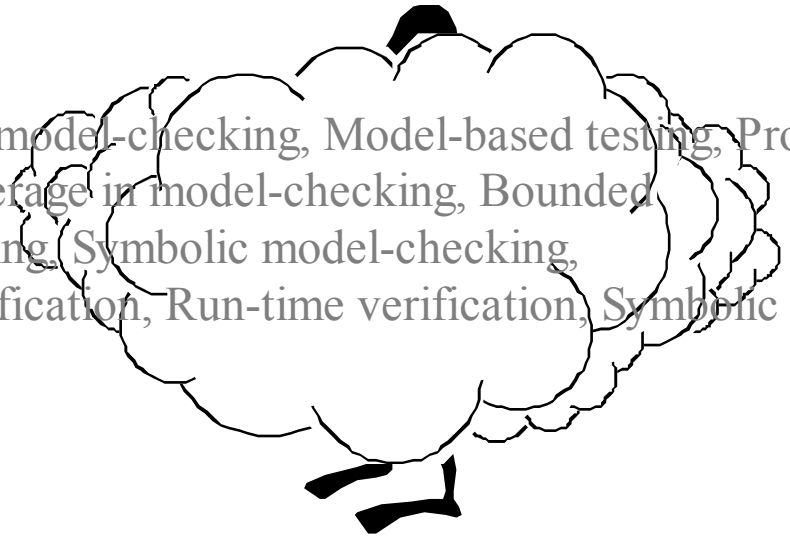
Checking Models, Proving Programs, and Testing Systems

Marie-Claude Gaudel

LRI, Université de Paris-Sud &
CNRS

It's a talk on terminology…

ProSoftware model-checking, Model-based testing, Property Testing, Coverage in model-checking, Bounded model-checking, Symbolic model-checking, program Verification, Run-time verification, Symbolic execution…

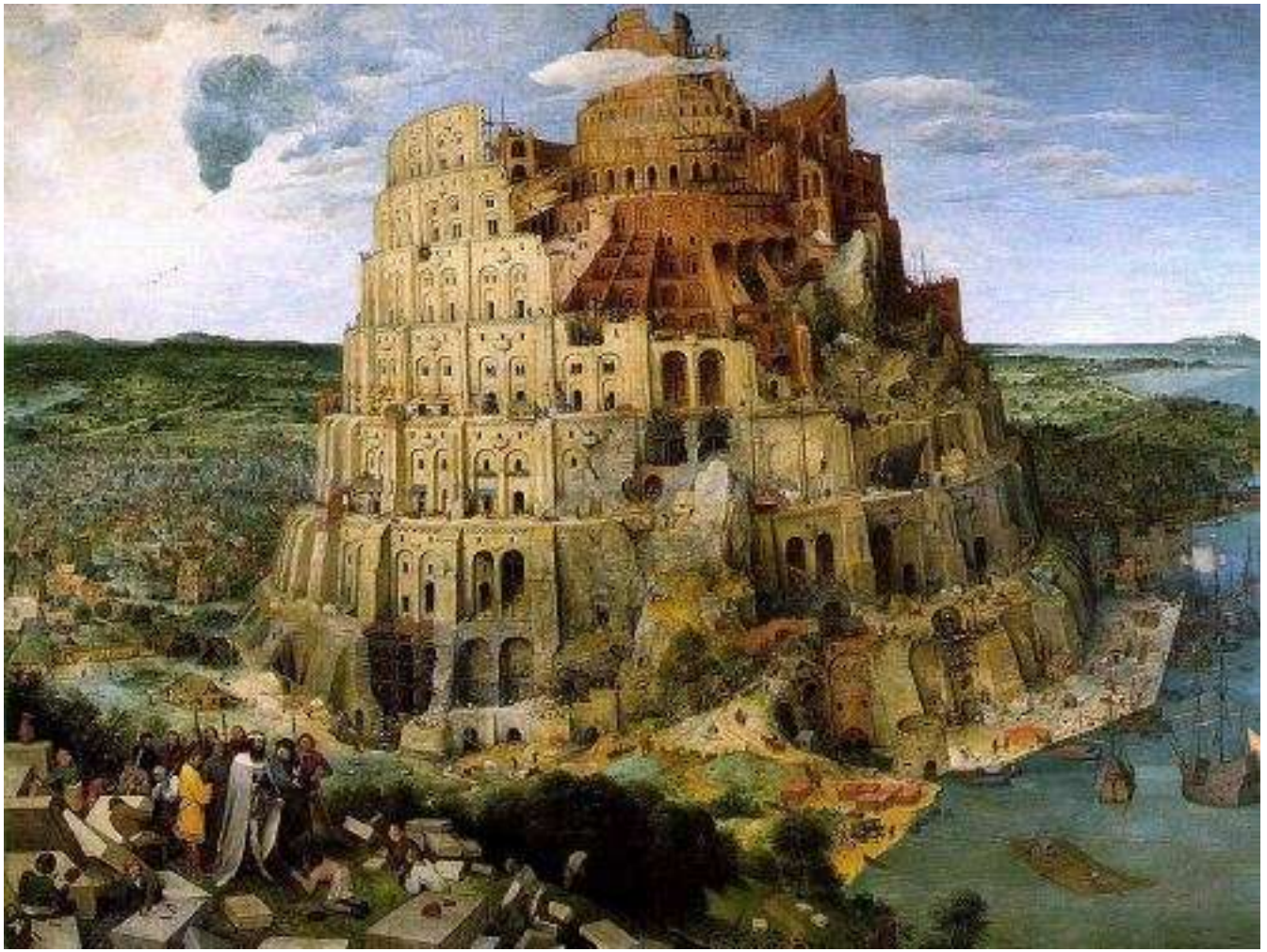


Do we understand each other ?

Yes of course, but it could be better.

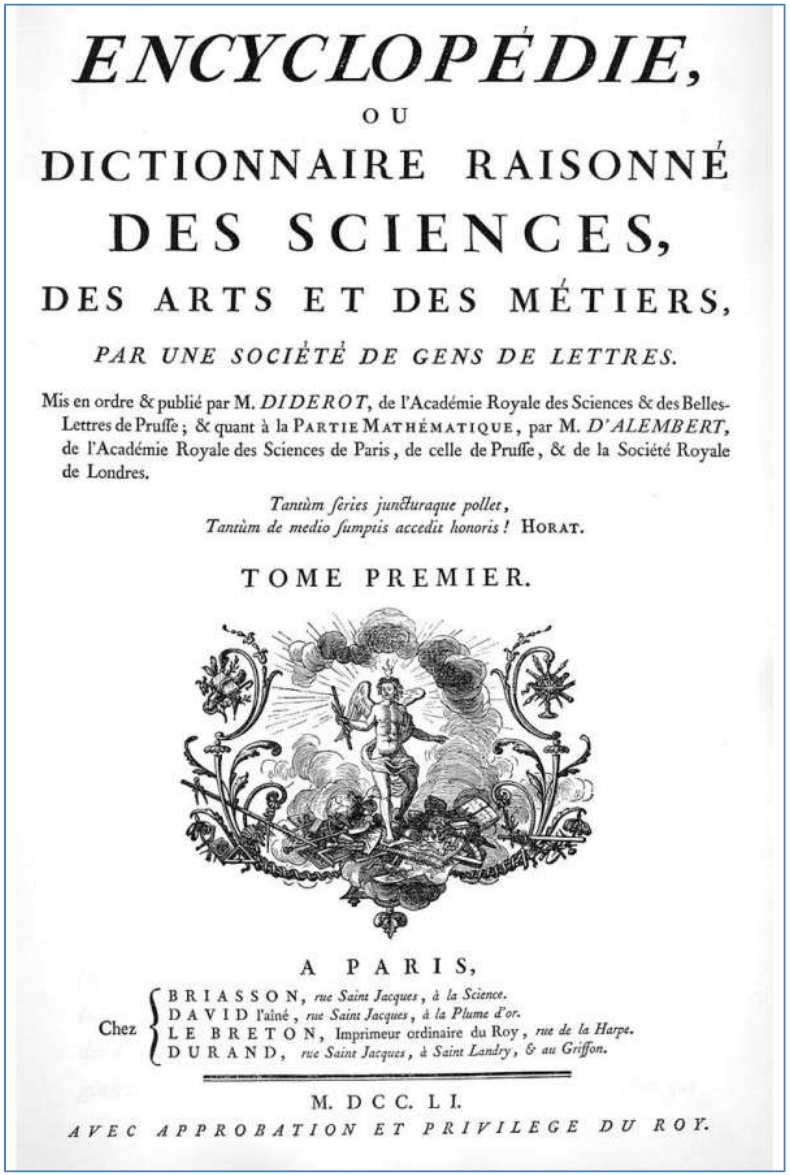
LABORATOIRE DE RECHERCHE EN INFORMATIQUE

