Languages & Automata
Some Open Questions

Ciprian.TEODOROV@ENSTA-Bretagne.fr

Context

Autonomous systems

- Mars Rover
- Drones
- Voilier robot

Medical

- Robot chirurgien
- Stimulateur cardiac (Pacemaker)

Aero-space

- Flight control system
- Landing gear

ENSTA Bretagne

Why do we need formal techniques?

- **Quality**
  - Safety = human lifes
  - Security = access control; online banks
  - Legal = electronical signature
- **Productivity**
  - Early error detection
  - Re-use
  - Test generation

But we are testing…

- *Execute* the system with *a predefined set of **inputs** and observe* the **outputs**
  - Random inputs -> coverage problems
  - Smart inputs -> high cost
  - Automatically choosen smart inputs -> need of formal models
- What does it means execute for a plane ?
- What can we say about the inputs that were not tested ?
- How do we observe ? Oracle…

- The formal methods do help

Proof and techniques

Test [Angle bisection – Ancient Greece]

- Proves the **existence** of **given** situations

Proof [Angle trisection Pierre Wantzel in 1837]

- Proves the **absence** of **given** situations

Proof techniques

- Statical analysis: type-checking
- Proof assistants: equivalences
- **Model-checking : exhaustive analysis (counter-example proof).**

Formal verification : Model-Checking

System

Abstraction

Model

satisfies

Specification

Abstraction

Properties

satisfies

Model-checking

ENSTA Bretagne

Towards formal methods

{System, Hypotheses, Domain Laws} |= Requirements

Necessary conditions :

The **system** satisfies the **requirements** if and only if:

- The *hypotheses* are satisfied by the real **environment**
- The *domain laws* are **true**
- The preceeding elements are **consistents**

Model-checking

Idea : **Exhaustive search** of a counter-example

{Requirements, Hypotheses, Domain Laws} |= Spec

Design under study (DUS)

Environment

Properties

{ DUS || Environment} |= Properties

Model

ENSTA Bretagne

Formal verification of critical systems: Complexity race

- **Bigger** systems vs **more powerful** verification techniques

- Model-checking:

Advantages

- intuitive
- generic
- automated
- counter-examples

Disadvantages

- State-space explosion
- Manual model reduction
- Temporal logics

[Partial] Solutions: POR, SR, BMC, etc.

ENSTA Bretagne

Industrial Challenges

- Notations *[OK]*
  - Requirements (Doors, Doc)
  - Models (UML, AADL, SDL)
- Tools *[OK]*
  - Model-checkers
  - Other formal analysis tools
- Problems, Solutions et Questions
  - Requirements
  - Environment (the model should be closed 4 verification)
  - System model
  - Abstraction & Modular Decomposition

Industrial Challenges

- Notations *[OK]*
  - Requirements (Doors, Doc)
  - Models (UML, AADL, SDL)
- Tools *[OK]*
  - Model-checkers
  - Other formal analysis tools
- Problems, Solutions et Questions
  - **Requirements**
  - Environment (the model should be closed 4 verification)
  - System model
  - Abstraction & Modular Decomposition

ENSTA Bretagne

The Rise and Fall of LTL

Vardi, 2011

old history

[https://youtu.be/Ayg0V1qiJwc]

- 1928 – First order logic decidable ? ‘36 – No, but some fragments Yes
- The *declarative* “logic” is connected to the *imperative* “machine” (automaton)
- ‘57-58 [Buchi, Elgot, Trakhtenbrot] proved *finite* MSO ≡ NFA ≡ *DFA* ≡ *RegExp*,
  - ‘59 NFA complementation is hard, ’78 & ’93 $\mathbf{2^n}$ upper bound, L(A)≠∅ linear in size A
  - ’74 finite words MSO non-elementary satisfiability
- ‘60 [Buchi] MSO ≡ Buchi ≡ ω-Reg; ‘60 [Church] Model-checking is decidable
- ’54, ’57 [Prior] linear & modal logic; ’58 [Kripke$_{<18\ years\ old}$] **branching time**.
- Linear time = a set of linear trace vs Branching time = a trace tree
- ‘77 [Pnueli] LTL for program specification, model-checking via automata

ENSTA Bretagne

The Rise and Fall of LTL

Vardi, 2011

modern history

[https://youtu.be/Ayg0V1qiJwc]

- ‘79, ‘80 Expresity : LTL ≡ FO ≡ star-free ω-RE < *MSO ≡ ω-RE*
- ’81, ’82 LTL satisfiability is PSPACE-complete vs FO which is non-elementary
- ’85 Past LTL : expressivity & satisfiability PLTL ≡ LTL but **PLTL exponentially more succinct** than LTL (shorter phrases)
- ’83, ‘89 ETL (LTL + automata) ≡ µTL (LTL + fixpoints) ≡ MSO
- ‘98, ’01 @ IBM : TCTL & Sugar – branching time logics
- ’90 – ’00 @ Intel : LTL & RETL & ForSpec – linear time logics (RETL LTL + RegExp)
- **’00 : PSL industrial standard = LTL + RexExp + branching + clocks + resets**
- ’11 [Vardi] : Linear dynamic logic : linear time, LDL ≡ MSO, RegExp + Past + dynamic modalities, exponentially more succinct than RETL
- **Open question : Whats next ?**

ENSTA Bretagne

Property Specification Language: Great success

- Textual requirement :
  - "every **request** which is immediately followed by an **ack** signal, should be followed by a complete data transfer, where a complete data transfer is a sequence starting with signal **start**, ending with signal **end** in which busy holds at the meantime"
- **PSL property :**
  - (true[*]; req; ack) |=> (start; busy[*]; end)
- But : Is it readable ?

Property patterns

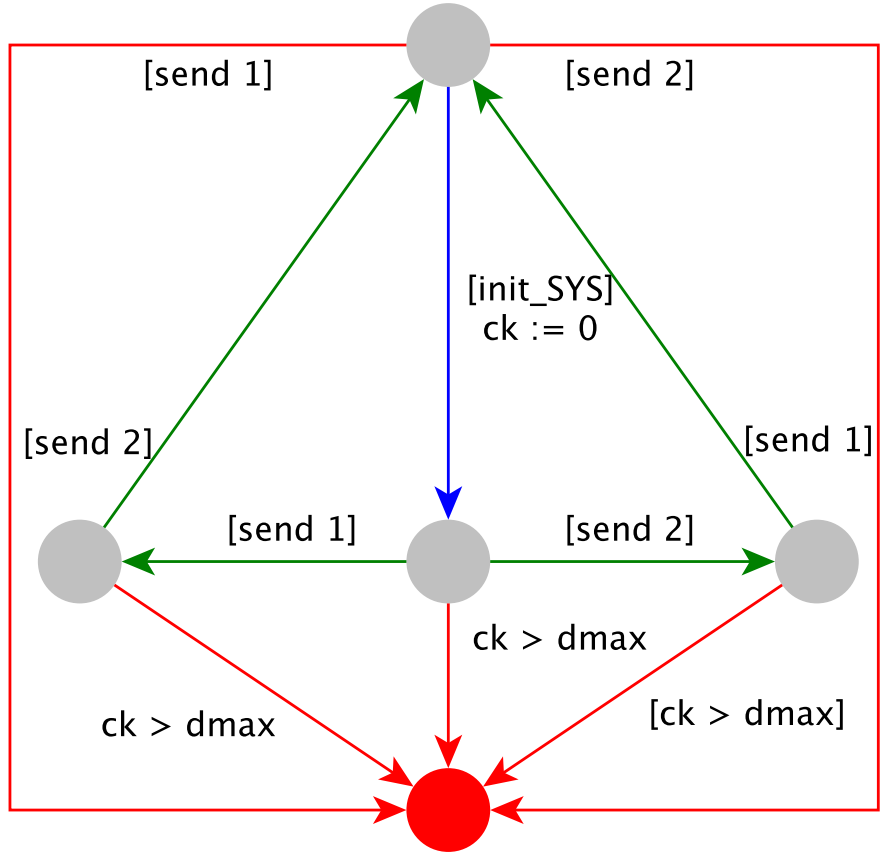
[Dwyer] + [Cheng] + [Smith]

SYST-DP-REQ-6-1: During initialization procedure, the SYS shall associate an identifier to NC console (IHM), before dmax time units.

An **exactly one** occurrence of init_SYS
eventually leads-to [0 .. dmax [

All combined
exactly one occurrence of send 1
exactly one occurrence of send 2

init_SYS may never occur
one of send 1 **cannot occur** before the first one of init_SYS
one of send 2 **cannot occur** before the first one of init_SYS
repeatability: true



Response, *Precedence*, *Absence*, *Existence*, *Pre-arity*, *Post-arity*, *Immediacy*, *Precedence*, *Nullity*, *Repeatability*

AFADL'10

ENSTA Bretagne

Industrial Challenges

- Notations *[OK]*
  - Requirements (Doors, Doc)
  - Models (UML, AADL, SDL)
- Tools *[OK]*
  - Model-checkers
  - Other formal analysis tools
- Problems, Solutions et Questions
  - Requirements
  - **Environment (the model should be closed 4 verification)**
  - System model
  - Abstraction & Modular Decomposition

ENSTA Bretagne

Context-aware Verification

- Clear separation between the **system** and the **environment**
- Extraction of ***verification guides*** from the environment model

- **Properties** + ***verification guides*** = **verification context**
  - The verification guides are acyclic interaction scenarios

www.obpcdl.org

- **Reduction** axes:
  1. Decomposition through **contexts**: ex. operating modes
  2. **Environment-guided analysis**

- **Complementarity** with traditional reduction techniques: POR, SR

ENSTA Bretagne

Context decomposition: ex. *operating modes*

SYST-DP-REQ-6

During initialization *procedure, the SYST_DP shall associate a generic equipment identifiers to one or several role in the system (MainSensor, OtherSensor, IFF, Actuator, ...). It shall also associate an identifier to each console.*
The SYST_DP shall send an evtEquipmentRole message, in ***preparation mode****, for each connected generic equipment, to each connected console.*
Initialization procedure shall end successfully, when the SYST_DP has set all the generic equipment identifiers and all console identifiers and all evtEquipmentRole message have been sent.

End

SYST-DP-REQ-2

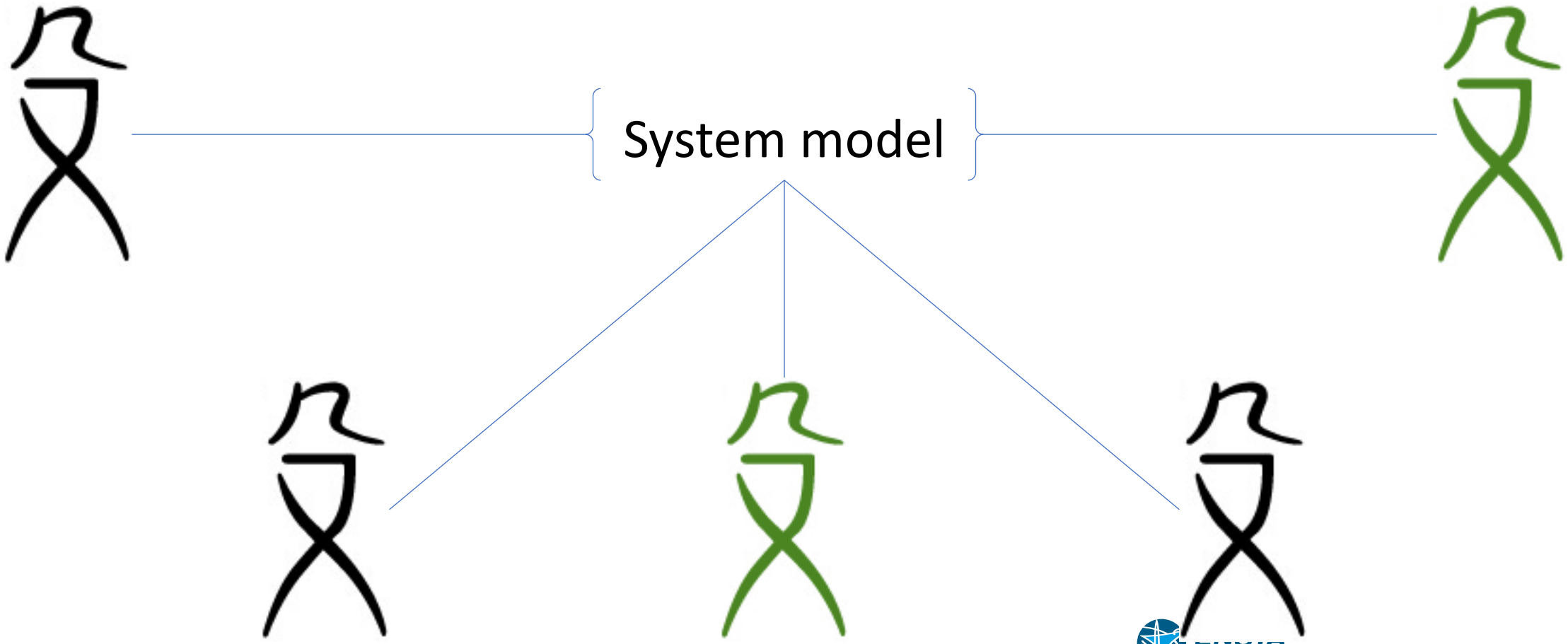
Once initialization is achieved*, the SYST_DP shall send to each console an evtCurrentMission with curMission set to IDLE, to set current mission to idle, followed by an evtCurrentActivity with curActivity to LOGIN and status to TRUE to activate login.*

