Formal Simulation and Visualisation of Hybrid Programs in LINCE

Slides based on a presentation given in FMAS @ iFM 2024

Pedro Mendes, Ricardo Correia, Renato Neves, José Proença

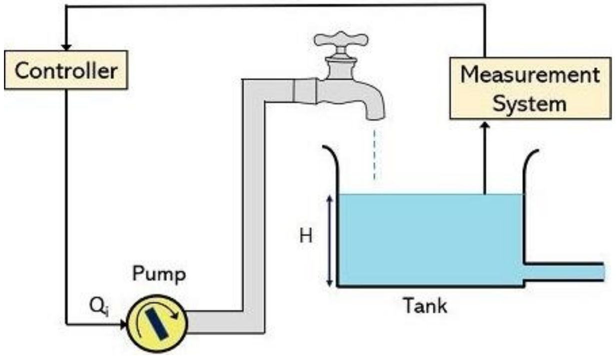
https://jose.proenca.org/publication/fmas-lince-2024/

José Proença
System Verification (CC4084) 2024/2025
CISTER – U.Porto, Porto, Portugal

https://fm-dcc.github.io/sv2425

Hybrid systems

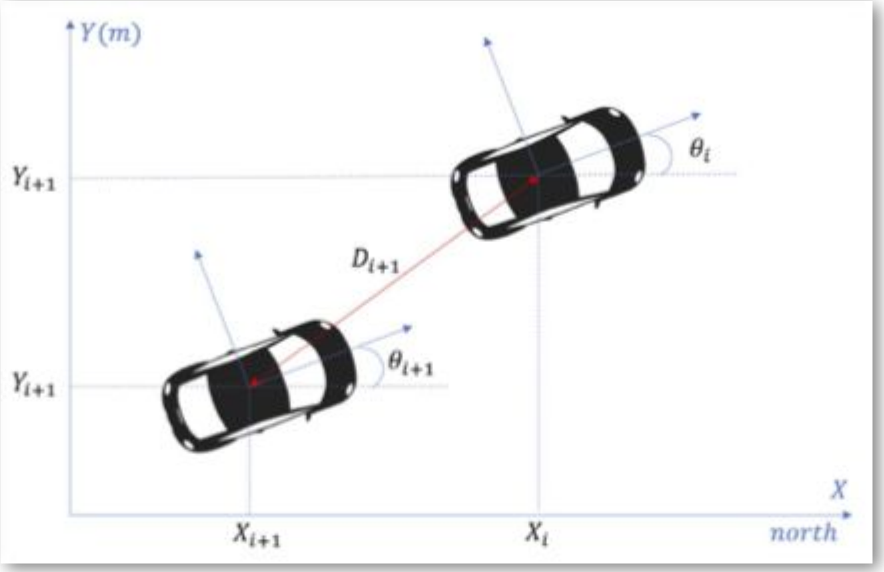
Computational devices that interact with **physical** environment



