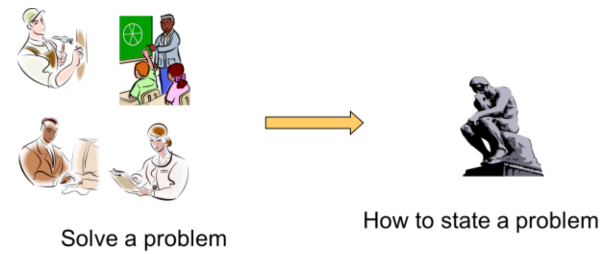
Basic Introduction to Systems Engineering

Master of science in Aerospace
ISAE SUPAERO

Jean-Charles Chaudemar
jean-charles.chaudemar@isae-supaero.fr

Institut Supérieur de l'Aéronautique et de l'Espace
SUPAERO

Objectives and competencies



Goal

To know a few methods to model and design a complex system

Competencies

- To be able to analyze a complex problem
- To be capable of stating or specifying formally right solutions

Outline

- Formal method concepts and rationale
- Event-B introduction
- Abstract model
- Refined model
- Simulation and theorem proving

Failure “is expensive”

- ARIANE 501 Failure: see full report on *http://sunnyday.mit.edu/nasa-class/Ariane5-report.html*
- Reminder of accident scenario
  - H0-3s: start of Flight Mode of Inertial Reference System (SRI)
  - H0: end of count-down, all conditions nominal, sparking engine ignition
  - H0+7s: lift-off
  - H0+36.7s: back-up SRI failed, then active SRI too, because of same fault, overflow on one variable for horizontal velocity, Horizontal Bias (BH) indicator, in a not needed function of alignment of SRI platform
  - H0+39s: curt change of attitude due to erroneous flight data to flight controller
  - H0+40: disintegration, then self-destruction automatically

What went wrong?

What would you propose to solve it?

Context

- Possible answers

Design based on models correct by construction

The actual causes of the failure are design errors in capturing the application needs and environmental assumptions relating to Ariane 5, along with design and sizing errors in the on-board computer system of SRI. These faults stem from a lack of rigorous method in Systems Engineering e.g., the absence of V&V methods based on proof obligations.

INRIA research report, member of Inquiry Board

Req.: One single fault shall not lead to a SRI failure

Invariant property:

$$\cancel{BH_{value} \in [x_{min}; x_{max}] \rightarrow SRI_{status} = nominal}$$

or

Invariant property:

$$SRI_{status} = nominal \rightarrow BH_{value} \in [x_{min}; x_{max}]$$

Notions in systems engineering

Is this a system?

Water | Toy | Car

- Main features of these objects
  - **Water:** monolithic
  - **Toy**: faithful copy of real car including many elements, but no interaction between elements except permanent contact
  - **Car**: complex mission, lots of constraints (socio, technical, economical, environmental,…)

Definitions

- “A system is a **set of elements** in **dynamic interaction**, organized **to achieve a goal**” (J. de Rosnay – *Le Macroscope*)
  - **Structural** aspects
  - **Behavioral** aspects
  - **Goal** aspects
- “An integrated set of elements that accomplished a defined objective” (INCOSE)
- “A combination of interacting elements organized to achieve one or more purposes” (ISO 15288)
- **3P** definition: “An integrated composite of **people**, **products**, and **processes** that provide a capability to satisfy a stated need or objective” (MIL-STD-499B)

Main characteristics

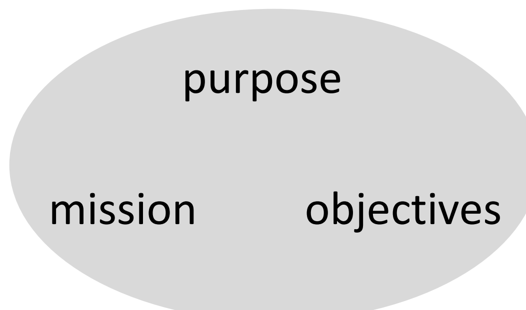
5 basic features identifying a system

- **Wholeness**
  - each element has a function which can’t be considered separately from the others
- **Structure**
  - elements are linked each other physically or logically, also they shape a topology
- **Complexity**
  - complex = interactions, reliance vs. complicated = difficult, tricky
- **Evolution**
  - is born (its conceptual birth), operates (its behavior), eventually dies (its disposal)
- **Emergence**
  - properties or behavior due to interactions, e.g. reliability (positive emergence) or EMI (negative emergence)

Definition of a system to engineer

In early conceptual phase, since a system to engineer does not exist yet, its first definition stems from need, or expectation, or requirement viewpoint

- **Purpose or final goal**
  - **Why** does the system exist? What is its raison d’être within the context?
- **Mission**
  - **What** does it do? What does it transform? What kind of services does it provide?
- **Objectives**
  - **How** many inputs are transformed into outputs? How often? How well… ? What are the measures of effectiveness?



