Principles of AltaRica Language and tools for system safety assessment

January 2016

Pierre.Bieber@onera.fr, Christel.Seguin@onera.fr

ONERA

THE FRENCH AEROSPACE LAB

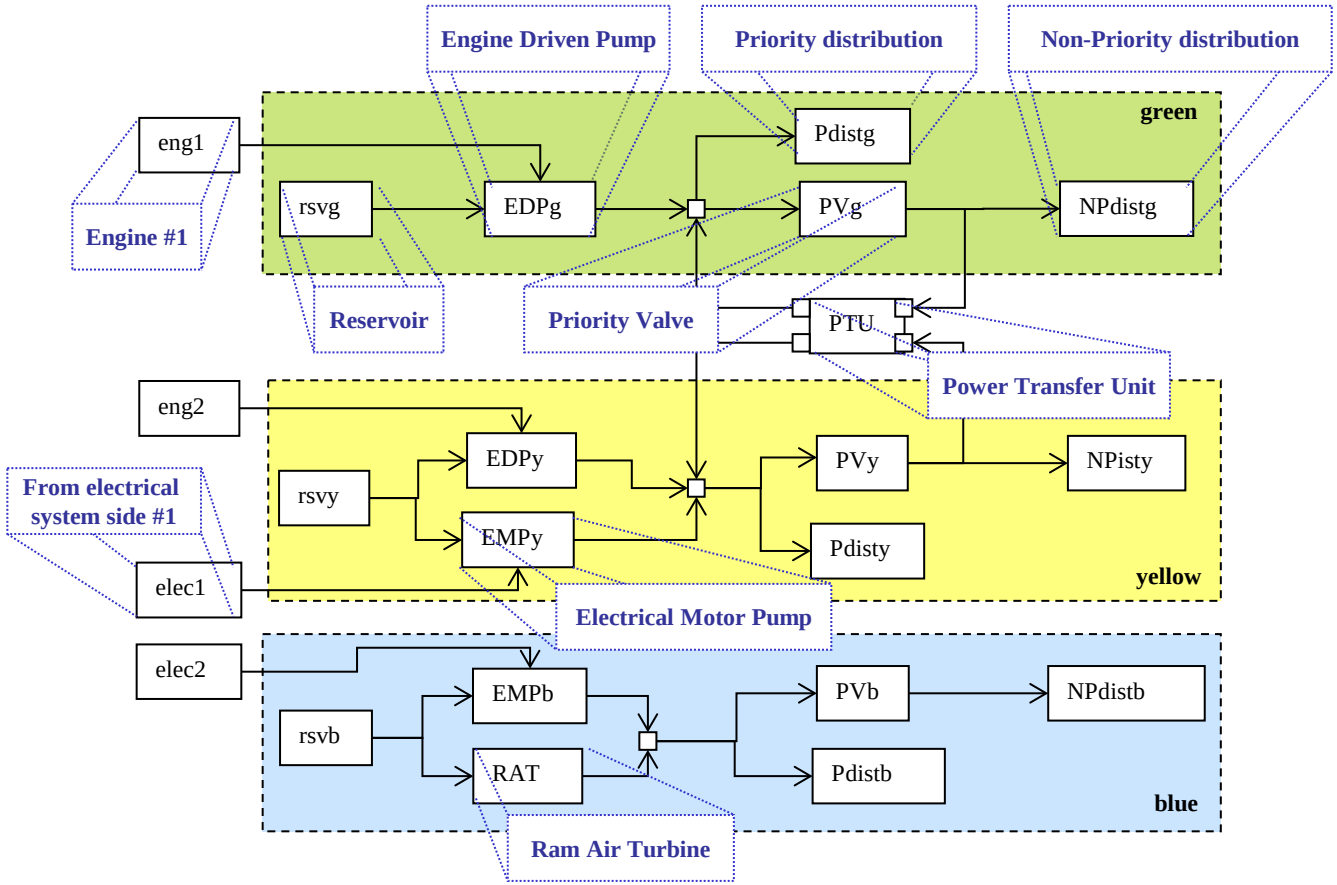
retour sur innovation

Outline

- System Safety Assessment
- AltaRica Basics
  - AltaRica Data Flow Language
  - Fault tree generation
- DAL Allocation