End

3 different verification contexts

ENSTA Bretagne

CaV: Selection of Verification Guides



Verification Guide Syntax & Semantics

- G ::= a | C;C | C [] C | C||C | C? | C+ | C* | C{i, j} | {i,j} of [C_1, .., C_n]

What does it mean ?
How do we compile ?

$$\frac{a \in A^+}{a \xrightarrow{a} \bot}\ [\text{atom}] \qquad \frac{a \in A^+}{a;C \xrightarrow{a} C}\ [\text{seq}_1] \qquad \frac{C_1 \xrightarrow{a} C_1' \wedge C_1 \neq a}{C_1;C_2 \xrightarrow{a} C_1';C_2}\ [\text{seq}_2]$$

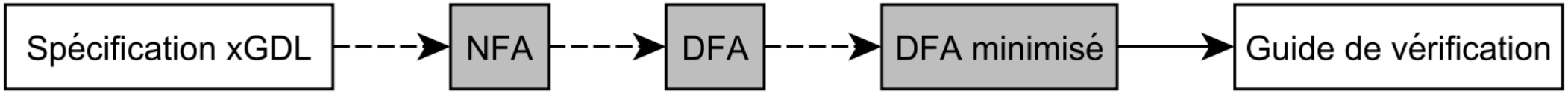
$$\frac{}{C_1 \Box C_2 \xrightarrow{\tau} C_1}\ [\text{alt}_1] \qquad \frac{}{C_1 \Box C_2 \xrightarrow{\tau} C_2}\ [\text{alt}_2] \qquad \frac{C_1 \xrightarrow{a} C_1'}{C_1 \| C_2 \xrightarrow{a} C_1' \| C_2}\ [\text{par}_1]$$

$$\frac{C_2 \xrightarrow{a} C_2'}{C_1 \| C_2 \xrightarrow{a} C_1 \| C_2'}\ [\text{par}_2] \qquad \frac{}{\bot \| C \xrightarrow{\tau} C}\ [\text{par}_3] \qquad \frac{}{C \| \bot \xrightarrow{\tau} C}\ [\text{par}_4]$$

$$\frac{}{C? \xrightarrow{\tau} \bot \Box C}\ [\text{opt}] \qquad \frac{}{C* \xrightarrow{\tau} (C;C*)?}\ [\text{star}] \qquad \frac{}{C+ \xrightarrow{\tau} C;C*}\ [\text{plus}]$$

$$\frac{0 < i \leq j}{C\{i,j\} \xrightarrow{\tau} C;C\{i-1,j-1\}}\ [\text{rep}_1] \qquad \frac{i = 0 \wedge j > 0}{C\{i,j\} \xrightarrow{\tau} (C;C\{0,j-1\})?}\ [\text{rep}_2] \qquad \frac{i = j = 0}{C\{i,j\} \xrightarrow{\tau} \bot}\ [\text{rep}_3]$$

Similar to RegExp



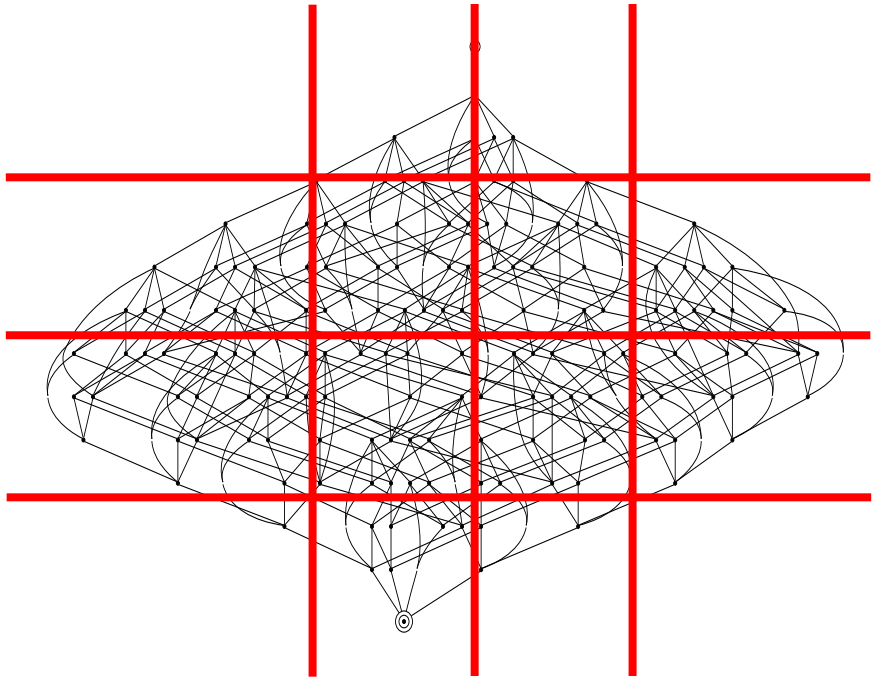
ENSTA Bretagne

CaV : Environment-guided analysis

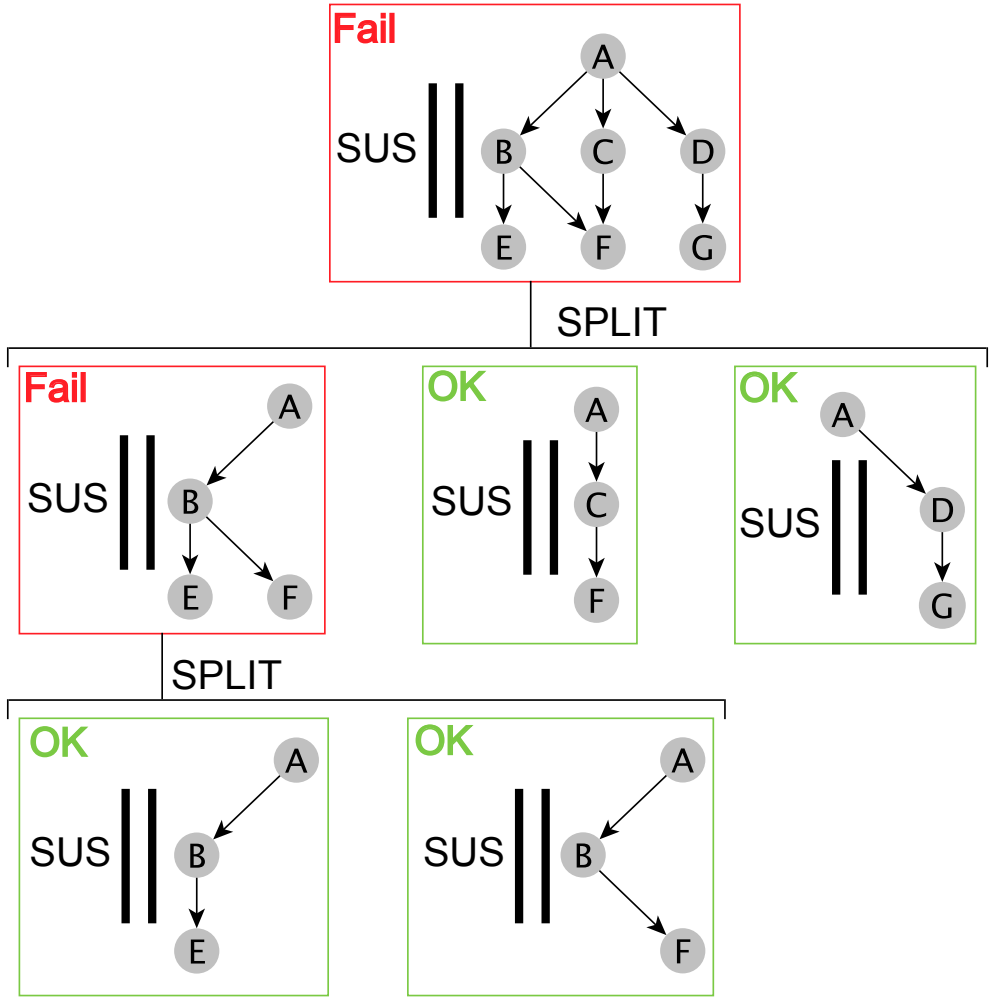
- If the verification guides are acyclic
- Exploited during analysis

Recursive scenario decomposition

PastFree[ze] reachability

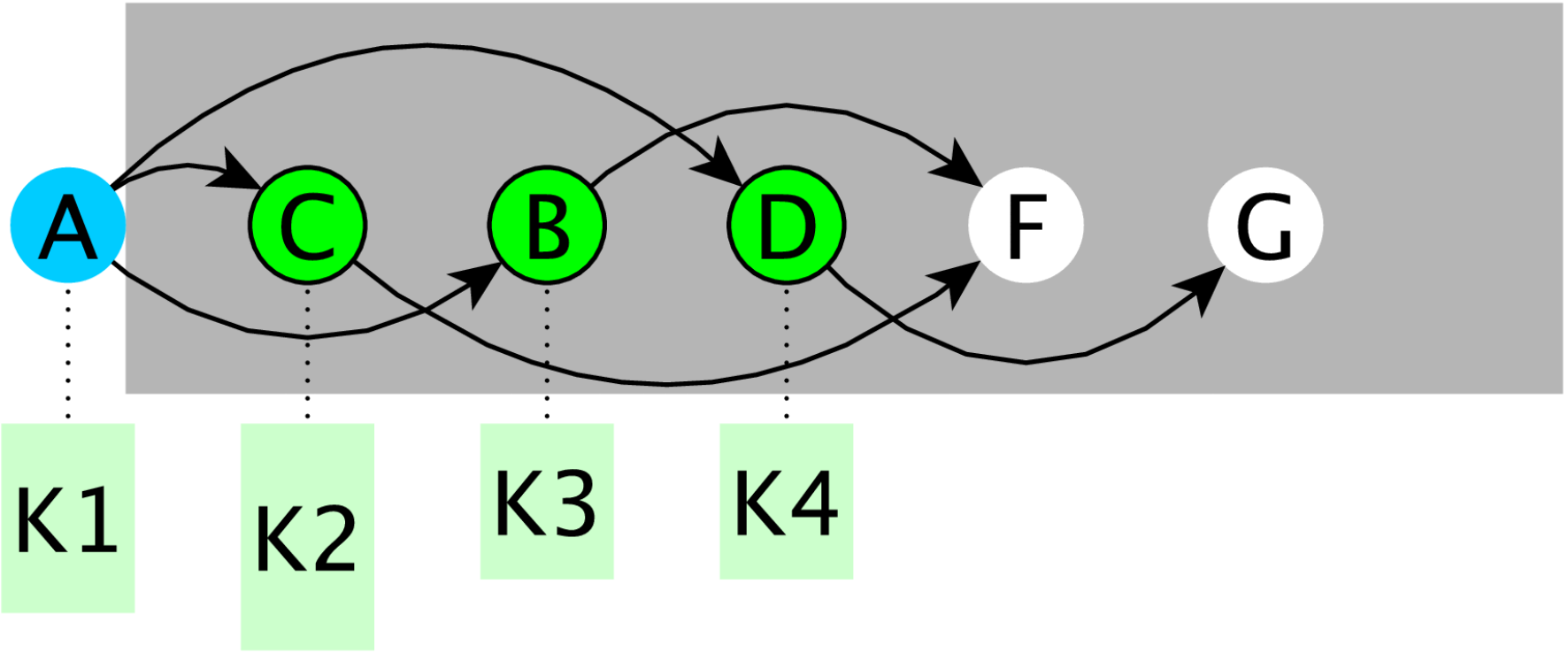


CaV: Recursive Scenario Decomposition (split)