The Babel Tower



LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Another solution ?



ENCYCLOPÉDIE,

OU

DICTIONNAIRE RAISONNÉ

DES SCIENCES,

DES ARTS ET DES MÉTIERS,

PAR UNE SOCIÉTÉ DE GENS DE LETTRES.

Mis en ordre & publié par M. *DIDEROT*, de l'Académie Royale des Sciences & des Belles-Lettres de Prusse ; & quant à la PARTIE MATHÉMATIQUE, par M. *D'ALEMBERT*, de l'Académie Royale des Sciences de Paris, de celle de Prusse, & de la Société Royale de Londres.

Tantùm series junctúraque pollet,
Tantùm de medio sumptis accedit honoris ! HORAT.

TOME PREMIER.

A PARIS,

Chez
BRIASSON, *rue Saint Jacques, à la Science.*
DAVID l'aîné, *rue Saint Jacques, à la Plume d'or.*
LE BRETON, Imprimeur ordinaire du Roy, *rue de la Harpe.*
DURAND, *rue Saint Jacques, à Saint Landry, & au Griffon.*

M. DCC. LI.

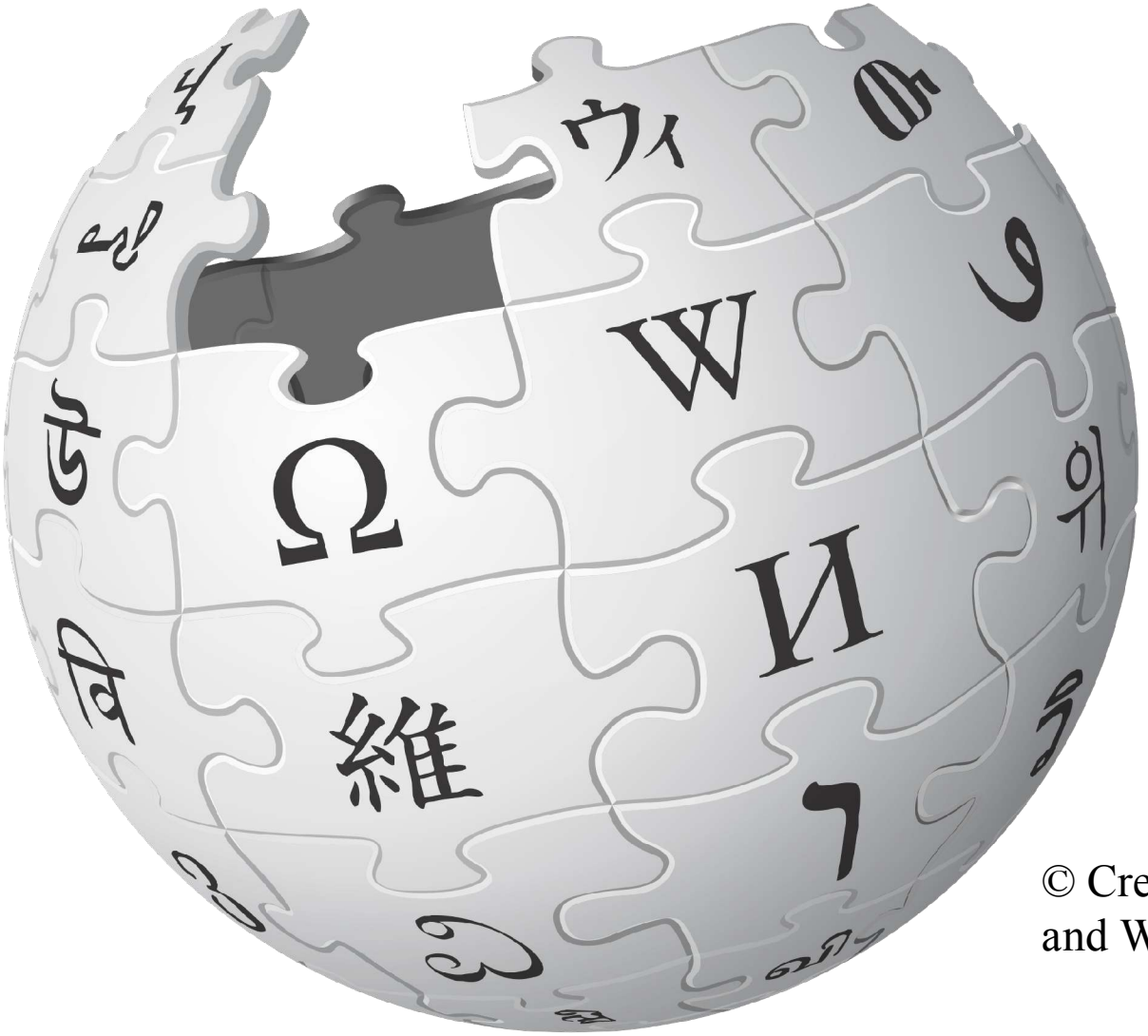
AVEC APPROBATION ET PRIVILEGE DU ROY.

