HybridSAL Relational Abstracter

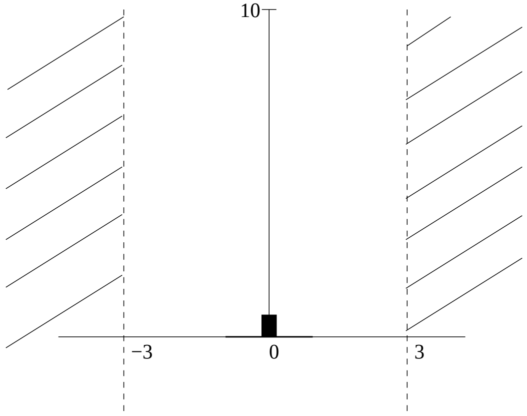
Ashish Tiwari

SRI International

Menlo Park

CA 94025

HybridSAL = SAL + ODEs



The goal is to prove that the robot remains inside `Safe` starting from `Init`:

$$
\begin{aligned}
\texttt{Init} &:= (x \in [-1,1],\ y = 0,\ v_x = 0,\ v_y = 0) \\
\texttt{Safe} &:= (|x| \leq 3)
\end{aligned}
$$

The robot can move in 2 modes:

- Mode 1: Force applied in $(1,1)$-direction (NE)

$$\frac{dx}{dt} = v_x, \quad \frac{dv_x}{dt} = 1.2(1-v_x)+0.1(v_y-1), \quad \frac{dy}{dt} = v_y, \quad \frac{dv_y}{dt} = 1.2(1-v_y)+0.1(v_x-1)$$

- Mode 2: Force applied in $(-1,1)$-direction (NW)

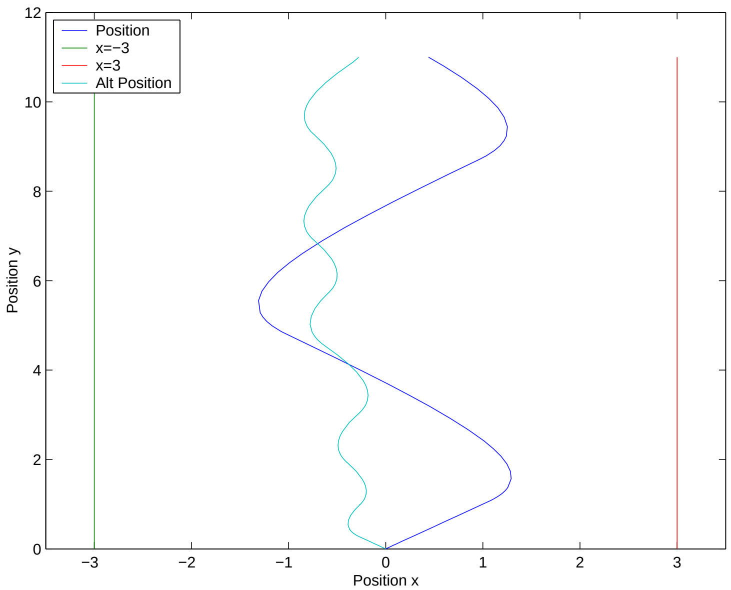
$$\frac{dx}{dt} = v_x, \quad \frac{dv_x}{dt} = -1.2(1+v_x)+0.1(v_y-1), \quad \frac{dy}{dt} = v_y, \quad \frac{dv_y}{dt} = 1.2(1-v_y)+0.1(v_x-1)$$

Example: Driving a Robot

Consider a non-deterministic controller:

- Switch to NE mode when moving left and $-1.5 \leq x \leq -1$
- Switch to NW mode when moving right and $1 \leq x \leq 1.5$

A possible simulation trajectory:



Does the robot ever hit the wall – for all initial states and switchings?

HybridSAL: Modeling the Plant

```
plant:  MODULE =
BEGIN
  INPUT direction :  BOOLEAN
  OUTPUT x, vx, y, vy :  REAL
  INITIALIZATION
    x IN {z:  REAL | -1 ≤ z AND z ≤ 1};
    vx = 0; vy = 0; y = 0
  TRANSITION
  [ direction = TRUE -->
      xdot' = vx; vxdot' = -12/10*(1 + vx) + 1/10*(vy - 1);
      ydot' = vy; vydot' = 12/10*(1 - vy) + 1/10*(vx + 1)
  [] direction = FALSE -->
      xdot' = vx; vxdot' = 12/10*(1 - vx) + 1/10*(vy - 1);
      ydot' = vy; vydot' = 12/10*(1 - vy) + 1/10*(vx - 1)
  ]
END;
```

HybridSAL: Modeling the Controller

```
controller: MODULE =
BEGIN
  OUTPUT direction, flag: BOOLEAN
  INPUT x, vx : REAL
  TRANSITION
  [ vx ≤ 0 AND vx' ≤ 0 AND x' ≤ −1 AND x' ≥ −3/2 -->
        direction' = FALSE
  [] vx ≥ 0 AND vx' ≥ 0 AND x' ≥ 1 AND x' ≤ 3/2 -->
        direction' = TRUE
  [] ...
  ]
END;
```

Note: the initial value of `direction` is unconstrained

HybridSAL: Modeling the System

```
robotnav:  CONTEXT
BEGIN
  plant:  MODULE = ...

  controller:  MODULE = ...

  system:  MODULE = plant || controller ;

  correct:  THEOREM
    system ⊢ G( -3 ≤ x AND x ≤ 3 );
END
```

Is the property `correct` true or false?