Why?

- It corresponds to tacit needs considering socio economical technical environment
- Typical sentence : “*the system_s enables goal_x*”
- Examples of *goal_x* : reduction of consumption, reduction of pollution, improvement of safety, employment, development of a new market

purpose

mission objectives

What?

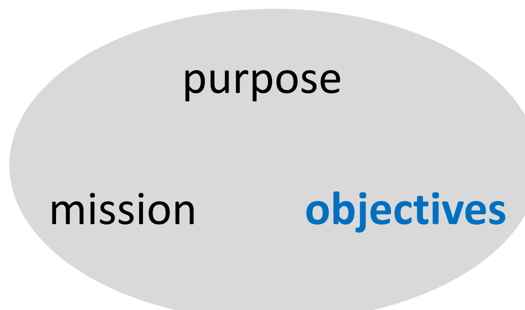
- It provides users with high level functions or services
- Typical sentence : “*the system_s does function_x in order to meet goal_x*”
- Examples of *function_x* : to fly, to move from A to B, to control combustion, to connect C with D

purpose

mission objectives

How?

- It corresponds to constraints on time, delay, performances
- Typical sentence : "*the system_s does function_x in order to meet goal_x in objective_x way*"
- Examples of *objective_x* : autonomously, maximum speed, cost, capacity

