# ETMF 2023 – FORMAL METHODS: WHAT ARE THEY? WHAT ARE THEY USED FOR?

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil



Centro de Informática UFPE

Universidade Federal de Pernambuco

# Lecturer @ CIn-UFPE, Brazil

Research interests: formal methods (theory and practice)

**CIn-TRUST**

trust.cin.ufpe.br

cin.ufpe.br/~ghpc

ROBOSTAR

robostar.cs.york.ac.uk


Project: **RoboTIC@**

ines.org.br/projects-2

**RoboWorld**: more information (publications, examples, tool, etc.)

robostar.cs.york.ac.uk/roboworld

Centro de Informática UFPE

Universidade Federal de Pernambuco

# A connected (software-based) society

Software and systems should behave as expected

But, do we trust them?

Formal methods to the rescue!

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Small digression: aeroplanes

Source: https://ccntservice.airbus.com/apps/cockpits/a380


Centro de Informática UFPE

Universidade Federal de Pernambuco

# Small digression: aeroplanes

KLM cockpit tales: https://www.youtube.com/watch?v=XXMO1nnlnxs


Centro de Informática UFPE

Universidade Federal de Pernambuco

# Small digression: aeroplanes

Flight tests: https://www.youtube.com/playlist?list=PLjl-u2y72YNwiW9CQ1Tp6yOsR-WkL_QIX


Centro de Informática UFPE

Universidade Federal de Pernambuco

# Formal methods

What about the rigour in:
software system specification, design, and verification?

# Formal methods

What about the rigour in:
software system specification, design, and verification?

> "*Formal methods are mathematical approaches to software and system development which support the* ***rigorous*** *specification, design and verification of computer systems.*"
>
> Formal Methods Europe – FME

An important role on achieving high trustworthiness levels

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Formal methods: not only about aeroplanes

**AWS**: How Amazon web services uses formal methods
https://doi.org/10.1145/2699417

**Google**: Using Formal Methods at Google
https://datatracker.ietf.org/meeting/118/session/ufmrg

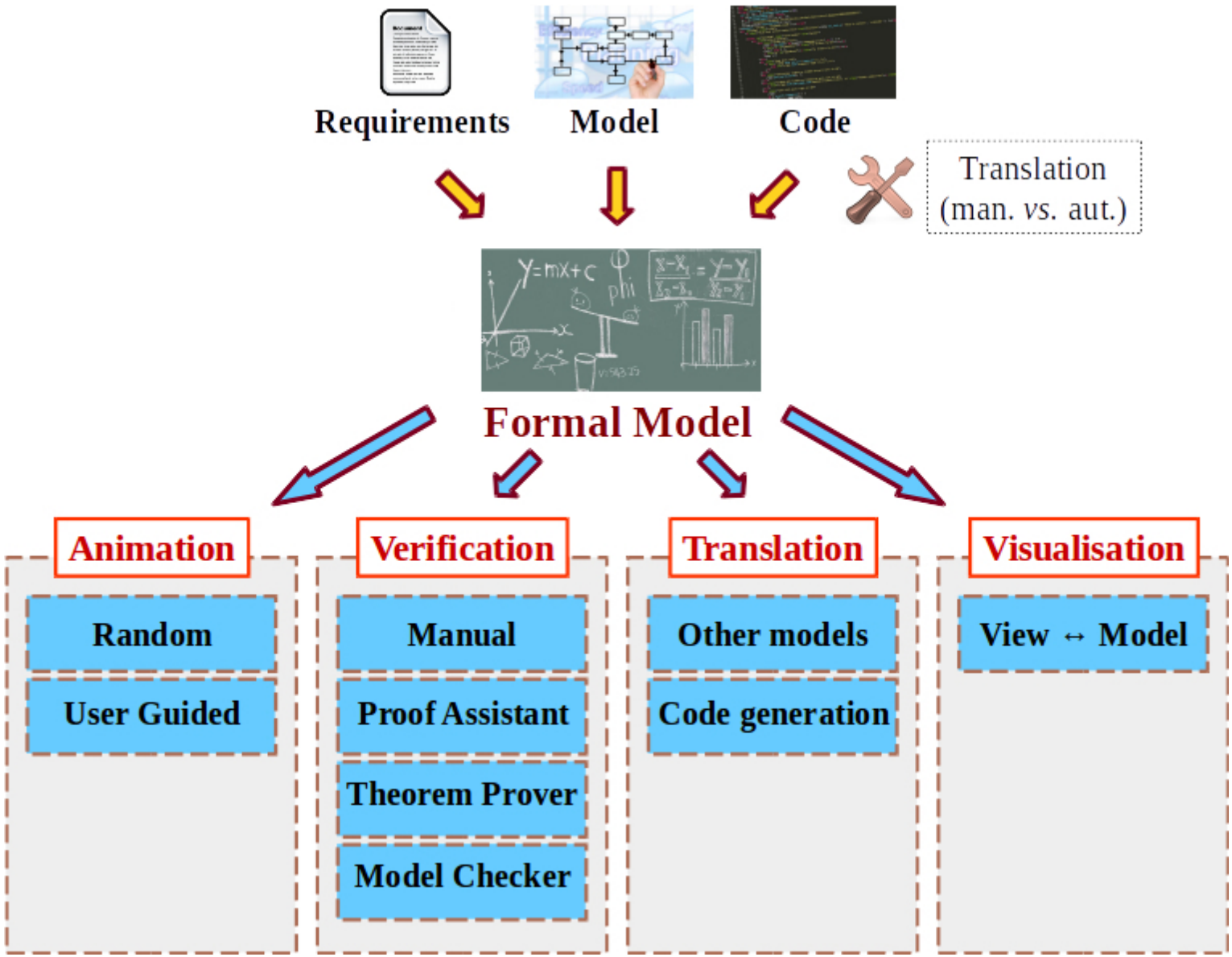
**Meta**: Moving Fast with Software Verification
https://doi.org/10.1007/978-3-319-17524-9_1