Demo: File examples/robotnav.hsal

HybridSAL Analysis

Verification of HybridSAL models is done in two steps:

Abstract:	filename.hsal	$\xrightarrow{\text{hsal2hasal}}$	filename.sal
Model Check:	filename.sal	$\xrightarrow{\text{sal-inf-bmc -i filename property}}$	Proved/Invalid

If Proved, then property is valid over the concrete system

If Invalid, then property may be false over the concrete system

If failed to prove and failed to find a CE, then property is most likely valid over the concrete system, but need to find an k-inductive invariant

Demo: `bin/hsal2hasal examples/robotnav.hsal`

Demo: File `examples/robotnav.sal`

HybridSAL to SAL

The HybridSal Relational Abstracter

- creates a discrete infinite-state abstraction
- does not abstract the state-space;
  only the ODE transitions are over-approximated by discrete transitions
  $\vec{x} \rightarrow \vec{x}'$ if there is a solution F of the ODE s.t. $F(0) = \vec{x}$ and $F(t) = \vec{x}'$ for some $t \geq 0$
- HybridSAL finds an over-approximation $\rightarrow$ without finding F
- completely automatic for linear ODEs

Relational Abstraction: Examples

continuous-time continuous-space concrete system	continuous-space discrete-time relational abstraction
$\dot{x} = 1, \dot{y} = 1$	$x' - x = y' - y \quad \wedge \quad y' \geq y$
$\dot{x} = 2, \dot{y} = 3$	$(x' - x)/2 = (y' - y)/3 \quad \wedge \quad y' \geq y$
$\frac{dx}{dt} = -x$	$x \geq x' > 0 \vee x \leq x' < 0 \vee x = x' = 0$
$\frac{dx}{dt} = -x + y$ $\frac{dy}{dt} = -x - y$	$\mathtt{max}(\lvert x\rvert, \lvert y\rvert) \geq \mathtt{max}(\lvert x'\rvert, \lvert y'\rvert) \wedge$ $x^2 + y^2 \geq {x'}^2 + {y'}^2$
$\frac{d\vec{x}}{dt} = A\vec{x}$	$(c^T\vec{x} \geq c^T\vec{x'} > 0 \;\vee$ $c^T\vec{x} \leq c^T\vec{x'} < 0 \;\vee$ $c^T\vec{x} = c^T\vec{x'} = 0) \quad \wedge \ldots$

WHY Relational Abstraction

Concept: Analyze hybrid systems by first replacing ODEs by their relational abstraction

Why is this a good idea?

- separation of concerns
  - use knowledge from control/system theory/linear algebra/Lyapunov functions/barriers to construct high-quality relationalizations of ODEs
  - then use verification techniques for infinite-state systems
- accuracy improves as we get closer to decidable classes
  - relationalization is lossless for timed automata, LHAs
  - almost lossless for other decidable classes of CDSs
- good quality abstractions automatically computed for linear ODEs
- generalizes to timed relational abstraction etc.

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

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

- For linear ODEs, both real and complex left eigenvectors yield high quality relational abstractions
- For nonlinear ODEs, there are generic methods that are not fully automated

Relational Abstraction: Definition

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:

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

$$\Downarrow \tag{5}$$

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

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:

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

  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 $a > 0$

HybridSAL: Old vs New

Old HybridSAL:

$$\text{HybridSAL} \overset{\texttt{QualitativeAbstraction}}{\Longrightarrow} \text{SAL}$$

Resulting SAL was finite-state model, could be model checked

New HybridSAL:

$$\text{HybridSAL} \overset{\texttt{RelationalAbstraction}}{\Longrightarrow} \text{SAL}$$

Resulting SAL is infinite-state model, can be infinite bounded model checked

Model Checking Relational Abstraction

The output of relational abstracter is an infinite-state SAL model

- How to verify the abstract system?
  - k-induction and infinite BMC

    `sal-inf-bmc --help`
  - scalability?

    Relational abstracter is very fast.

    `sal-inf-bmc` is the bottleneck

    One reason is disjunctive relational abstraction
- Can we generate nonlinear relational abstractions?
  - Yes, they will be more precise
  - But, current SMT solvers can't analyze those abstractions

Demo Continued

Demo: `sal-inf-bmc -i -d 2 robotnav correct`

No counter example is found, but unable to prove either

Demo: `sal-inf-bmc -i -d 4 robotnav correct`

Proved!