Another hybrid system

Platooning

- Acceleration (1D)
- Steering (2D)
- Failures



By Ênio Filho, Anis Koubâ, Ricardo Severino, Eduardo Tovar @ CISTER

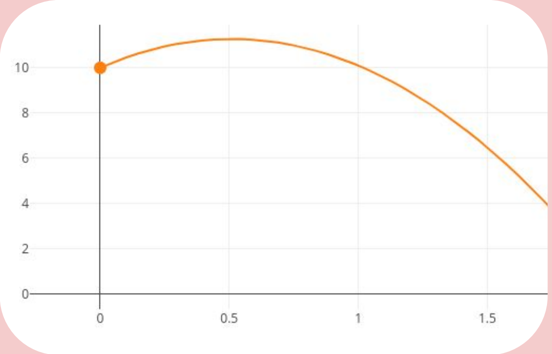
Discrete behaviour

+

Continuous behaviour

```
...
v := 10;
while v <= 10 do {
  ...
}
```

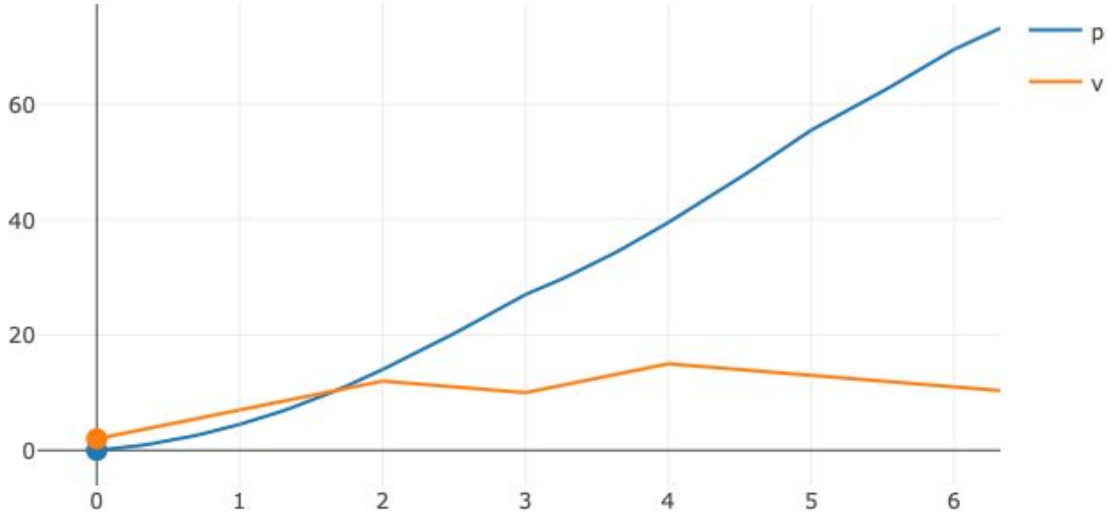
$$\begin{cases} v' = -9.8 \\ p' = v \end{cases}$$




```
p:=0; v:=2;
while true do {
  if v<=10
  then p'=v,v'=5  for 1
  else p'=v,v'=-2 for 1
}
```

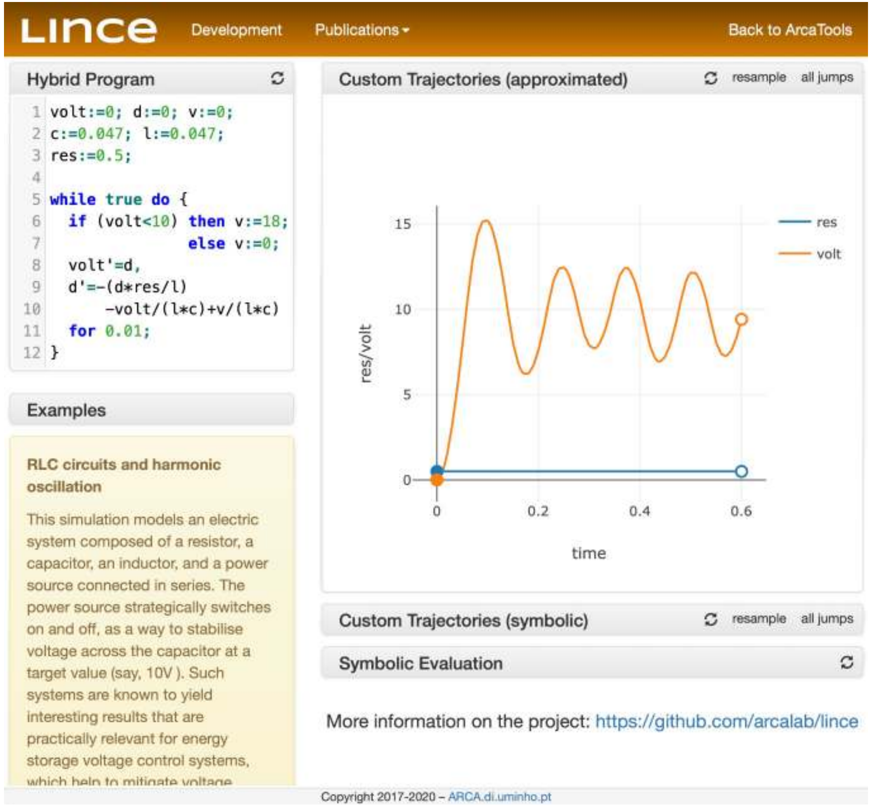
A cruise controller in Lince

```
// Cruise control
p:=0; v:=2;
while true do {
  if v<=10
  then p'=v,v'=5  for 1
  else p'=v,v'=-2 for 1
}
```



The expected output

Why Lince?