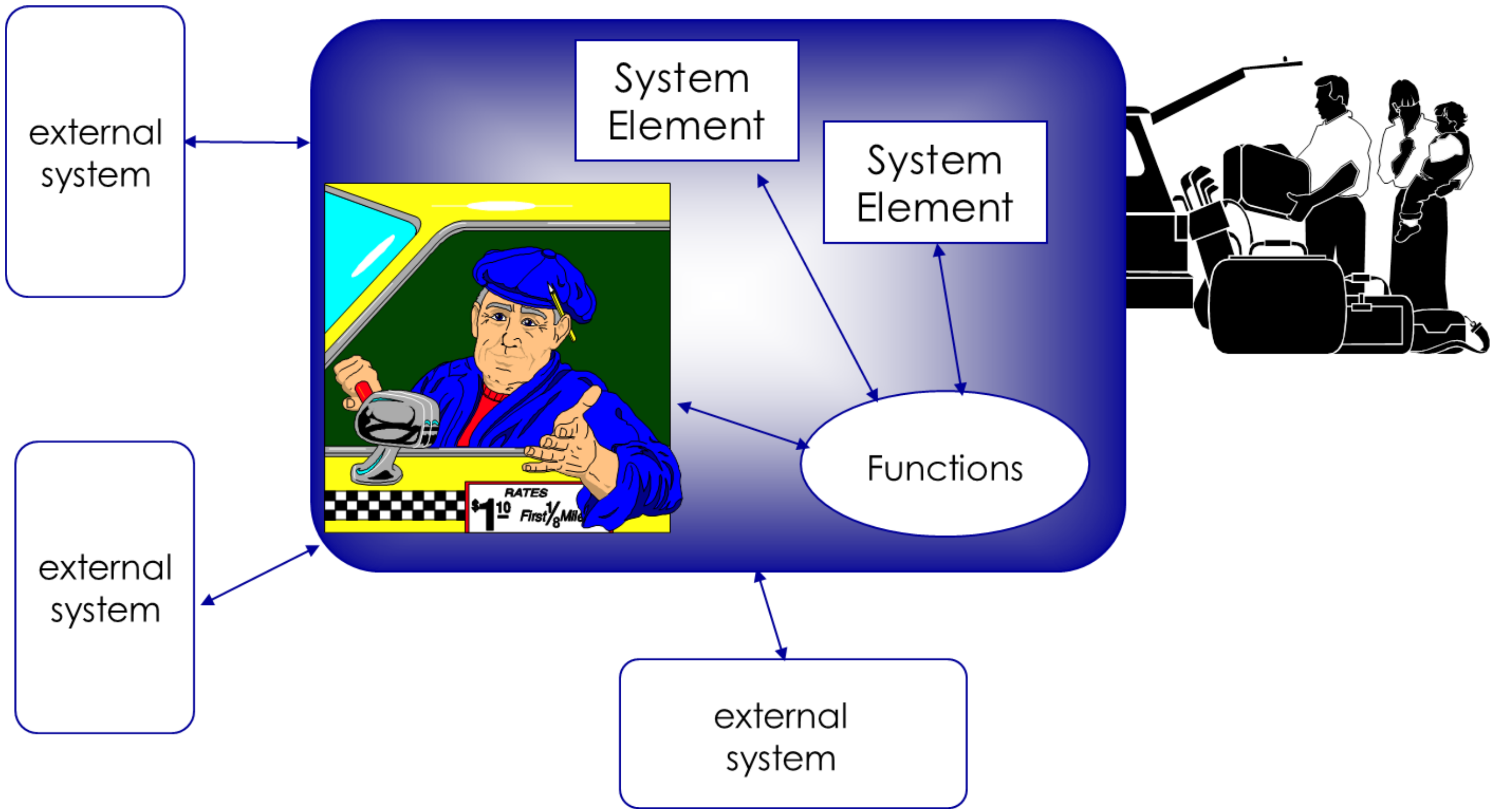


Example #1

system	purpose	mission	objectives
A car manufacturer	?	?	?
One production line	?	?	?
	why ?	what ?	how much ?

Example #2

Let consider a transportation system with a driver



Example #2

Let consider a transportation system with a driver

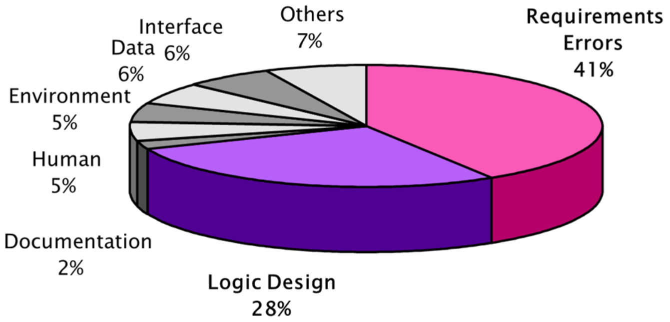
Purpose : to provide people transportation for various destinations

Mission : to transport people, goods from one point to another

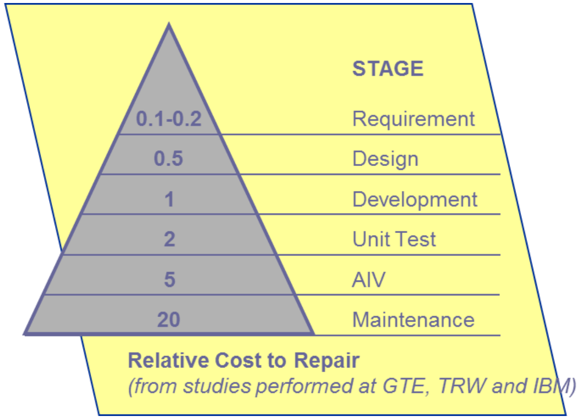
Objectives : to transport 1 to 6 people at the same time

Rationale for system modeling

- System is an abstract notion with no physical existence
- Shared representation in order to increase its understanding, communication
- Costly to develop products at early design phase, so reduction of cost and reduction of time to market
- Reusability of models
- More and more relevant
- Design errors have costly fallout when undetected, but many are easily detected through models



Source of Errors, US Air Force project
(Sheldon, F et al., Reliability Measurement from Theory to Practice, IEEE software, july 92)



Definition of modeling

Model

An object A is a model of an object B if an observer can use A to answer questions that interest him about B.

Minsky,1985

Model

Abstraction of the reality which represents some aspects of the actual process considered as important by the "modeler".

Marquardt, 1994

Modeling

Intended action to work out and build models (using sets of symbols).

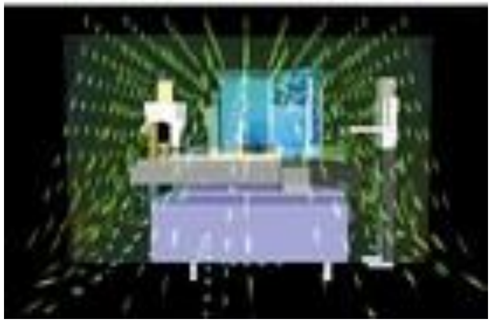
These models might be capable to make understandable a system perceived as complex. They allow reasoning in order to anticipate consequences of potential system's actions.

