

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 Abtractor | | HybridSAL | | HSAL RelAbtractor |
|--|--------------|--|---------------|--|
| Constructs quali- tative 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 Abtractor

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 RelAbtractor** 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 Abtractor

Is developed **compositionally**

Independently usable **components of HSAL Relational Abtractor**:

- **hsal2html**: A parser for HybridSAL, creates HSAL model in XML
- **html2hsal**: Pretty printer for HSAL XML
- **hsal2hasal**: HSAL relational abtractor, from .hsal, or .html to .hasal
The original model and its abstraction are **both** stored in .hasal file
hsal2html can parse .hasal file
html2hsal can also pretty print .html 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, $\rightarrow = \bigcup_{t \geq 0} \xrightarrow{t}$ 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:

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

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 |
| $\dot{\vec{x}} = A\vec{x} + \vec{b}$ | ... |

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

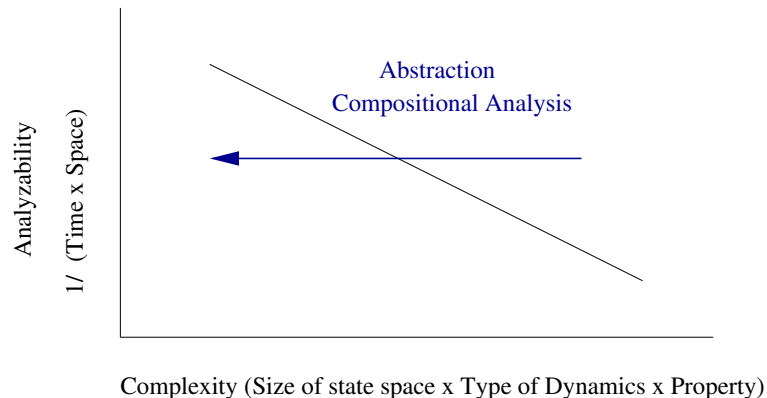
More generally, we can use the **certificate-based** approach to generate relational abstractions using **constraint solving**

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

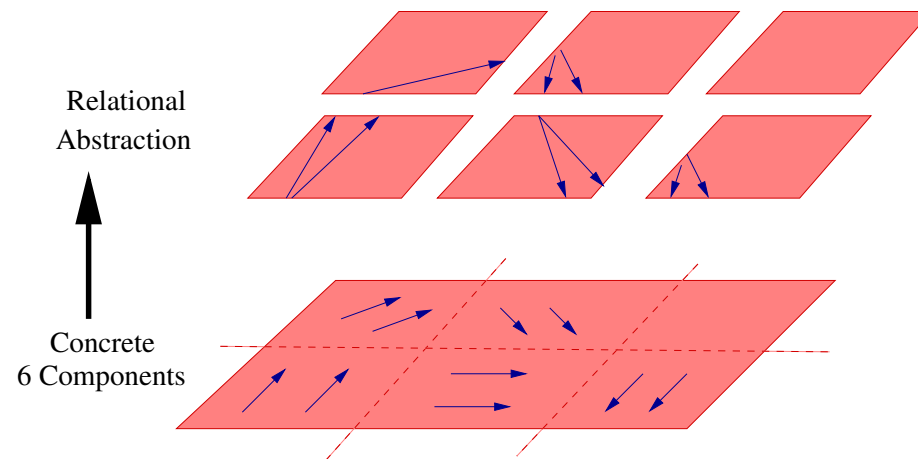
Benefit: Enables analyzability of complex systems