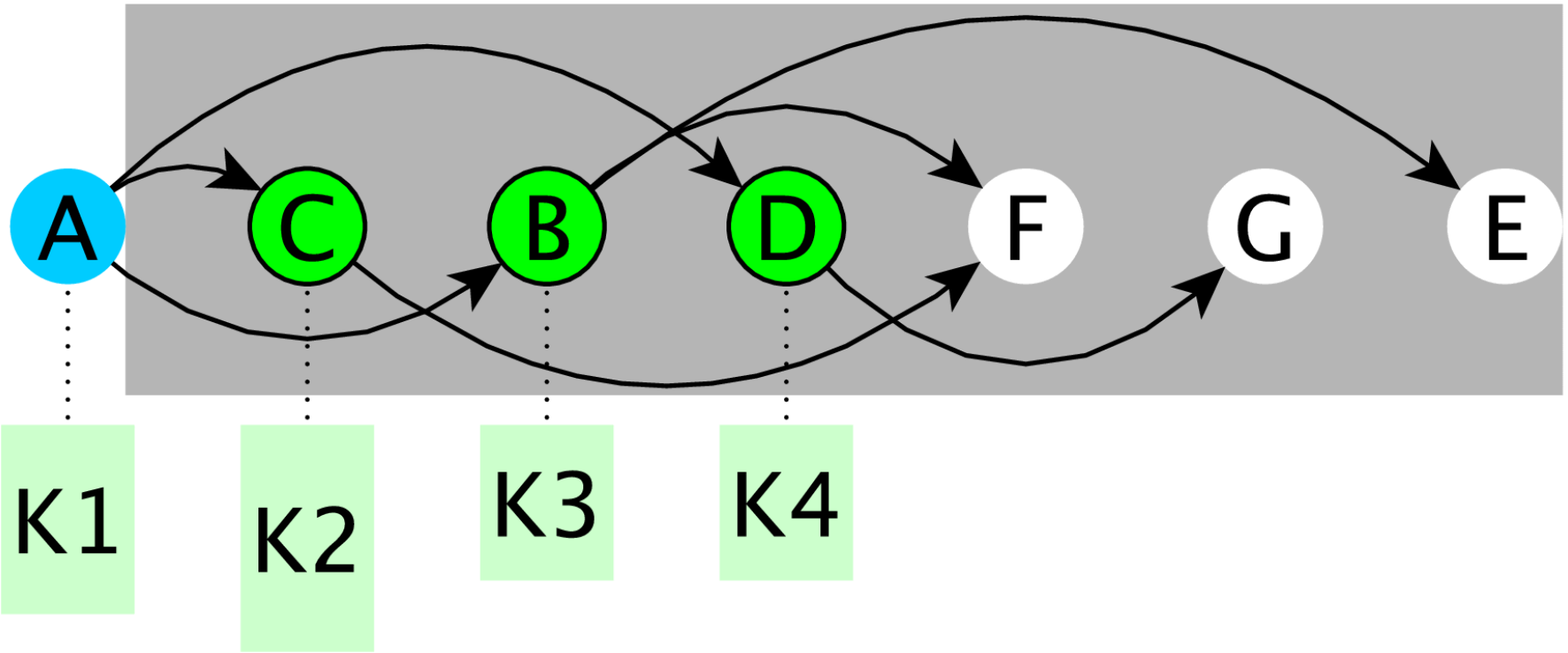


ENSTA Bretagne

CaV: Reachability Algorithm

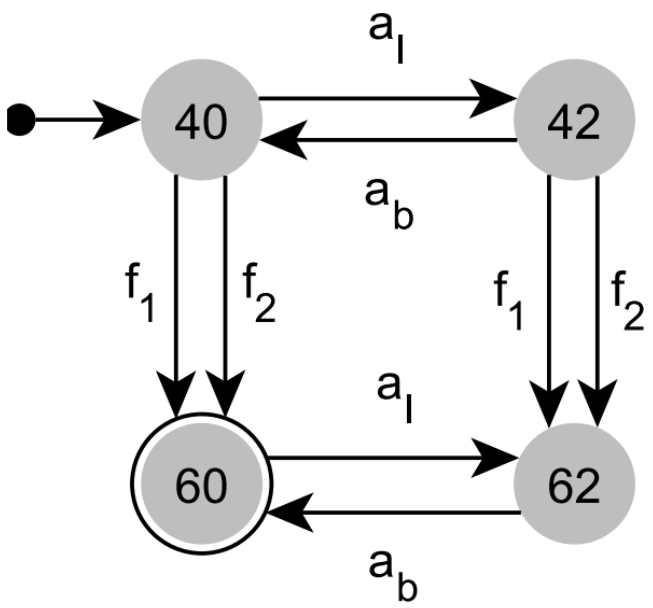


CaV: PastFree[ze] Reachability Algorithm



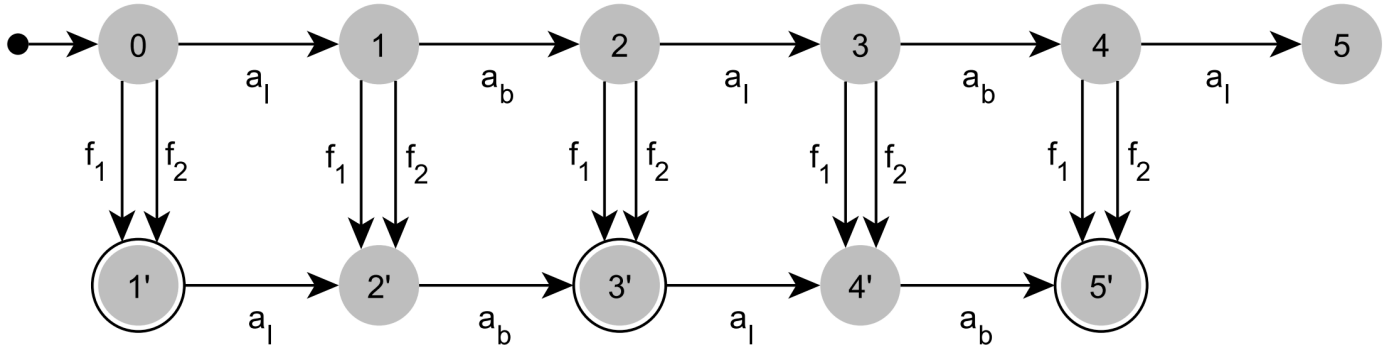
Partially-Bounded Model Checking

How can we get an acyclic verification guide ?