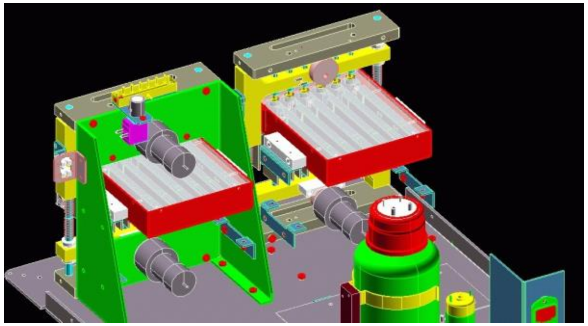
from J.L LeMoigne

Definition of modeling

For analyzing complex systems, we use modeling techniques with formal and graphical representations.

- A model is a representation of reality (physical phenomenon, process…) with symbols organized according to conventions
- A "logical" model is independent from any implementation
- A model is a restricting abstraction of a real system (e.g. : a map, a mock-up, mathematical laws ...)





Restricting the reality is not a model defect; this allows

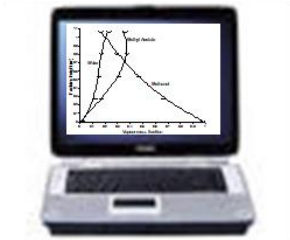
- to remove useless details for comprehension
- to focus on main points

Modeling and Simulation

Actual system

EXPERIMENTS (REAL TESTS)

Simulated system



MODELLING

$$L^{I}_{n} \cdot (U^{I}_{n} + U^{II}_{n}) - U^{I}_{n} \cdot (L^{I}_{n} + L^{II}_{n}) = 0$$

$$\sum_{i}^{nc} x^{II}_{n,i} + \tau^{II}_{n} - \sum_{i}^{nc} y_{n,i} - \tau^{V}_{n} = 0$$

$$\sum_{i}^{nc} x^{I}_{n,i} + \tau^{I}_{n} - \sum_{i}^{nc} y_{n,i} - \tau^{V}_{n} = 0$$

$$U^{II}_{n} \cdot \frac{d\left(\tau^{II}_{n}\right)}{dt} + \tau^{II}_{n} \cdot \frac{d\left(U^{II}_{n}\right)}{dt} = 0$$

$$U^{I}_{n} \cdot \frac{d\left(\tau^{I}_{n}\right)}{dt} + \tau^{I}_{n} \cdot \frac{d\left(U^{I}_{n}\right)}{dt} = 0$$

$$U^{I}_{n} + U^{II}_{n} - \mathrm{mod}(U_{n}) = 0$$

Mathematical model

SIMULATION (VIRTUAL TESTS)

Process which establishes a formalized structure used to explain a set of phenomena which have relationships between them.

Process which conducts experiments using the model in order, either to understand the behavior of the system, or assess various operating strategies of the system.

Formal specification

Example of requirements

- System Requirements
  - 1 Introduction
  - 2 Functional Requirements
    - 2.1 Power car
    - 2.2 Control car
    - 2.3 Illuminate car
    - 2.4 Control windows
    - 2.5 Control sun roof
    - 2.6 Maintain visibility
    - 2.7 Stabilize occupants
    - 2.8 Protect passengers
    - 2.9 Protect environmental
    - 2.10 Modularity
    - 2.11 Control entertainment
    - 2.12 Communicate
    - 2.13 Calculate
    - 2.14 Accommodate
  - 3 System constraints
    - 3.1 Reliability
    - 3.2 Modularity
    - 3.3 Failure modes
    - 3.4 Fuel efficiency
    - 3.5 Fuel input mechanism
    - 3.6 Braking
    - 3.7 Steer car