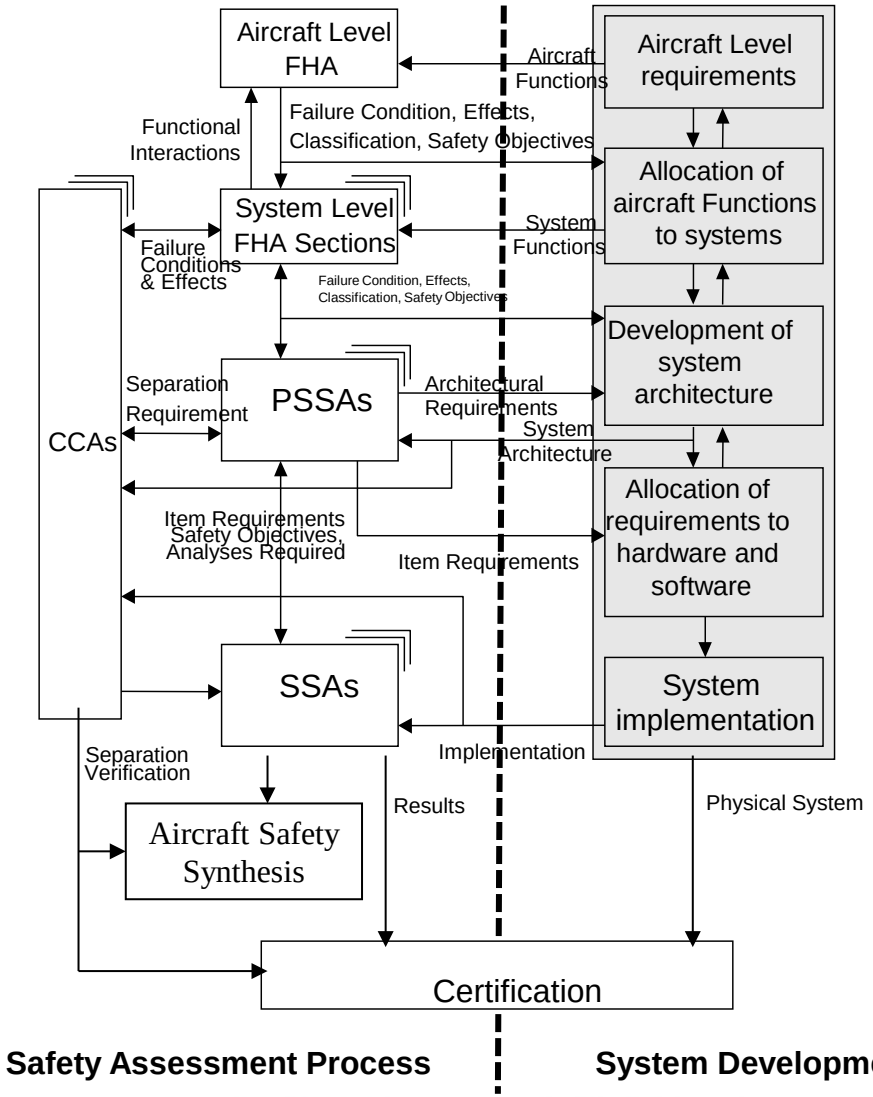
System safety analysis
and limits of current approaches

Hydraulic System



- **Safety architecture**: 3 independent lines
- About 20 components of 8 classes: reservoir, pumps, pipes, valves

ARP 4754 Safety Assessment Process



FHA: Functional Hazard Analysis

PSSA: Preliminary System Safety Assessment

CCA: Common Cause Analysis

Classical failure propagation models and safety assessment techniques (cf ARP 4761)

- Failure mode and effect analysis (FMEA)
  - Model: from a local failure to its system effects / natural languages

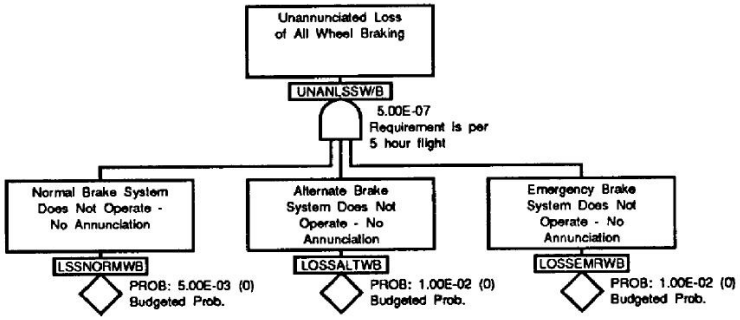
FAILURE MODES AND EFFECTS ANALYSIS (FMEA)

System:	FMEA Description:	Date:
Subsystem:		Sheet of
Item ATA:	FTA References:	File:
	Author:	Rev:

FUNCTION NAMES	FUNCTION CODE	FAILURE MODE	MODE FAILURE RATE	FLIGHT PHASE	FAILURE EFFECT	DETECTION METHOD	COMMENTS

Functional FMEA template

- Fault tree analysis (FTA)
  - Model: from a system failure to its root causes / boolean formulae
  - Computation: minimal cut sets / probability of occurrence of top event



FT unannunciated loss of wheel braking

Drawbacks of the classical Safety Assessment Approaches

- Fault Tree, FMEA
  - Give failure propagation paths without referring explicitly to a commonly agreed system architecture / nominal behavior =>
    - Misunderstanding between safety analysts and designers
    - Potential discrepancies between working hypothesis

- Manual exhaustive consideration of all failure propagations become more and more difficult, due to:
  - increased interconnection between systems,
  - integration of multiple functions in a same equipment
  - dynamic system reconfiguration