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

A more recent version

© Creative Common

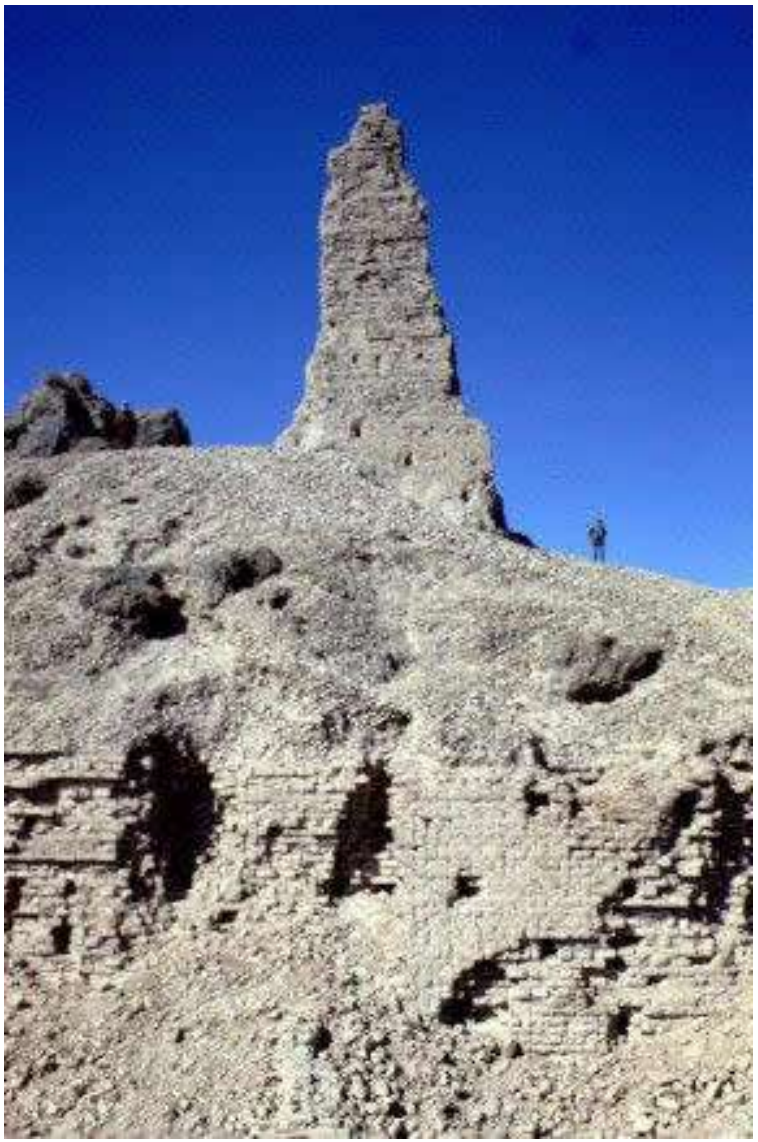
An even more recent version



© Creative Common
and WIKIPEDIA

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

How the Babel tower ended…



An example of what can be done

- Avizienis A., Laprie J.-C., Randell B., Landwehr C.: ***Basic concepts and taxonomy of dependable and secure computing***. IEEE Trans. on Dependable and Secure Computing 1, 11--33 (2004)
- Joint committee by the TC on Fault-Tolerant Computing of the *IEEE CS* and *IFIP WG 10.4* "Dependable Computing and Fault Tolerance."
- Several special sessions in the main conferences of the domain.
- Continuous update, following the state of the art.

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Outline of the talk

- Discussion of some hopefully "clear" definitions
  - Models, Programs, Systems
  - Model-checking, Program proving, Testing
- Not so clear variants of the definitions above
  - *Run-time verification, Software model-checking, Coverage in model-checking, Bounded model-checking, Model-based testing, Program checking, Proof approximation…*
- A few words on:
  - Model-based testing
  - Concolic testing

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Some “clear” definitions

- **Models**
- Programs
- Systems

- Model-checking
- Program proving
- System testing

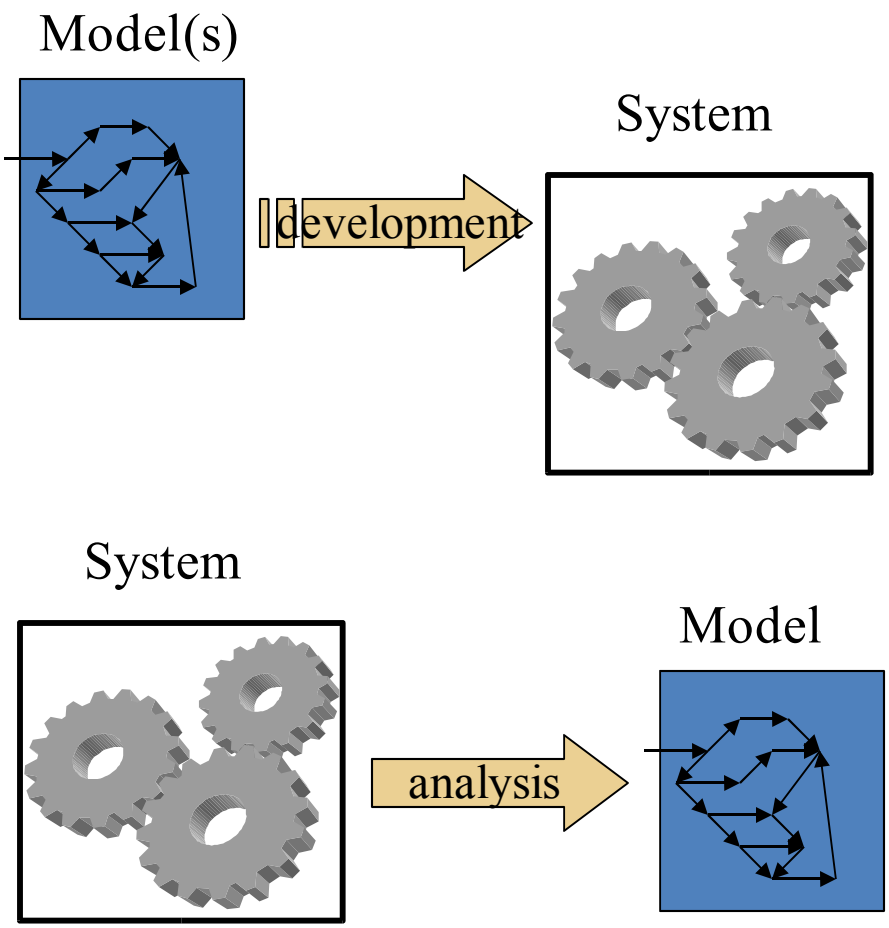
Models: an heavily overloaded* term

- In this talk, models – as they are used for model-checking – are just *annotated graphs*
  - A finite set of states, S
  - Some initial state, s_0
  - A transition relation between states, $T \subseteq S \times S$
  - A finite set of atomic propositions, AP
  - A labelling function $L : S \rightarrow \mathscr{P}(AP)$
  - Richer similar notions:
    - Labelled Transition systems, LTS
    - Finite State machines, FSM
    - State charts, …

** For a physicist a "model" is a differential equation; For a biologist, it may be … mice or frogs*

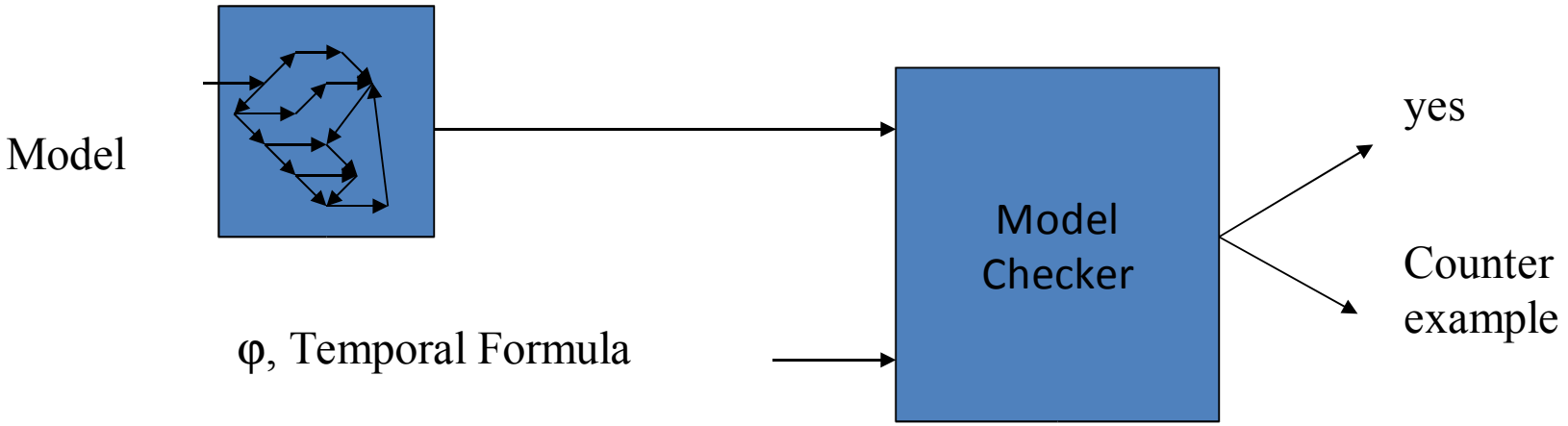
Why are models useful?