- **No installation**
  - just a website (+ server)
- **Easy** to experiment
- **Simple** language
  - No need for complex frameworks
- **Precise** semantics
- Low effort to **extend**
  - new extensions
  - involve students
  - involve partners

What we will see

```
// Cruise control
p:=0; v:=2;
while true do {
  if v<=10
  then p'=v,v'=5  for 1
  else p'=v,v'=-2 for 1
}
```

1. Syntax & semantics

2. Examples

3. Recent extensions

4. Roadmap

Syntax

Discrete control

```
p,q ∋ x := t
      | p ; q
      | if b then p else q
      | while b do { p }
      | deq
```

Continuous control

deq ∋ $x_1' = t_1$, … , $x_n' = t_n$ for t

(systems of differential equations)

```
t,s ∋ real | real * x | t + s | x
```

(linear terms)

Syntax

Discrete control

```
p,q ∋ x := e
      | p ; q
      | if e then p else q
      | while b do { p }
      | deq
```

Continuous control

```
deq ∋ x₁' = t₁,
      … ,
      xₙ' = tₙ
      for e
```

(systems of differential equations)

```
t,s  ∋  real | real * x | t + s | x
```

(linear terms)

new

```
e    ∋  e | f(e₁,...,e₁)
```

(non-linear terms)

(Big-Step) Semantics

$$\textbf{(diff-skip)}\quad \frac{\llbracket e \rrbracket(\sigma) = t}{\vec{x}' = \vec{\ell}\ \texttt{for}\ e, \sigma, t \Downarrow \mathit{skip}, \sigma[\vec{x} \mapsto \phi(t)]}$$

$$\textbf{(diff-stop)}\quad \frac{\llbracket e \rrbracket(\sigma) > t}{\vec{x}' = \vec{\ell}\ \texttt{for}\ e, \sigma, t \Downarrow \mathit{stop}, \sigma[\vec{x} \mapsto \phi(t)]}$$

$$\textbf{(diff-err)}\quad \frac{\llbracket e \rrbracket(\sigma)\ \text{undefined}}{\vec{x}' = \vec{\ell}\ \texttt{for}\ e, \sigma, t \Downarrow \mathit{err}}$$

$$\textbf{(asg-skip)}\quad \frac{\llbracket e \rrbracket(\sigma)\ \text{defined}}{x := e, \sigma, 0 \Downarrow \mathit{skip}, \sigma[x \mapsto \llbracket e \rrbracket(\sigma)]}$$

$$\textbf{(asg-err)}\quad \frac{\llbracket e \rrbracket(\sigma)\ \text{undefined}}{x := e, \sigma, t \Downarrow \mathit{err}}$$

$$\textbf{(seq-skip)}\quad \frac{\mathtt{p}, \sigma, t \Downarrow \mathit{skip}, \tau \qquad \mathtt{q}, \tau, u \Downarrow v}{\mathtt{p;q}, \sigma, t + u \Downarrow v}$$

(if-rules) … **(while-rules) …**

(Big-Step) Semantics

(diff-stop)

$$\frac{[\![e]\!](\sigma) > t}{\vec{x}' = \vec{\ell}\ \texttt{for}\ e, \sigma, t \Downarrow \mathit{stop}, \sigma[\vec{x} \mapsto \phi(t)]}$$

Statement (syntax)

State (var -> value)

Target time (real value)

(Big-Step) Semantics

$$\textbf{(diff-stop)} \quad \frac{[\![e]\!](\sigma) > t}{\vec{x}' = \vec{\ell} \ \texttt{for} \ e, \sigma, t \Downarrow \boxed{stop}, \boxed{\sigma[\vec{x} \mapsto \phi(t)]}}$$

Statement
(syntax)

State
(var -> value)

Target time
(real value)

Reached the end?

