FROM RESEARCH TO INDUSTRY

cea tech

list

FORMAL METHODS

HIGH-ASSURANCE APPROACHES FOR SAFETY AND SECURITY

This document is the property of CEA. It can not be copied or disseminated without its authorization.

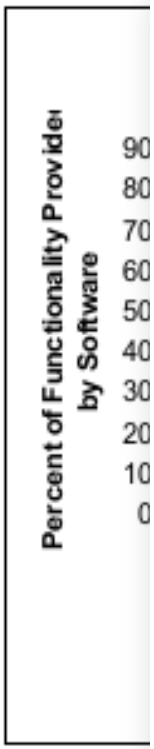


Figure 7. Growth in

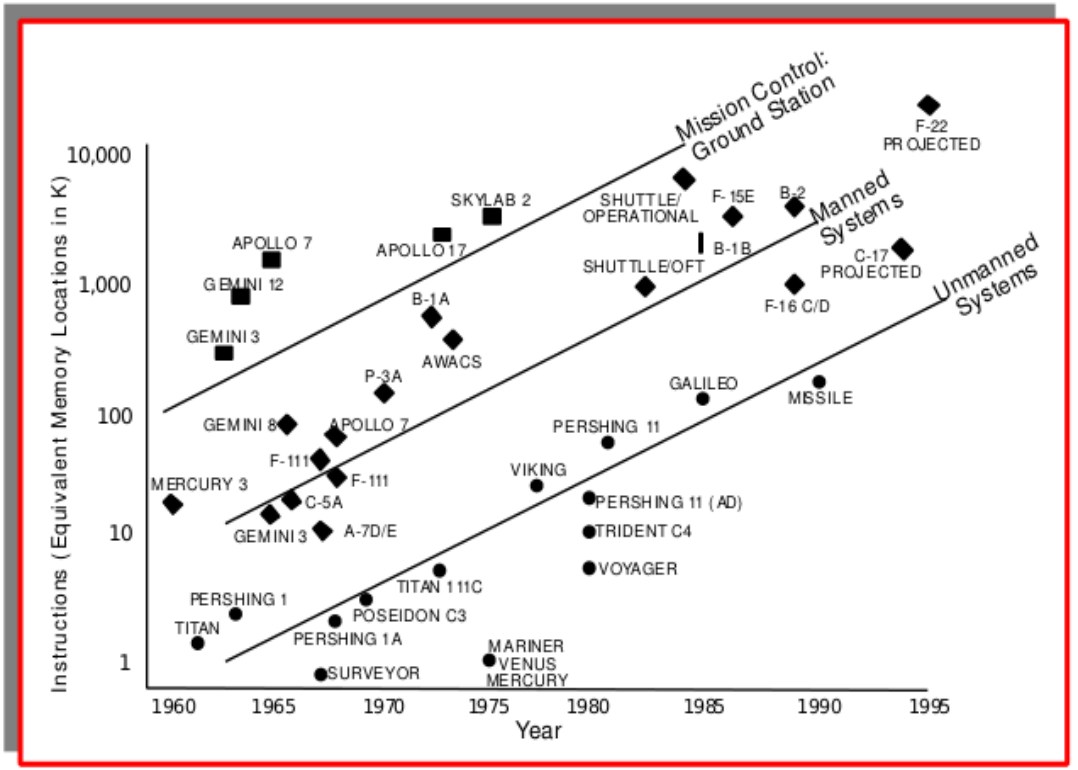
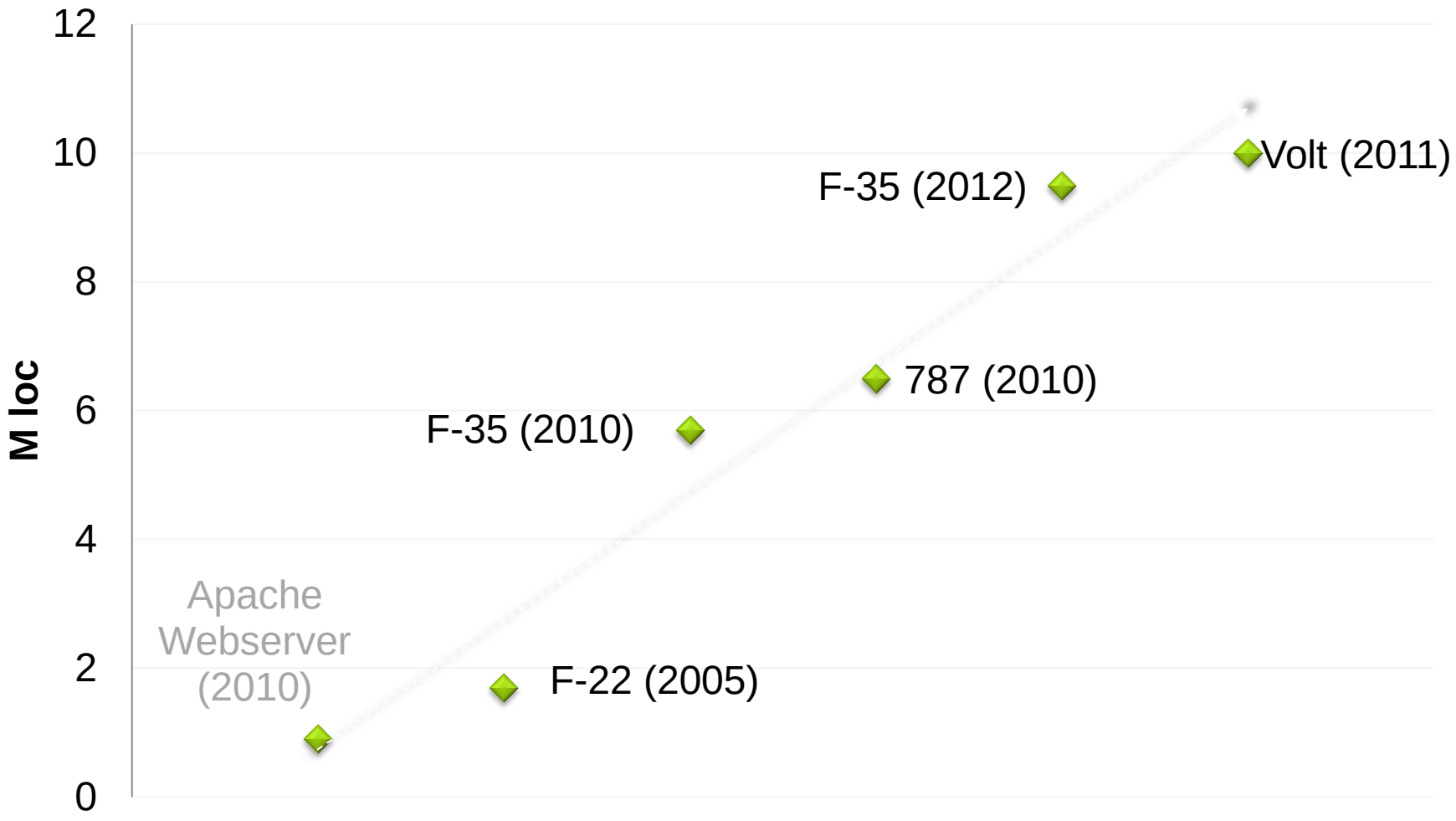