- System description and design:
  - The future system must conform to the model(s)
  - The model(s) is used as a starting point for (±automatic) development
- System analysis
  - Observing the existing system, one extracts a model and studies it
- Essential role in V and V and quality assessment



Model-checking

Does the *model* satisfy a formula?
Techniques: exhaustive search (or variants)



If yes, does the *system* satisfy the property?
Exhaustivity, but it is w.r.t. the model…

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Some “clear” definitions (cont.)

- Models
- **Programs**
- Systems

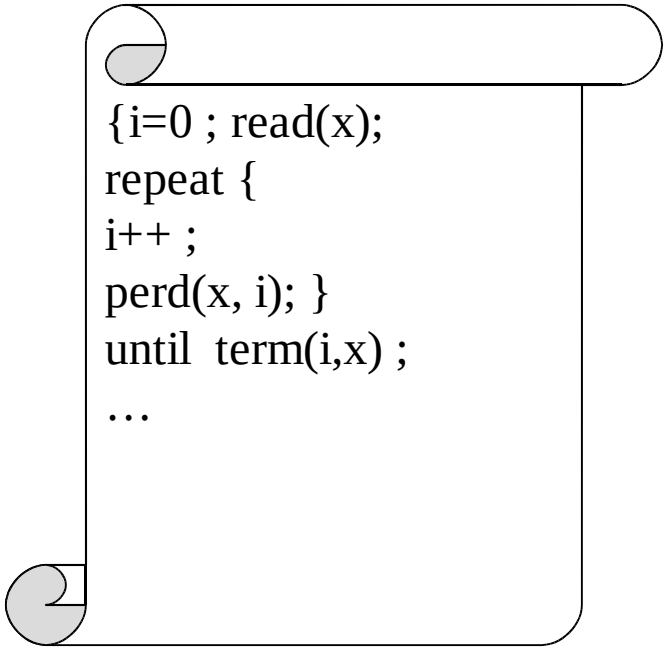
- Model-checking
- Program proving
- Testing

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Programs

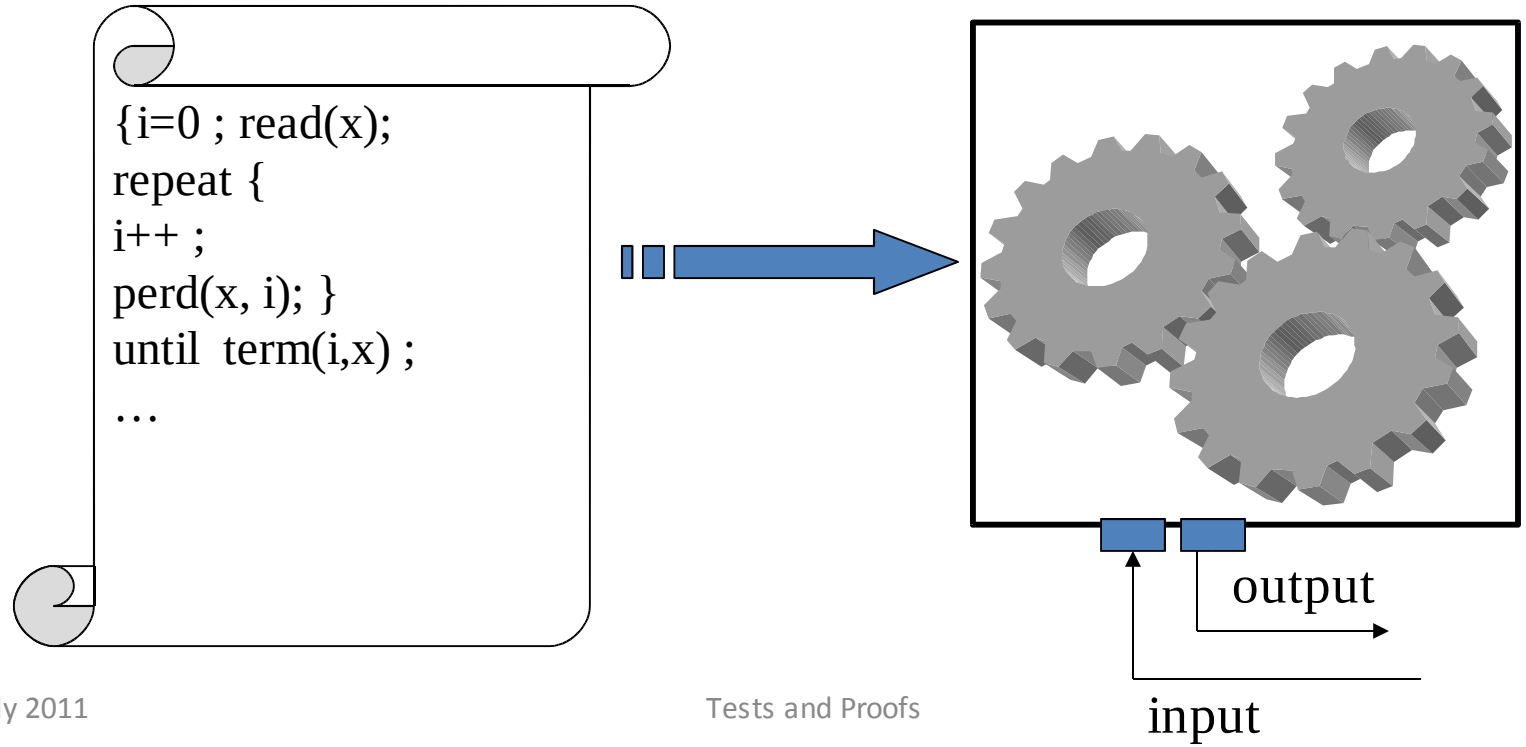
- Everybody knows what it is ☺
  - A program is a piece of text in a (hopefully) well defined language
  - There is a syntax, some semantics, and **compilers**

- "A program is a very detailed solution to a much more abstract problem" [Ball, 2005]

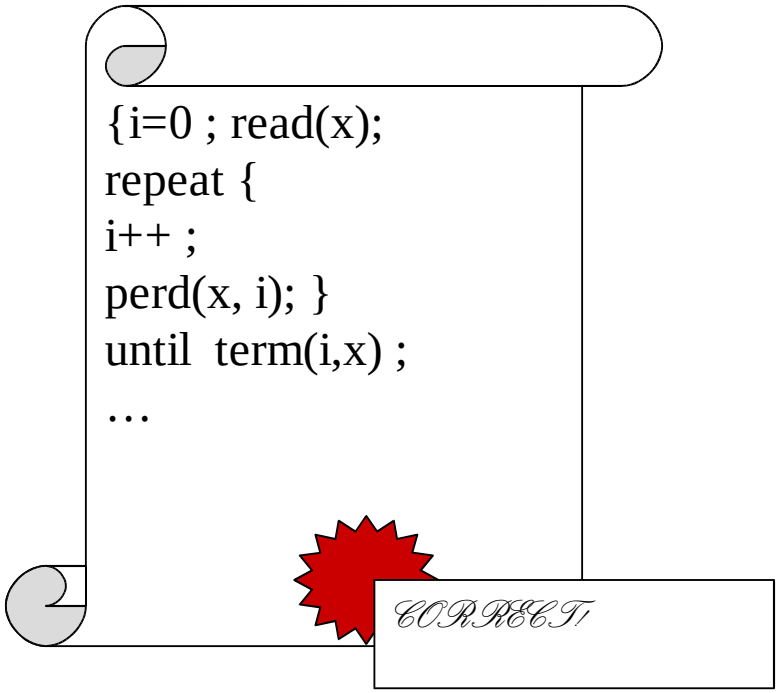


Why are programs useful?

