

# Formal Simulation and Visualisation of Hybrid Programs in LInce

Pedro Mendes, Ricardo Correia, Renato Neves, **José Proença**

*FMAS @ iFM, 11 Nov 2024*



This work is financed by National Funds through FCT - Fundação para a Ciência e a Tecnologia, I.P. (Portuguese Foundation for Science and Technology) within the project IBEX, with reference 10.54499/PTDC/CCI-COM/4280/2021. This work is also partially supported by National Funds through FCT/MCTES, within the CISTER Unit (UIDP/UIDB/04234/2020); and by the EU/Next Generation, within the Recovery and Resilience Plan, within project Route 25 (TRB/2022/00061 -- C645463824-00000063).



# Formal Simulation and Visualisation of Hybrid Programs in LInce



**José Proença**  
CISTER & U. Porto  
Portugal



Renato Neves  
INESC-TEC & U. Minho  
Portugal



Ricardo Correia  
U. Minho  
Portugal

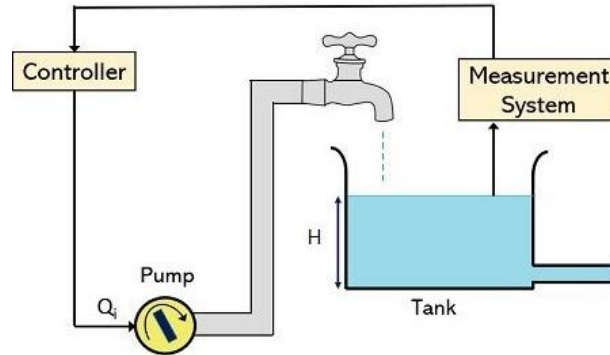


Daniel Mendes  
U. Minho  
Portugal

# Hybrid systems



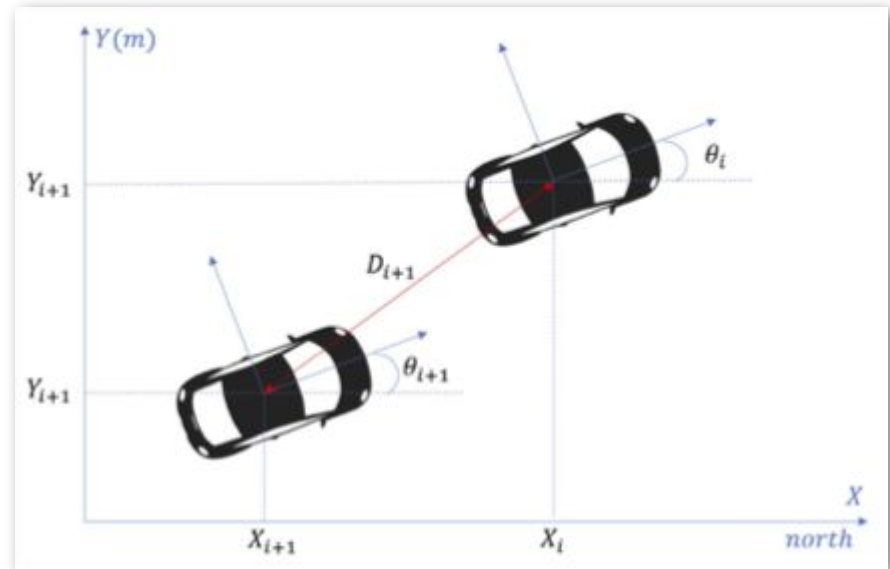
**Computational** devices that interact with **physical** environment



# Another hybrid system

## Platooning

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



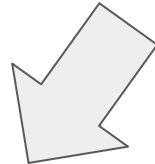
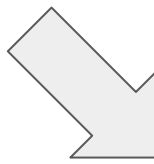
By Ênio Filho, Anis Koubâ, Ricardo Severino, Eduardo Tovar @ CISTER

# Discrete behaviour

+

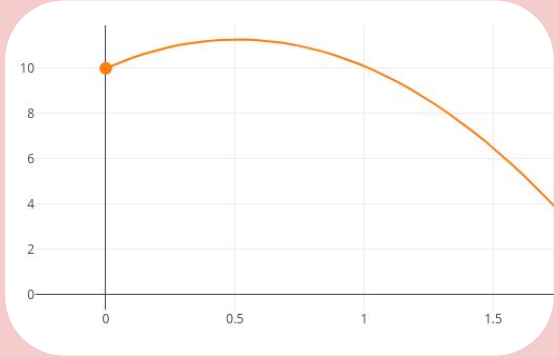
# Continuous behaviour

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



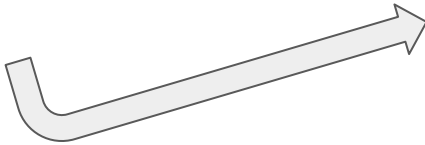
```
p:=0; v:=2;  
while true do {  
  if v<=10  
  then p'=v, v'=5 for 1  
  else p'=v, v'=-2 for 1  
}
```