Model based safety assessment rationales

- Goals
  - Propose formal failure propagation models closer to design models
  - Develop tools to
    - Assist model construction
    - Analyze automatically complex models
  - For various purposes
    - FTA, FMEA, Common Cause Analysis, Human Error Analysis, …
    - since the earlier phases of the system development

- Approaches

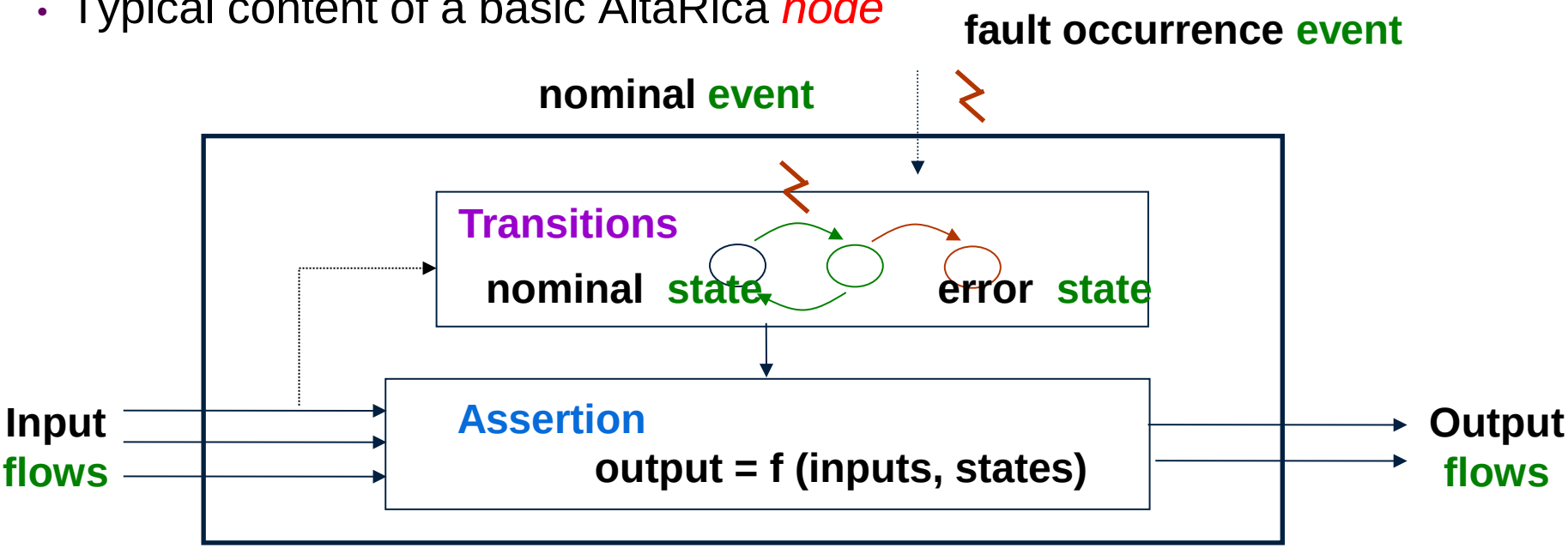
Extend design models (Simulink, SysML, AADL...) with failure modes	Build dedicated failure propagation models (Figaro, AltaRica, Slim...)

Basics of AltaRica dataflow language

AltaRica language at a glance

- Language designed in late 90's at University of Bordeaux
  - for modelling both *combinatorial* and *dynamic* aspects of *failure propagation*
  - in a *hierarchical* and *modular* way
  - *formally*.

- Typical content of a basic AltaRica *node*

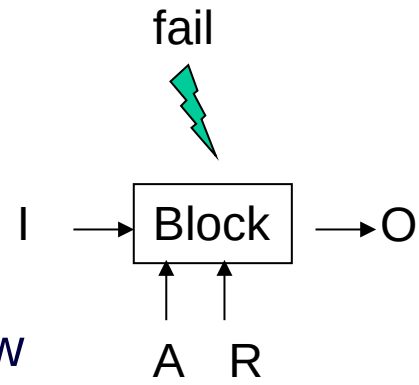


A leading example: the basic reliability block

- Let be a basic system component *Block* that
  - receives
    - one Boolean input I,
    - an activation signal A and
    - a resource signal R.
  - produces
    - a Boolean output O

Component name

Component interfaces



2 internal states:
- 1 ok
- 1 not ok

- Block performs *nominally* the following transfer law
  - O is true iff I, A and R are true.