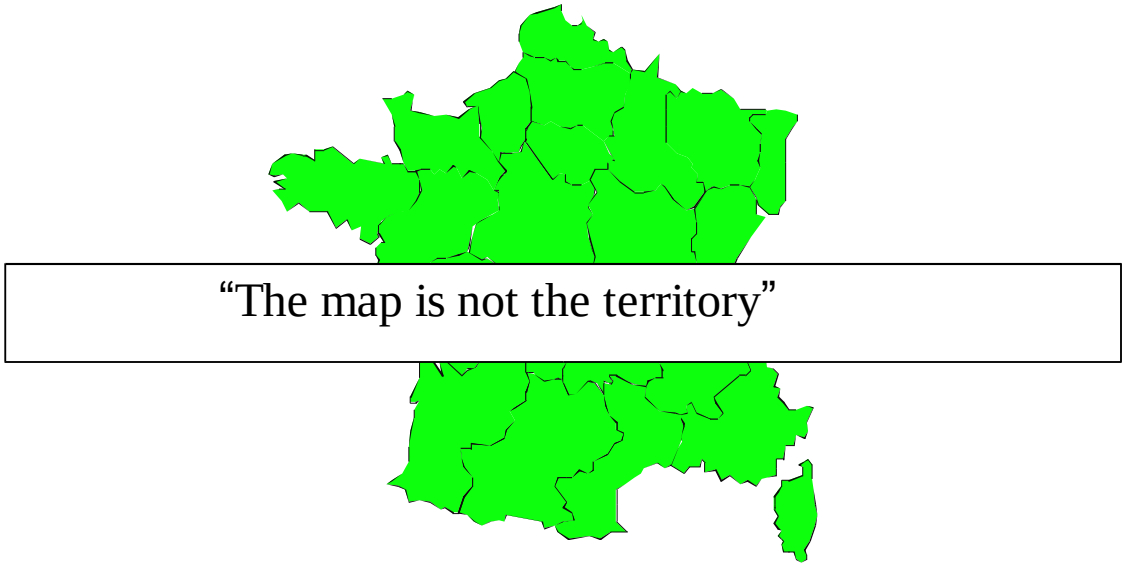
- They can be compiled and embedded into some system



Interlude

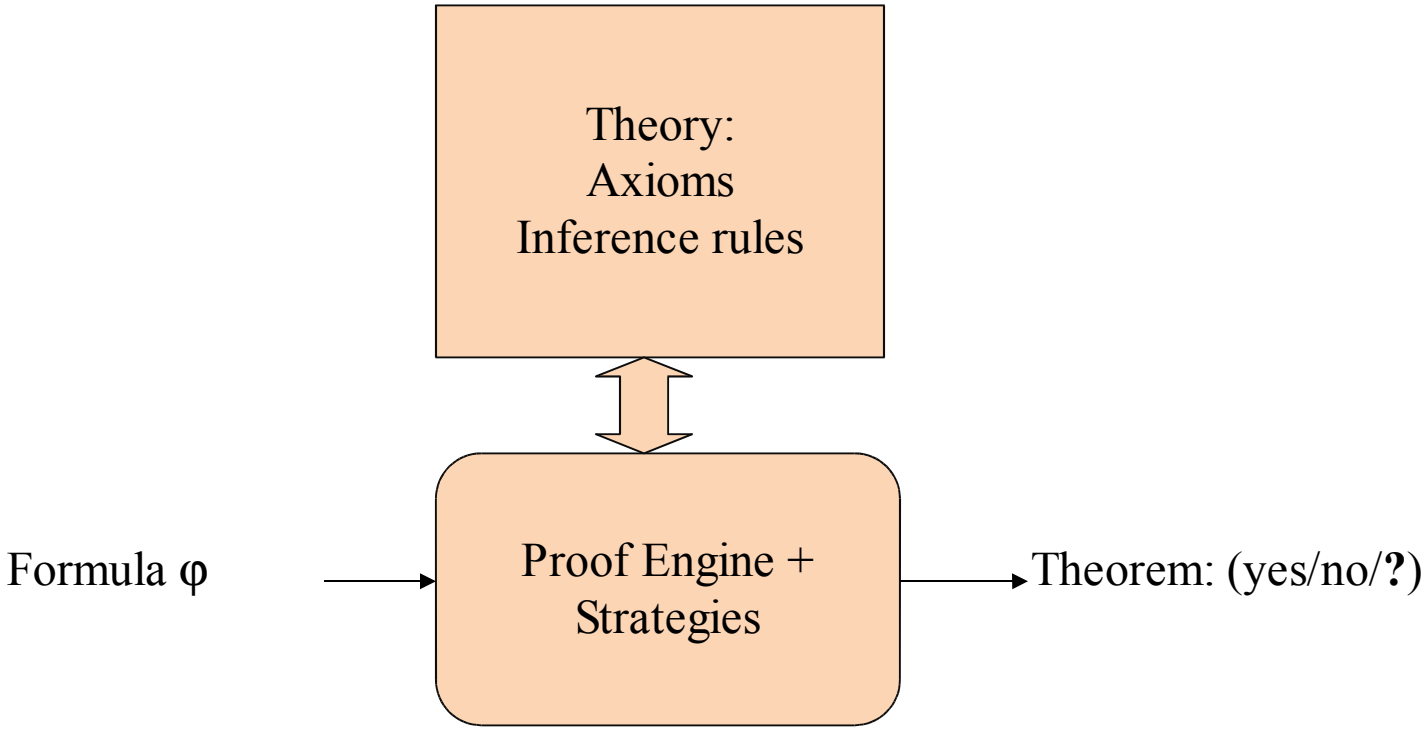


Interlude (cont)