$$(a_l; a_b) * \|(f_1 \square f_2)$$

Cyclic

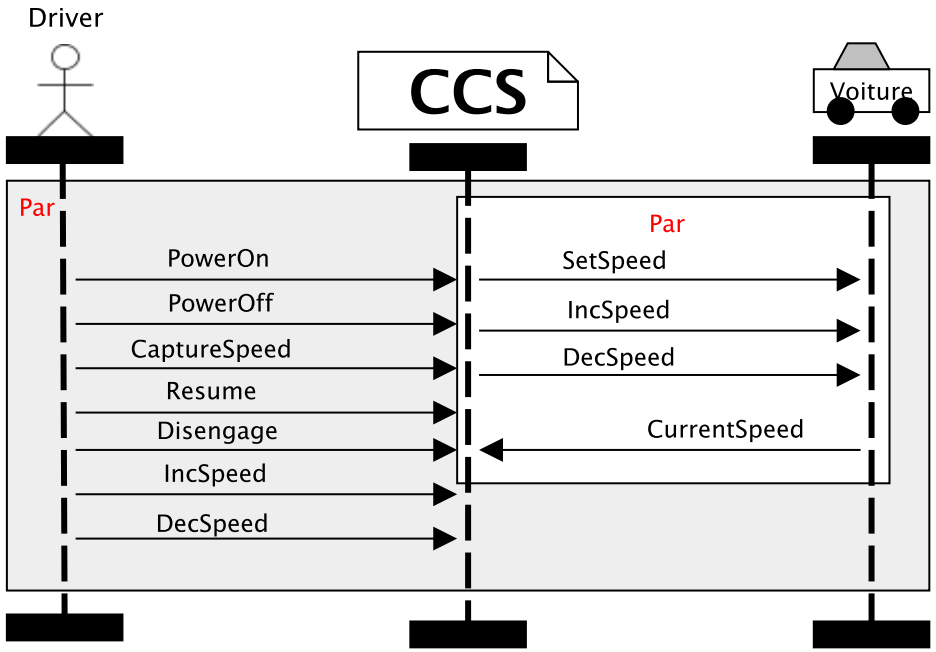


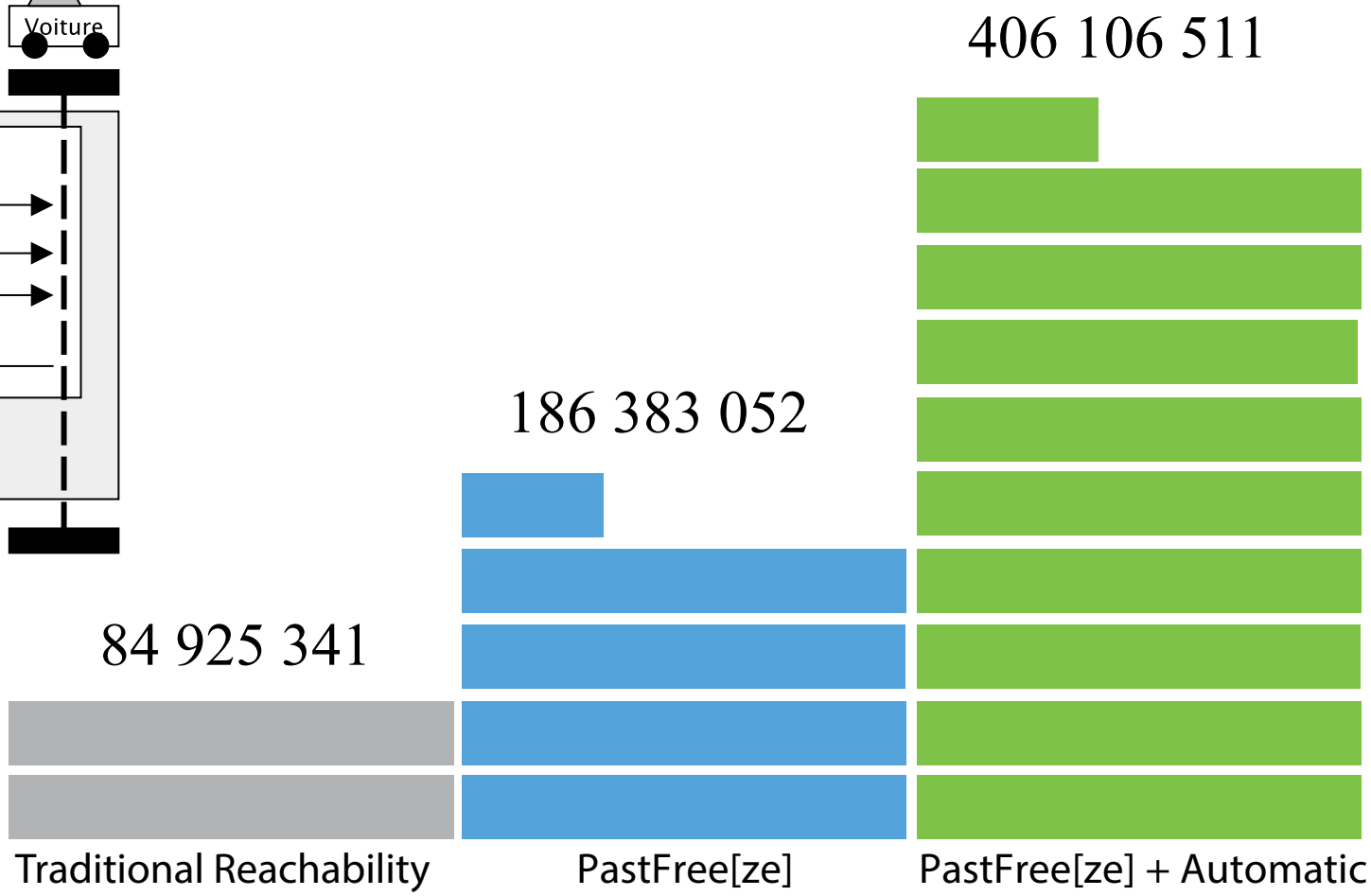
5 step unrolling

acyclic

ENSTA Bretagne

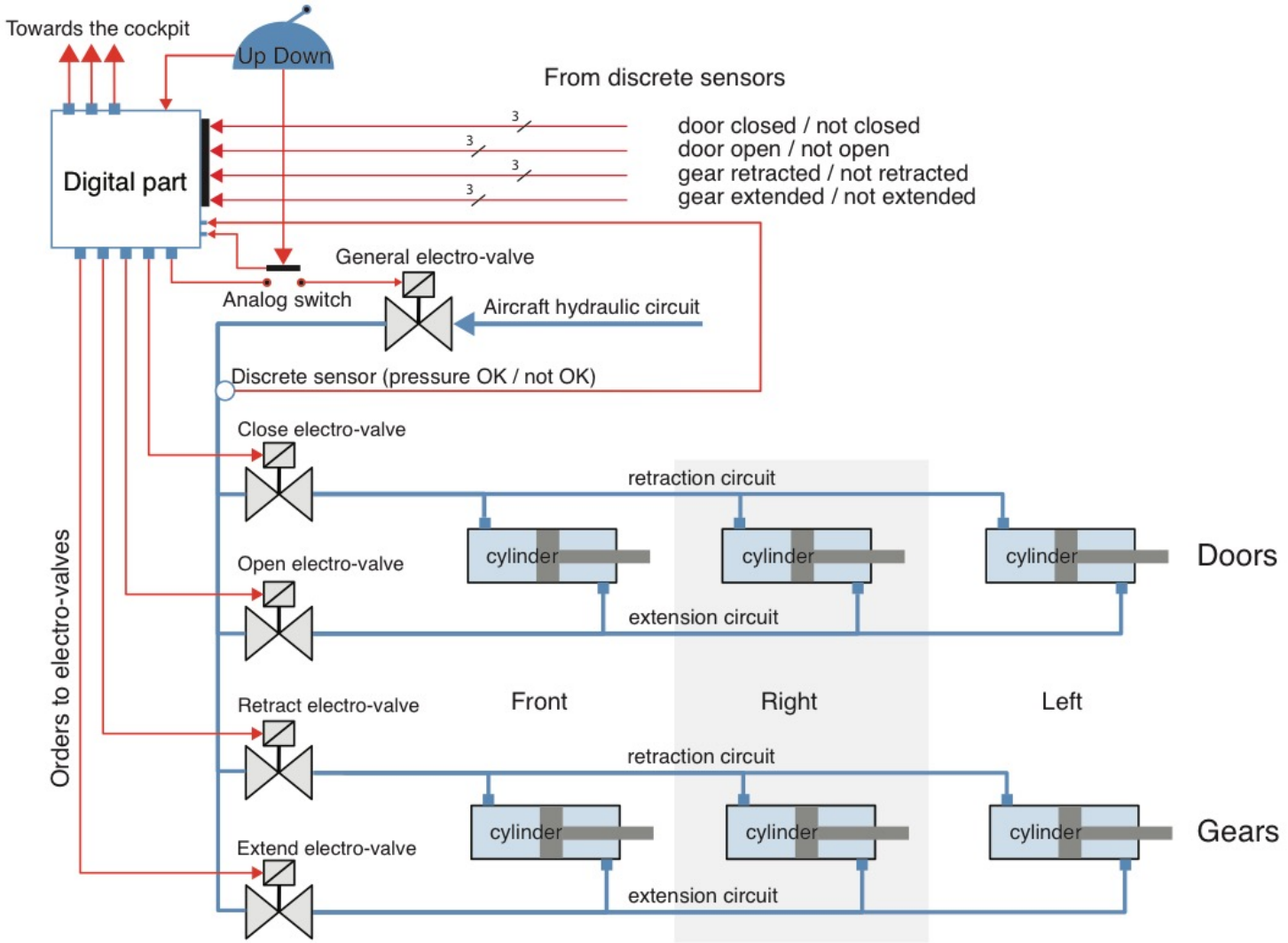
Case-study 1: Cruise Control System

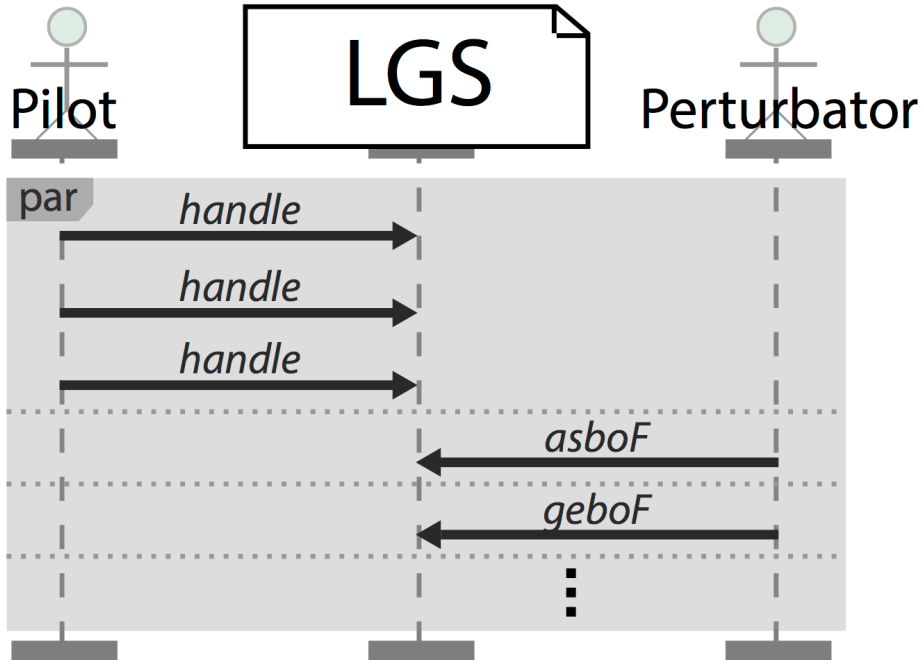




ENSTA Bretagne

Case-study 2: **L**anding **G**ear **S**ystem

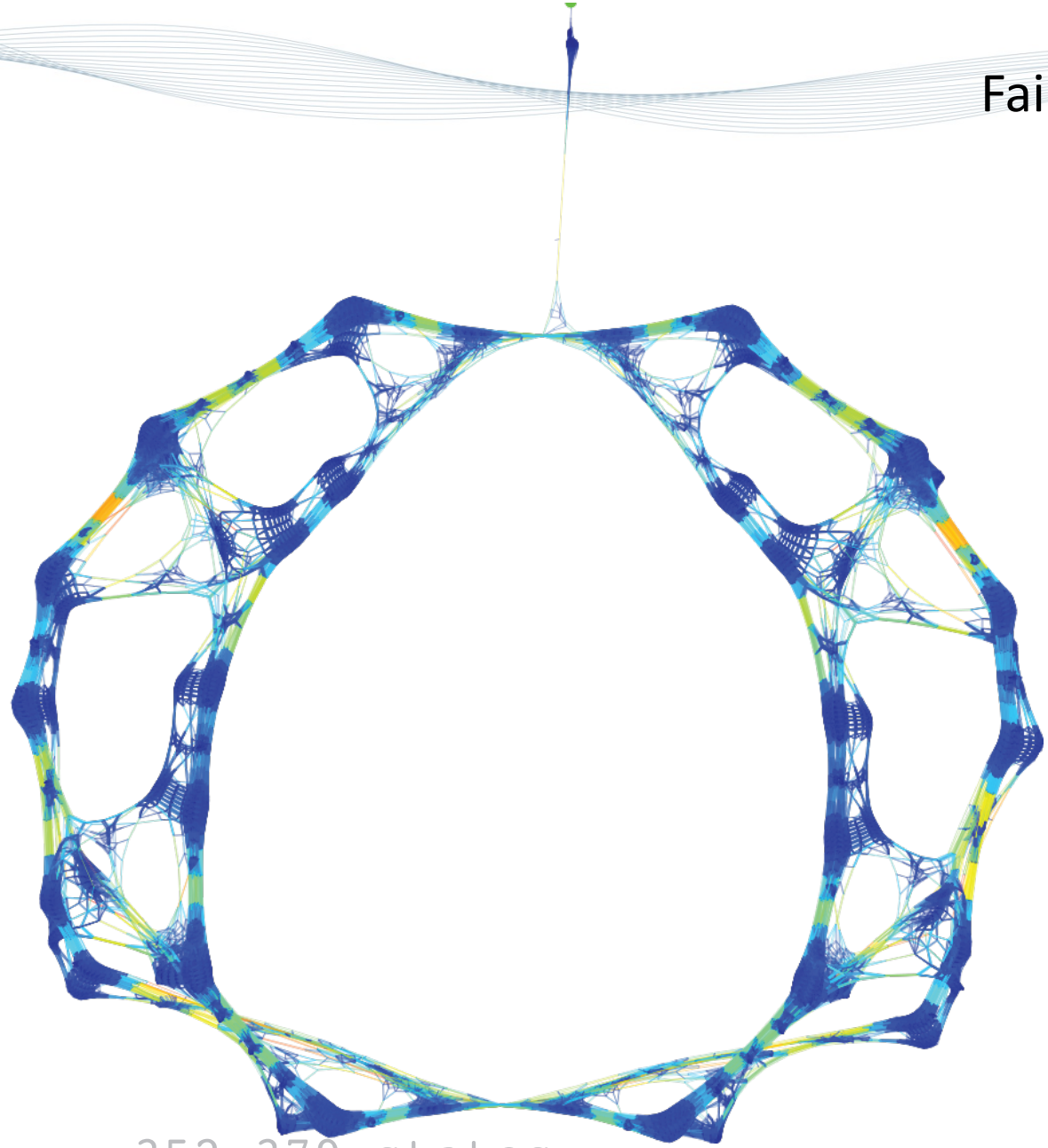




STTT'15

ENSTA Bretagne

Failure-free state-space

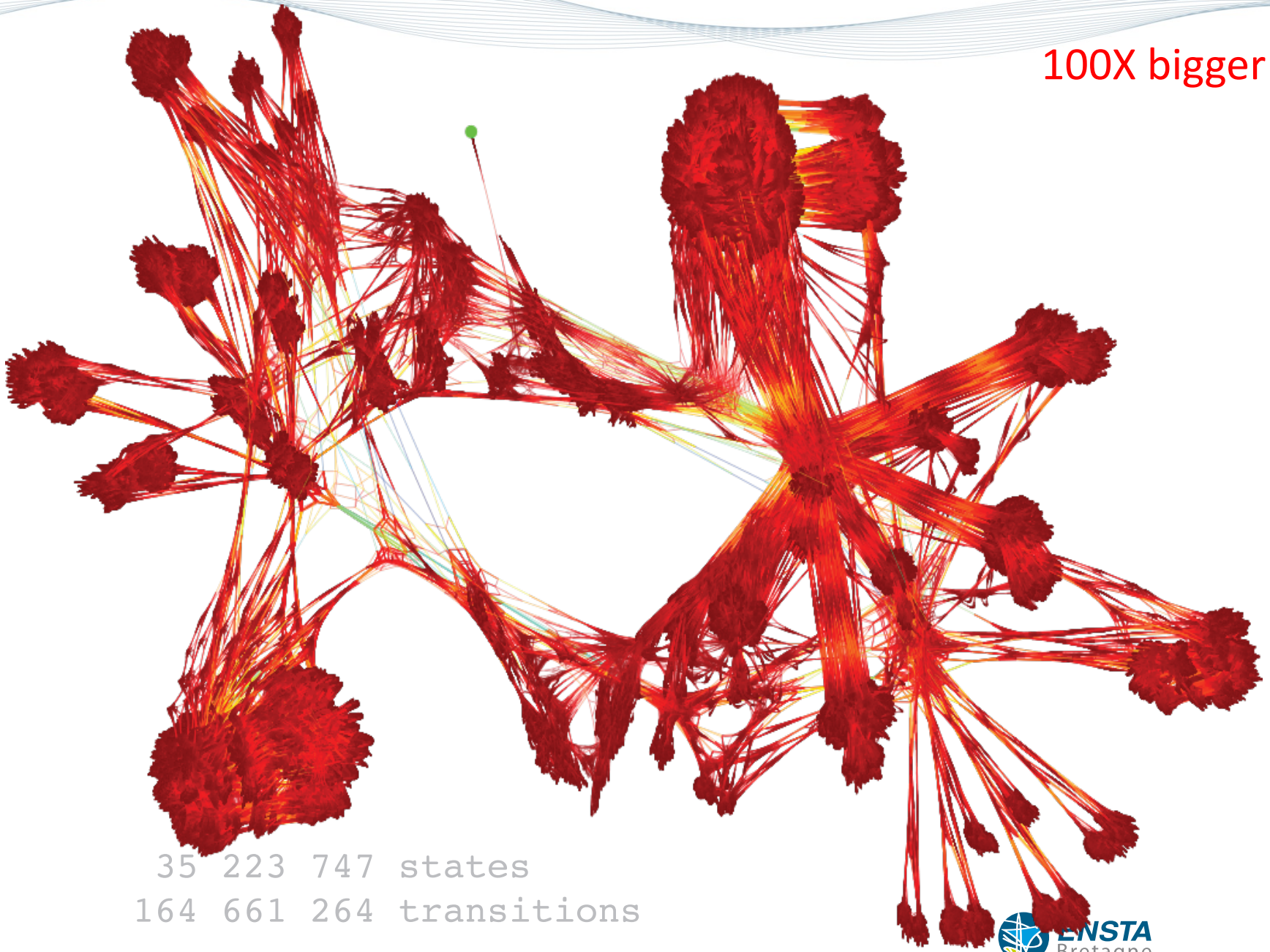


352 379 states
1 477 197 transitions

ENSTA Bretagne

Failure: Electro-valve blocked open

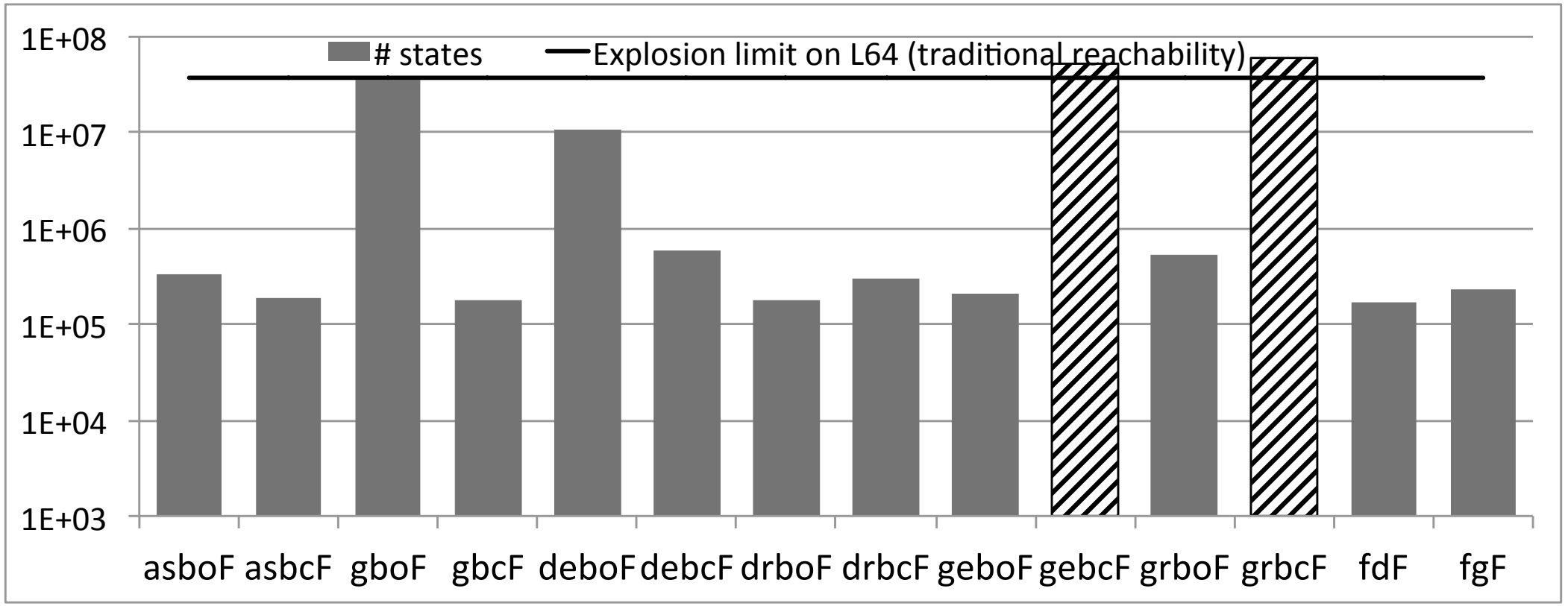
100X bigger state-space



35 223 747 states
164 661 264 transitions

ENSTA Bretagne

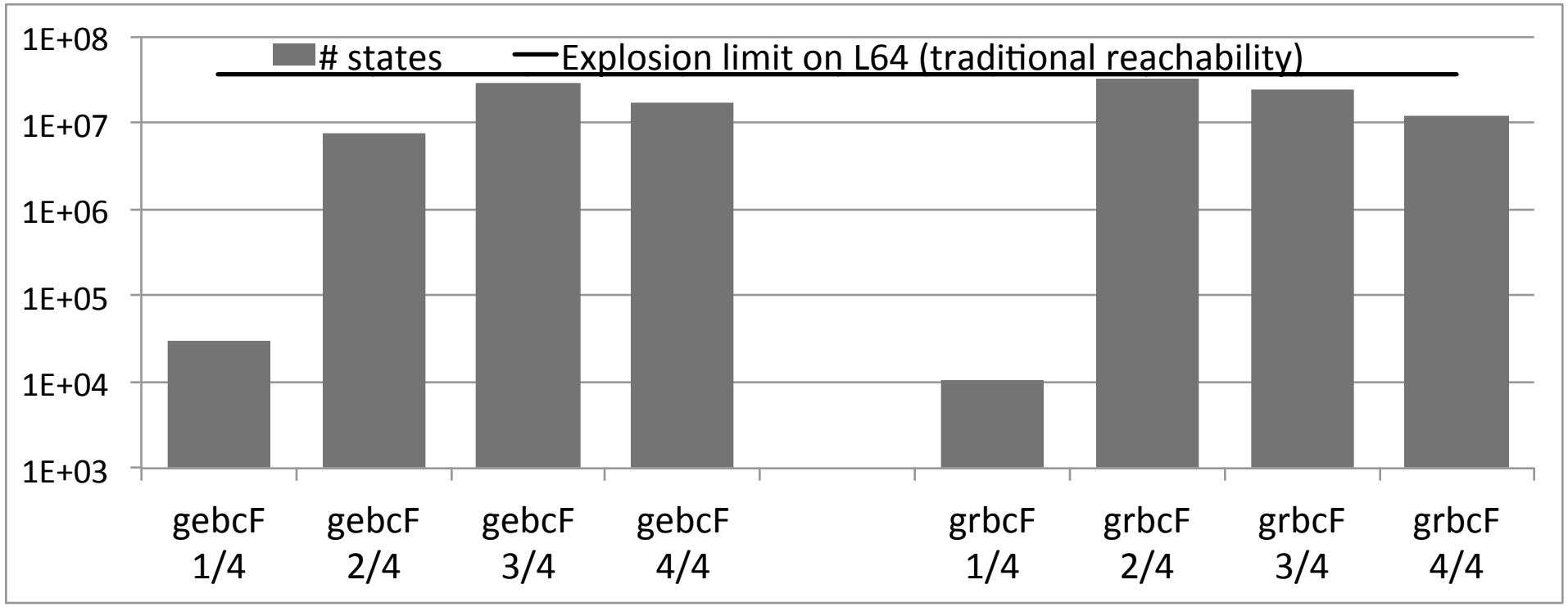
Traditional reachability: 1 Failure