Demo: `sal-inf-bmc -i -d 12 robotnav wrong`

Counter-example reported.

Timed Relational Abstraction

Why Timed Relational Abstraction?

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

Timed relational abstraction is an approach we are developing to analyze designs in the presence of platform constraints

Timed Relational Abstraction: Definition

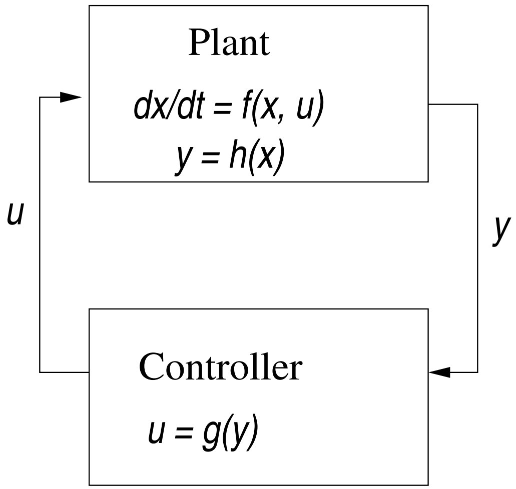
What is TRA?

A timed relational abstraction of a component is a relation between the initial state of the component and the state of the component after time T

Timed relational abstraction captures what a component can do in T time units

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

Timed Relational Abstraction: Illustration



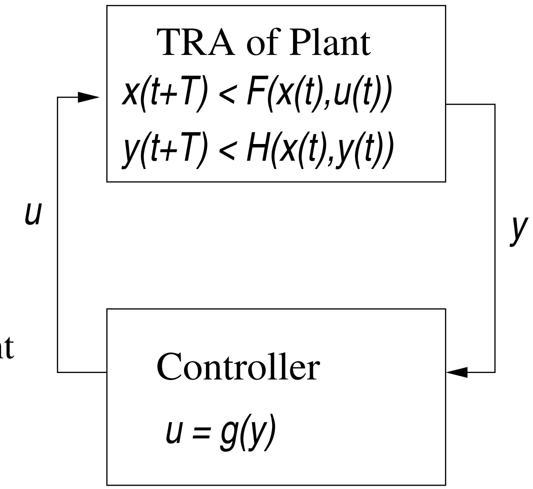
Controller implemented on TTA

```
Every T time units:
1. Controller reads y
2. Controller computes u
3. Controller sends u to plant
```

cont. time plant
disc. time controller

$\xrightarrow{\text{TRA}}$

disc.time non–det plant
disc.time controller



Relational vs. Timed Relational Abstraction

Consider a system consisting of a P/PI controller + plant

- Relational abstraction can be used verify safety of the system
  But it assumes the controller is running in continuous time
- In reality, the controller is implemented in software running on some platform
- Suppose the platform imposes the restriction that the controller executes once every T seconds
- Timed relational abstraction can be used to verify safety/stability of such a system
- Results: The system can be safe/stable for certain T, but fail to be safe/stable for larger T.

Timed Relational Abstraction in HybridSAL

HybridSAL can analyze controllers running on a time-triggered platform

At command-line, we specify the sampling period T

Demo: `examples/PTimed.hsal`: A simple P controller in HybridSAL

Demo: `bin/hsal2hasal -t 0.01 examples/PTimed.hsal`

Demo: `sal-inf-bmc -i -d 10 PTimed stable`

Proved!

Demo: `bin/hsal2hasal -t 0.1 examples/PTimed.hsal`

Demo: `sal-inf-bmc -i -d 10 PTimed stable`

Counter-example

Another Demo of TRA in HybridSAL

Demo: `examples/PISatTimed.hsal:`

A PI controller, whose integrator is saturated, in HybridSAL

Demo: `bin/hsal2hasal -t 0.01 examples/PISatTimed.hsal`

Demo: `sal-inf-bmc -i -d 10 PISatTimed stable`

Proved!

Demo: `sal-inf-bmc -i -d 10 PISatTimed wrong`

Counter-example returned.

Demo: `bin/hsal2hasal -t 0.1 examples/PISatTimed.hsal`

Demo: `sal-inf-bmc -i -d 10 PISatTimed stable`

Counter-example

More About HybridSAL

`bin/hsal2hasal -h`

Other options:

`-n`:	creates nonlinear abstract models
`-mdt <T>`:	assume minimum dwell time of T units in each mode
	(system forced to spend at least T units in each mode)

Other examples:

`nav.hsal`: Hybrid system navigation benchmark

`powertrain.hsal`: Powertrain from Ford

`drivetrain.hsal`: Simple drivetrain in HybridSal

`InvPenTimed.hsal`: Inverted pendulum in HybridSal

HybridSAL: Restrictions

All ODEs should be linear

Not full syntax of SAL supported
Actively developing

Careful of deadlocks

Alternative to `sal-inf-bmc` ?
Generating (helper) invariants