$v' = -9.8$   
 $p' = v$

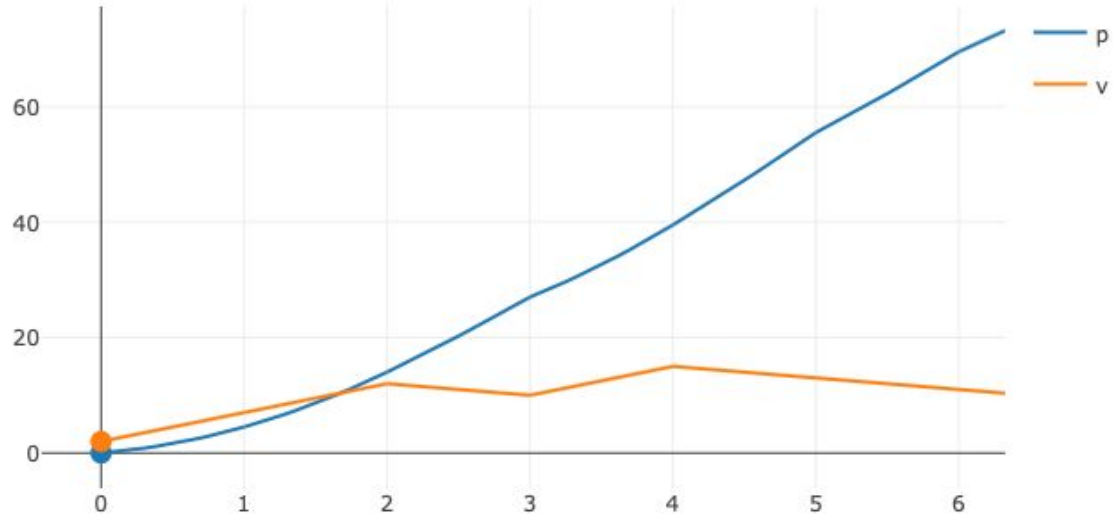


# A cruise controller in **Lince**

```
// 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?

Lince Development Publications ▾ Back to ArcaTools

Hybrid Program ↻

```
1 volt:=0; d:=0; v:=0;
2 c:=0.047; l:=0.047;
3 res:=0.5;
4
5 while true do {
6   if (volt<10) then v:=18;
7     else v:=0;
8   volt'=d,
9   d'=- (d*res/l)
10  -volt/(l*c)+v/(l*c)
11  for 0.01;
12 }
```

Examples

**RLC circuits and harmonic oscillation**

This simulation models an electric system composed of a resistor, a capacitor, an inductor, and a power source connected in series. The power source strategically switches on and off, as a way to stabilise voltage across the capacitor at a target value (say, 10V). Such systems are known to yield interesting results that are practically relevant for energy storage voltage control systems, which help to mitigate voltage

Custom Trajectories (approximated) ↻ resample all jumps

Custom Trajectories (symbolic) ↻ resample all jumps

Symbolic Evaluation ↻

More information on the project: <https://github.com/arcab/lince>

Copyright 2017-2020 – ARCA.di.uminho.pt

- **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

# Hybrid programs - **today**

```
// 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  $\ni$  x := t  
| p ; q  
| if b then p else q  
| while b do { p }  
| deq
```

## Continuous control

```
deq  $\ni$   $x_1' = t_1,$   
... ,  
 $x_n' = t_n$   
for t
```

(systems of differential equations)

