HybridSAL Relational Abstracter

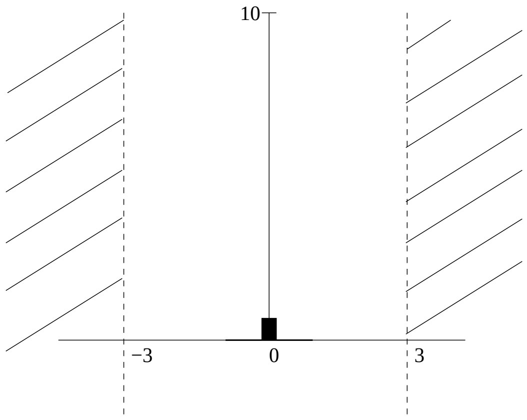
Ashish Tiwari

SRI International

Menlo Park

CA 94025

HybridSAL = SAL + ODEs



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

$$\texttt{Init} \quad := \quad (x \in [-1,1],\ y = 0,\ v_x = 0,\ v_y = 0)$$

$$\texttt{Safe} \quad := \quad (|x| \leq 3)$$

The robot can move in 2 modes:

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

$$\frac{dx}{dt} = v_x, \quad \frac{dv_x}{dt} = 1 - v_x, \quad \frac{dy}{dt} = v_y, \quad \frac{dv_y}{dt} = 1 - v_y$$

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

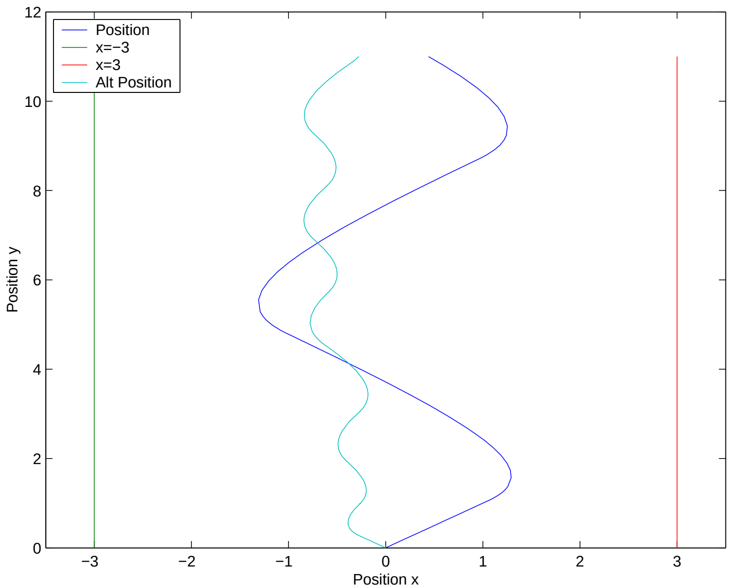
$$\frac{dx}{dt} = v_x, \quad \frac{dv_x}{dt} = -1 - v_x, \quad \frac{dy}{dt} = v_y, \quad \frac{dv_y}{dt} = 1 - v_y$$

Example: Driving a Robot

Consider a non-deterministic controller:

- Switch from Mode 1 to Mode 2 when $-2 \leq x + v_x \leq -1$
- Switch from Mode 2 to Mode 1 when $1 \leq x + v_x \leq 2$

Two possible simulation trajectories:



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 AND x + vx ≥ -2 -->
      xdot' = vx; vxdot' = -1 - vx;
      ydot' = vy; vydot' = 1 - vy
  [] direction = FALSE AND x + vx ≤ 2 -->
      xdot' = vx; vxdot' = 1 - vx;
      ydot' = vy; vydot' = 1 - vy
  ]
END;
```

HybridSAL: Modeling the Controller

```
controller:  MODULE =
BEGIN
  OUTPUT direction:  BOOLEAN
  INPUT x, vx :  REAL
  TRANSITION
  [ direction = TRUE AND x' + vx' ≤ -1 -->
        direction' = FALSE
  [] direction = FALSE AND x' + vx' ≥ 1 -->
        direction' = TRUE
  ]
END;
```

Note: the initial value of `direction` is unconstrained

HybridSAL: Modeling the System

```
robot:  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/robot.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/robot.hsal`

Demo: File `examples/robot.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
$\dot{x} = 1, \dot{y} = 1$	$x' - x = y' - y \quad \wedge \quad y' \geq y$
$\dot{x} = 2, \dot{y} = 3$	$(x' - x)/2 = (y' - y)/3 \quad \wedge \quad y' \geq y$
$\frac{dx}{dt} = -x$	$x \geq x' > 0 \vee x \leq x' < 0 \vee x = x' = 0$
$\frac{dx}{dt} = -x + y$ $\frac{dy}{dt} = -x - y$	$\mathtt{max}(\lvert x\rvert, \lvert y\rvert) \geq \mathtt{max}(\lvert x'\rvert, \lvert y'\rvert) \wedge$ $x^2 + y^2 \geq {x'}^2 + {y'}^2$
$\frac{d\vec{x}}{dt} = A\vec{x}$	$(c^T\vec{x} \geq c^T\vec{x'} > 0 \ \vee$ $c^T\vec{x} \leq c^T\vec{x'} < 0 \ \vee$ $c^T\vec{x} = c^T\vec{x'} = 0) \quad \wedge \ldots$

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

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 4 robot correct`

No counter example is found, but unable to prove either

Demo: `sal-inf-bmc -i -d 4 robot helper`

Proved!

Demo: `sal-inf-bmc -i -l helper -d 4 robot correct`

Proved!

Demo: `sal-inf-bmc -i -l helper -d 4 robot 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

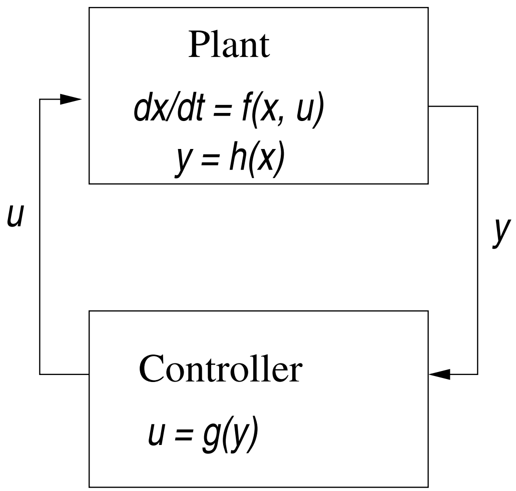
What is TRA?

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

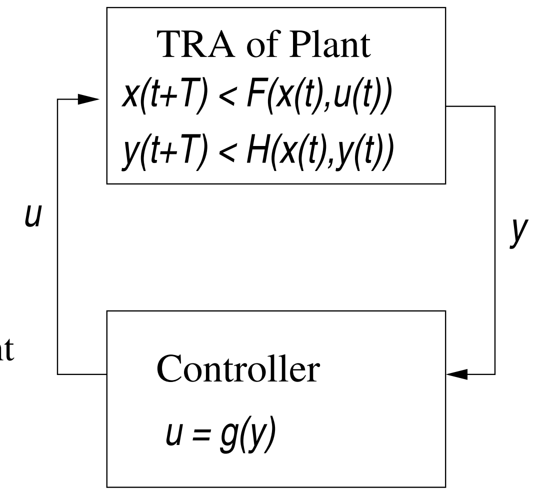
Timed Relational Abstraction: Illustration



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