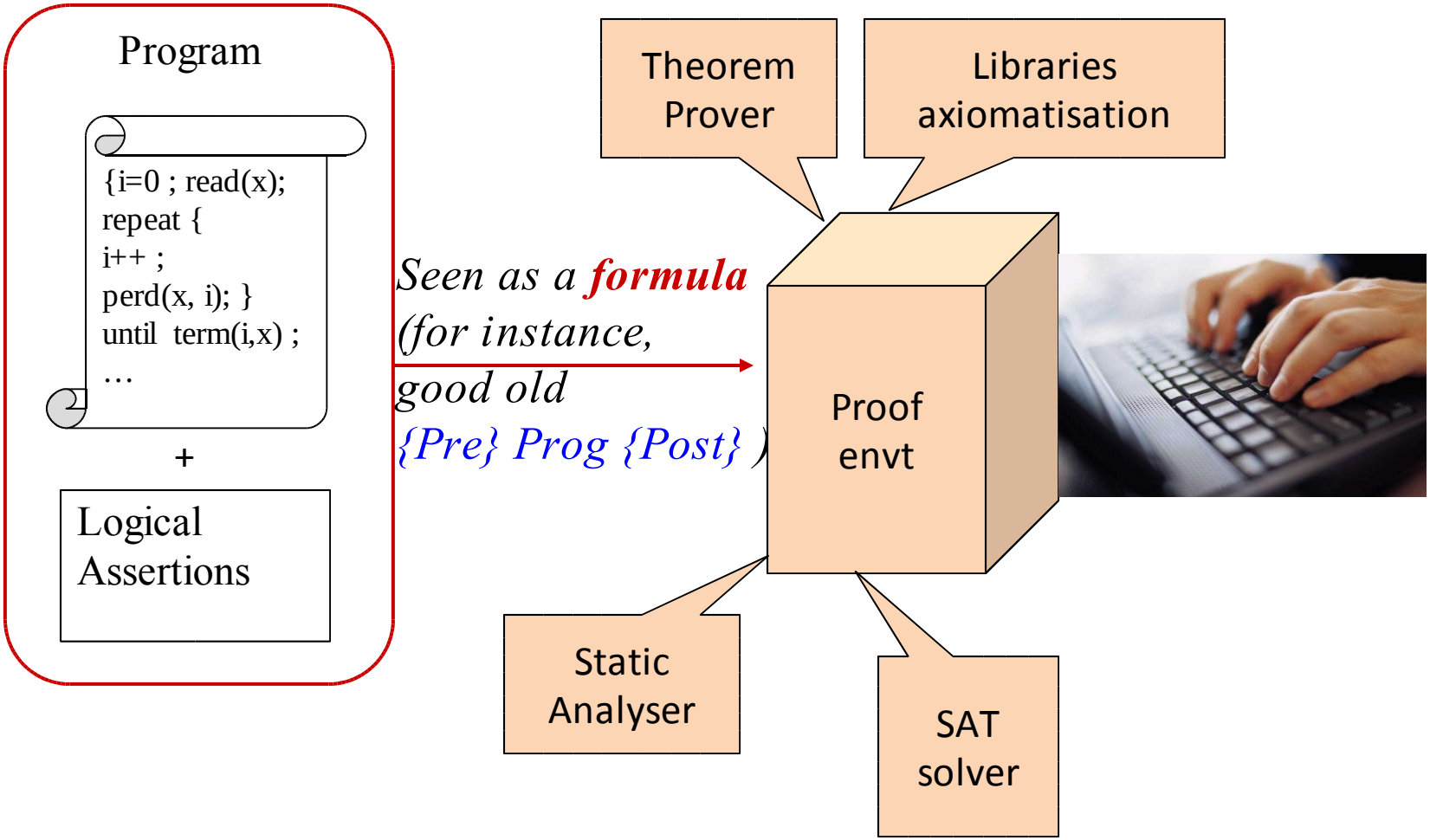
- A program text, or a specification text, or a model, *is not the system*
- Beware of hidden assumptions…
- "You cannot trust the compiler"[Godefroid 2011] … nor the execution environment.

What is a proof?



Basically: a sophisticated syntactic process

Program Proving



LABORATOIRE DE RECHERCHE EN INFORMATIQUE

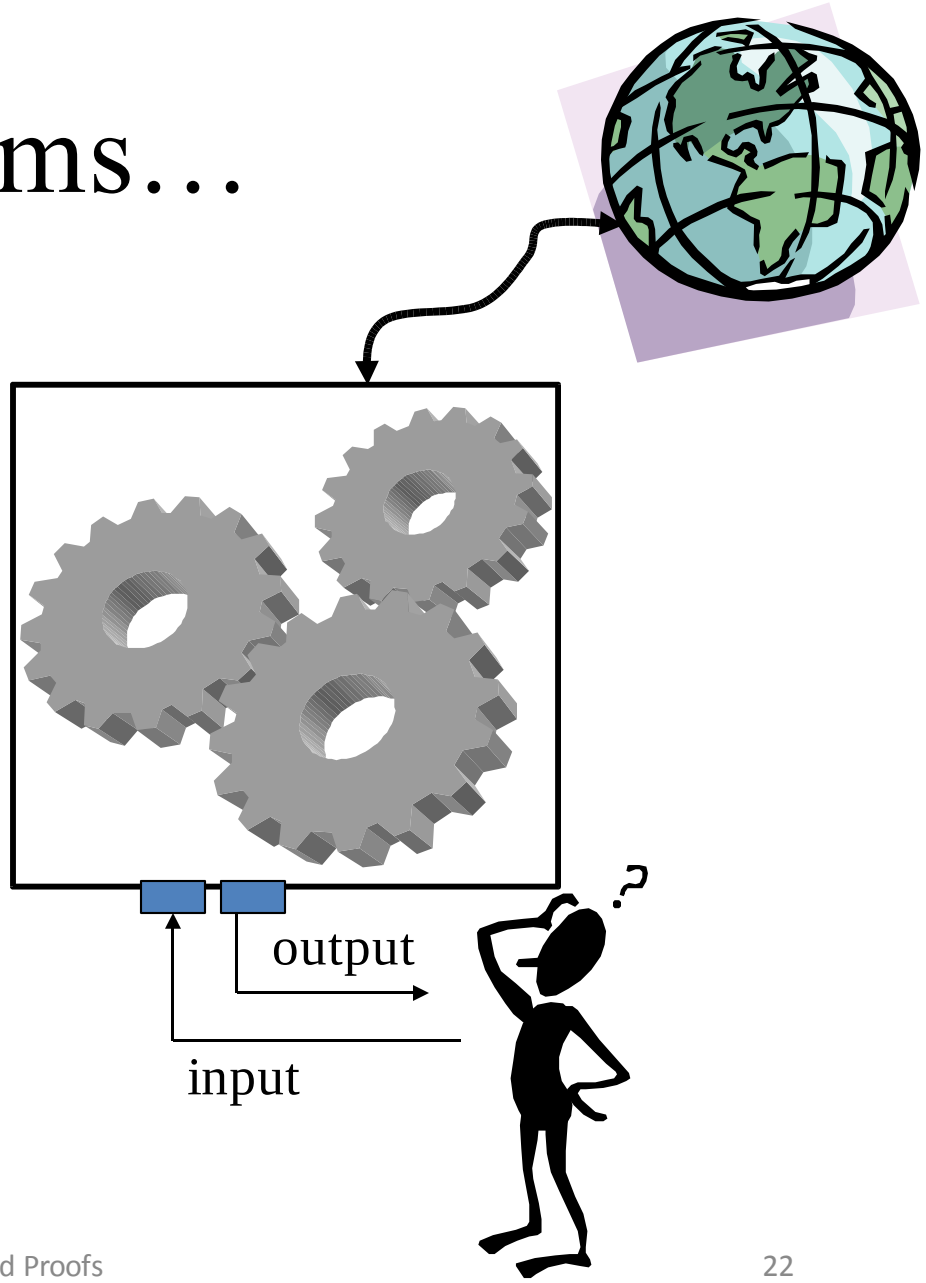
Some last “clear” definitions

- Models
- Programs
- **Systems**

- Model-checking
- Program proving
- Testing

Systems…

- A system is a dynamic entity, embedded in the physical world
- It is **observable** via some limited interface/procedure
- It is not always **controllable**
- Quite different from a piece of text (formula, program) or a diagram (model)