**Microsoft**: SMT in Verification, Modeling, and Testing at Microsoft
https://doi.org/10.1007/978-3-642-39611-3_3

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Formal methods: a big picture



# Agenda



Centro de Informática UFPE

Universidade Federal de Pernambuco

# B language

Focus: safety-critical software (single threaded)

Application: numerous applications, particularly in the railway domain (https://link.springer.com/chapter/10.1007/978-3-030-58298-2_8)

Tools

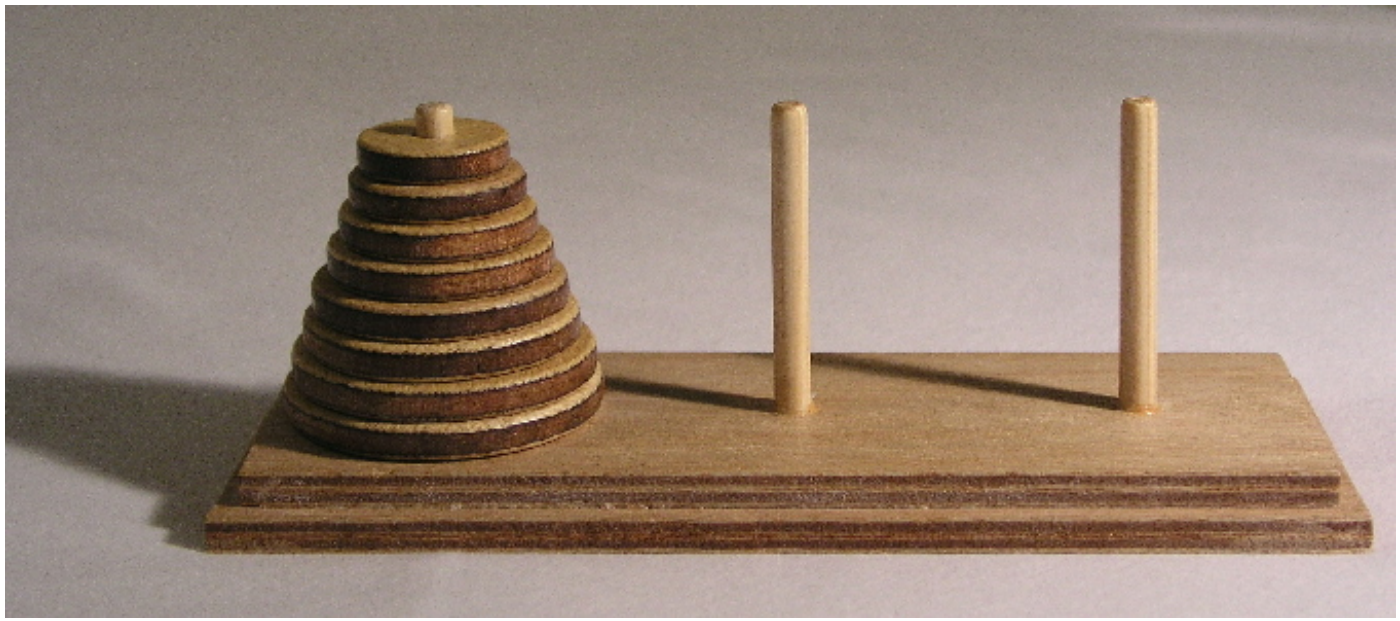
- Atelier B: https://atelierb.eu/
- ProB: https://prob.hhu.de/w/index.php
- BMotionWeb: https://prob.hhu.de/w/index.php?title=BMotionWeb

