Hybrid Systems

Ashish Tiwari
SRI International

Hybrid Dynamical Systems

A hybrid dynamical system consists of

- hybrid-space: $\mathsf{X} \subset \mathbb{N}^n \times \mathbb{R}^m$
- That is, some variables take values in a discrete domain $\mathbb{N}$
- Other variables take values in a continuous domain $\mathbb{R}$

The trajectories are defined over

- hybrid-time: $\mathsf{T} = \mathbb{R} \times \mathbb{N}$
- That is, at some time instants $t \in \mathbb{R}$, the system makes $n \in \mathbb{N}$ jumps

Useful for modeling systems having complex, nonlinear, multimodal behavior

Or systems involving interaction between physical system and software

Specifying the Dynamics

Dynamics are typically specified using local rules

A dynamical system can be specified as a tuple $(X, \rightarrow)$ where
X : variables defining state space of the system
$\rightarrow$: binary relation over state space defining system dynamics

A "run" of such a system is a sequence of states related by $\rightarrow$:

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \cdots$$

Now, we can talk about temporal properties of dynamical systems

But what about continuous-time systems?

Continuous Dynamical Systems

We give semantics to continuous-space continuous-time systems by mapping them to continuous-space, discrete-time systems

Continuous dynamics are specified using ordinary differential equations $\frac{d\vec{x}}{dt} = F(\vec{x})$, where $F : \mathbb{R}^n \mapsto \mathbb{R}^n$

Discrete-time semantics: $\vec{x}_0 \rightarrow \vec{x}_1$ iff there exists a $f : \mathbb{R}^+ \mapsto \mathbb{R}^n$ and $\delta \geq 0$ such that

$$
\begin{aligned}
\vec{x}_0 &= f(0) \\
\vec{x}_1 &= f(\delta) \\
\frac{df(t)}{dt} &= F(f(t))
\end{aligned}
$$

A state can have uncountably many successors

Now we can make sense of temporal logic properties of continuous-time systems

Hybrid Systems

For hybrid systems

- X includes Boolean- and Real-valued variables; hence, a hybrid state space
- executions are in hybrid-time, hence its semantics $\rightarrow$ relates a state to all its hybrid-time successors

$$\rightarrow \; := \; \rightarrow_{disc} \cup \rightarrow_{cont}$$

Example of a Hybrid System

$$
\begin{pmatrix} \frac{dx}{dt} & = & v_x \\ \frac{dy}{dt} & = & v_y \\ \frac{dv_x}{dt} & = & -1 - v_x \\ \frac{dv_y}{dt} & = & 1 - v_y \\ x + v_x & \geq & -2 \end{pmatrix}
\begin{array}{c} \xrightarrow{x + v_x \leq -2} \\ \\ \xleftarrow[x + v_x \geq 2]{} \end{array}
\begin{pmatrix} \frac{dx}{dt} & = & v_x \\ \frac{dy}{dt} & = & v_y \\ \frac{dv_x}{dt} & = & 1 - v_x \\ \frac{dv_y}{dt} & = & 1 - v_y \\ x + v_x & \leq & 2 \end{pmatrix}
$$

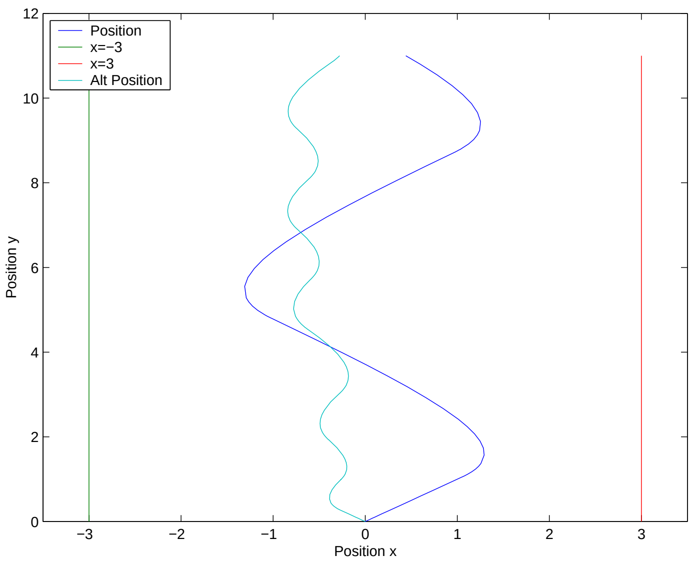
Starting from a region $-1 \leq x \leq 1, y = 0, v_x = v_y = 0$, how to prove $G(-3 \leq x \leq 3)$ for this system?