Feature: Compositional analysis: Abstracts open components with hybrid dynamics

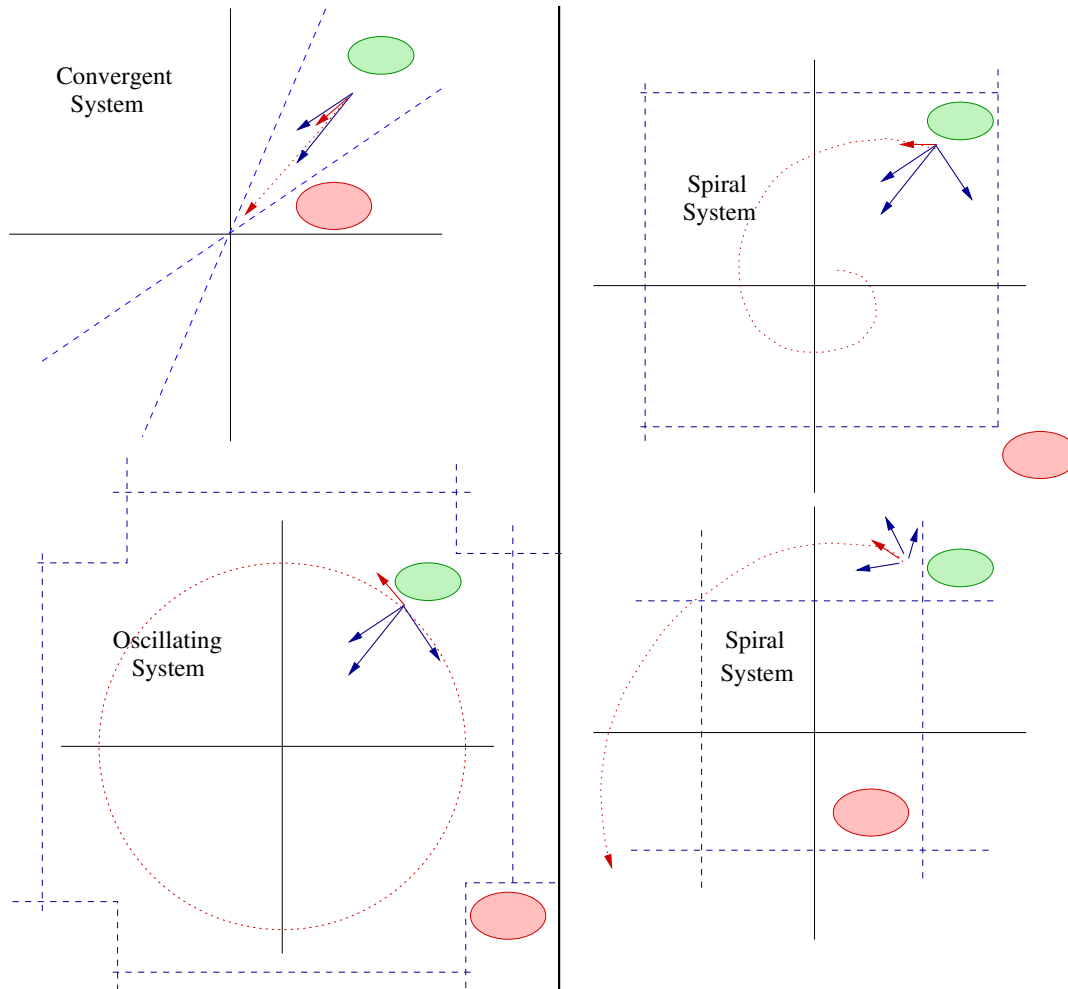
Feature: Compatible with other abstraction and model checking techniques

