec{v}} &= A(\vec{v} - \vec{v}_d) \end{aligned}$$

The direction $\vec{v}_d$ depends on the position in the grid

Can verify instances in minutes using HSAL RelAbs and sal-inf-bmc

From [Ansgar and Ivancic, 2004]

Backup: Abstraction vs RelAbstraction



Two methods for abstracting continuous/hybrid systems

- predicate abstraction: Implemented in Hybrid-SAL
- relational abstraction: New approach that we will demonstrate here

Abstraction for Open Systems

Relational Abstraction

Abstract model defines how the input relates to the output

$$\frac{d\vec{x}}{dt} = f(\vec{x}) \tag{1}$$

$$\Downarrow \tag{2}$$

$$\vec{x} \rightarrow \vec{y} \quad \text{if} \quad \vec{x}, \vec{y} \text{ are related by } R(\vec{x}, \vec{y}) \tag{3}$$

Example:

$$\frac{dx}{dt} = -x \tag{4}$$

$$\Downarrow \tag{5}$$

$$\vec{x} \rightarrow \vec{y} \quad \text{if} \quad (x \leq y < 0) \vee (0 < y \leq x) \tag{6}$$

Computing Relational Abstractions

Suppose dynamics are $\frac{d\vec{x}}{dt} = A\vec{x}$

- Compute left eigenvector $\vec{c}^T$ of A

$$\vec{c}^T A = \lambda \vec{c}^T$$

- Note that

$$\frac{d(\vec{c}^T \vec{x})}{dt} = \vec{c}^T \frac{d\vec{x}}{dt} = \vec{c}^T A\vec{x} = \lambda \vec{c}^T \vec{x}$$

- Thus, we can relate the initial value of $c^T \vec{x}$ and its future value $c^T \vec{x}'$ as follows:

$$0 < \vec{c}^T \vec{x}' \leq \vec{c}^T \vec{x} \ \vee \ 0 > \vec{c}^T \vec{x}' \geq \vec{c}^T \vec{x}$$

  if $\lambda < 0$. And if $\lambda > 0$, then $\vec{x}, \vec{x}'$ swap places.

This idea generalizes to $\frac{d\vec{x}}{dt} = A\vec{x} + \vec{b}$

Computing Relational Abstractions 2

Suppose dynamics are $\frac{d\vec{x}}{dt} = A\vec{x}$

Suppose we have generated relations for all real eigenvalues

Now suppose there is a complex eigenvalue $a + b\iota$

- Find two vectors $\vec{c}^T$ and $\vec{d}^T$ such that

$$\begin{pmatrix} \frac{d\vec{c}^T\vec{x}}{dt} \\ \frac{d\vec{d}^T\vec{x}}{dt} \end{pmatrix} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} \frac{d\vec{c}^T\vec{x}}{dt} \\ \frac{d\vec{d}^T\vec{x}}{dt} \end{pmatrix}$$

- Thus, the values of $\vec{c}^T\vec{x}$ and $\vec{d}^T\vec{x}$ spiral in (or spiral out) if $a < 0$ (respectively if $a > 0$)

- Hence, we can relate their initial values to their future values

$$(\vec{c}^T\vec{x})^2 + (\vec{d}^T\vec{x})^2 \geq (\vec{c}^T\vec{x}')^2 + (\vec{d}^T\vec{x}')^2$$

  if $a < 0$, and the inequalities are reversed if $a > 0$

Qualitative vs Relational Abstraction

Consider $\dot{x} = -x$

Qualitative abstraction:

if $qx = pos$ then $qx' \in \{pos, zero\}$

if $qx = neg$ then $qx' \in \{neg, zero\}$

if $qx = zero$ then $qx' = zero$

Relational abstraction:

$0 \leq x' \leq x \;\vee\; 0 \geq x' \geq x$

If initially $x = 5$, then qualitative abstraction can prove x is never neg

If initially $x = 5$, then relational abstraction can prove x remains between 0 and 5

Demo of (Prototype) Timed Relational Abstraction

New: First version of Timed Relational Abstraction

Why TRA?

- A controller is designed, and verified for stability, in the continuous domain
- The controller is implemented on a time triggered architecture
- Is the system still stable?

What is TRA? Abstraction of $\frac{dx(t)}{dt} = f(x)$ by a relation $R(x(0), x(T))$ that relates all possible pairs $x(0), x(T)$, where T is the sampling period

Example 1: Timed Proportional Controller

Consider plant $\frac{dx}{dt} = 5 * x + u$

Consider a P-controller $u = -30 * x$

This is clearly stabilizing in the continuous domain.

Time-triggered implementation of this controller need not be provably stabilizing.

When $T = 0.01$, the controller is still stabilizing

When $T = 0.1$, it is not so

Example 2: Timed PI Controller

Consider plant $\frac{dx}{dt} = 5 * x + u$

Consider a PI-controller $u = -30 * x - y$, where $\frac{dy}{dt} = x$

When $T = 0.05$, the controller is stabilizing

When $T = 0.1$, the controller is stabilizing

When $T = 0.5$, it is not so

Example 3: Inverted Pendulum

$$\text{A linearized plant} \quad := \quad \left\{ \begin{array}{rcl} \frac{dx}{dt} & = & y \\ \frac{dy}{dt} & = & 20 * x + 16 * y + 4 * u \end{array} \right.$$

$$\text{A controller} \quad := \quad \left\{ \begin{array}{rcl} y \geq 2 & \longrightarrow & u = -16 \\ y \leq -2 & \longrightarrow & u = 16 \\ 16 * x - y \leq -10 & \longrightarrow & u = -16 \\ 16 * x - y \geq 10 & \longrightarrow & u = 16 \\ \text{Else} & \longrightarrow & u = u \end{array} \right.$$

A continuous controller results in a stable system

For any sampling period T, resulting system is not stable