Centro de Informática UFPE

Universidade Federal de Pernambuco

# B language: example

Tower of Hanoi puzzle[1]



[1] Source: Wikimedia Commons

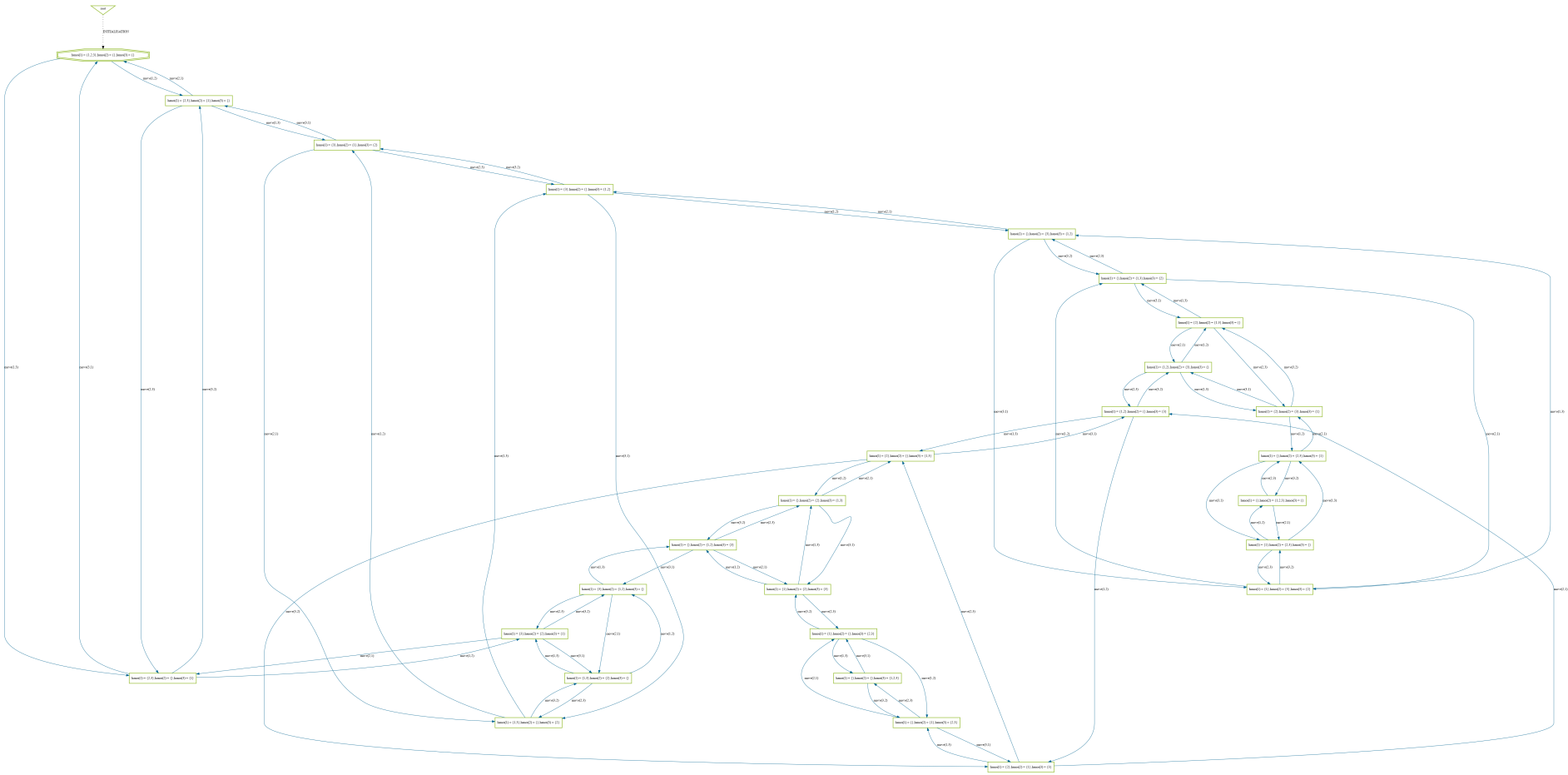
# B language: specification

```
 7- VARIABLES
 8      hanoi
 9- INVARIANT
10      // System state is described as a function from pegs (1..3) to disks (1..3)
11      // 1 denotes the smaller disk, whereas 3 denotes the biggest one
12      hanoi : 1..3 --> POW(1..3)
13      // There is no disk repetition between pegs
14      & ! (p1,p2) . ((p1 : dom(hanoi) & p2 : dom(hanoi) & p1 /= p2 => hanoi(p1) /\ hanoi(p2) = {}))
15      // Is there a solution?
16      & not((hanoi(1) = {}) & (hanoi(2) = {}) & (hanoi(3) = 1..3))
17- INITIALISATION
18      // Initially, all disks on the first peg
19      hanoi := {1 |-> {1,2,3}, 2 |-> {}, 3 |-> {}}
20- OPERATIONS
21      // Moving disks between pegs
22      move(from,to) =
23-     PRE
24          // The source and target pegs shall be different
25          from : dom(hanoi) & to : dom(hanoi) & from /= to
26          // The source peg is not empty
27          & hanoi(from) /= {}
28          // You cannot place a bigger disk over a smaller one
29          & (hanoi(to) /= {} => min(hanoi(from)) < min(hanoi(to)))
30-     THEN
31          hanoi := hanoi <+
32                   // Updating the state of the source and target pegs
33-                {
34                     from |-> (hanoi(from) - {min(hanoi(from))}),
35                     to |-> (hanoi(to) \/ {min(hanoi(from))})
36                   }
37      END
```