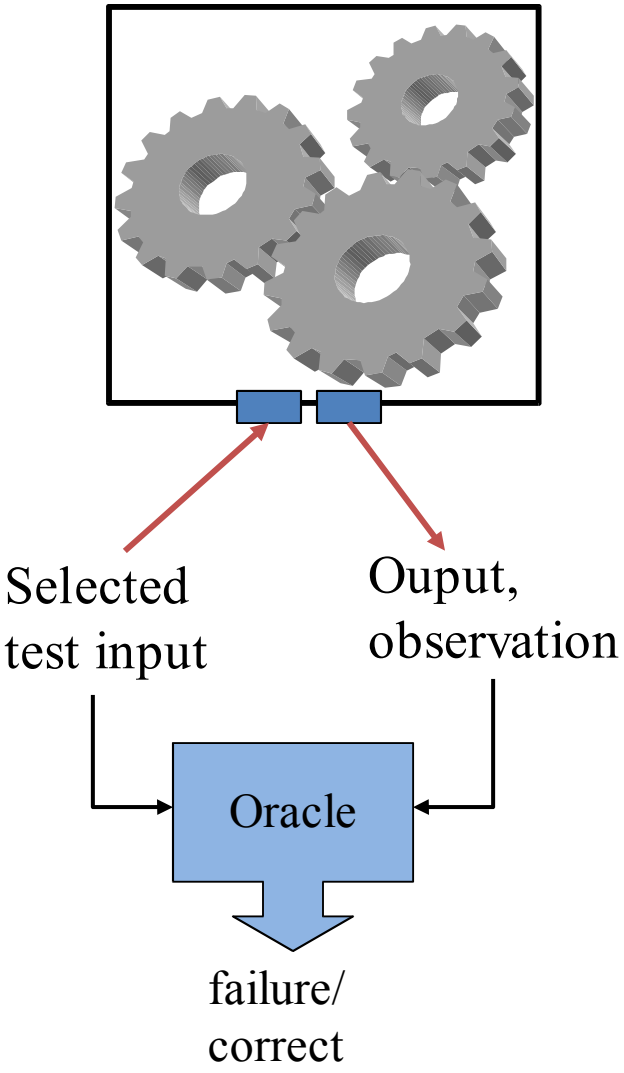


Systems are the actual objects of interest

- How to guarantee that a system satisfies certain properties?
- *Properties?*
  - Texts in natural languages…
    - "whenever a request is made it holds continuously until it is eventually granted"
    - "no buffer overflows"
  - Formulas in a given specification logic
    - $G(\neg request \vee (request\ U\ grant))$
  - Sets of mandatory or forbidden execution paths in some model
  - Constraints on some well-known quantities: WCET, MTTF, *etc, etc*
- Beware the *abstraction gap* (between the entities of the system environment and the concepts mentioned in the properties)

Testing

- The actual system is executed for a finite set of selected inputs
  - NB: selected test sequences for reactive systems
- These executions are observed, and a decision is made on their conformance w. r. t. some specification
- Issues:
  - Selection
  - Oracle
  - Control, non-determinism
  - Assessment of the result of a test campaign



Selection (randomised or not)

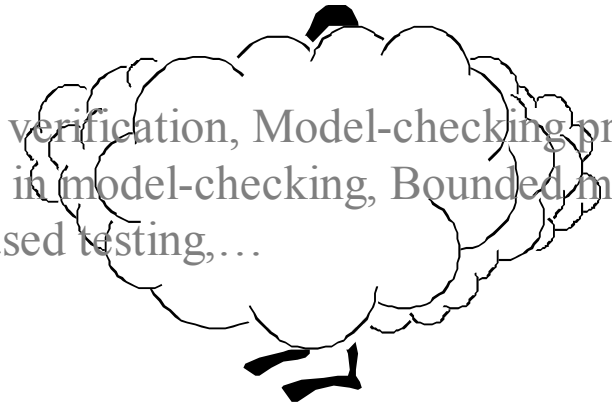
- *Infinite* input domain → *finite* test set, likely to lead to as many failures as possible.
- The selection process can be based on:
  - Some characteristics of the input domain,
  - The structure of the system, or of the program,
  - Some specification/model of the system, and or its environment,
  - Test purposes.
- *Yes, testing is generally incomplete!*
  - Even worst: "exhaustive" testing is exhaustive w.r.t. the bases of the selections, not w. r. t. the system: there are implicit *testability hypotheses*.

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Outline of the talk

- Discussion of some hopefully "clear" definitions
  - Models, Programs, Systems
  - Model-checking, Program proving, Testing
- Not so clear variants of the definitions above
  - Run-time verification, Model-checking programs, Coverage in model-checking, Bounded model-checking, Model-based testing, Program checking, Proof approximation…
- A few words on:
  - Model-based testing
  - Concolic testing

Not so clear variants and cross-fertilisations



• use of an object/subject O_1 for performing an activity on another object/subject O_2: f.i. *model-based testing*.

• use of a method developed for activity A_1 for performing activity A_2: f.i. *"tests from proofs"*.

• integrated use of *static* and *dynamic* methods.

Sometimes confusing!

Classification of methods?

- What inputs?
  - Model, program, system, several of them,
  - Properties, formulas, …
- What results ? What guarantees? Under what hypotheses?
  - Certainty? No certainty? Probability?
- What kind of software and systems?
- Scalability, complexity?

TENTATIVE

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Outline of the talk

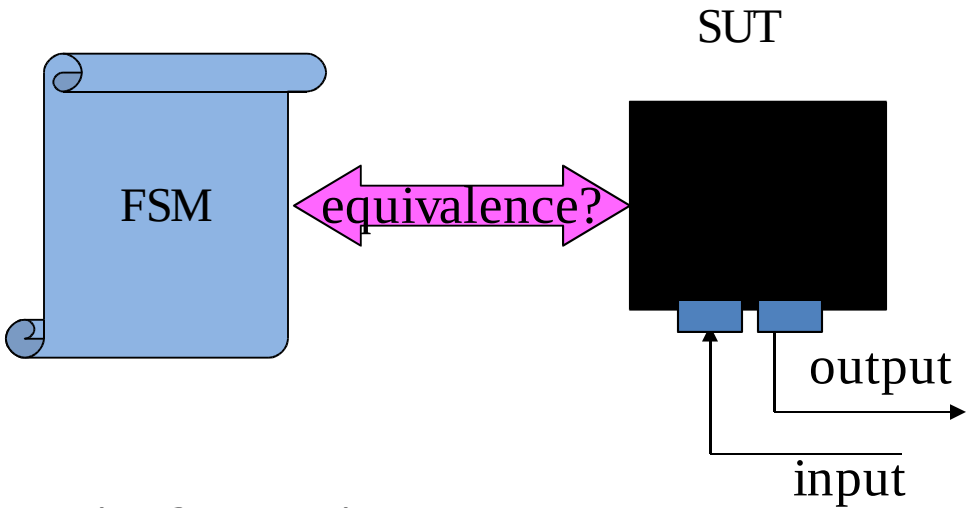
- Discussion of some hopefully "clear" definitions
  - Models, Programs, Systems
  - Model-checking, Program proving, Testing
- Not so clear variants of the definitions above
    - Run-time verification, Model-checking programs, Coverage in model-checking, Bounded model-checking, Model-based testing, Program checking, Proof approximation…
- A few words on:
  - Model-based testing
  - Concolic testing