Final state (var -> value)

(Big-Step) Semantics

$$\textbf{(diff-err)} \quad \frac{[\![e]\!](\sigma)\ \boxed{\text{undefined}}}{\vec{x}' = \vec{\ell}\ \texttt{for}\ e, \sigma, t\ \Downarrow\ \boxed{err}}$$

new

Partial functions (can throw errors)

(Small-Step) Semantics

$$(\mathbf{asg}^{\rightarrow}) \qquad \mathtt{x} := \mathtt{t}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathit{skip}\,,\sigma\nabla[\mathtt{t}\sigma/\mathtt{x}]\,,\mathtt{t}$$

$$(\mathbf{diff\text{-}stop}^{\rightarrow}) \qquad \bar{\mathtt{x}}' = \bar{\mathtt{u}}\;\mathtt{for}\;\mathtt{t}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathit{stop}\,,\sigma\nabla[\phi_\sigma(\mathtt{t})/\bar{\mathtt{x}}]\,,0 \qquad (\mathit{if}\;\; t < \mathtt{t}\sigma)$$

$$(\mathbf{diff\text{-}skip}^{\rightarrow}) \qquad \bar{\mathtt{x}}' = \bar{\mathtt{u}}\;\mathtt{for}\;\mathtt{t}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathit{skip}\,,\sigma\nabla[\phi_\sigma(\mathtt{t}\sigma)/\bar{\mathtt{x}}]\,,\mathtt{t} - (\mathtt{t}\sigma) \qquad (\mathit{if}\;\; t \geqslant \mathtt{t}\sigma)$$

$$(\mathbf{if\text{-}true}^{\rightarrow}) \qquad \mathtt{if}\;\mathtt{b}\;\mathtt{then}\;\mathtt{p}\;\mathtt{else}\;\mathtt{q}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathtt{p}\,,\sigma\,,\mathtt{t} \qquad (\mathit{if}\;\; \mathtt{b}\sigma = \top)$$

$$(\mathbf{if\text{-}false}^{\rightarrow}) \qquad \mathtt{if}\;\mathtt{b}\;\mathtt{then}\;\mathtt{p}\;\mathtt{else}\;\mathtt{q}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathtt{q}\,,\sigma\,,\mathtt{t} \qquad (\mathit{if}\;\; \mathtt{b}\sigma = \bot)$$

$$(\mathbf{wh\text{-}true}^{\rightarrow}) \qquad \mathtt{while}\;\mathtt{b}\;\mathtt{do}\;\{\,\mathtt{p}\,\}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathtt{p}\,;\mathtt{while}\;\mathtt{b}\;\mathtt{do}\;\{\,\mathtt{p}\,\}\,,\sigma\,,\mathtt{t} \qquad (\mathit{if}\;\; \mathtt{b}\sigma = \top)$$

$$(\mathbf{wh\text{-}false}^{\rightarrow}) \qquad \mathtt{while}\;\mathtt{b}\;\mathtt{do}\;\{\,\mathtt{p}\,\}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathit{skip}\,,\sigma\,,\mathtt{t} \qquad (\mathit{if}\;\; \mathtt{b}\sigma = \bot)$$

$$(\mathbf{seq\text{-}stop}^{\rightarrow}) \quad \frac{\mathtt{p}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathit{stop}\,,\sigma'\,,\mathtt{t}'}{\mathtt{p}\,;\mathtt{q}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathit{stop}\,,\sigma'\,,\mathtt{t}'} \qquad (\mathbf{seq\text{-}skip}^{\rightarrow}) \quad \frac{\mathtt{p}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathit{skip}\,,\sigma'\,,\mathtt{t}'}{\mathtt{p}\,;\mathtt{q}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathtt{q}\,,\sigma'\,,\mathtt{t}'}$$

$$(\mathbf{seq}^{\rightarrow}) \quad \frac{\mathtt{p}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathtt{p}'\,,\sigma'\,,\mathtt{t}'}{\mathtt{p}\,;\mathtt{q}\,,\sigma\,,\mathtt{t} \;\rightarrow\; \mathtt{p}';\mathtt{q}\,,\sigma'\,,\mathtt{t}'} \qquad (\mathit{if}\;\; \mathtt{p}' \neq \mathit{stop}\;\; \mathit{and}\; \mathtt{p}' \neq \mathit{skip})$$