```
t,s  $\ni$  real | real * x | t + s | x
```

(linear terms)

# Syntax

## Discrete control

$p, q \ni x := e$   
|  $p ; q$   
| **if**  $e$  **then**  $p$  **else**  $q$   
| **while**  $b$  **do** {  $p$  }  
| **deq**

## Continuous control

**deq**  $\ni$   $x_1' = t_1,$   
 $\dots,$   
 $x_n' = t_n$   
**for**  $e$

(systems of differential equations)

$t, s \ni$  real | real \*  $x$  |  $t + s$  |  $x$  (linear terms)

*new*  $e \ni e$  |  $f(e_1, \dots, e_1)$  (non-linear terms)

# (Big-Step) Semantics

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

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

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

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

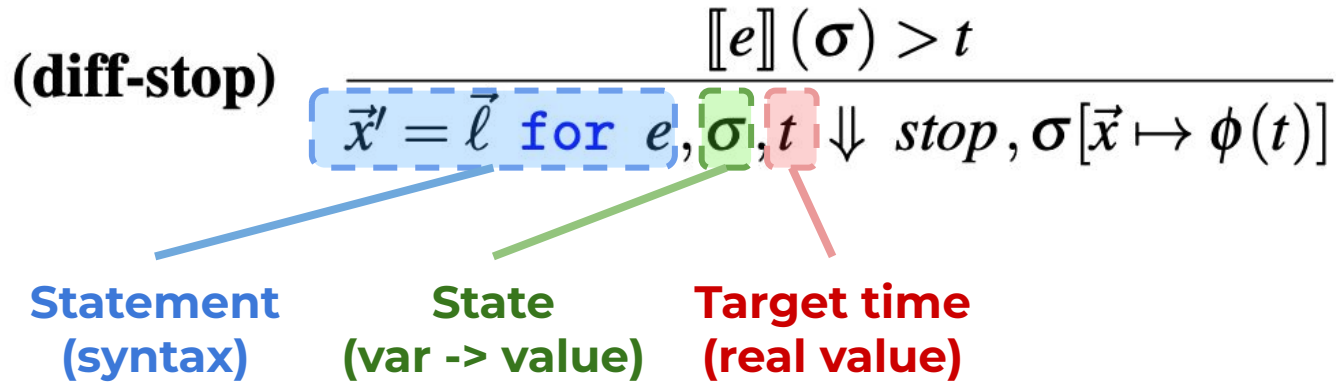
$$\text{(asg-err)} \quad \frac{\llbracket e \rrbracket(\sigma) \text{ undefined}}{x := e, \sigma, t \Downarrow \text{err}}$$

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

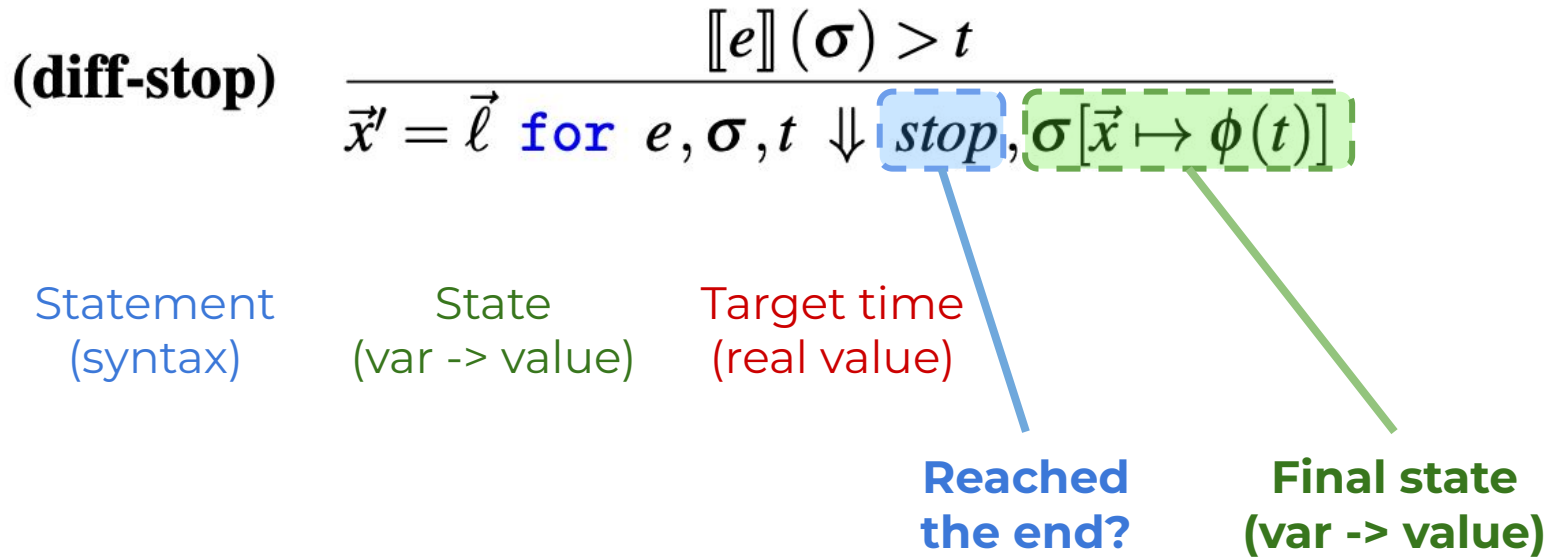
(if-rules) ...

(while-rules) ...

# (Big-Step) Semantics



# (Big-Step) Semantics



# (Big-Step) Semantics

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

**Partial functions  
(can throw errors)**

# 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;
```

```
}
```