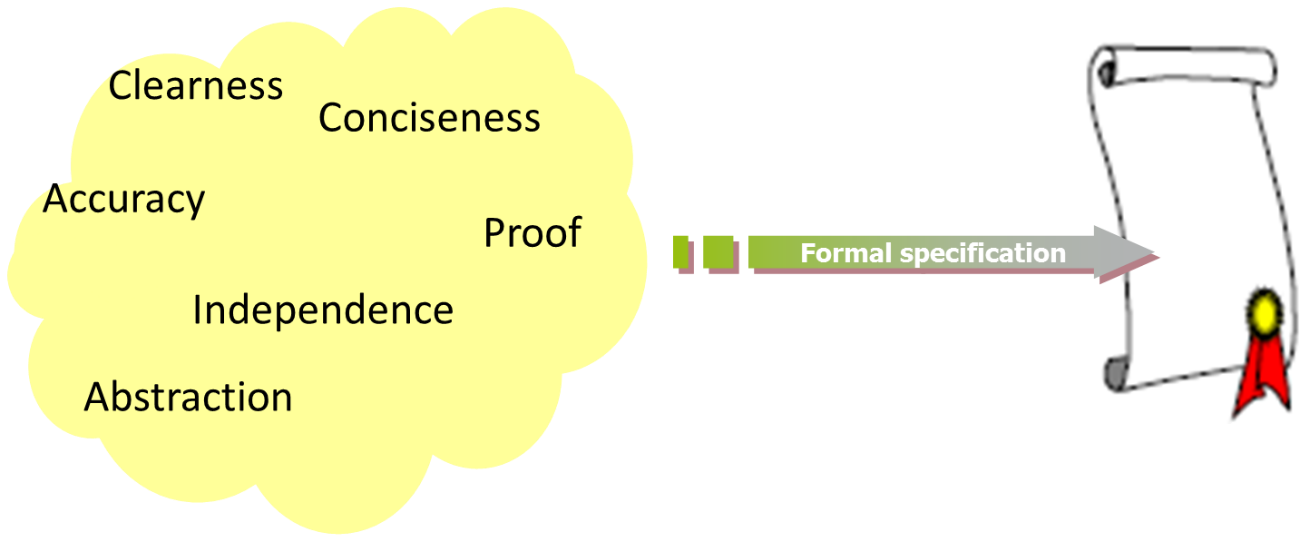
Object Identifier	System requirements for passenger car	Incoming Links
SR-1	**2 Functional Requirements**	
SR-2	**2.1 Power car**	
SR-3	**2.1.1 Move car**	
SR-4	**2.1.1.1 Move forwards**	
SR-5	The car shall be able to move forwards at all speeds from 0 to 200 kilometers per hour on standard flat roads with winds of 0 kilometers per hour, with 180 BHP.	User Requirements: UR25 3.1.3.1.1.0-1 User Requirements: UR27 3.1.3.1.2.0-1
SR-6	**2.1.1.2 Move backwards**	
SR-7	The car shall be able to move backwards to a maximum speed of 20 Kilometers per hour on standard flat roads with winds of 0 kilometers per hour, with 180 BHP.	User Requirements: UR25 3.1.3.1.1.0-1
SR-8	**2.1.2 Accelerate car**	
SR-9	The car shall be able to accelerate from 0 to 100 Kilometers per hour in 10 seconds on standard flat roads with winds of 0 kilometers per hour.	User Requirements: UR28 3.1.3.1.2.0-2
SR-10	The car shall be able to accelerate from 100 to 150 kilometers per hour at a rate of 5 kilometers per second on standard flat roads with winds of 0 kilometers per hour.	
SR-11	The car shall be able to accelerate from 150 to 200 kilometers per hour at a rate of 3 kilometers per second on standard flat roads with winds of 0 kilometers per hour.	
SR-12	**2.2 Control car**	

Requirement or specification

- Specification is a domain of problem comprehension, it describes a set of implementations, of potential solutions, but it doesn't give the solution
- Requirement is a contractual description between a customer and a supplier for the sake of design and development
- Controlled natural language (CNL) for requirement:
  - "set of linguistic rules constraining the lexicon, the syntax and the semantics" (M. Warnier, 2018)
  - "shall" modal verb in order to state an obligation
  - Short sentence

Formal specification

- Formal specification is the expression in formal language of requirements with respect to a set of properties that the system must meet
- Formal language relies on syntax (mathematical notations) and semantics (mathematical handling technics for meaning)
- Formal methods represent a rational framework consisting of modelling tools and reasoning technics



Abstraction level

- Mastering the complexity by dividing the problem into sub problems while focusing on specific aspects or viewpoints
- Building an abstraction model is an observation of the system along with intrinsic (natural) laws
- Refinement enables to add more details progressively

Precaution for designer

- Modelling supposes some pitfalls to be overcome such as
  - To deal with complicated notations to be handled properly
  - To find out new issues revealed by the models

- In the specification practice, 3 notions are crucial
  - Level of granularity for abstraction
  - Interesting properties
  - Type of the formal language