Exercises

1. Verify if the transitions exist, and justify.

```
x:=3, [], 4 ⟶ stop, [x=3], 4
x:=5, [], 0 ⟶ skip, [x=5], 0
x:=5;y:=6, [], 0 ⟶ skip, [x=5,y=6], 0
```

2. Evolve twice each of these states

```
x:=3;x:=5;y:=7, [], 4

if (x>2)
then y:=0
else y:=1, [x=25], 0

while (x>2) do
  {x:=x-1}     , [x=25], 0
```

3. Run the program

```
x:=3;x’=1 for 1; x’=-2 for 2, [], 2
```

Hybrid programs in Lince

- Demo -

Run in **our server**

https://arcatools.org/lince

- *Internet browser*

Download and run **your server**

https://github.com/arcalab/lince

- SageMath (http://www.sagemath.org/)
- SBT (https://www.scala-sbt.org)
- Java runtime
- *Internet browser*

Examples

```
// Simple composition
p:=0; v:=0;
p'=v,v'=-2  for 1;
p'=v,v'=-2  for 1;
```

```
// Initial values of the water tank
level := 5; drain := -1/2;

while true do {
  // keep level between 3..10
  if      level<=3  then drain:= 1/2;
  else if level>=10 then drain:=-1/2;
  else    skip;
  level'= drain, drain'=0 for 0.1;
}
```

Exercise

4. Re-implement the water tank, but calculating the optimal duration at each step (when to reach the maximum or minimum level)

```
// Initial values of the water tank
level := 5; drain := -1/2;

while true do {
  // keep level between 3..10
  if      level<=3  then drain:= 1/2;
  else if level>=10 then drain:=-1/2;
  else    skip;
  level'= drain, drain'=0 for 0.1;
}
```

More Examples

```
// Cruise control
p:=0; v:=2;
while true do {
  if v<=10
  then p'=v,v'=5  for 1
  else p'=v,v'=-2 for 1
}
```

```
// Adaptive Cruise control
p:=0;    v:=0;   a:=5; b:=-2; // follower
pl:=50; vl:=10; al:=0;        // leader
period:=1;

while true do {
  ???
}
```

More Examples

```
// Cruise control
p:=0; v:=2;
while true do {
  if v<=10
  then p'=v,v'=5  for 1
  else p'=v,v'=-2 for 1
}
```

```
// Adaptive Cruise control
p:=0;    v:=0;   a:=5; b:=-2; // follower
pl:=50; vl:=10; al:=0;        // leader
period:=1;

while true do {
  if ((p+v*period+ a/2*period^2 <
        pl+vl*period+al/2*period^2) &&
       (((v-vl+(a-al)*period)^2 -
         4*(p-pl+(v-vl)*period +
         (a-al)/2*period^2)*(b-al)/2) < 0))
  then p'=v,v'=a,pl'=vl,vl'=al for period;
  else p'=v,v'=b,pl'=vl,vl'=al for period;
}
```

More Examples

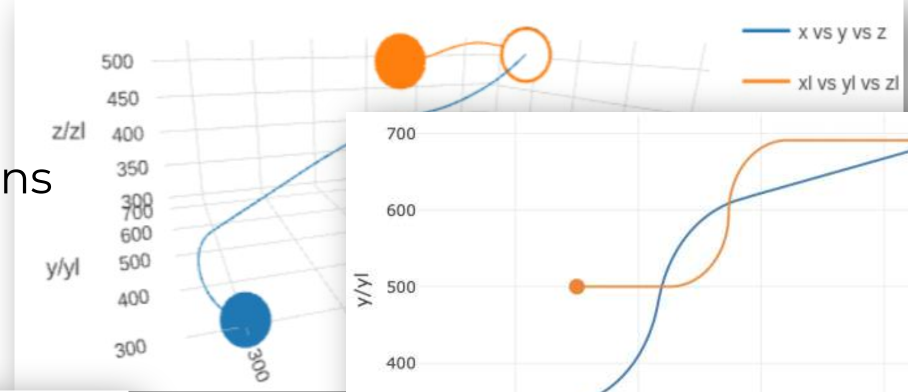
```
// Cruise control
p:=0; v:=2;
while true do {
  if v<=10
  then p'=v,v'=5  for 1
  else p'=v,v'=-2 for 1
```