3 Pilot Interactions + 1 Failure on 64 GB RAM

ENSTA Bretagne

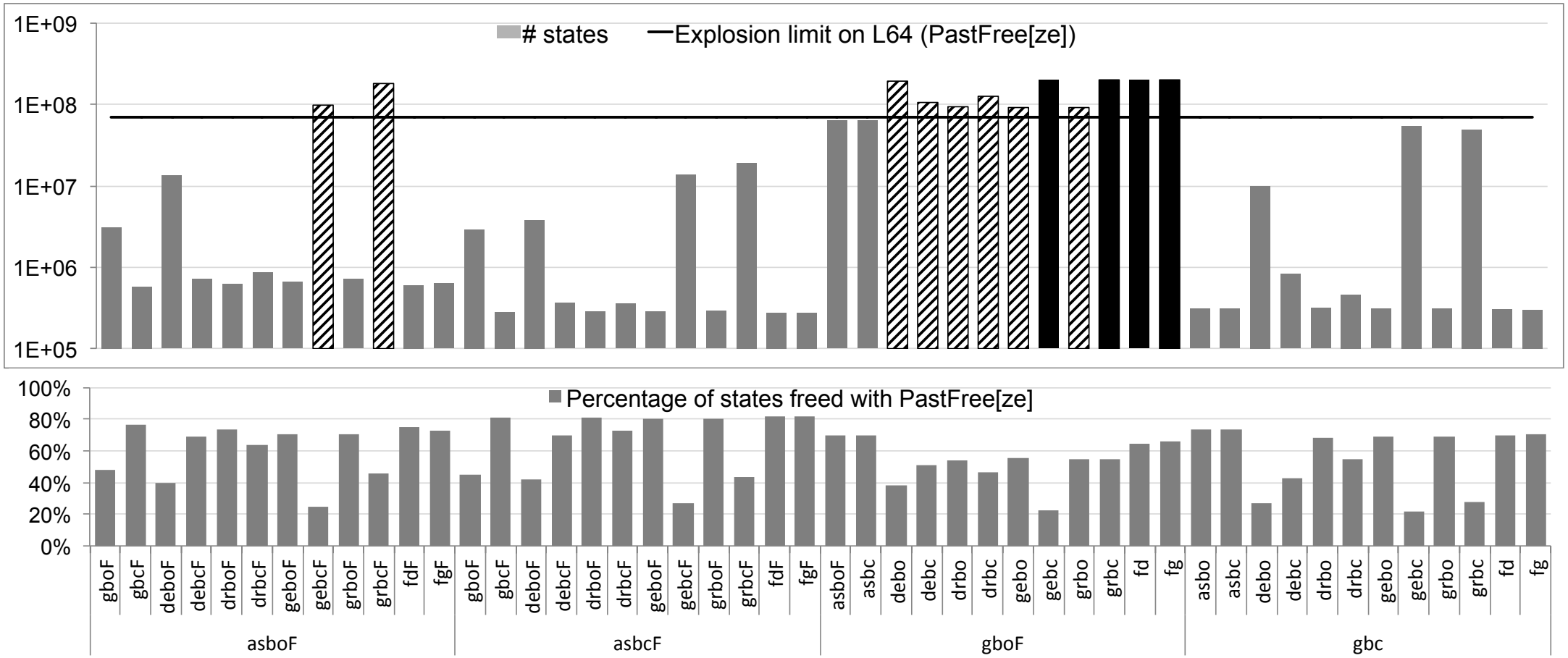
Traditional reachability + **SPLIT**: 1 Failure



3 Pilot Interactions + 1 Failure on 64 GB RAM

ENSTA Bretagne

Reachability: 2 Failures – PastFree[ze]



ENSTA Bretagne

Context-aware Verification: **Completeness**

- CaV is not complete (not exhaustive)
  - some states remain undiscovered (e.g. the states unraveled by a longer acyclic verification guide – 6-steps unrolling)
- A completeness bound should be proved. This is very difficult.
  - Can this completeness bound be proved automatically ?

Yes, using PastFree[ze] in some LGS cases

	asbo	asbc	gbo	gbc	debo	debc	drbo	drbc	gebo	gebc	grbo	grbc	fd	fg
b_{guide}	16	16	18	17	20	20	18	20	20	X	18	X	20	20

Open question : In which cases can the proof be automated ? In general ?