Centro de Informática UFPE

Universidade Federal de Pernambuco

# B language: state space



28 states and 79 transitions
Graph generated with ProB

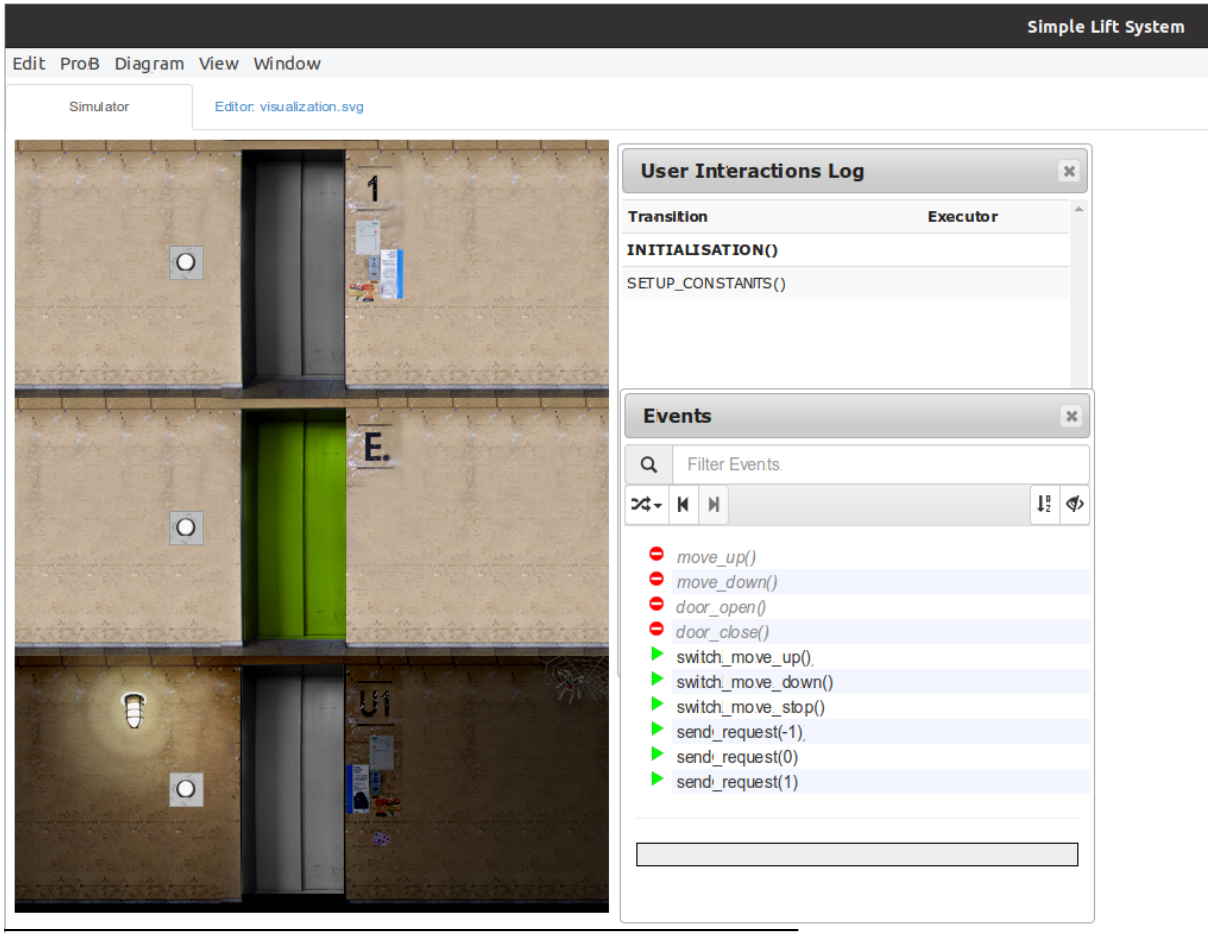
```
$ dot -Tpng Hanoi.dot > Hanoi.png
```