Event-B method

Basic concepts

Principles of formal language

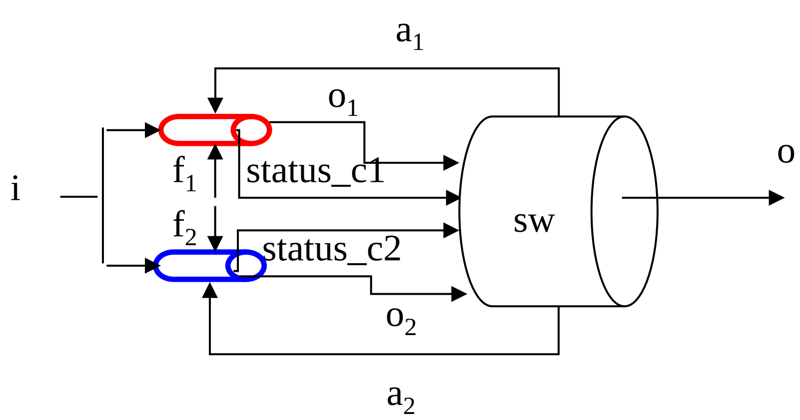
Event-B method at a glance

- B language designed in mid-80's by Jean-Raymond Abrial for development of reliable software as a safety engineer on behalf of RATP
  - Specifications and programs modeled through notation called *abstract machine*
  - Properties stated through logic theory and set theory
  - Refinement principle until concrete implementation model which is able to generate C or ADA code
- Event-B
  - extension of B method for system specification, design, and coding of the associated software
  - Formal modelling of complex systems with preponderant software
- Supported tools
  - B-tool developed in England by B-core company
  - Atelier B developed in France by ClearSy company
  - Rodin, which is an open source platform (http://www.event-b.org/)

Case study: FDIR mechanism

- FDIR objective: to maintain the availability and the safety of a system
  - Identify fault-classes
  - Set up a health monitoring to diagnose abnormal state
  - Recover to tolerate fault-classes

- Implementation of FDIR strategy to tolerate one single fault: redundant architecture



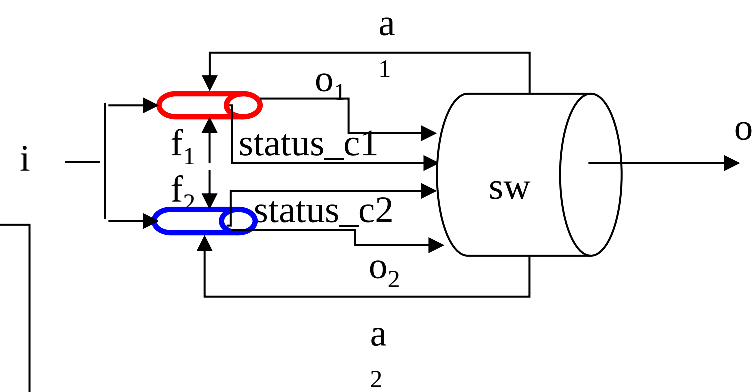
Safety objectives:

• qualitative: “one single fault shall not lead to the total loss of the function”

• quantitative: the probability of failure occurrence is less than 10^{-x} per flight hour

Illustration in Event-B

- Idea: modelling of fault tolerant architecture based on redundancy concept



COSSAP_0 :

• safety requirement definition: “one single fault shall not lead to the total loss of the function”

• events : disconnection of primary component (called 1) and secondary component (called 2) if a fault occurred

refines

COSSAP_1 :

• take into account activation and switch process

refines

COSSAP_2 :

• take into account impact of failure on output data

Event-B Model description

- Event-B model = MACHINE + CONTEXT
- COSSAP_0 Machine consists of 5 sub-sections
  - Name of machine: machine deals with behavioral part of a model
  - SEES: stood for link to CONTEXT part of a model, where are defined all static parameters (model domain) and axioms
    - Defined domains of values like SETS (abstract set or carrier set, enumerated set)
    - Other sets are predefined (Boolean, Integer, Natural number)
    - CONSTANTS: name of symbolic constants
    - AXIOMS: properties or logic statements/expressions on model domain

CONTEXT *Name*

SETS *list of sets*
CONSTANTS *list of constants*
AXIOMS *axioms*

MACHINE *Name*

SEES *Context*
VARIABLES *list of variables*
INVARIANTS *Invariants*
EVENTS *Events*

Event-B Machine description

- ❑ Event-B model = MACHINE + CONTEXT
- ❑ COSSAP_0 Machine consists of 5 sub-sections
  - Name of machine: machine deals with behavioral part of a model
  - SEES: stood for link to CONTEXT part of a model, where are defined all static parameters (model domain) and axioms
  - VARIABLES: name of new state variables
    - state variables used to model the state of the system,
    - i.e., system state is value of all variables at a given time

CONTEXT	*Name*
SETS	*list of sets*
CONSTANTS	*list of constants*
AXIOMS	*axioms*

MACHINE	*Name*
SEES	*Context*
VARIABLES	*list of variables*
INVARIANTS	*Invariants*
EVENTS	*Events*