Figure 1-2. Software Systems Size Growth 1960 to 1995

SOFTWARE SIZE



TRUST

WHY DO WE (OUR) SYSTEMS?

STRINGENT CERTIFICATION STANDARDS REQUIRE STRONG GUARANTEES

EN 50128 for rail safety

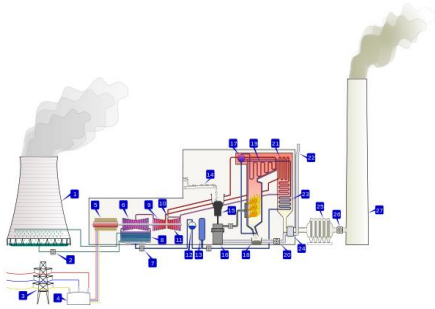
ISO 26262 for road vehicles

DO-178B/C for aerospace safety

IEC ISO 61513 for nuclear power plants

IEC 62304 for medical equipment

IEC 61508 reference for E/E/PE systems



ISO 15408 (Common Criteria)

IEC 62443 (ISA 99)

NIS Directive

ISO

Common Criteria

QUESTION | WHAT ARE THE MEANS TO BUILD TRUST?

- **Process-based assurance**

  Based on testing, V&V tools designed in the 1980s
  Familiar, but **expensive** to scale up for software-intensive systems

  Inapplicable to COTS software components

- **Product-based assurance**

  Using formal methods to provide strong guarantees regarding:
  - Compliance with requirements
  - Absence of vulnerabilities

  Disruptive, but **reduce the cost** of meeting mandatory requirements

  New-generation verification tools are reaching maturity in terms of cost effectiveness

FORMAL TOOLS | TO UNDERSTAND SOFTWARE PROPERTIES

- **Properties are captured using precise specifications**

$$0 <= i,j < N \rightarrow \textbf{valid_read}\ (\&(buff.data[i]).cell[j].next \rightarrow val)$$