Centro de Informática UFPE

Universidade Federal de Pernambuco

# B language: visualisation

Another example: lift[2]




[2]Source: BMotionWeb examples

# Agenda



Centro de Informática UFPE

Universidade Federal de Pernambuco

# CSP

Focus: concurrent systems

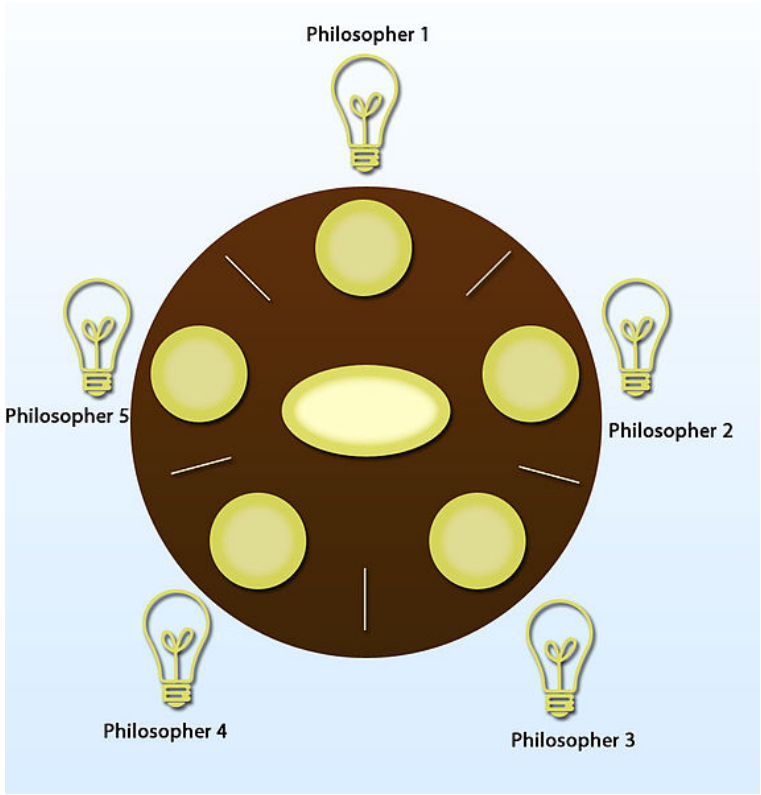
Application: numerous applications
(https://doi.org/10.1007/978-3-319-51046-0_4)