Nominal mode definition

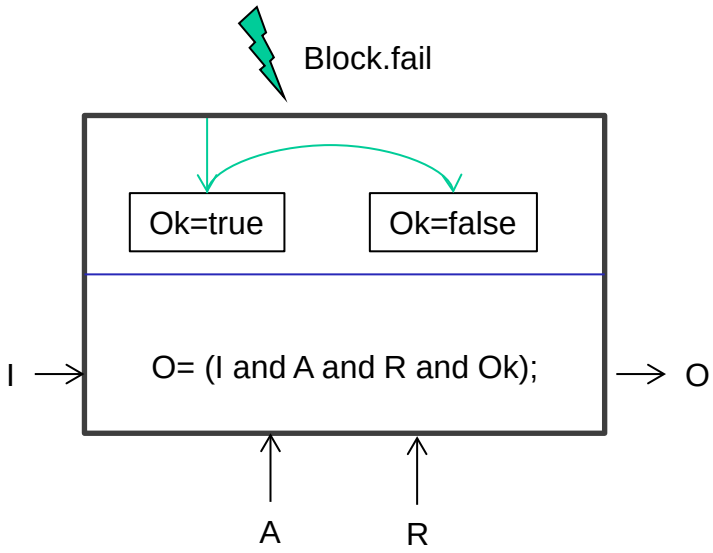
- Block may *fail*.
  - In this case, the output O is false.

Failure mode definition

- Initially, the block performs the nominal function

AltaRica basic block

From concepts to a concrete syntax:




```
node Block
  flow
    O:Bool:out;
    I, A, R :Bool: in;
 state
    ok: Bool;
 event
    fail;
 init
    ok:= true;
 trans
    ok |- fail -> ok := false;
  assert
    O = (I and A and R and ok);
edon
```