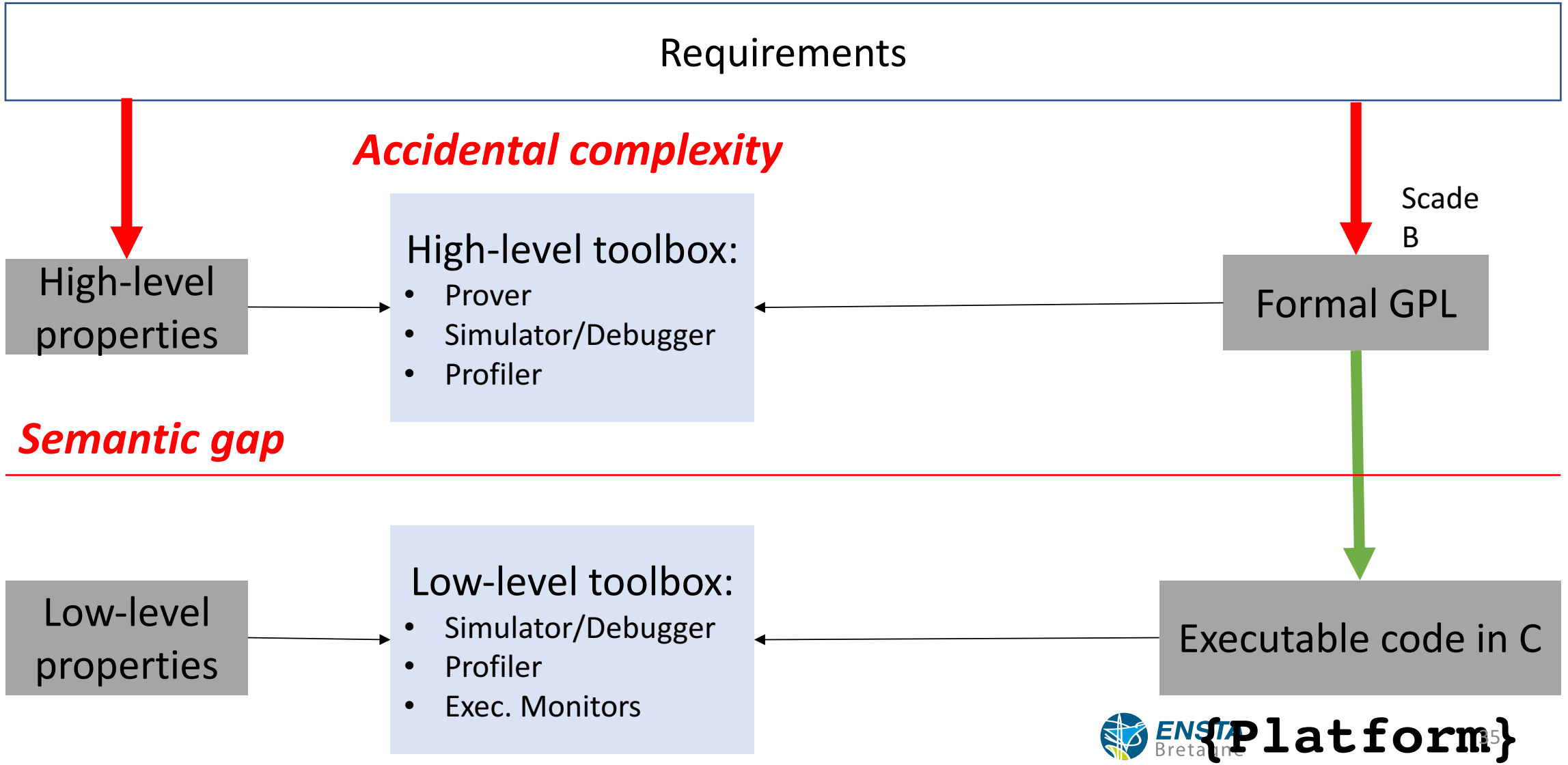
ENSTA Bretagne

Industrial Challenges

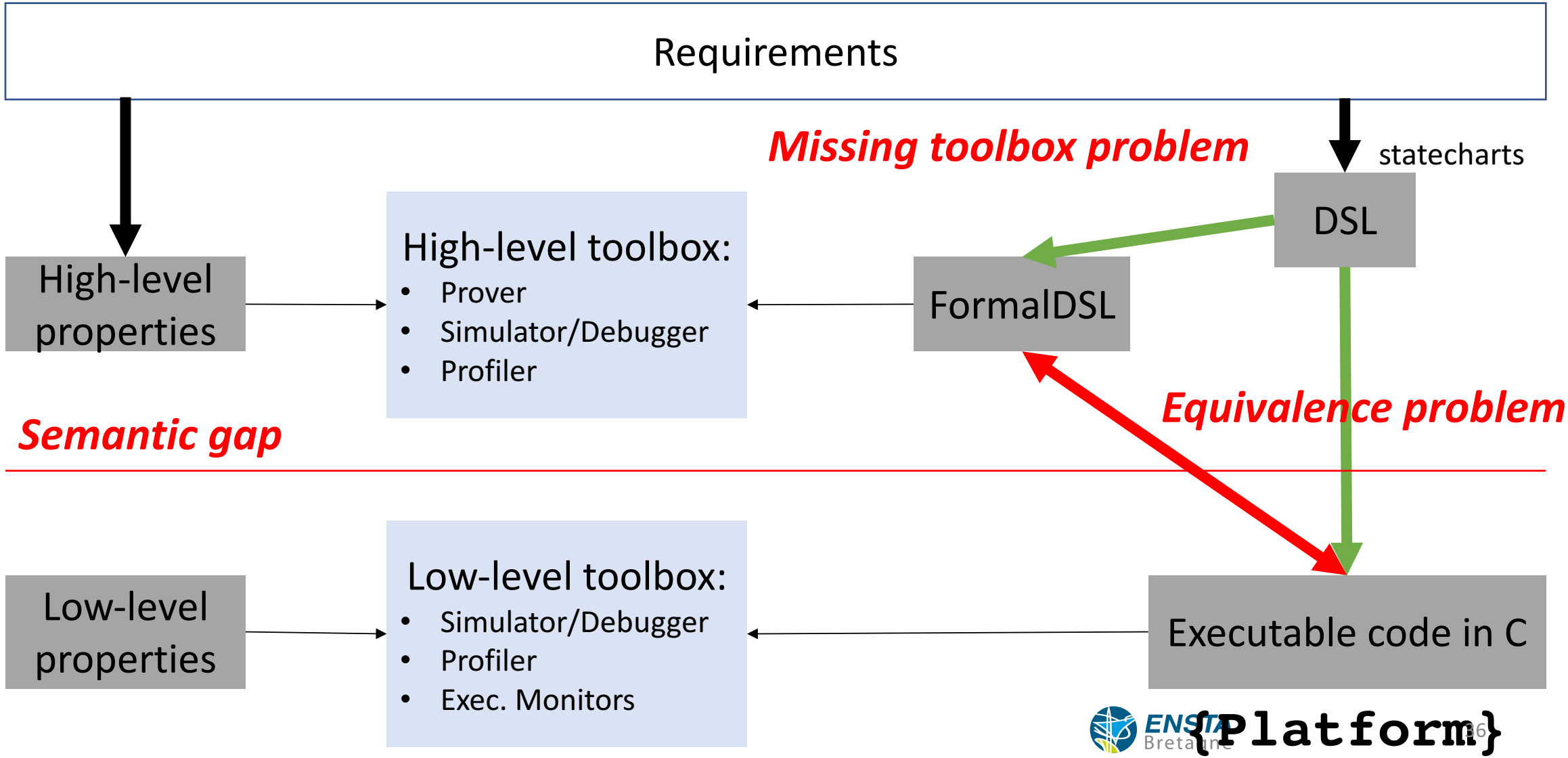
- Notations *[OK]*
  - Requirements (Doors, Doc)
  - Models (UML, AADL, SDL)
- Tools *[OK]*
  - Model-checkers
  - Other formal analysis tools
- Problems, Solutions et Questions
  - Requirements
  - Environment (the model should be closed 4 verification)
  - **System model**
  - Abstraction & Modular Decomposition

ENSTA Bretagne

DSL-based Diagnosis 4 Critical Systems

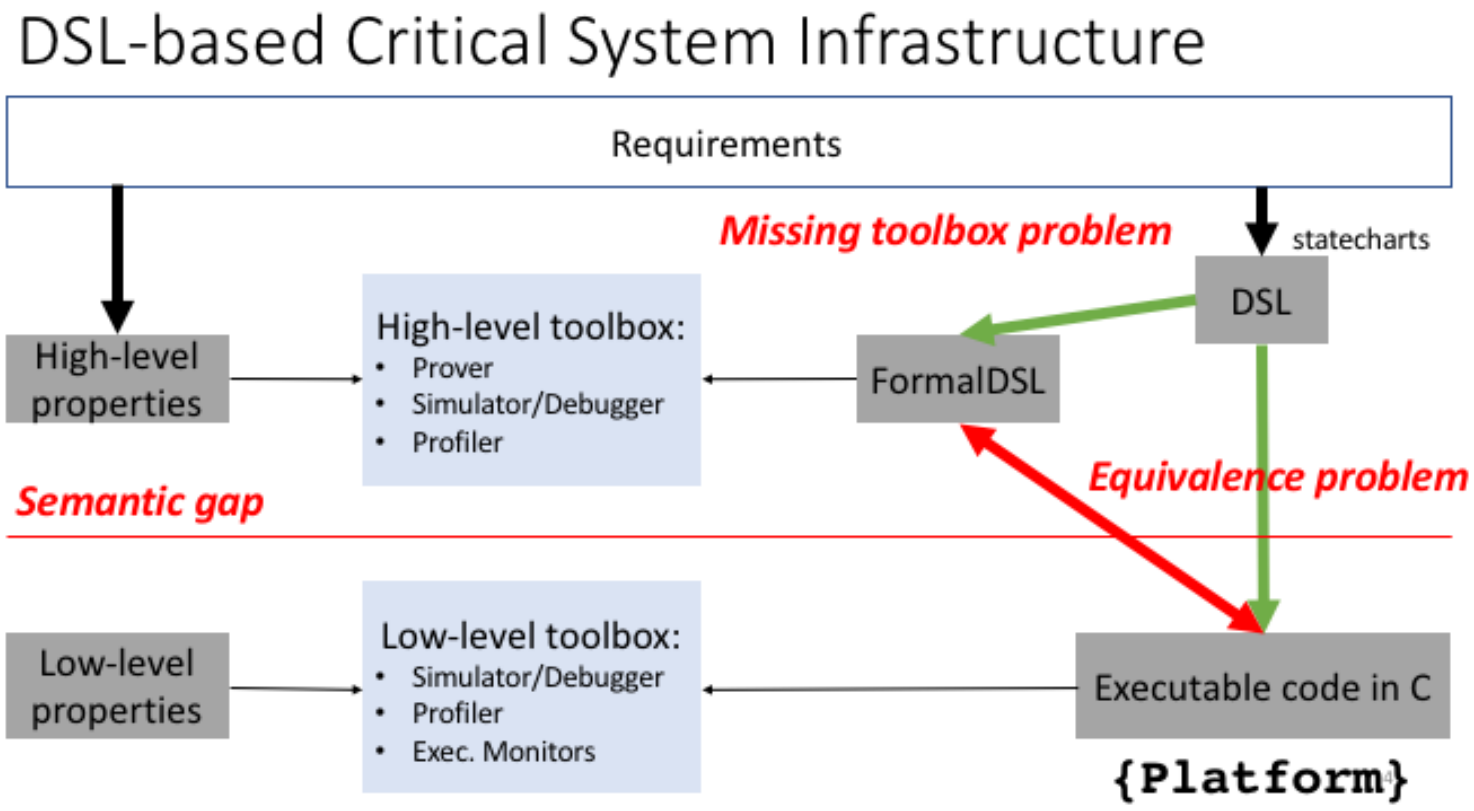


DSL-based Critical System Infrastructure



DDASCA DEPARTS

Model-checking scalability with Context-aware Verification
- Explicit verification guides
- Dedicated algos : Split, PastFree[ze], Folding
- Realistic case studies



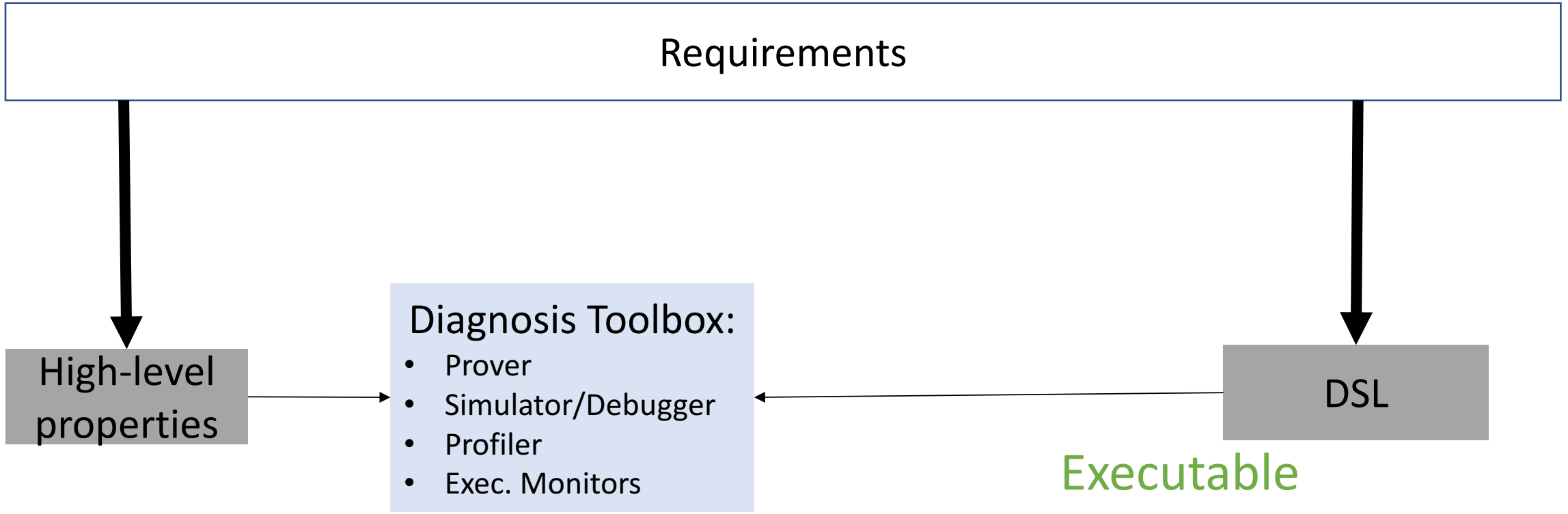
UML Statechart → Fiacre
$AEFD_{SNCF}$ → Fiacre
Fiacre with embedded Scade

New language : ABCD

ABCD

ENSTA Bretagne

DSL-based Critical System Infrastructure



Missing toolbox problem

The Problem: How to **make** the ***connection*** ?