Tools

- FDR: https://cocotec.io/fdr/

# CSP: example

## Dining philosophers problem[3]



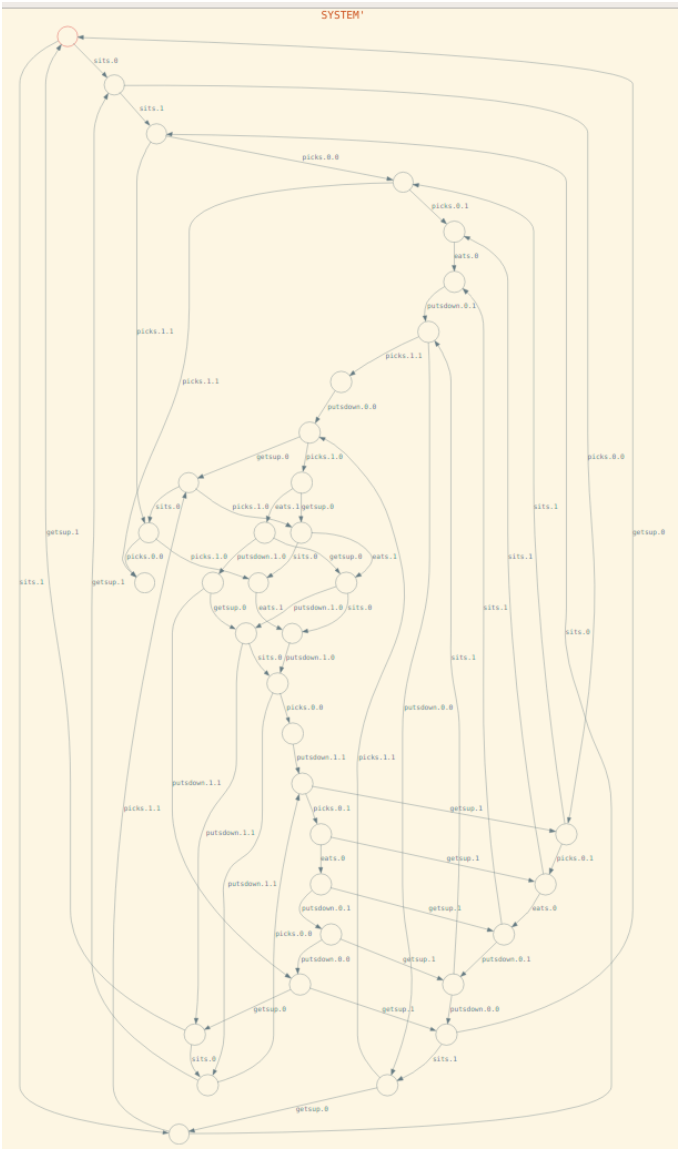

[3] Source: Wikimedia Commons

# CSP: specification[4]

```
1  -- Bill Roscoe
2  N = 4
3
4  PHILNAMES= {0..N-1}
5  FORKNAMES = {0..N-1}
6
7  channel sits, eats, getsup : PHILNAMES
8  channel picks, putsdown : PHILNAMES.FORKNAMES
9
10 PHIL(i) = sits!i -> picks!i!i -> picks!i!((i+1)%N) ->
11           eats!i -> putsdown!i!((i+1)%N) -> putsdown!i!i -> getsup!i -> PHIL(i)
12
13 FORK(i) = picks!i!i -> putsdown!i!i -> FORK(i)
14          []  picks!((i-1)%N)!i -> putsdown!((i-1)%N)!i -> FORK(i)
15
16 PHILS = ||| i:PHILNAMES @ PHIL(i)
17 FORKS = ||| i:FORKNAMES @ FORK(i)
18
19 SYSTEM' = PHILS[|{|picks, putsdown|}|]FORKS
20
21 -- The potential for deadlock is illustrated by
22
23 assert SYSTEM' :[deadlock free [F]]
24
```

[4] Code from FDR Documentation

# CSP: state space




N = 2

- States: 36
- Transitions: 65

N = 3

- States: 214
- Transitions: 565

N = 4

- States: 1,296
- Transitions: 4,569

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Agenda



Centro de Informática UFPE

Universidade Federal de Pernambuco

# Timed automata

Focus: timed systems

Application: numerous applications
(https://uppaal.org/casestudies/)