Example: Simulations of the Robot

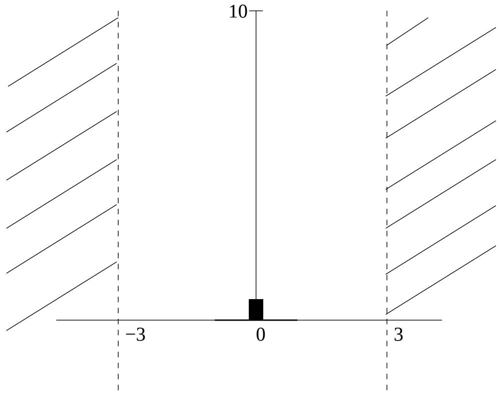
The controller is non-deterministic:

- Switches from Mode 1 to Mode 2 when $x + v_x + 2 \leq 0$
- Switches from Mode 2 to Mode 1 when $x + v_x - 2 \geq 0$

Two possible simulation trajectories:



HybridSAL: Modeling



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

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

The robot can move in 2 modes:

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

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

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

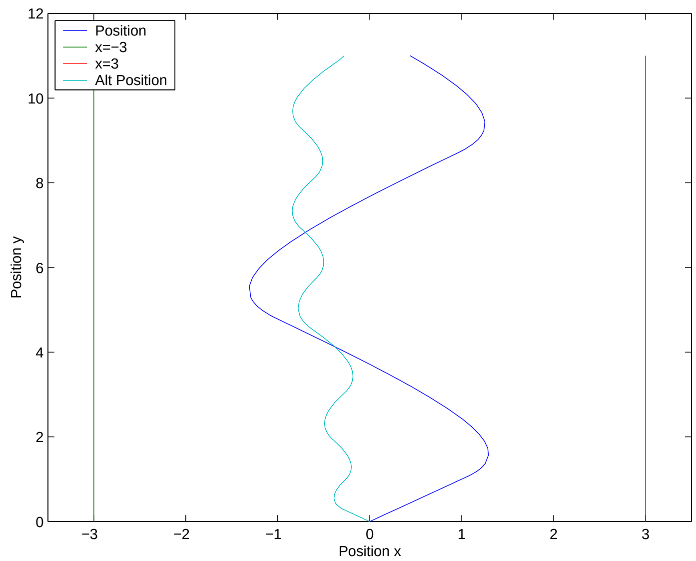
$$\frac{dx}{dt} = v_x, \quad \frac{dv_x}{dt} = -1 - v_x, \quad \frac{dy}{dt} = v_y, \quad \frac{dv_y}{dt} = 1 - v_y$$

Example: Driving a Robot

Consider a non-deterministic controller:

- Switch from Mode 1 to Mode 2 when $x + v_x + 2 \leq 0$
- Switch from Mode 2 to Mode 1 when $x + v_x - 2 \geq 0$

Two possible simulation trajectories:



HybridSAL Model of Robot

```
robot:CONTEXT =
BEGIN
system: MODULE =
 BEGIN
 LOCAL direction : BOOLEAN  % moving left/right
 LOCAL x, vx, y, vy : REAL
 LOCAL xdot, vxdot, ydot, vydot : REAL
 INVARIANT TRUE
 INITFORMULA
  -1 <= x AND x <= 1 AND vx = 0 AND vy = 0 AND y = 0
 ...
```