Component interfaces

Component internal states

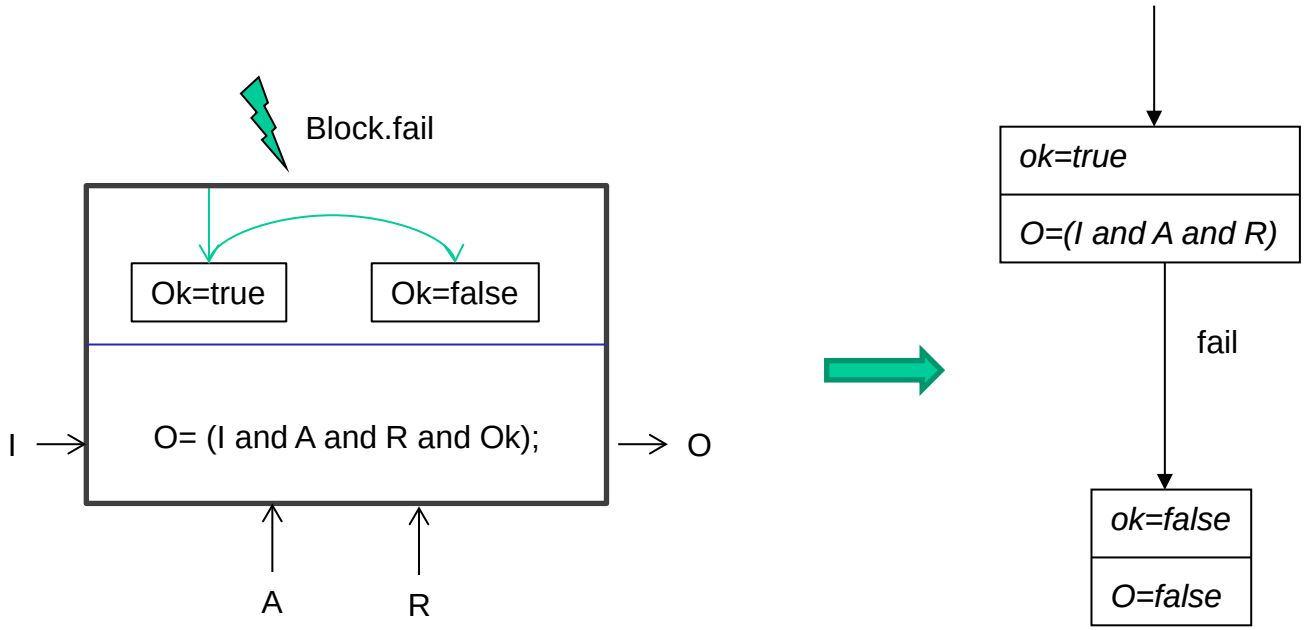
Observable event

Transitions:
Automata part

Assertion:
Combinatorial part

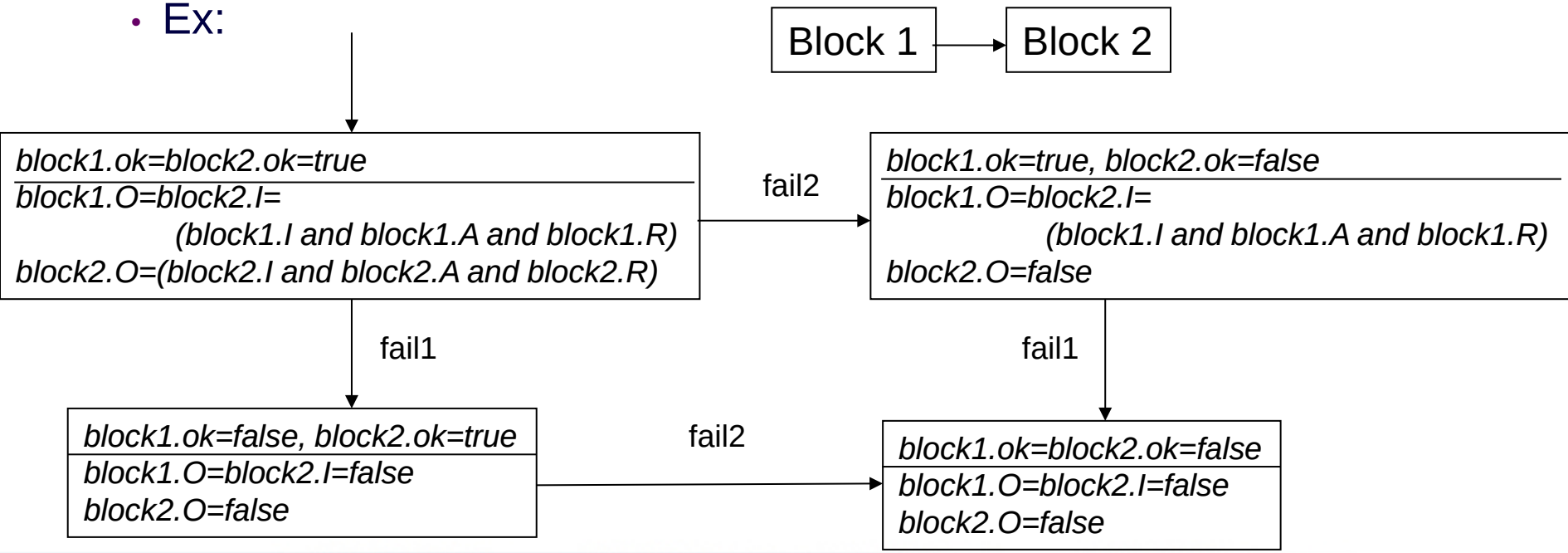
AltaRica semantics

From AltaRica code to mode automata

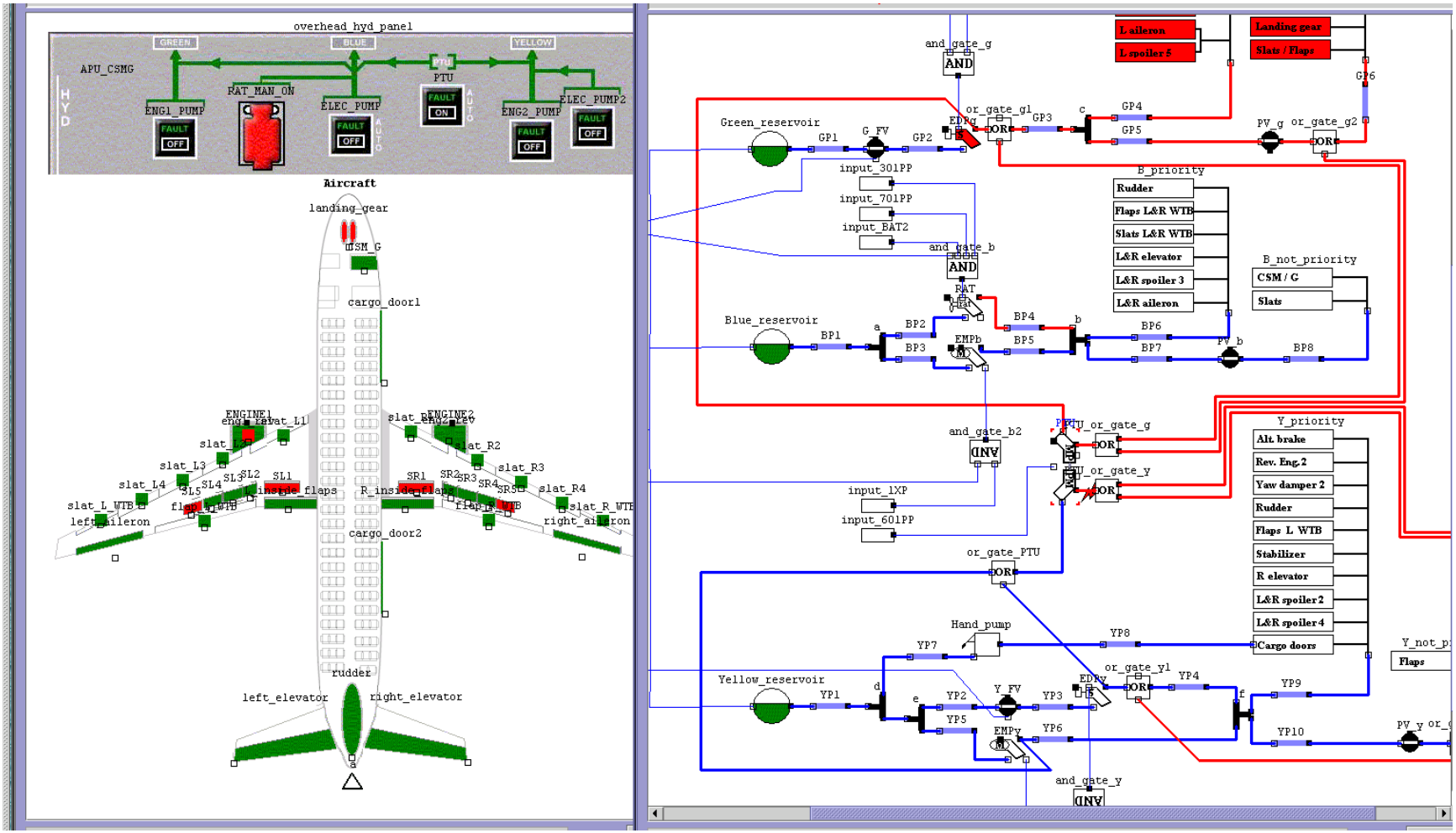


Internal operations on mode automata

- Interconnection : mapping an input of an automaton with an output of another automaton
  - preserves all states, variables, transitions, assertions
  - Introduces new assertions: Block2.I = Block1.O for all pairs of connected interfaces
  - interleaving parallelism (only one transition at a time)
  - ! allowed only if variables are not circularly defined