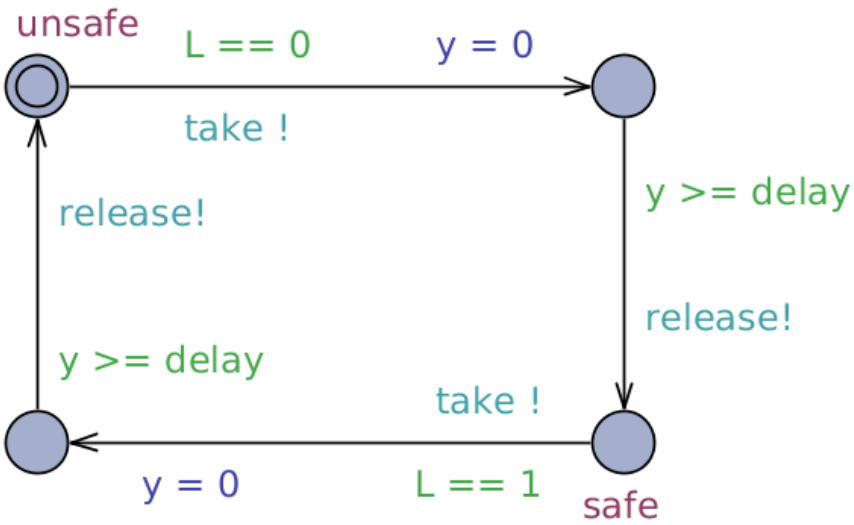
Tools

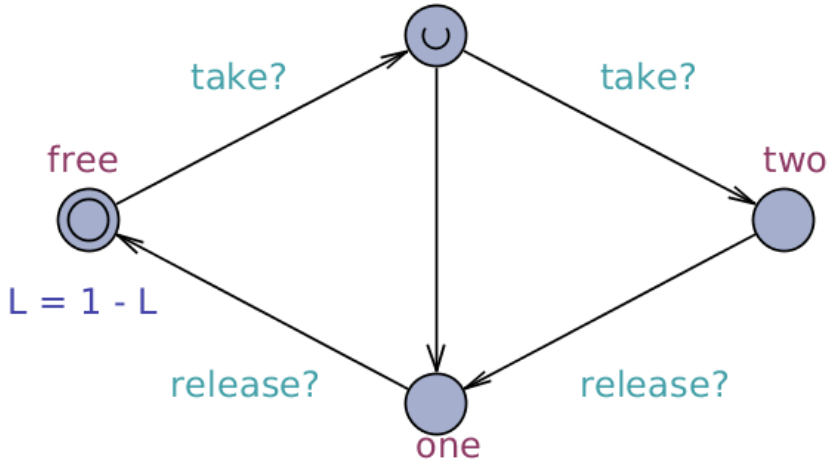
- UPPAAL: https://uppaal.org/

# Timed automata: example

Bridge crossing puzzle (Vikings version)

# Timed automata: specification[5]




[5] Source: UPPAAL demos

# Agenda



Centro de Informática UFPE

Universidade Federal de Pernambuco

# SMT-LIB v2

"*SMT-LIB is an international initiative aimed at facilitating research and development in Satisfiability Modulo Theories (SMT).*"

Focus: determining whether a mathematical formula is satisfiable
(https://en.wikipedia.org/wiki/Satisfiability_modulo_theories)

Application: numerous applications

Tools: numerous tools

- Z3:
  www.microsoft.com/en-us/research/project/z3-3/

Centro de Informática UFPE

Universidade Federal de Pernambuco

# SMT-LIB v2: example

$$3x + 2y = 6$$
$$2x - 3y + 5$$

```
(declare-const x Real)
(declare-const y Real)
(assert (= (+ (* 3 x) (* 2 y)) 6))
(assert (= (- (* 2 x) (* 3 y)) 5))
(check-sat)
(get-model)
(exit)
```

**Output**:

```
sat
(
  (define-fun y () Real (- (/ 3.0 13.0)))
  (define-fun x () Real (/ 28.0 13.0))
)
```

Centro de Informática UFPE

Universidade Federal de Pernambuco

# SMT-LIB v2: example

$$(p \Rightarrow q) \wedge (q \Rightarrow p)$$
$$\neg\,(p = q)$$

```
(declare-const p Bool)
(declare-const q Bool)
(assert (and (=> p q) (=> q p)))
(assert (not (= p q)))
(check-sat)
(exit)
```

**Output**:

```
unsat
```

# Agenda



Centro de Informática UFPE

Universidade Federal de Pernambuco

# Gallina

Focus: general purpose specification language

Application: numerous applications

(DeepSpec: https://deepspec.org/main)