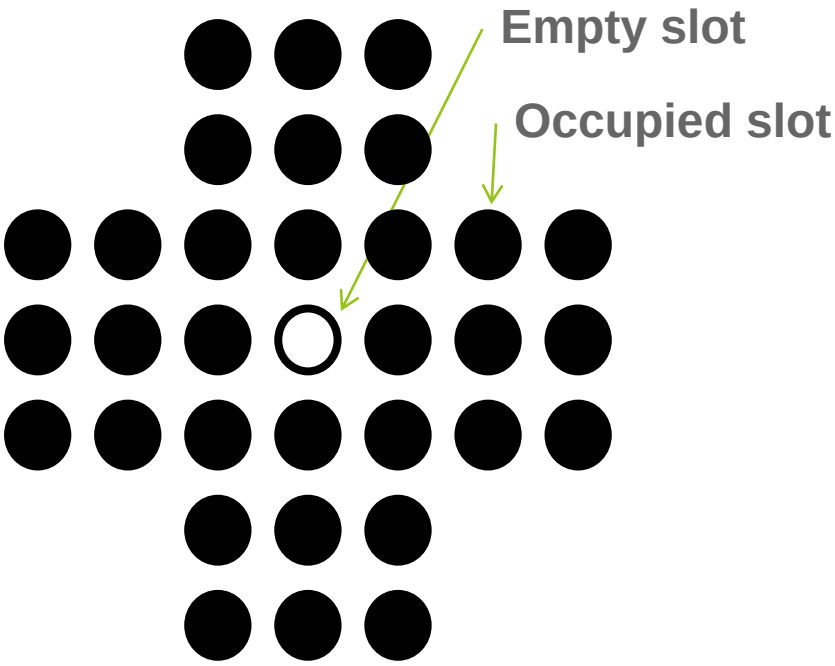
- **Software systems are represented as sets of rules**
  - At appropriate levels of abstraction
  - Transforming the system state
  - Satisfying certain properties

FORMAL GUARANTEES | ON A GIVEN PERIMETER

- **Innovative methods are used to show that some software properties hold…**

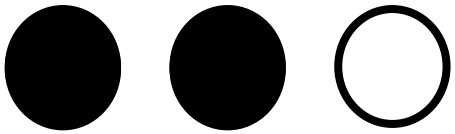
- **… or to provide insight on why other properties do not.**

PEG SOLITAIRE



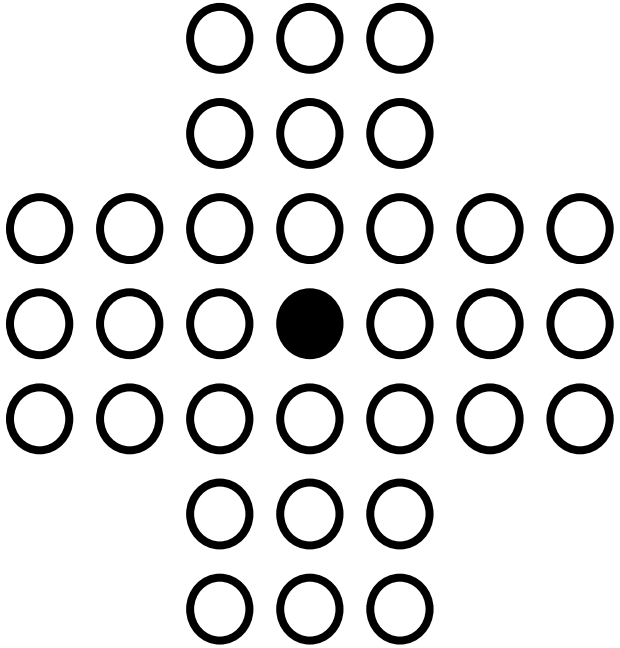
INITIAL STATE

ONE MOVE



AIM

Reach the following final state.