Domain-specific diagnosis ⟵ — — — ⟶ ***Language workbenches***

Moldable debugger — *Chis et al. CLSS'15*

DSProfile — *Sloane et al. SCP'16*

MetaSpy — *Ressia et al. JOT'02*

LTSMin — *Kant et al. TACAS'15*

Gemoc studio — *Bousse et al. SLE'15*

Spoofax — *Kats et al. OOPSLA'10*

MPS — *jetbrains.com/mps*

K Framework — *Rosu et al. JLAP'10*

The Problem: Requirements

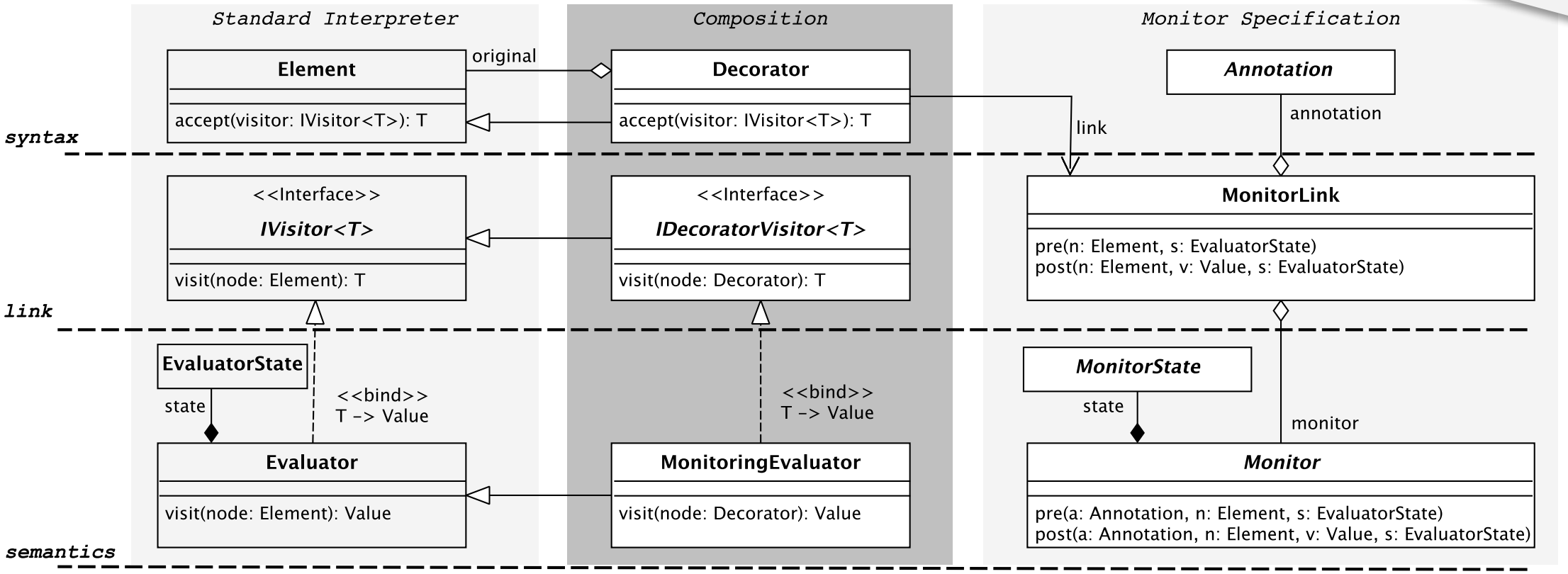
Domain-specific diagnosis  ***Language workbenches***

DSL monitoring is the process of **observing the execution of a program** expressed in a DSL.

[R01] Completeness	*[R06]* Portability
[R02] Non-Interference	*[R07]* DSL Runtime Integration
[R03] Genericity	*[R08]* Tool Integration
[R04] Composability	*[R09]* Minimize the Gap
[R05] Unanticipated Monitoring	*[R10]* Break the Rules

Object-Oriented Design Pattern for DSL Program Monitoring

Z.Drey et C.Teodorov @ SLE'16



ENSTA Bretagne

Steps towards integration:

Observation & Control: Object-Oriented Monitoring Pattern

[R01] Completeness

[R02] Non-Interference

[R03] Genericity

[R04] Composability

[R05] Unanticipated Monitoring

[R06] Portability

[R07] DSL Runtime Integration

[R08] Tool Integration

[R09] Minimize the Gap

[R10] Break the Rules

But only a pattern – no framework / no tools yet

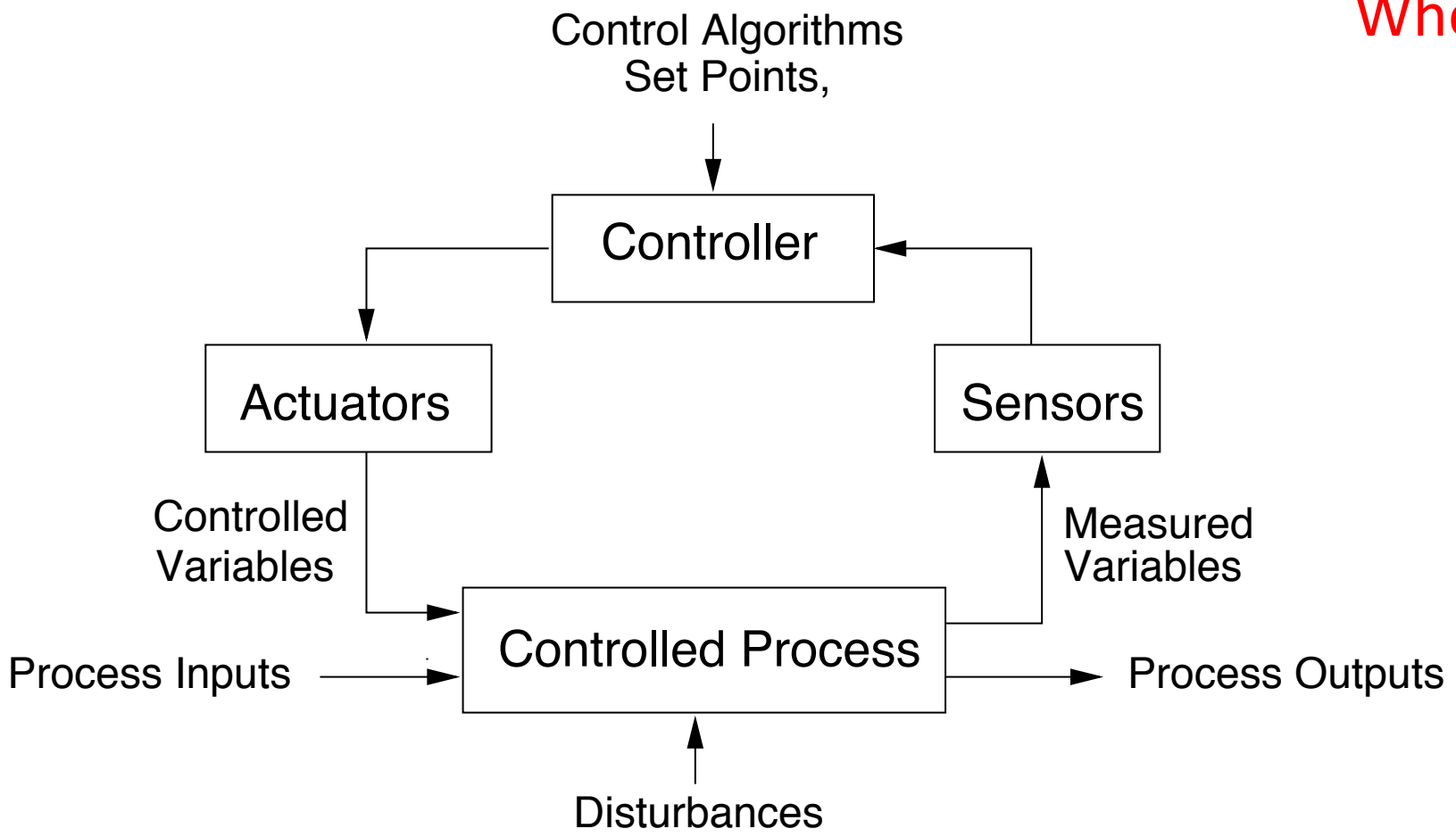
Industrial Challenges

- Notations *[OK]*
  - Requirements (Doors, Doc)
  - Models (UML, AADL, SDL)
- Tools *[OK]*
  - Model-checkers
  - Other formal analysis tools
- Problems, Solutions et Questions
  - Requirements
  - Environment (the model should be closed 4 verification)
  - System model
  - **Abstraction & Modular Decomposition**

ENSTA Bretagne

A standard control loop

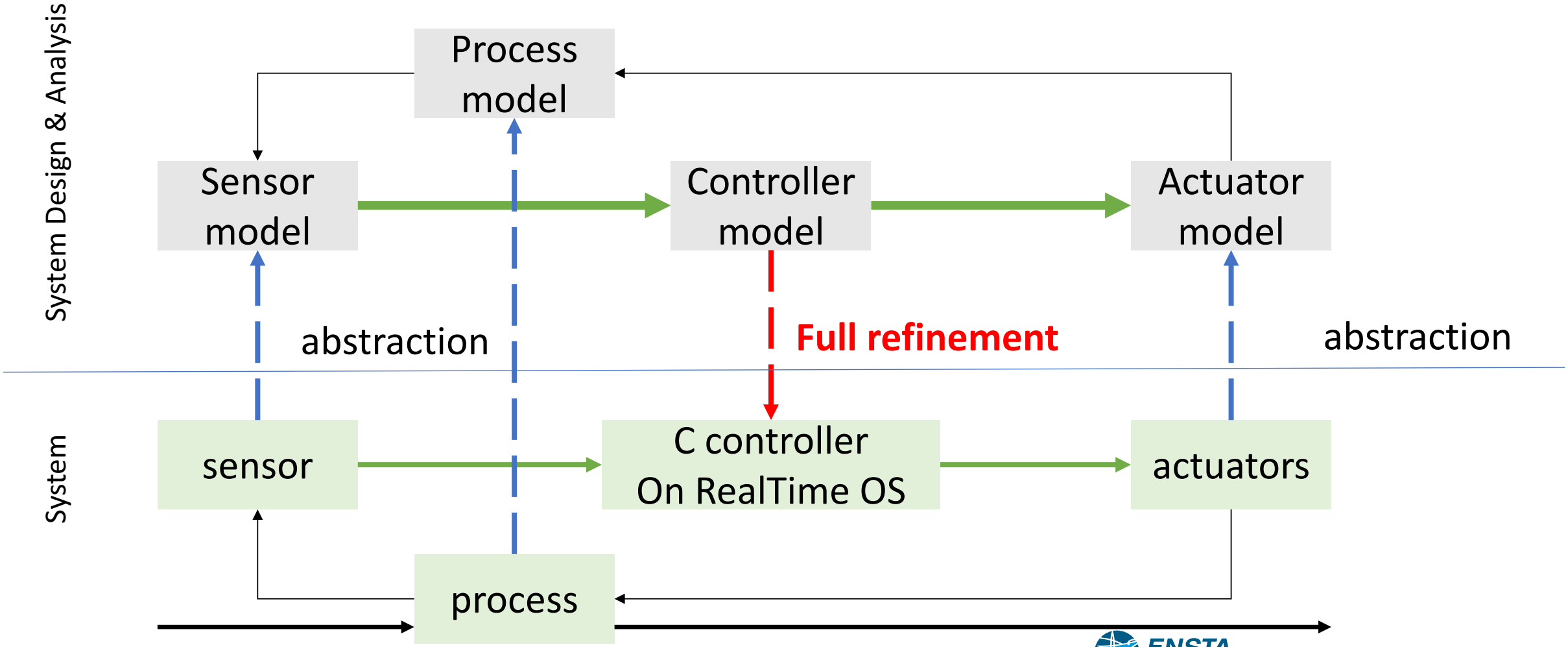
Where do we start ?