Event-B Machine description

- Event-B model = MACHINE + CONTEXT
- COSSAP_0 Machine consists of 5 sub-sections
  - Name of machine: machine deals with behavioral part of a model
  - SEES: stood for link to CONTEXT part of a model, where are defined all static parameters (model domain) and axioms
  - VARIABLES: name of new state variables
  - INVARIANTS: all properties to be hold anytime, before and after event occurring, expressed in
    - Set theory
    - First-order predicate

CONTEXT	*Name*
SETS	*list of sets*
CONSTANTS	*list of constants*
AXIOMS	*axioms*

MACHINE	*Name*
SEES	*Context*
VARIABLES	*list of variables*
INVARIANTS	*Invariants*
EVENTS	*Events*

Event-B Machine description

- Event-B model = MACHINE + CONTEXT
- COSSAP_0 Machine consists of 5 sub-sections
  - Name of machine: machine deals with behavioral part of a model
  - SEES
  - VARIABLES
  - INVARIANTS
  - EVENTS: operations that could occur
    - atomic action i.e. null execution time
    - when firing event, actions are carried out simultaneously
    - action updates value of 0..* state variables

CONTEXT	*Name*
SETS	*list of sets*
CONSTANTS	*list of constants*
AXIOMS	*axioms*

MACHINE	*Name*
SEES	*Context*
VARIABLES	*list of variables*
INVARIANTS	*Invariants*
EVENTS	*Events*

Event description

- Event
  - Statement of relevant properties to be met before and after modification of state variable(s)
  - Structure = parameter + guard + action
  - Parametric event: optional parameter; local or internal variable used in guard
  - Guard: pre condition to meet before firing event; definition of parameter type; expression using set theory and logic; conjunctive form i.e all guards related to each other with AND operator
  - Action: instruction updating value of state variable; expression using set theory and instruction formula
  - At least one action for defining an event like INITIALISATION

EVENTS

Reserve $\hat{=}$

ANY

nb

WHERE

nb ϵ $\mathbb{N}$

seat > nb

THEN

seat := seat - nb

END

EVENTS

INITIALISATION $\hat{=}$

THEN

seat := max

END

MACHINE CoSSAP_0

SEES CoSSAP_C0

VARIABLES

$status$
$status\_c1$
$status\_c2$
$fault\_c1$
$fault\_c2$

Health monitoring

INVARIANTS

$inv1 : status \in BOOL$
$inv2 : status\_c1 \in BOOL$
$inv3 : status\_c2 \in BOOL$
$inv4 : fault\_c1 \subseteq DEFAULT$
$inv5 : finite(fault\_c1)$
$inv6 : fault\_c2 \subseteq DEFAULT$
$inv7 : finite(fault\_c2)$
$inv9 : fault\_c1 \cap fault\_c2 = \varnothing$
$inv8 : status\_c1 = TRUE \Leftrightarrow fault\_c1 = \varnothing$
$inv10 : status\_c2 = TRUE \Leftrightarrow fault\_c2 = \varnothing$
$inv11 : status = TRUE \Leftrightarrow status\_c1 = TRUE \vee status\_c2 = TRUE$

Fault-classes

Safety property to be met

EVENTS

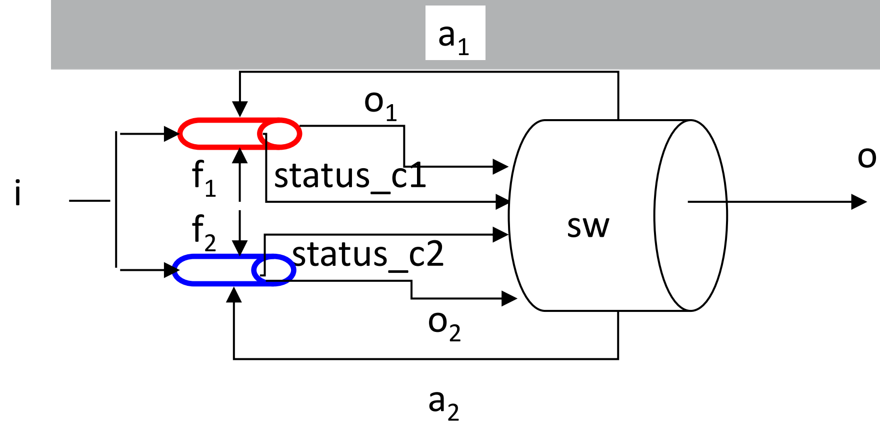
INITIALISATION

....