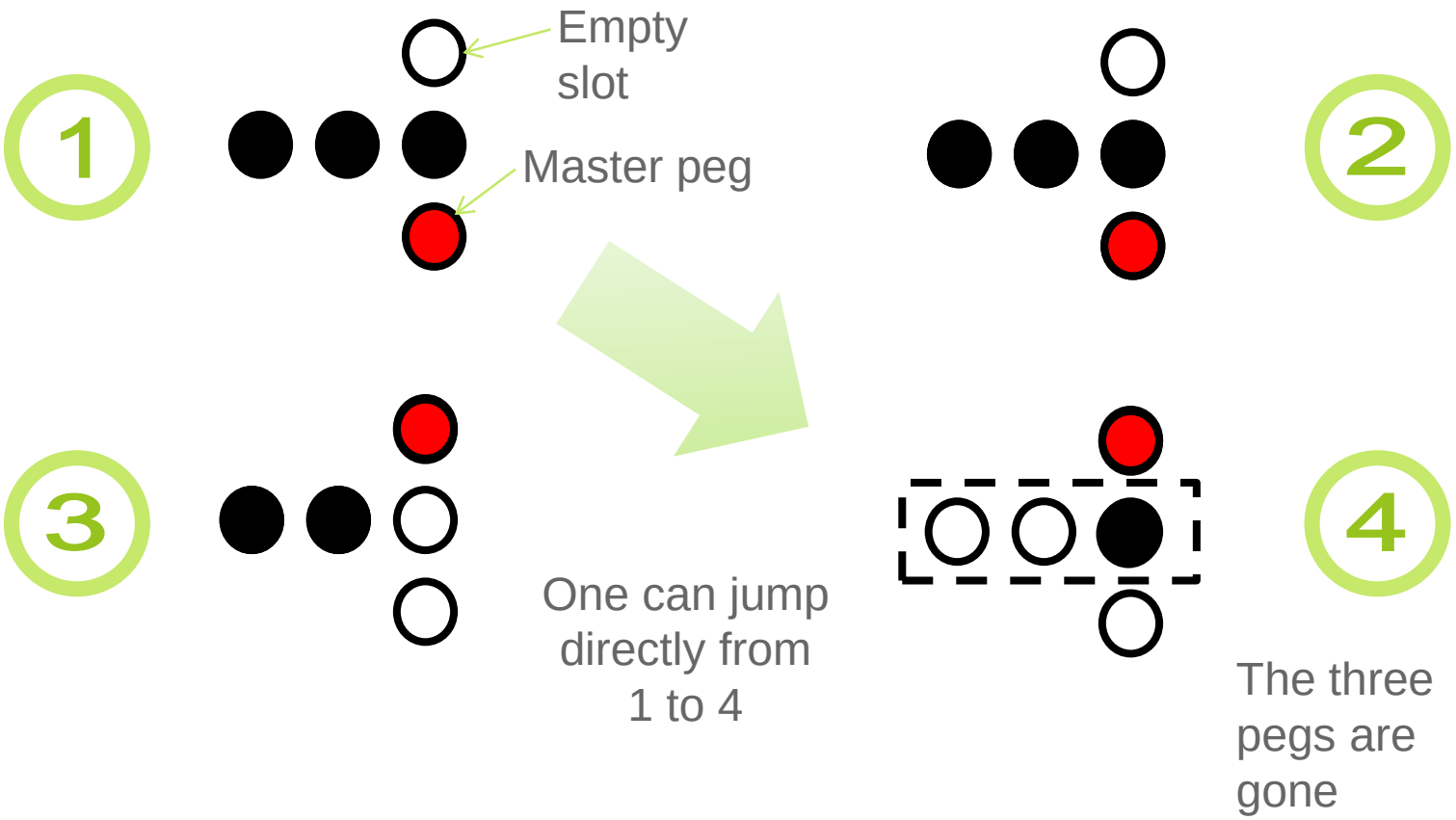
PEG SOLITAIRE | TRY AND PLAY

- There are many possible pegs and moves to consider
- It is hard to get a global vision for how to get to a given final state

We can leverage a *formalism* (semantics and algorithm) to analyze and verify the problem.