Model-based testing

- Widely used term
  - There are several MBT workshops:
  - *Almost everything is considered as a model* (may be not wrong ☺)
- Always present, but not always explicit: testability hypothesis.
  - *the system under test behaves like an unknown model of the same nature as the one used to guide testing*
- Back to [Chow 78] for FSM,
  - recommended reading : Lee and Yannakakis survey [1996]
  - and [Brinksma 88] for LTS, … and then many others

Back in history: the FSM approach

- System under test
  - unknown, but…



- **Testability Hypothesis for this approach:**

  The System Under Test behaves like some (unknown) FSM with the same number of states as the description
  - In other words, in the SUT, whatever the execution path leading to some state s, *the execution of transition* ***s –x:y-> s'*** *has the same effect (same output + same change of state)*

Back in history: control and observation

- Testability hypothesis => justification of transition coverage

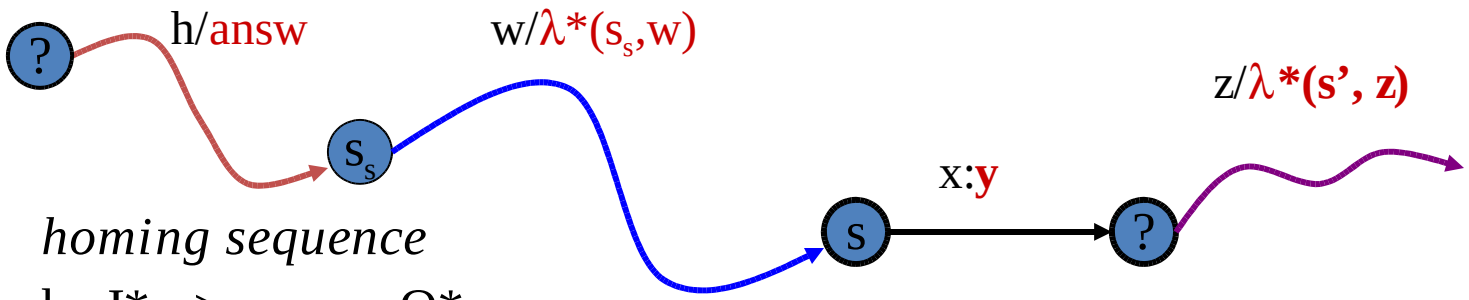
  Let ***s –x:y-> s'*** *be a transition.*

  In state s, input x must produce output y and move to state s'

  To check this once on the SUT is sufficient to ensure equivalence w. r. t. the original FSM

- Questions
  - **control:** how to put the System Under Test into a state equivalent to *s*?
    - solution 1: *reliable(??) reset*, and then adequate input sequence
    - solution 2: *"homing sequence"*, and then adequate input sequence
  - **observation:** how to check that after receiving *x* and issuing *y*, the SUT is in a state equivalent to *s'* ?
    - By observing this state via *distinguishing input/output sequences*
  - *see next slide*

One of the model-based tests for s -x:y-> s' : *h.w.x.z*



homing sequence

h∈ I* => answ ∈ O*: then the SUT state should be equivalent to s_s.

w ∈ I*: in the original description, *w leads from s_s to s*.

z belongs to the *separating set* of s'

preamble — transition execution — observation

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Elements for classification

- *Input*: a model
- *Output*: set of controllable and observable test sequences
- *Guarantee*: equivalence of the SUT and the model, provided that the testability hypothesis holds
- *Class of addressed systems*: finite

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Outline of the talk

- Discussion of some hopefully "clear" definitions
  - Models, Programs, Systems
  - Model-checking, Program proving, Testing
- Not so clear variants of the definitions above
    - Run-time verification, Model-checking programs, Coverage in model-checking, Bounded model-checking, Model-based testing, Program checking, Proof approximation…
- A few words on:
  - Model-based testing
  - Concolic testing (dynamic test generation)

Why to speak of concolic testing?

- It is an interesting combination of techniques.
- In short:
  - *The system is actually executed on some input,*
  - *It is instrumented in such a way that the symbolic execution of the program is recorded,*
  - *This symbolic execution is used to select the next test inputs using a constraint solver.*

Elements for classification

- It is an integration of
  - Static techniques: symbolic executions based on the *program*
  - Dynamic techniques: guided test executions of the *system*
  - Reasoning technique: use of constraint solvers
- *Both static and dynamic, with some flavour of proof…*
- Guarantee: *program paths coverage* below a given length (may be different from actual executions)

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Conclusion ? (1)

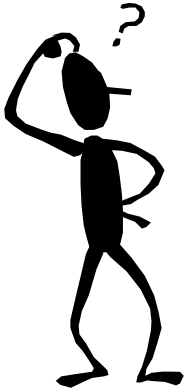
- Some politically correct and frequent comments:
  - "All these methods are now used together, and this convergence will lead to great results"
  - "Model-checking is very powerful and solves most problems in static analysis and model based testing and more generally in verification"
  - "Verification" is just model-checking
  - "When you prove, you don't need to test
  - …

Hum …

Conclusions (2)

- Many tricky scientific issues, among which:
  - Diverse logics
    - Standard temporal logics can specify only regular properties; correctness of procedures w. r. t. pre- and post conditions are not regular [Alur 2005]…
    - Model-checking is always "up to some bound"… limited input domain for specific properties … [Godefroid TAP 2011]…

    Integration of model-checking and program proving is not as clear as it may seem.



  - Diverse objects: model/specification, programs, systems
  - "Abstraction gap" between proving (reasoning with equality) and testing (oracle)
  - …

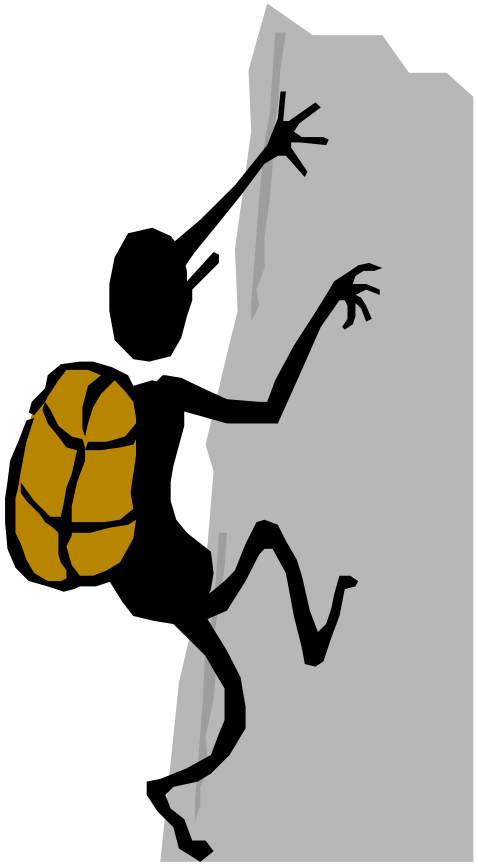
Some possible first steps towards an agreed terminology

- Distinction between *activities*
  - Model-checking, program proving, system testing
- and *methods* used to perform them.
  - For instance, using a testing-inspired strategy when model-checking is not testing: it is incomplete model-checking…
- Clarify a few numbers of notions:
  - Completeness versus incompleteness
  - Exhaustivity with respect to what?
  - What is static, what is dynamic?
- Careful use of the word "verification" etc.

TENTATIVE

LABORATOIRE DE RECHERCHE EN INFORMATIQUE

Final question



Is it worth climbing alone,
every day,
this sort of Babel tower?