Tools

- Coq proof assistant: https://coq.inria.fr/

# Gallina: example

Implementing dictionary ADT using binary search trees

```
1  clear      : dictionary -> dictionary
2  is_empty   : dictionary -> bool
3  size       : dictionary -> nat
4  insert     : key -> val -> dictionary -> dictionary
5  find       : key -> dictionary -> option val
```

# Gallina: specification, implementation and proofs

```
21 Module Type Dictionary.
22
23   (* [key]: the type of keys *)
24   Definition key              := nat.
25   (* [val]: the type of values *)
26   Definition val              := string.
27   (* [dictionary]: the type of dictionaries *)
28   Parameter dictionary        : Type.
29   (* [empty_dictioanry]: definition of an empty dictionary *)
30   Parameter empty_dictionary  : dictionary.
31
32 (**
33   Operations suppoted: [clear], [is_empty],
34                        [size], [insert], [find]
35 *)
36
37   Parameter clear     : dictionary -> dictionary.
38   Parameter is_empty  : dictionary -> bool.
39   Parameter size      : dictionary -> nat.
40   Parameter insert    : key -> val -> dictionary -> dictionary.
41   Parameter find      : key -> dictionary -> option val.
```

Centro de Informática UFPE

Universidade Federal de Pernambuco

# Agenda

# Conclusion

Mathematical approaches to support the **rigorous** specification, design and verification of computer systems

In this talk:

- B language: Atelier B, ProB, BMotionWeb
- CSP: FDR4
- Timed automata: UPPAAL
- SMT-LIB v2: Z3
- Gallina: Coq

Centro de Informática UFPE

Universidade Federal de Pernambuco

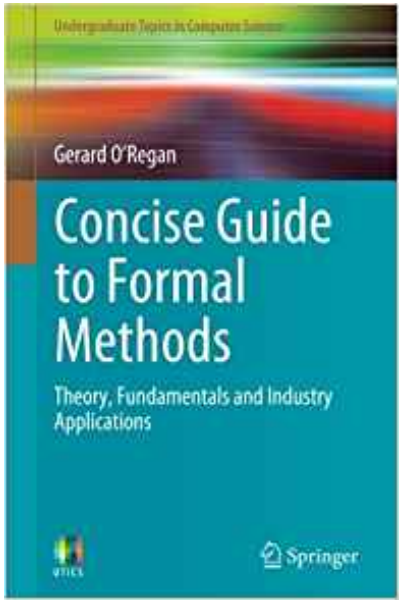
# Recommended reading: papers

**Formal methods in dependable systems engineering: a survey of professionals from Europe and North America**
Mario Gleirscher, Diego Marmsoler.
Empirical Software Engineering 25, pages 4473–4546. Springer 2020.

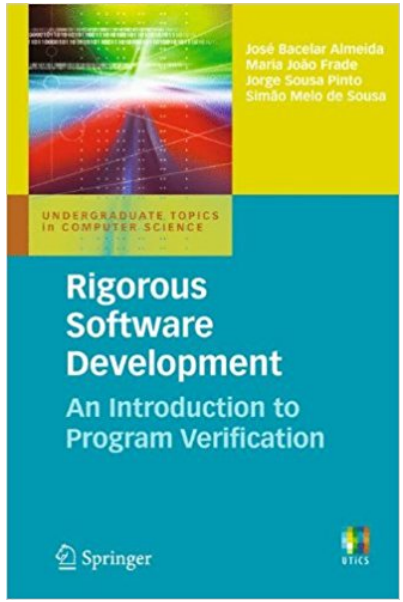
**The 2020 Expert Survey on Formal Methods**
Hubert Garavel, Maurice H. ter Beek, Jaco van de Pol.
Proc. Formal Methods for Industrial Critical Systems (FMICS 2020).
LNCS 12327, Springer 2020

Centro de Informática UFPE
Universidade Federal de Pernambuco

# Recommended reading: books




**Gerard O'Regan**
*Concise Guide to Formal Methods*
Springer, 2017.




**José Almeida et al.**
*Rigorous Software Development*
Springer, 2011.

Centro de Informática UFPE

Universidade Federal de Pernambuco

# ETMF 2023 – FORMAL METHODS: WHAT ARE THEY? WHAT ARE THEY USED FOR?

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil


Centro de Informática UFPE

Universidade Federal de Pernambuco