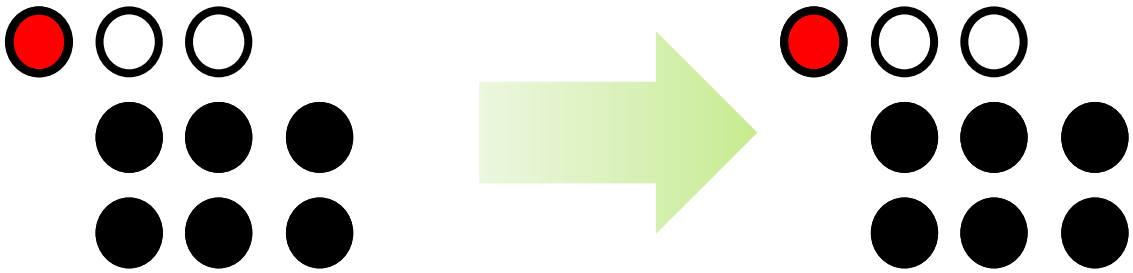
A THREE WORD VOCABULARY

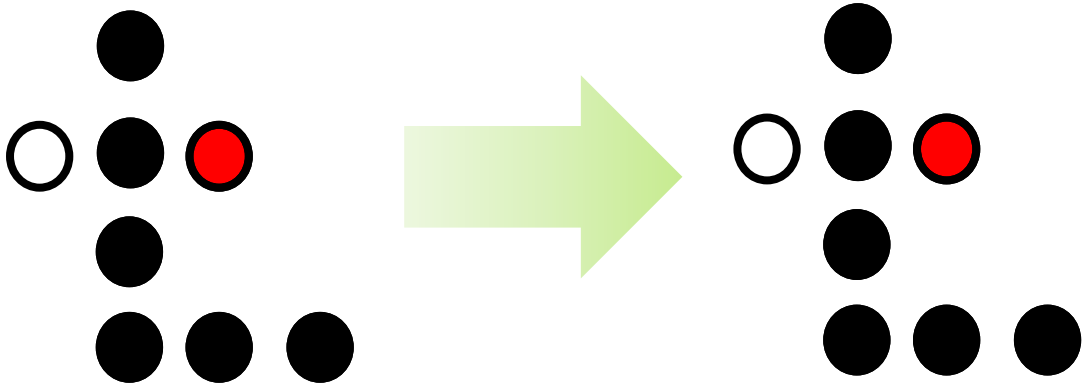
WORD 1 | USE THE RED PEG TO REMOVE A LINE OF THREE



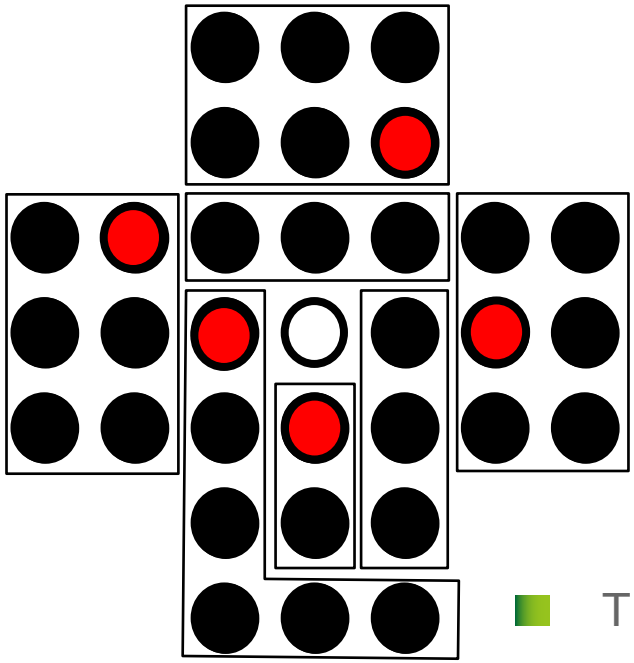
A THREE WORD VOCABULARY

WORDS 2 AND 3





AND BACK TO A SIMPLE PUZZLE

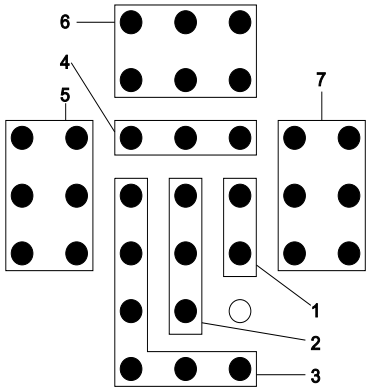


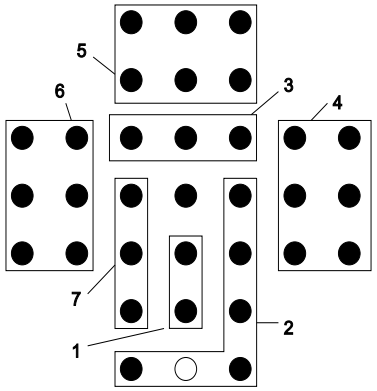
- This formalism has resulted in a more manageable problem, and guided us toward the solution.

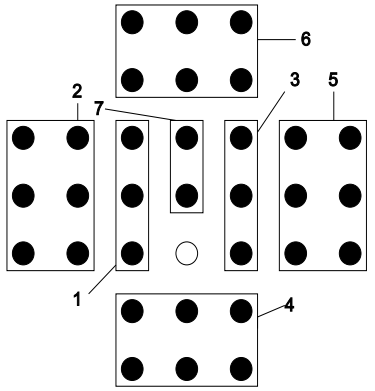
- The semantics : the three words
- The algorithm : the sequence of words used to solve the puzzle

