Lurette, Lucky, and co., Simulation tools.

But what about Simulink?

Erwan Jahier: Verimag/CNRS

Pascal Raymond: Verimag/CNRS

Synchron 2005

25 november - Malta

Lurette, Lucky, and co.

The dark side of Lutin

Erwan Jahier: Verimag/CNRS
Pascal Raymond: Verimag/CNRS

Synchron 2005

25 november - Malta

Lu . {rette | cky | tin}

- Lurette
  - Test and simulation of RP (feedback)
  - Black-box testing
  - Provide realistic inputs to the SUT (constraint solving)
- Lucky
  - The input sequence generator of Lurette V2
- Lutin
  - Higher-level language that compiles into Lucky

Non-deterministic reactive machines

- An interface
- An explicit control structure (weights)
- Constraint define one reaction of the machine
  - Boolean (bdd)
  - Numerics (polyhedra)

What about Simulink?

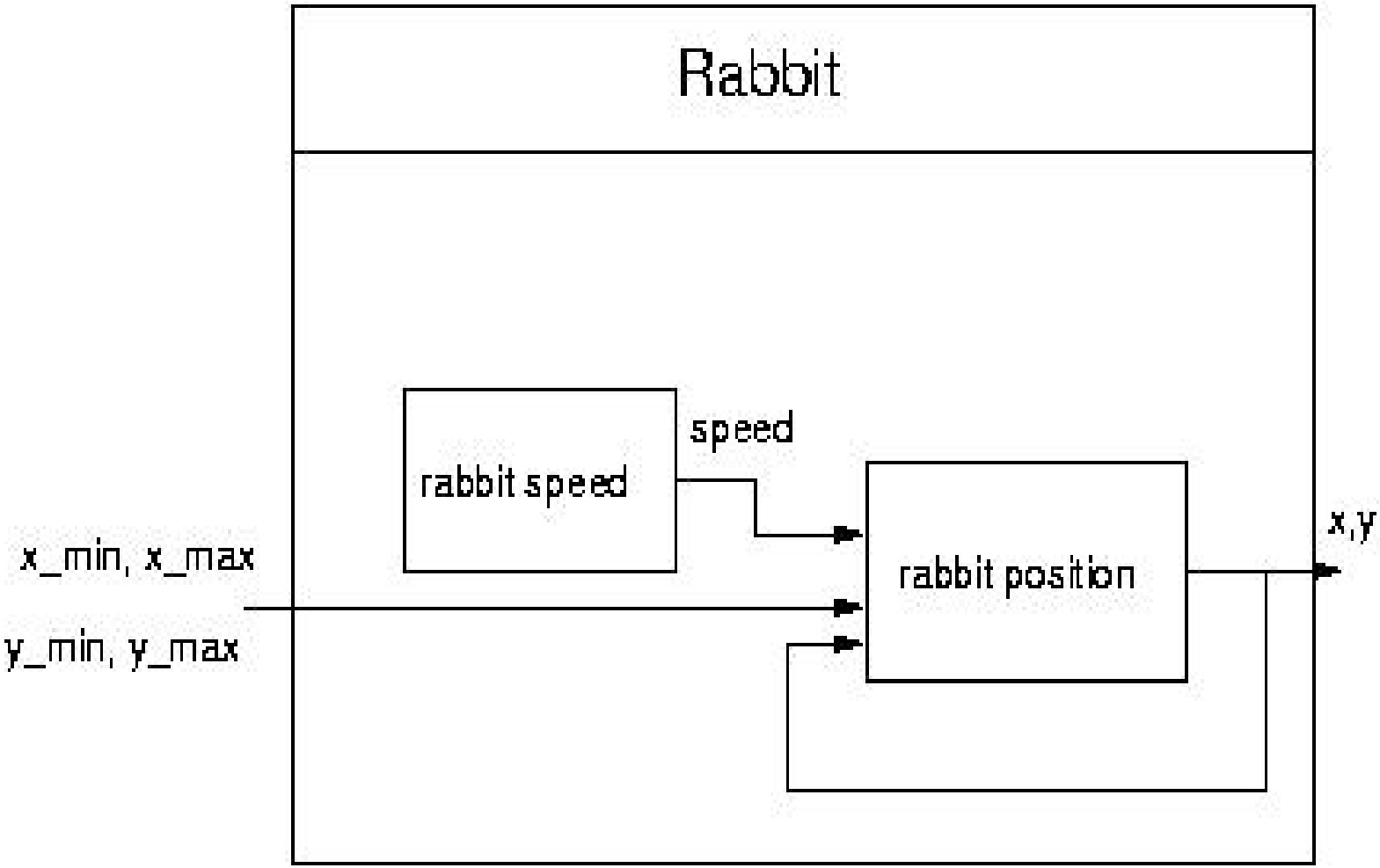
- Simulink
  - is a modelling environment
  - Have a fix set of (random-based) library procedures of stimuli
- Lucky
  - is dedicated to test and simulation
  - is a language to program stimuli