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

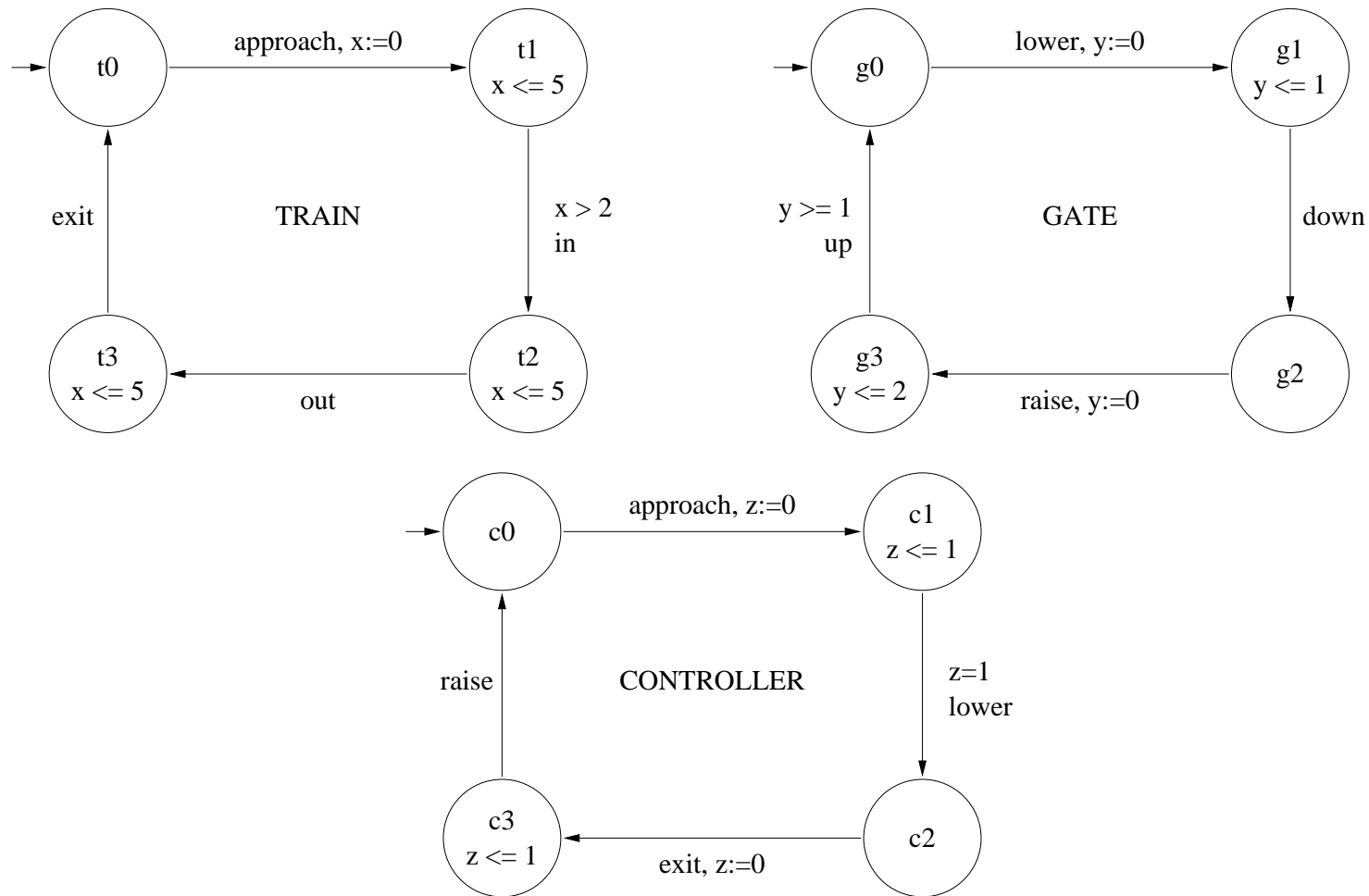


| Class | $\frac{d\vec{x}}{dt}$ | RelAbs |
|----------------------|---------------------------------|--|
| Timed System | $\dot{x} = 1,$ $\dot{y} = 1$ | $x' - x =$ $y' - y$ |
| Multirate System | $\dot{x} = 2,$ $\dot{y} = 3$ | $\frac{x' - x}{2} =$ $\frac{y' - y}{3}$ |
| Linear Hybrid System | $\dot{\vec{x}}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