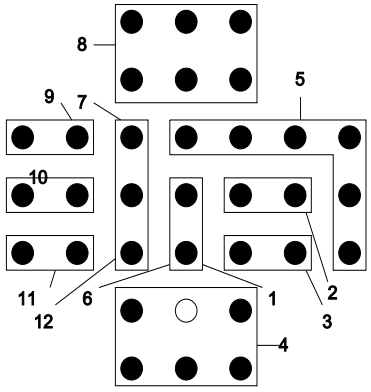
IT IS A GENERIC FORMALISM

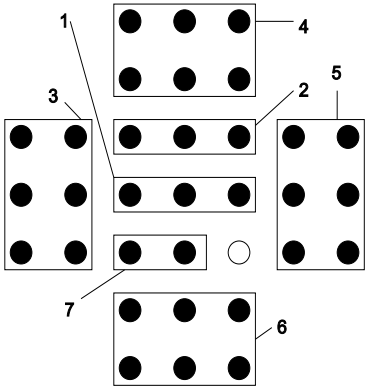
- One can use it to solve any puzzle with the same sort of initial state

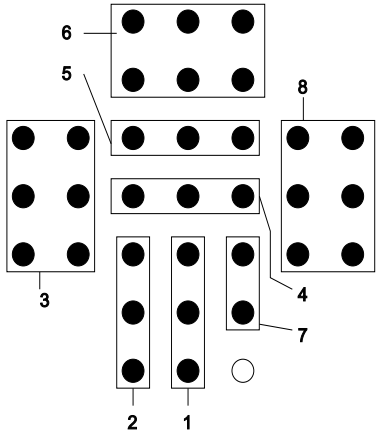












EXAMPLE | A FORMALISM FOR SOFTWARE VALIDATION

Handles logical formulas

ACSL: a mathematical universe

```
/*@ ensures \result >= x && \result >= y;
    ensures \result == x || \result == y; */
```

```
int max (int x, int y) {
  return (x > y) ? x : y;
}
```

Computational universe

Frame the computational universe using logics

Modifies transistor states on micro-chips during execution

TOOLS FOR HIGH CONFIDENCE SOFTWARE & SYSTEMS

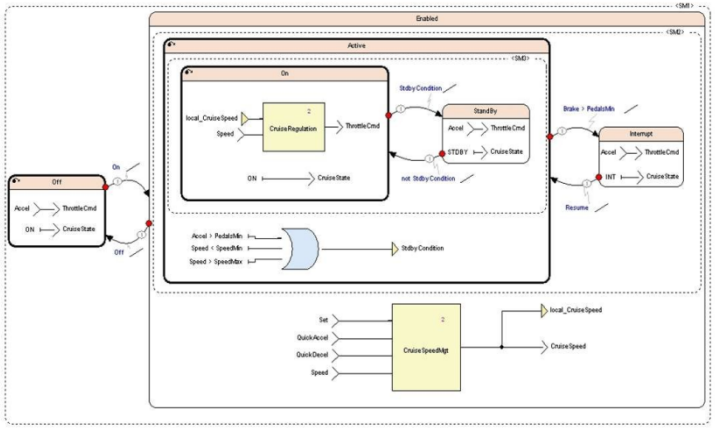
FORMAL METHODS

- Provide zero-defect guarantees
- Can work with existing components, tools, processes and certifications
- Understand cost-efficiency is a fundamental asset
- Require collaborating with stakeholders to
  - select the best tools
  - design innovative methods to guide their use
  - help move norms and standards forward

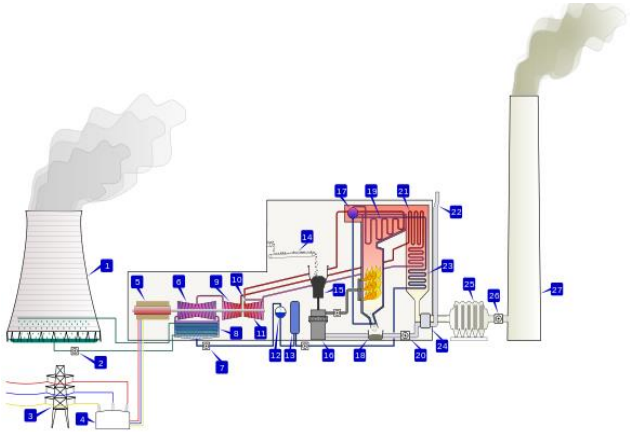
FORMAL MODEL EXPLORATION | #SCADASYSTEMS

How do we show that a set of inputs trigger all predefined model constructs?