ENSTA Bretagne

Steps towards integration:

Raffinement vs Abstraction

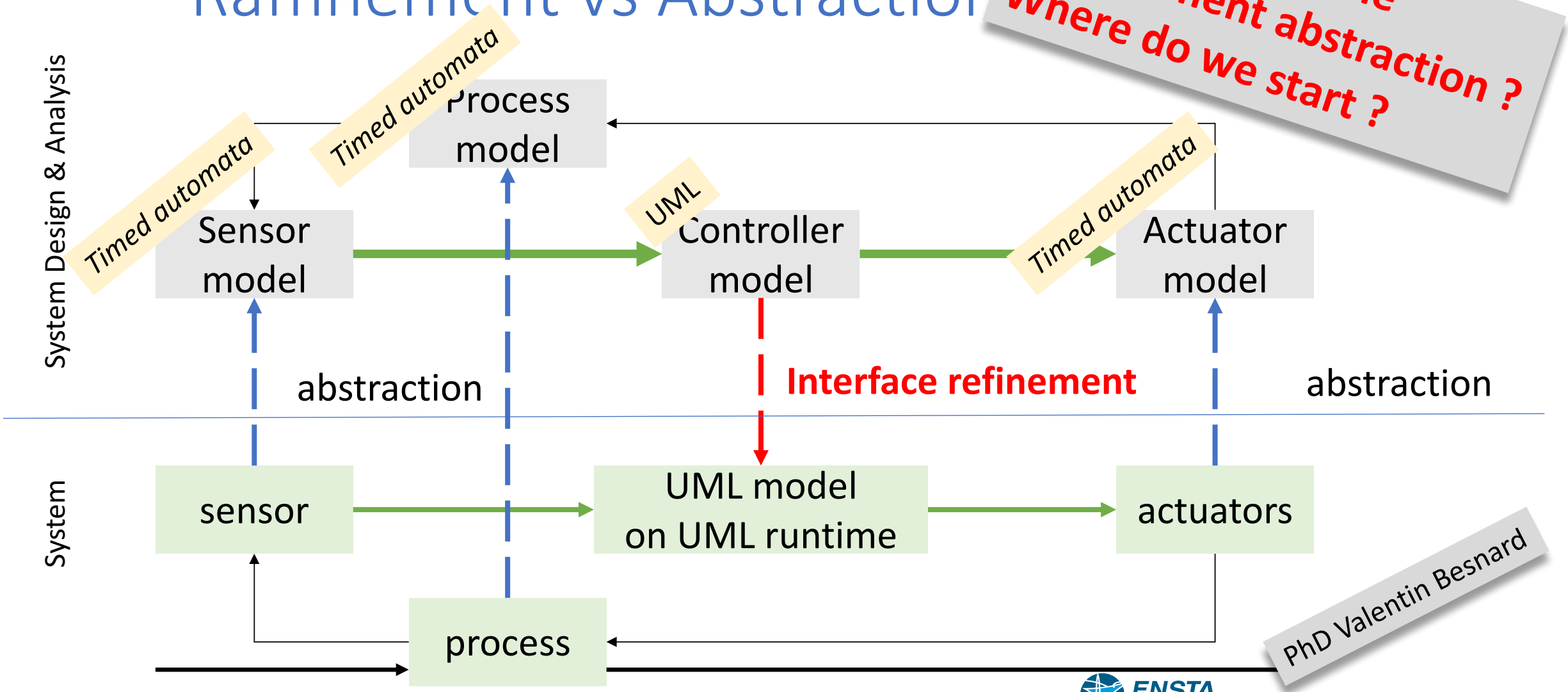


ENSTA Bretagne

Steps towards integration:

Raffinement vs Abstraction

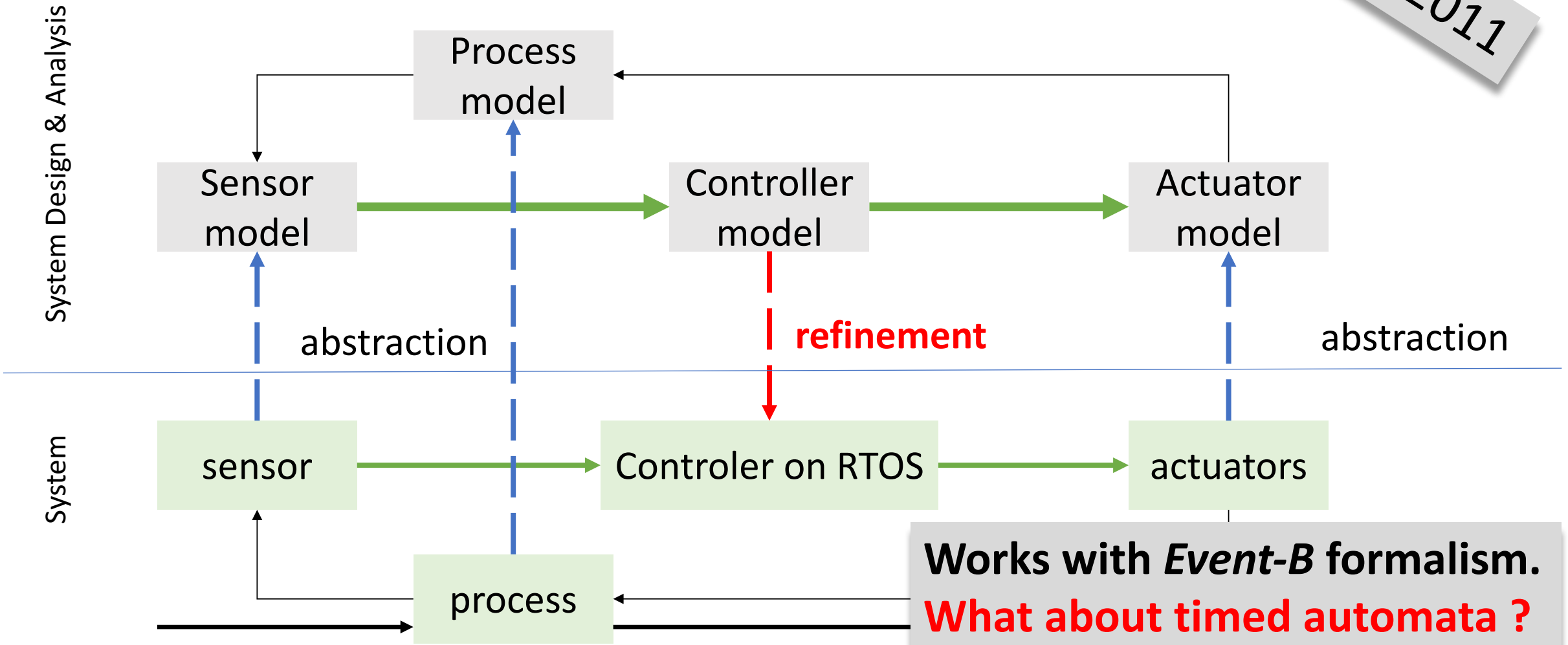
How do we get the environment abstraction ?
Where do we start ?



ENSTA Bretagne

Steps towards integration:
Raffinement vs Abstraction

Hudon et al. 2011



Works with *Event-B* formalism.

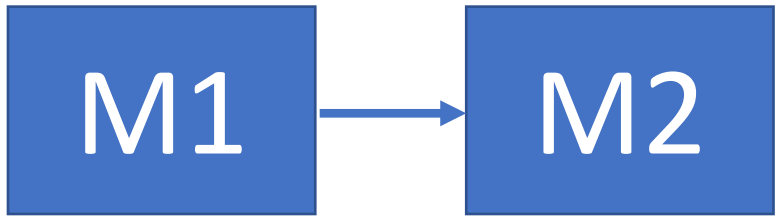
What about timed automata ?

Industrial Challenges

- Notations *[OK]*
  - Requirements (Doors, Doc)
  - Models (UML, AADL, SDL)
- Tools *[OK]*
  - Model-checkers
  - Other formal analysis tools
- Problems, Solutions et Questions
  - Requirements
  - Environment (the model should be closed 4 verification)
  - System model
  - **Abstraction & Modular Decomposition**

ENSTA Bretagne

Assume Guarantee Reasoning



$\models$ P

Yes – Cobleigh et al. 2003

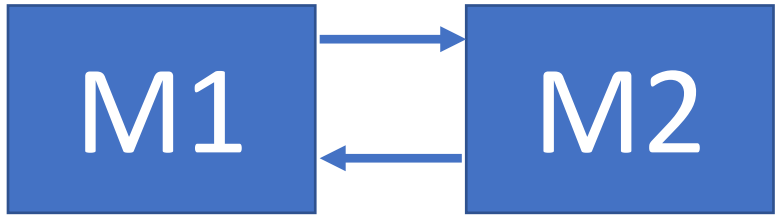
$$\frac{(\text{Premise 1}) \quad M_1 \vDash \quad true \quad \rhd g_1 \qquad (\text{Premise 2}) \quad M_2 \vDash \quad g_1 \quad \rhd P}{M_1 \parallel M_2 \vDash P}$$

Can we verify M1 independently of M2 ?

g1 is an abstraction of M1
We can compute it automatically.

ENSTA Bretagne

Circular Assume Guarantee Reasoning



$\models P$

Yes – Elkader et al. 2015

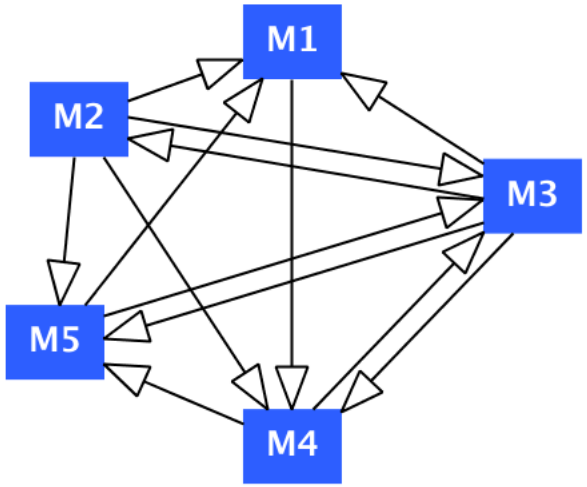
$$\frac{\begin{array}{ll}(\text{Premise 1}) & M_1 \models g_2 \triangleright g_1 \\ (\text{Premise 2}) & M_2 \models g_1 \triangleright g_2 \\ (\text{Premise 3}) & g_1 \| g_2 \models P\end{array}}{M_1 \| M_2 \models P}$$

Can we verify M1 independently of M2 ?
What are M1 & M2 ?

g1 & g2 are abstractions of M1 & M2.
We can compute them automatically.

ENSTA Bretagne

N-Way Circular Assume Guarantee Reasoning



$\models$ P

Can we verify each M_i independently the others?
M_i naturally maps to processes.s

Yes – Elkader et al. 2016

$$\frac{\begin{array}{ll}\text{(Premise 1)} & M_1 \models G_1 \triangleright g_1 \\ \text{(Premise 2)} & M_2 \models G_2 \triangleright g_2 \\ \quad\ldots & \\ \text{(Premise n)} & M_n \models G_n \triangleright g_n \\ \text{(Premise n+1)} & G_{n+1} \models P\end{array}}{M_1 \| M_2 \| \cdots \| M_n \models P}$$

$G_i \subseteq G - \{g_i\}$ for $i < n + 1$,
gi are abstractions,
compute them automatically

ENSTA Bretagne

N-Way Circular Assume Guarantee Reasoning

- Limitations / Open questions :
  - The « g_i » abstractions are computed using SAT.
    - Is there an better/specialized algorithm ?
    - What is the algorithmic complexity ?
  - Each component is developed independently under some assumptions.
    - Can we integrate these « development assumptions » in the approach ?
    - Does it help ?
  - This N-Way Circular Reasoning is limited to safety properties.
    - Can this approach handle **liveness properties** ?

ENSTA Bretagne

Questions ?

Happy research career !

ENSTA Bretagne