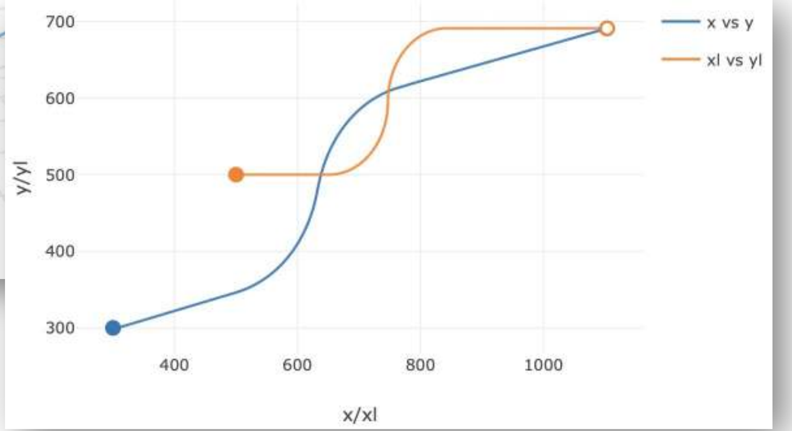
```
// Adaptive Cruise control
p:=0;    v:=0;    a:=5; b:=-2; // follower
pl:=50; vl:=10; al:=0;         // leader
period:=1;

while true do {                              new
  if ((p+v*period+ a/2*period^2 <
       pl+vl*period+al/2*period^2) &&
      (((v-vl+(a-al)*period)^2 -
        4*(p-pl+(v-vl)*period +
        (a-al)/2*period^2)*(b-al)/2) < 0))
  then p'=v,v'=a,pl'=vl,vl'=al for period;
  else p'=v,v'=b,pl'=vl,vl'=al for period;
```

```
// MORE:
// Automated braking system
// Pursuit games
//    (2D, 3D views)
// Electric RLC circuit
//    (many simulations, approx)
// ...
```

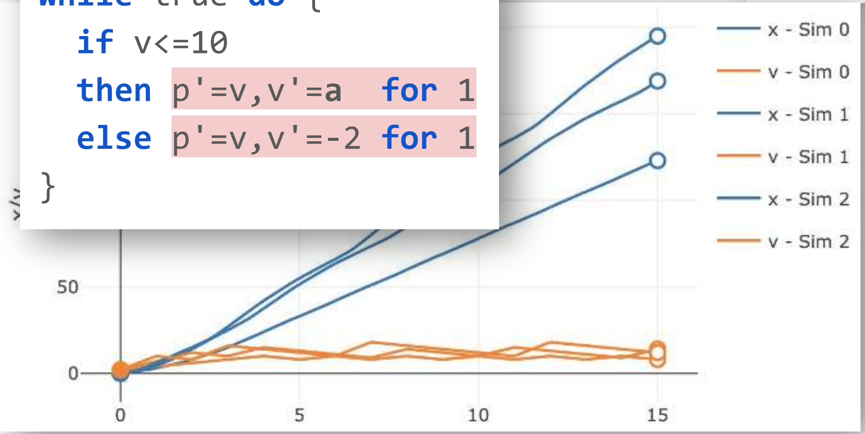
Recent improvements to Lince

- Customisable 2D/3D visualizations
- Overlay multiple simulations






```
// Cruise control
p:=0; v:=2; a:=[2,5,8];
while true do {
  if v<=10
  then p'=v,v'=a  for 1
  else p'=v,v'=-2 for 1
}
```



- More functions (and scenarios)
- Better error handling
- New approximated engine
- (Randomness)

Conclusions and challenges

What to do with hybrid programs?

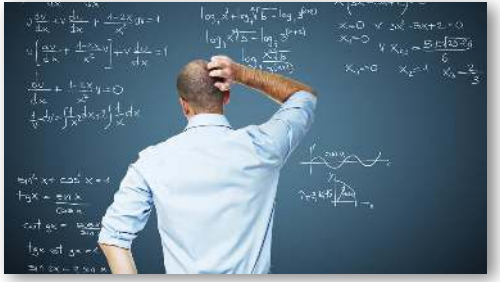
Generate software (e.g., controllers)

Predict physical behaviours

Simulate scenarios

Model checking
(verify properties of scenarios)

Theorem proving
(deductive reasoning to prove generic properties)



What to do with hybrid programs?