Simulate a Simulink model of a car driver (Caspi)

- Controls the steering wheel
- Feedback
- Looks just one point ahead (no anticipation)
- Therefore it is equivalent to a car following a target

- With Simulink, Paul Caspi stimulated this controller with a target that draws a square (ie, a circle in the Mauras topology)
- We proposed him a more elaborated trajectory

Demo 1: a rabbit running straight ahead



Demo 1 : a rabbit running straight ahead

-pi < a < pi ^

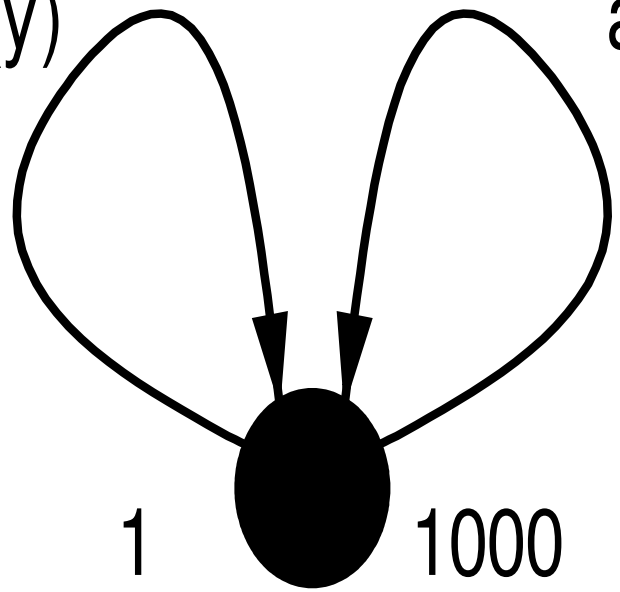
x = pre(x) ^

y = pre(y)

x = pre(x) + speed . cos(a) ^

y = pre(y) + speed . sin(a) ^

a = pre(a)