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

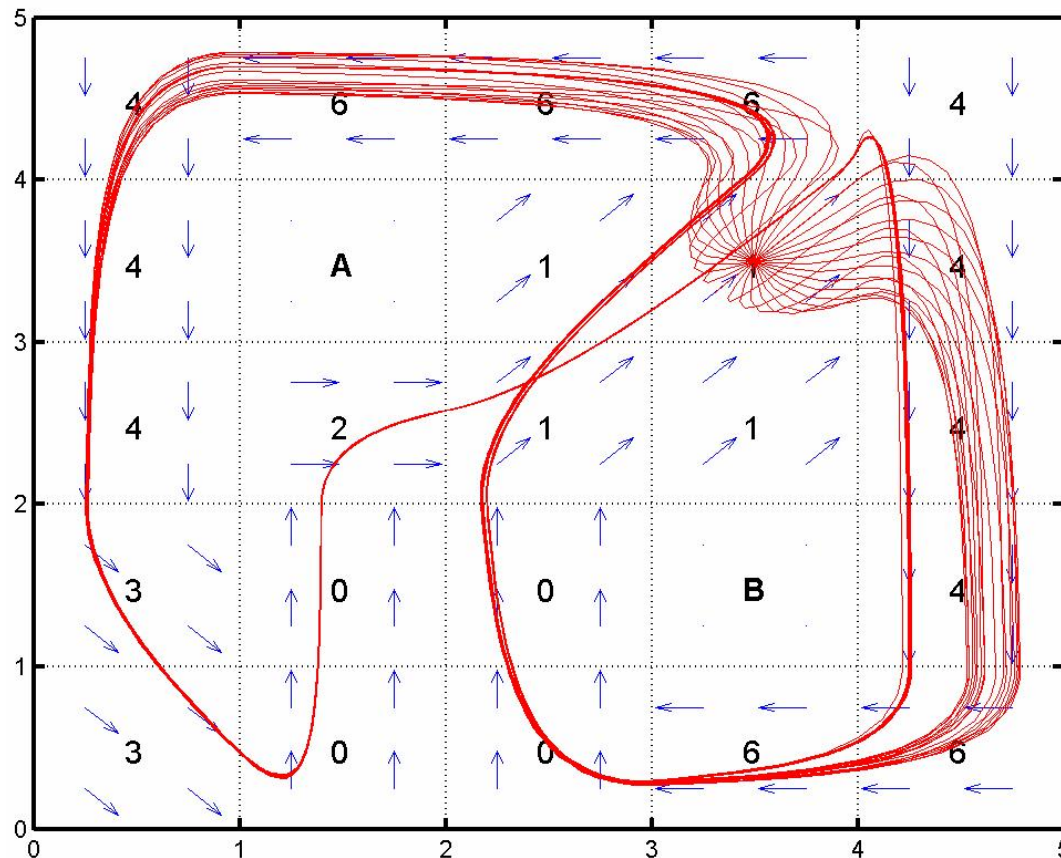
Demo: TGC Example

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:

$$\dot{\vec{x}} = \vec{v}$$

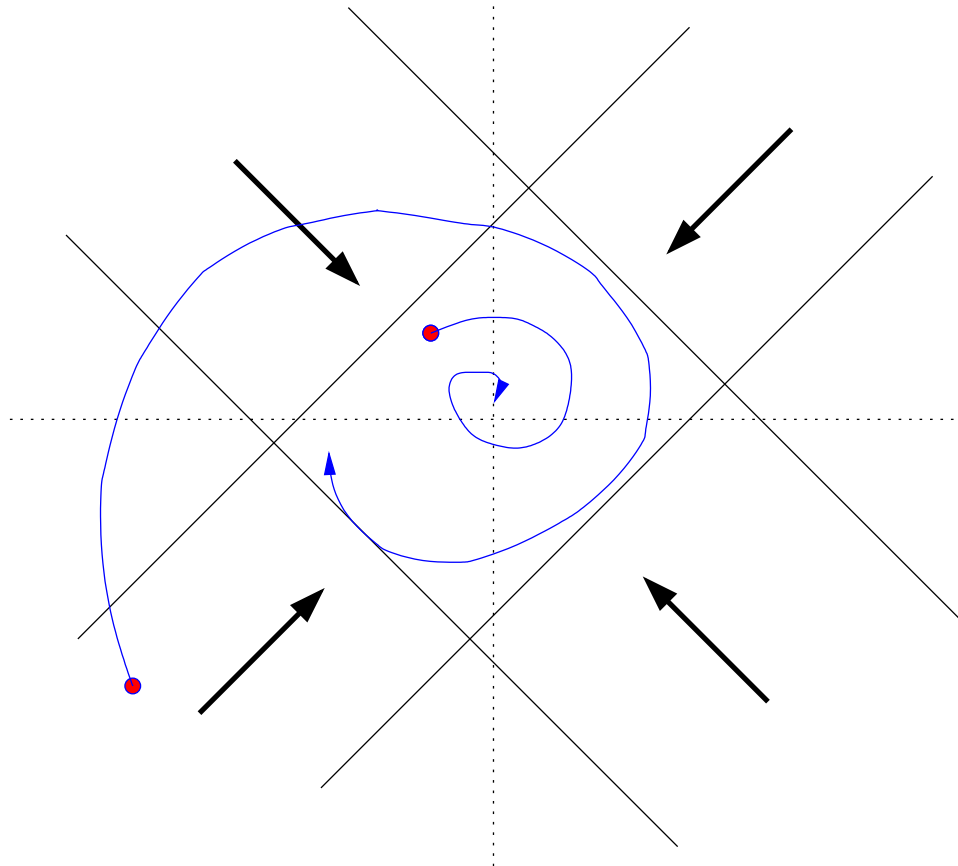
$$\dot{\vec{v}} = A(\vec{v} - \vec{v}_d)$$

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 HybridSAL
- **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**

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

$$\Downarrow \quad (2)$$

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

Example:

$$\frac{dx}{dt} = -x \quad (4)$$

$$\Downarrow \quad (5)$$

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