Generate software (e.g., controllers)

Predict physical behaviours

MATLAB SIMULINK

Simulate scenarios

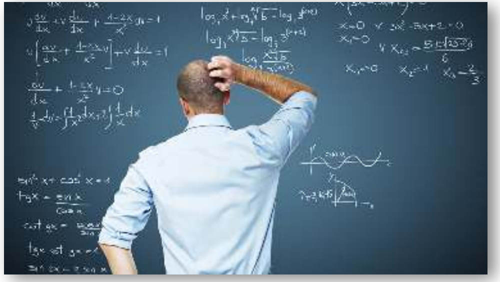
Lince

Model checking
(verify properties of scenarios)

Hybrid ABS

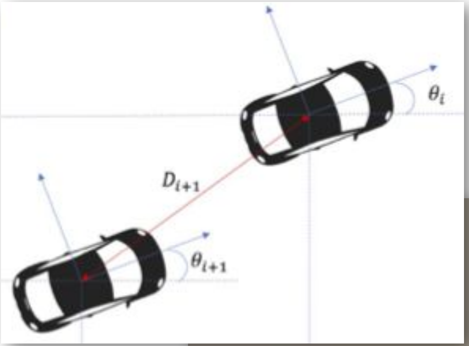
KeYmaera X

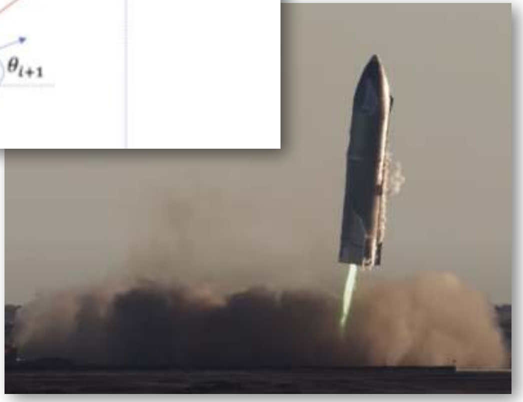
Theorem proving
(deductive reasoning to prove generic properties)



Challenge 1

Model more concrete
scenario in Lince





```
p:=0; v:=2;
while x<=50 do {
  // extend the language if needed
  x(t) = 5/2*t^2+10*t+x0  for 1
}
// Find the need for new extensions
```

Challenge 2

Export program to another tool

```
// Cruise control
p:=0; v:=2;
while true do {
  if v<=10
  then p'=v,v'=5  for 1
  else p'=v,v'=-2 for 1
}
```

MATLAB SIMULINK

KeYmaeraX.org

Show some property?

Extend Lince?

```
// Cruise control
p:=0; v:=2;
while true do {...}
check [v>=10?...] p>5
```

Challenge 3

Import program from another tool

```
// Cruise control
p:=0; v:=2;
while true do {
  if v<=10
  then p'=v,v'=5  for 1
  else p'=v,v'=-2 for 1
}
```

MATLAB SIMULINK

KeYmaeraX.org

Useful subset?

Extend Lince?

```
// Cruise control
p:=0; v:=2;
while true do {...}
check [v>=10?...] p>5
```

Challenge 4

Checking properties: using logics for runtime verification

```
// Cruise control
p:=0; v:=2;
while true do {
  if v<=10
  then p'=v,v'=5  for 1
  else p'=v,v'=-2 for 1

  assert <formula>
}

monitor <formula>
```

Metric temporal logic

RMTL-∫3 (three-valued restricted metric temporal logic with durations)

(Interval) duration logic

Challenge 5

Running many simulations:
Statistical Analysis

```
// Cruise control
p:=0; v:=2; a:=Unif[2..8];
while true do {
  if v<=10
  then p'=v,v'=5  for Exp[0..1]
  else p'=v,v'=-2 for Exp[0..1]
}
check Prob(conf=95%, v<=10)
```

Running multiple times

Aim at statistical relevance

Control number of runs, range of a run, confidence, error margin, etc.

Challenge 6

Improve framework (technology)

JVM

(Java Virtual Machine)

JavaScript