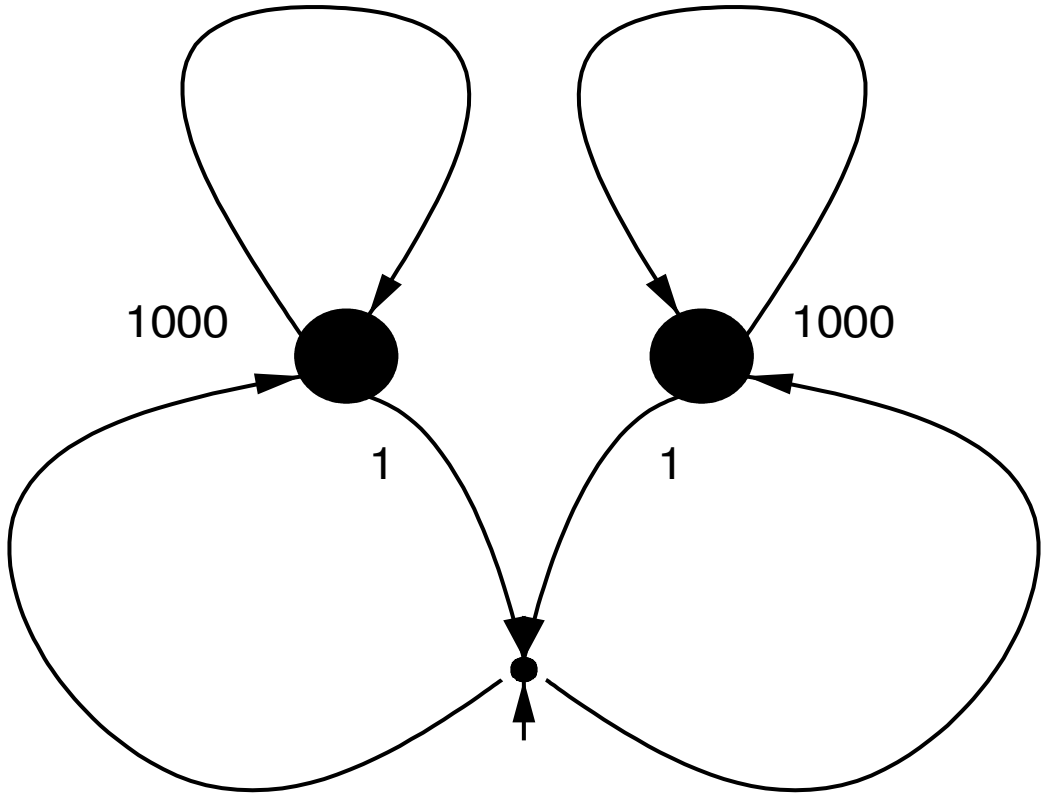


demo...

Demo 2 : a rabbit running drawing lines and curves

x = pre(x) + speed . cos(a) ^
y = pre(y) + speed . sin(a) ^
a = pre(a) + b

x = pre(x) + speed . cos(a) ^
y = pre(y) + speed . sin(a) ^
a = pre(a)



-pi < a < pi ^
-0.3 < b < 0.3 ^
x = pre(x) ^
y = pre(y)

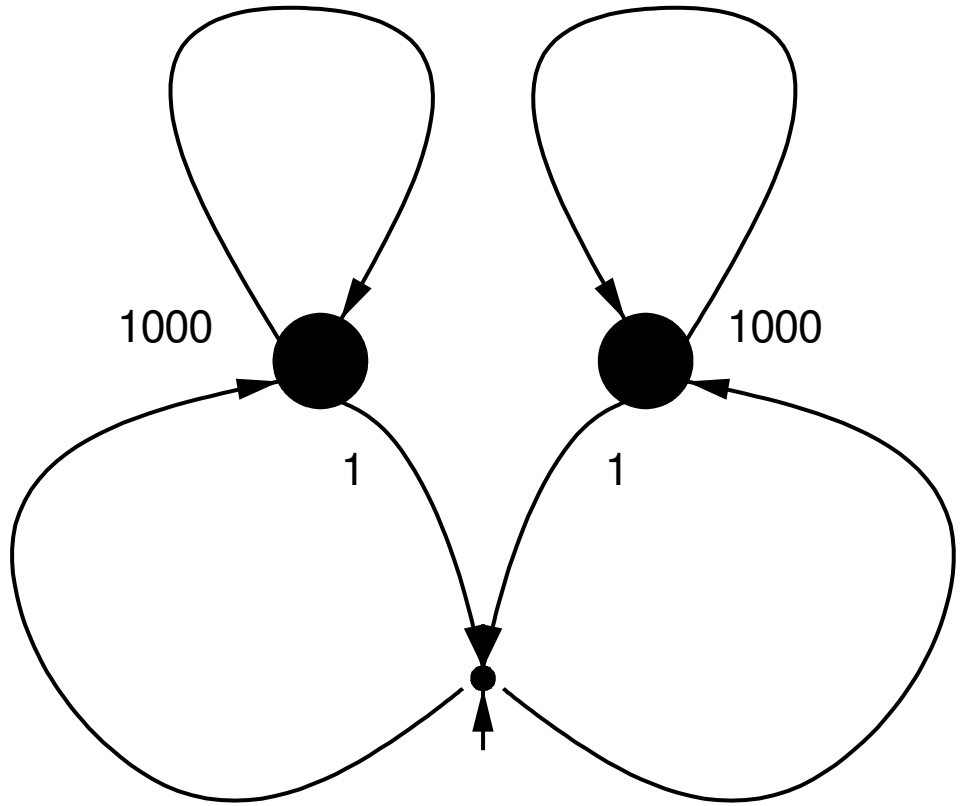
-pi < a < pi ^
x = pre(x) ^
y = pre(y)

demo...

Demo 3 : lines, curves, around an obstacle

obstable ^
x = pre(x) + speed . cos(a) ^
y = pre(y) + speed . sin(a) ^
a = pre(a) - b

obstable ^
x = pre(x) + speed . cos(a) ^
y = pre(y) + speed . sin(a) ^
a = pre(a)



-pi < a < pi ^
-0.3 < b < 0.3 ^
x = pre(x) ^
y = pre(y)

-pi < a < pi ^
x = pre(x) ^
y = pre(y)