- Ex:



AltaRica Model of the Hydraulic System



Safety assessment tools

Formal Requirement Modeling

Example of safety requirement

- **Requirement :** "*Total loss of hydraulic power is classified Catastrophic, the probability rate of this failure condition shall be less than 10^{-9} /FH. No single event shall lead to this failure condition* " (SSA ATA29)

- **Extended qualitative requirements could be added to reveal architecture design concerns:**

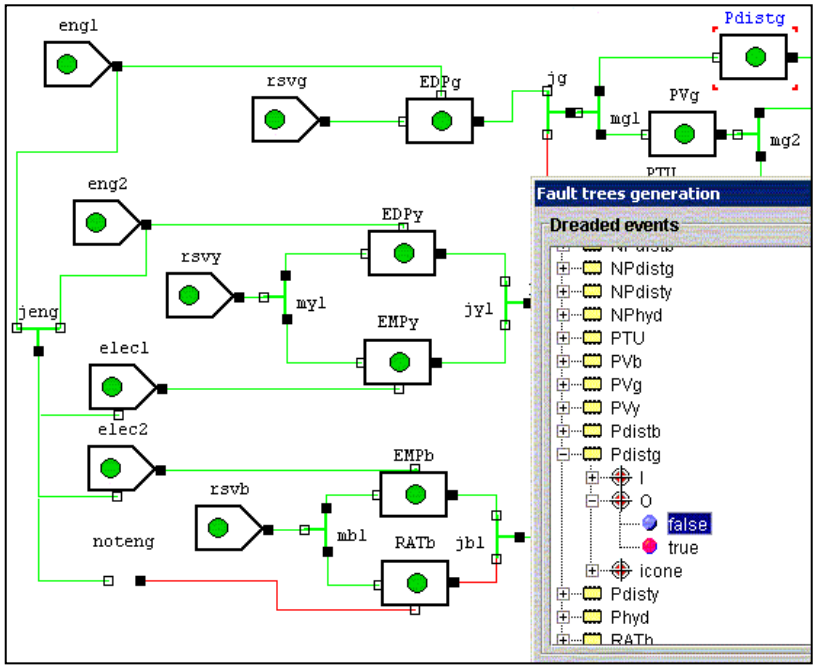
  "if up to N individual failures occur then failure condition FC should not occur",

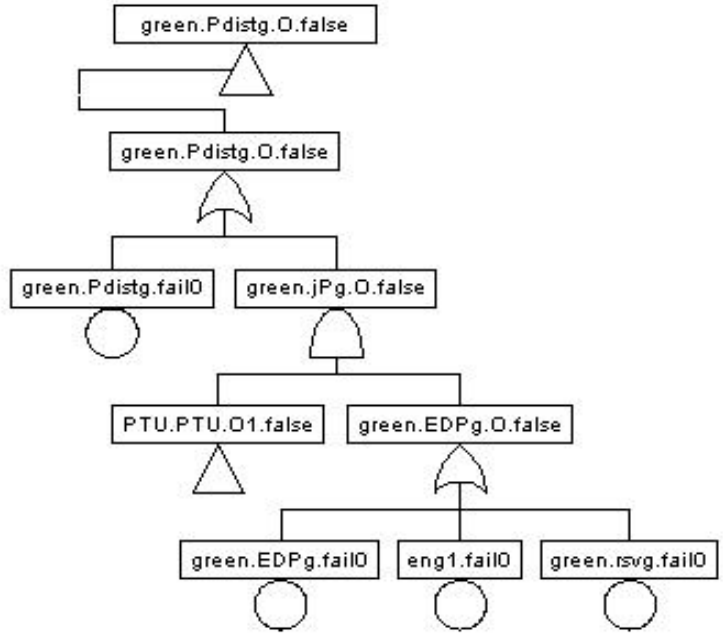
  with N= 0, 1, 2 if FC is Minor, Major or Hazardous, Catastrophic.