# 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
```

```
p1:=50; v1:=10; a1:=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
p1:=50; v1:=10; a1:=0; // leader
period:=1;

while true do {
  if ((p+v*period+ a/2*period^2 <
      p1+v1*period+a1/2*period^2) &&
      (((v-v1+(a-a1)*period)^2 -
        4*(p-p1+(v-v1)*period +
          (a-a1)/2*period^2)*(b-a1)/2) < 0))
  then p'=v,v'=a,p1'=v1,v1'=a1 for period;
  else p'=v,v'=b,p1'=v1,v1'=a1 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
```

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

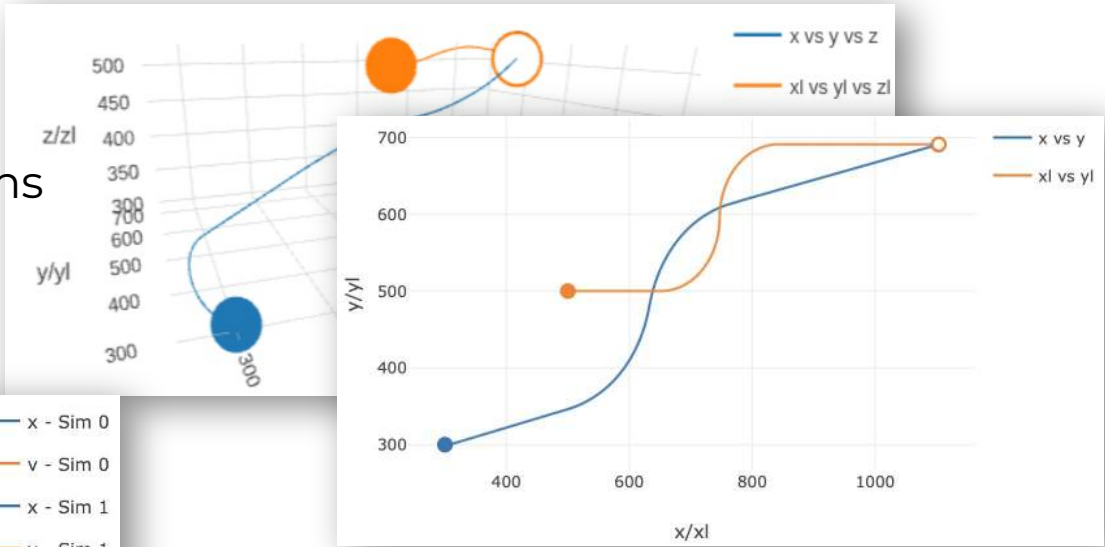
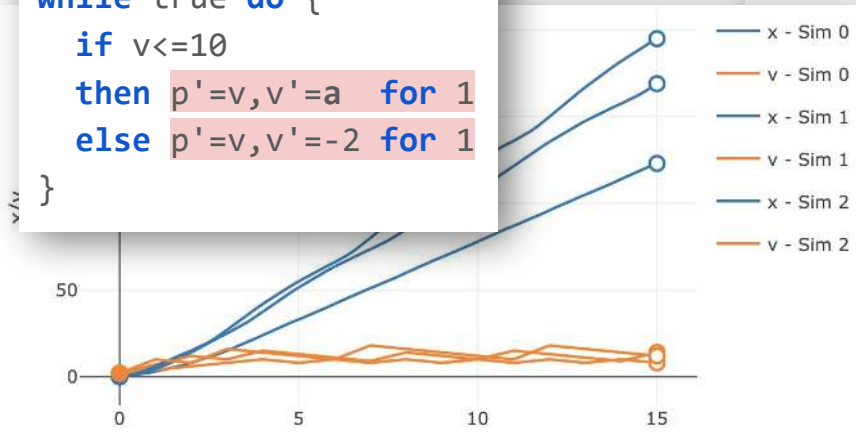
```
// Adaptive Cruise control
p:=0; v:=0; a:=5; b:=-2; // follower
pl:=50; vl:=10; al:=0; // leader
period:=1;

while true do {
  if ((p+v*period+new 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;
```

# 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



**Simulate scenarios**

**Lince**

Model checking  
*(verify properties of scenarios)*



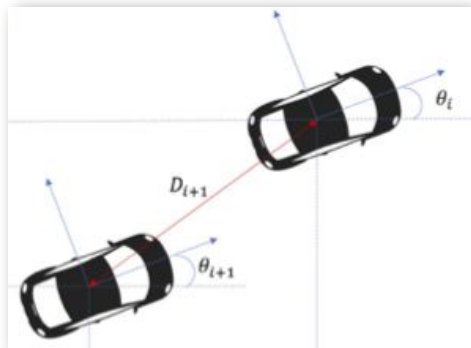
**KeYmaera X**

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
}
```



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
}
```



KeYmaeraX.org

Extend **Lince**?

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

Useful subset?

# 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- $\int$ 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)



# Wrap up

```
// 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 improvements



4. Challenges