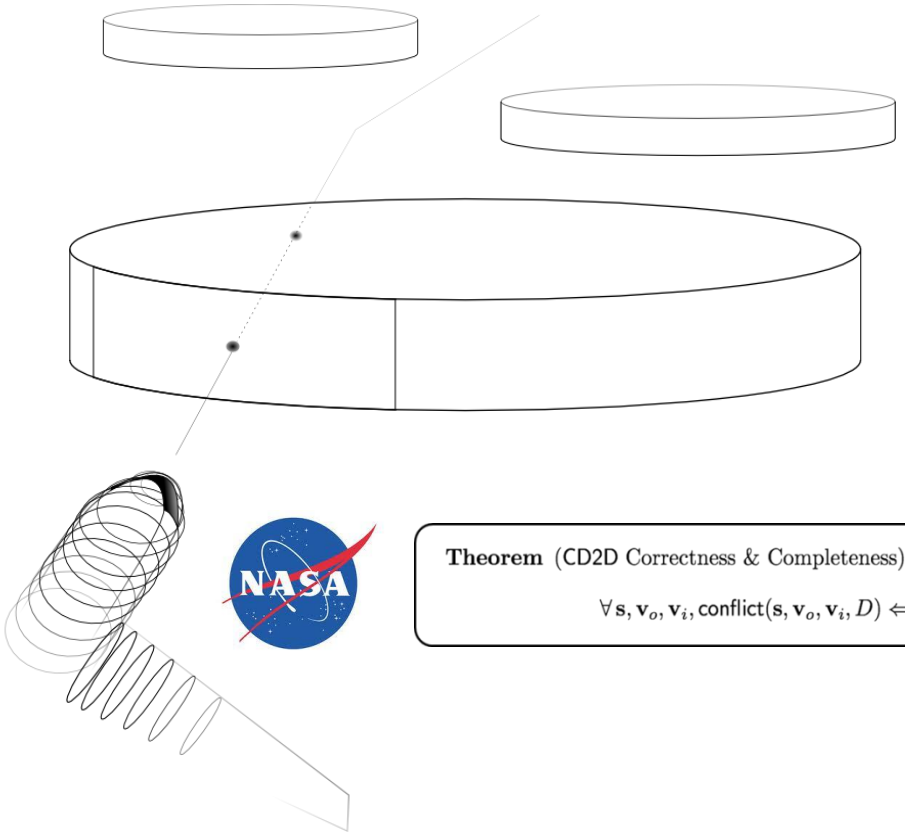
- **Check SCADE 6 models for reachability**
- **Scaling up to long temporizations and complex computations**
- **Justification and certification to IEC60880 Class 1**



Top Level of the Cruise Control application

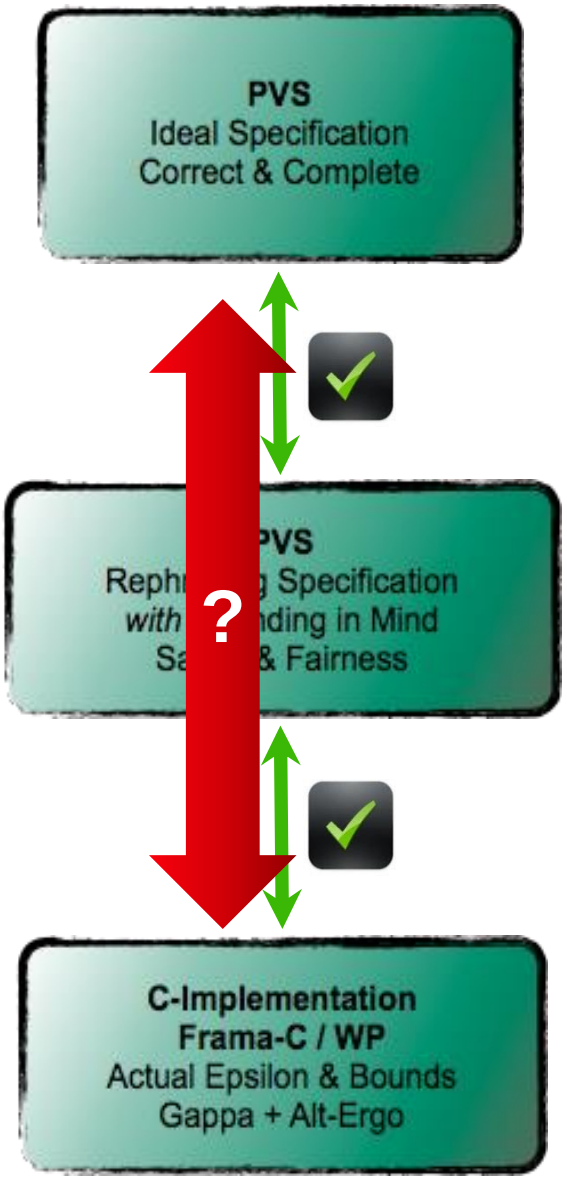


ALGORITHMIC CONFORMANCE PROOF | #ATC



Theorem (CD2D Correctness & Completeness)

$$\forall \mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, \mathsf{conflict}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D) \iff \mathsf{cd2d}(\mathbf{s}, \mathbf{v}_o, \mathbf{v}_i, D)$$