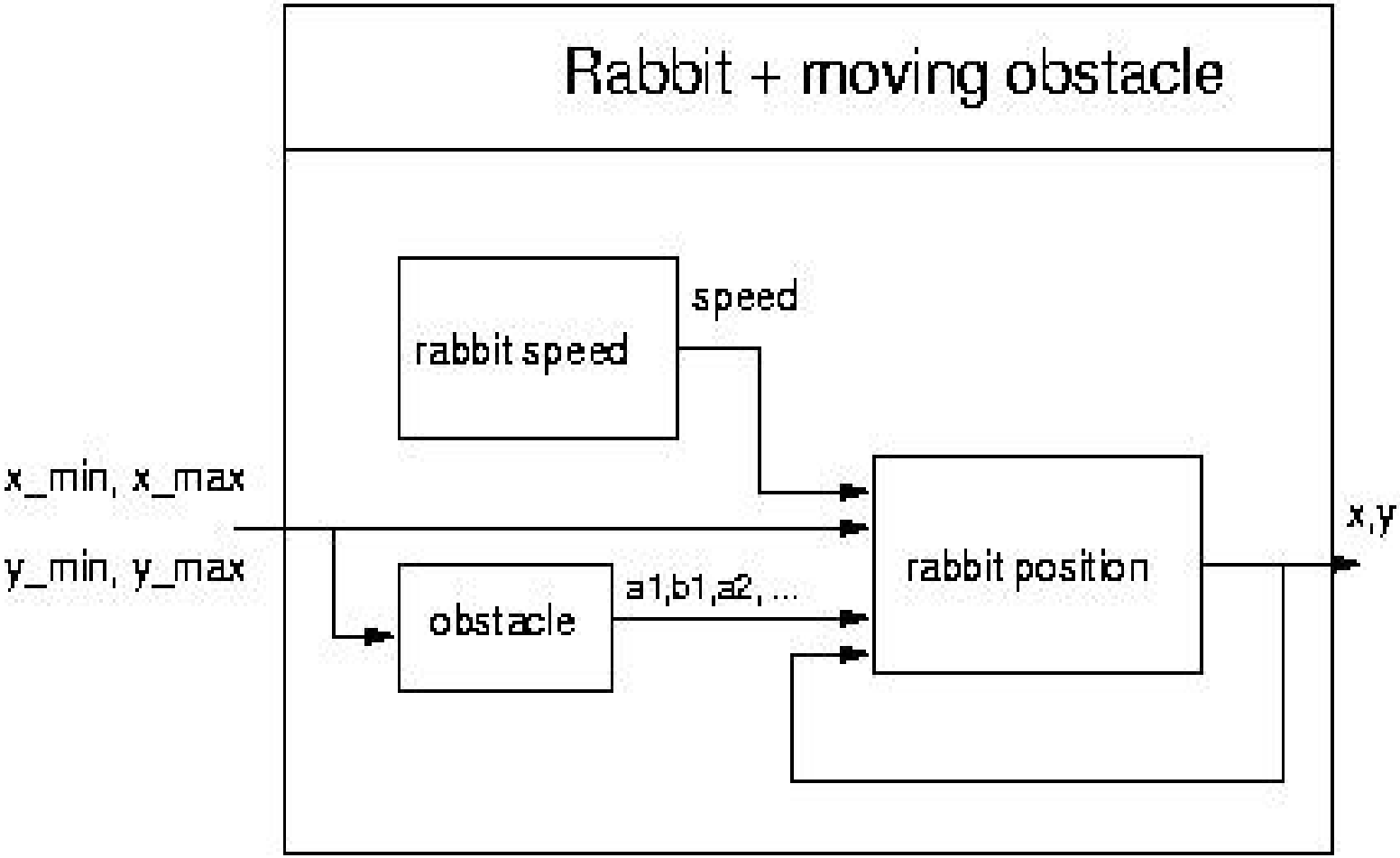
Demo 3 : lines, curves, around an obstacle

Where:

```
Obstacle =
   a1.X + b1 > Y ^
   a2.X + b2 > Y ^
   a3.X + b3 > Y ^
   a4.X + b4 > Y ^
```

demo...

Demo 4: The obstacle is moving



demo...

Conclusion of the demo

- Probably this could have been done with matlab and the Simulink stimuli lib, but...

Entry points

- Lurette
- Lurette/Scade
- Lucky interpreter
- Luc4ocaml (RML)
- Luc2c (Simulink)
- LuckyDraw (API for Ocaml and C)
- Lutin compiler

Conclusion

- Complementary to Simulink
- Several entry points
- Freely available

  `http://www-verimag.fr/~synchron`

Thank you for your attention

Questions?