Observer nodes are added into the model to detect requirement violation

Fault-Tree generation

- A pair (output variable, target value) is selected
- A Fault Tree of faults leading to this situation is generated
- The fault tree can be exported to other tools (e.g. Arbor,...) to compute of minimal cut sets

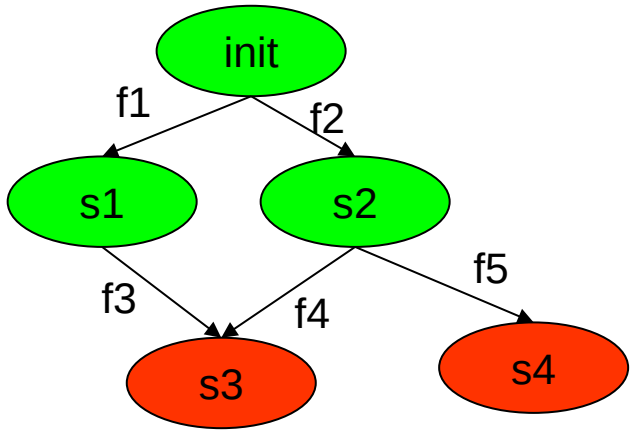


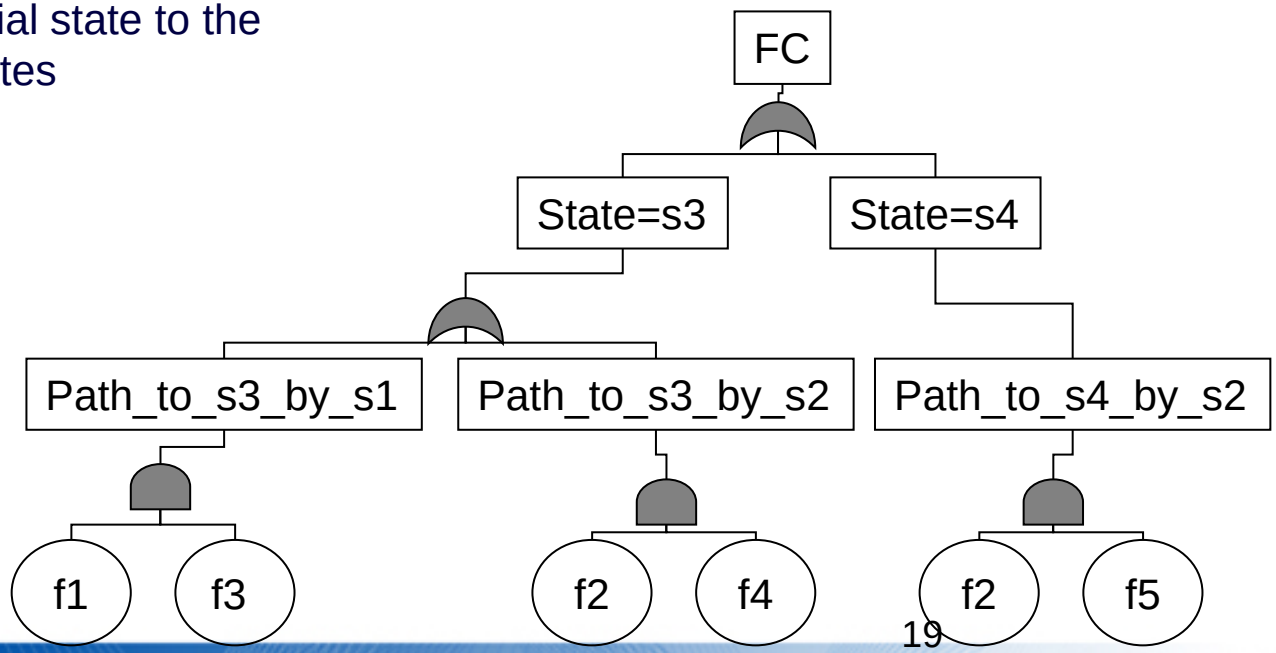


Principles of Fault-Tree computation

- To compute a fault-tree for a Failure Condition (FC) from an AltaRica Model:

1. Generate the model automaton
2. Select states where the FC holds
3. Compute event paths that leads from the initial state to the selected states