VERIFYING PROPERTIES | #AVIONICS

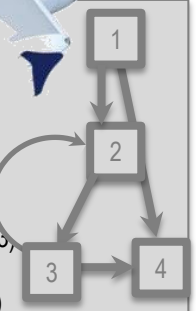
How do we verify that low-level requirements are met?

- **Automate unit proofs on critical code**
- **Using C code interactive verification**
- **A380, A350 & A400M programs**
- **Qualification and certification to DO178B/C level A**

Code Analysis :

```
1 : while (x>=1000) {
2 :   x= x-a;
3 :   }
4 :
```

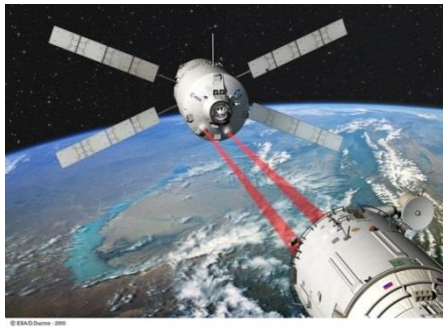
P1 : init(x)
P2 : (1000 ≤ x) ∧ (P1 ∨ P3)
P3 : P2[x+a]
P4 : (x<1000) ∧ (P1 ∨ P3)



Research of pre-conditions for infinite loops (Threat example) ➔ $x \geq 1000 \wedge a \leq 0$

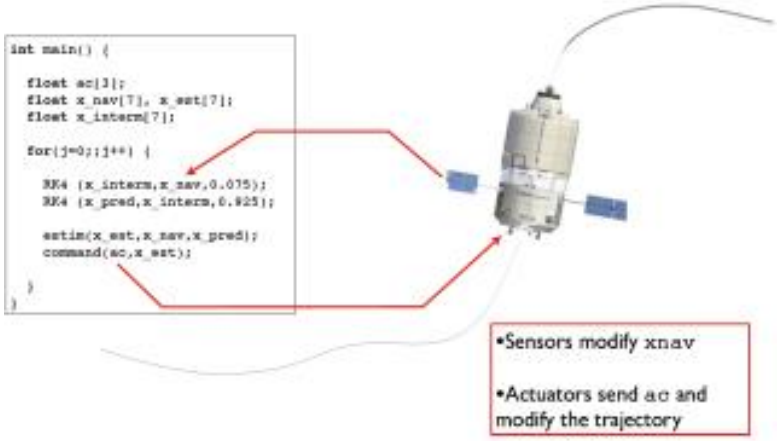
SAFETY

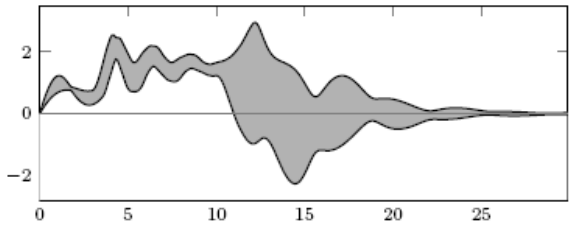
PILOTING LAWS VERIFICATION | #CYBERPHYSICALSYSTEMS



esa

AIRBUS DEFENCE & SPACE





Can we demonstrate that the system converges toward a safe "escape" state?

- **Build abstractions of both software and physical systems**
- **Propagate sets of numerical values across the system**
- **Formal over-approximation of physical behaviors with uncertain inputs**

METHODS AND TOOLS FOR HCS&S

- **Scientific roots and community**
  - Formal proof
  - Model checking
  - Constraint solving
  - Simulation
  - Abstract interpretation
  - Test case generation
  - Architecture Exploration
  - Synchronous languages
- **Prototyping and development of industrial-strength tools**
  for academia
  for the industry
- **Objectives**
  Raise the level of confidence in software
  Lower the costs of verification

Guided by industrial requirements

Scaling & Performance

WHAT WE DO | LIST

- Research and technology: formal methods and tools
- For intrinsically secure, high-confidence software and systems
- Contributing to normative and defacto standards
- Supporting component certification and tool deployment

Software Security Laboratory
Software & Systems Engineering Department
CEA LIST

Florent Kirchner
florent.kirchner@cea.fr

This document is the property of CEA. It can not be copied or disseminated without its authorization.