HybridSAL Model of Robot

```
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
[] direction = TRUE AND x + vx <= -2 -->
    direction' = FALSE
[] direction = FALSE AND x + vx >= 2 -->
    direction' = TRUE  ]
END;
...
```

HybridSAL Model of Robot

```
robot: CONTEXT
BEGIN
system: MODULE =
 BEGIN
   LOCAL ...
   INVARIANT ...
   INITFORMULA ...
   TRANSITION
   [ ... [] ... [] ... ]
 END;

 correct: THEOREM
     system |- G( 0 <= x+3 AND x <= 3 );
END
```

HybridSAL Analysis

HybridSAL provides an abstractor that takes a HybridSAL model and outputs a finite state SAL model

HybridSAL is written in Lisp has a command-line interface:

```
mlisp
(load "load.lisp")
(in-package :sal)
(abstract "robot" 'system :property 'correct)
```

This creates a file "robotABS.sal"

Abstract Model in SAL

```
%% Abstract variable to Polynomial Mapping:
%% g11 --> -1*x - 3
%% g10 --> x - 3
%% g9 --> -1*x - 1
%% g8 --> x - 1
%% g7 --> vx
%% g6 --> vy
%% g5 --> y
%% g4 --> x + vx + 2
%% g3 --> x + vx - 2
%% g2 --> -1*vy + 1
%% g1 --> -1*vx - 1
%% g0 --> -1*vx + 1
...
```

Abstract Model in SAL

```
robotABS: CONTEXT =
 BEGIN
  SIGN: TYPE = {pos, neg, zero};

  ASSVP(x0: SIGN, x1: SIGN): [SIGN -> BOOLEAN] = ...
  ASSVN(x0: SIGN, x1: SIGN): [SIGN -> BOOLEAN] = ...

  INV12(g11: SIGN, ..., g0: SIGN): BOOLEAN = ...

  system: MODULE = BEGIN
    GLOBAL g0, ..., g11: SIGN
    LOCAL  direction: BOOLEAN
    INITIALIZATION g11 = neg; ... ; g0 = pos
    ...
```

Abstract Model in SAL

```
TRANSITION
[(direction = TRUE AND (g4 = pos OR g4 = zero)) AND
  INV12(g11', ..., g0') AND (g4' = pos OR g4' = zero) -->
 g11' IN ASSVN(g11, g7); ...; g0' IN ASSVN(g0, g1)
 []
 (direction = FALSE AND (g3 = neg OR g3 = zero)) AND
  INV12(g11', ..., g0') AND (g3' = neg OR g3' = zero) -->
 g11' IN ASSVN(g11, g7); ...; g0' IN ASSVN(g0, g0)
 []
 (direction = TRUE AND (g4 = neg OR g4 = zero)) AND
  INV12(g11', ..., g0') --> direction' = FALSE
 []
 (direction = FALSE AND (g3 = pos OR g3 = zero)) AND
  INV12(g11', ..., g0') --> direction' = TRUE
 ]
```

Abstract Model in SAL

```
robotABS: CONTEXT =
BEGIN
  ...
  system: MODULE = ...

  correct: THEOREM
    system |- G((g11 = neg OR g11 = zero) AND
                (g10 = neg OR g10 = zero));
END
```

Model Check the Abstract Model

If SAL is installed, then we can analyze the abstract SAL model

```
sal-deadlock-checker robotABS system
sal-smc -v 3 robotABS correct
```

We can thus verify the safety property of the hybrid robot model.

If property is not true of abstract model, then we get a counter-example in the abstract

Which may be spurious

HybridSAL: Discussion

- Predicates for abstraction are chosen automatically
- This choice is crucial, and can be influenced by command-line input
- The abstraction process is completely automatic, but it can take long
- Tool is still work in progress: several features of SAL are not supported in HybridSAL
- Such as compositional abstraction
- `http://sal.csl.sri.com/hybridsal/`