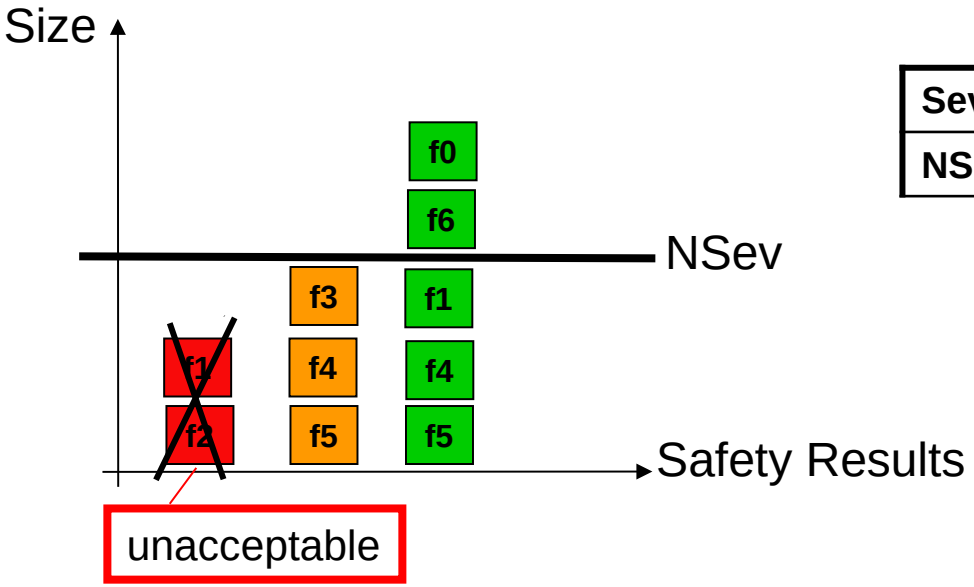


ONERA

Verification of Qualitative Requirements

- Generate Minimal cut sets from the Fault Tree
  - Loss of Green Hydraulic : {{distg.fail}, {rsvg.fail}, {empg.fail, edpg.fail}, {empg.fail, eng1.fail}, {elec.fail, edpg.fail}, {elec.fail, eng1.fail}}

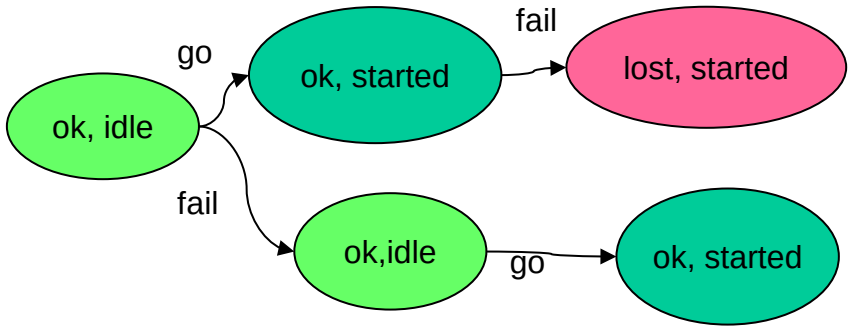
- The size of minimal cut sets for a FC in *Sev* should be greater or equal to *NSev*.



Sev	MIN	MAJ	HAZ	CAT
NSev	1	2	2	3

! Classes of model

- Static/Dynamic Model
  - **Static** Model: the order of the events in the sequence as no influence on the current configuration
  - **Dynamic** Model : the last property is not verified => use sequence generation rather than fault tree generation



DAL

Development Assurance Level

- DAL
  - DAL ranges from E to A
  - The DAL is the level of rigor of development assurance tasks performed on functions and items (software, hardware)

- DAL allocation
  - DAL of a function depends on the severity of the most severe Failure Condition that this function fault contributes to.

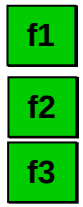
  - A Qualitative analysis of the Minimal Cut Sets of the system has to be performed

DAL Allocation

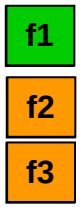
- Basic Allocation rule
  - If f1 appears in a MCS for of FC with severity HAZ then the DAL of f1 is B

Sev	DAL
CAT	A
HAZ	B
MAJ	C
MIN	D

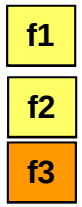
- DAL downgrading rules
  - If f1 appears in a MCS in combination with f2 and f3 then the DAL of f1 could be downgraded if there is independence between f1, f2 and f3.



No independence



Independence + Option 1



Independence + Option 2

Sev(FC)=HAZ
DAL(FC) =B

AltaRica Tools available

- Cecilia OCAS from Dassault Aviation
  - Used for the first time for certification of flight control system of Falcon 7X in 2004
  - Tested by contributors of ARP 4761 (cf MBSA appendix)
- AltaRica free suite from Labri
  - compatible with data flow restriction, http://altarica.labri.fr/wp/
- Other tools
  - Safety Designer from Dassault System, Simfia from APSYS Airbus group, RAMSES from Airbus, AltaRica 3.0 (under development at IRT Systemix)
- And plugins to independent tools
  - NU-SMV (FBK Trento), MOCA-RP (Satodev Bordeaux), Arc (LaBri Bordeaux), EPOCH (ONERA)….
- DAL allocation
  - DALculator (ONERA)