EVENT disc2

.....

END



Proof obligations discharged automatically

EVENT disc1
  ANY
    $f1$
  WHERE
    $grd1 : f1 \in DEFAULT$
    $grd2 : f1 \notin fault\_c1$
    $grd3 : f1 \notin fault\_c2$
  THEN
    $act1 : status\_c1 := FALSE$
    $act2 : status := status\_c2$
    $act3 : fault\_c1 := fault\_c1 \cup \{f1\}$
  END

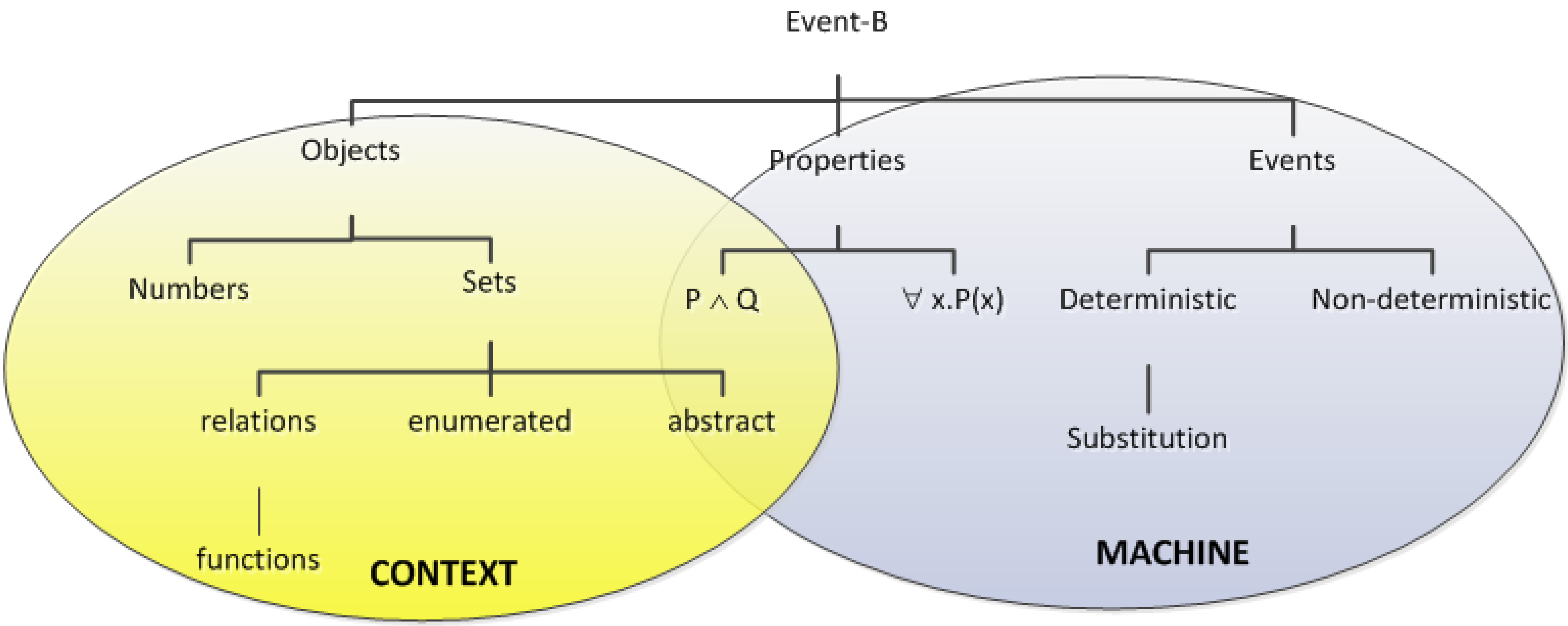
Effect of a fault

Formal definitions

Event-B

- Model defined by n-tuple M = (c, P, v, I, K, E)
  - c set of constant objects
  - $P(c)$ collection of axioms
  - v set of state variables
  - $I(c, v)$ collection of invariant properties
  - K initialization event
  - E set of events including guards $G(c, v)$ and action $A(c, v, v')$

Definition


Event-B
Objects
Properties
Events
Numbers
Sets
P ∧ Q
∀ x.P(x)
Deterministic
Non-deterministic
relations
enumerated
abstract
Substitution
functions
CONTEXT
MACHINE

Logic basics

- Logic is a language bearing a deduction mechanism. It provide with a syntax and semantics so as to state a property or expression or proposition or formula
- The smallest proposition is called atomic proposition
  - Constant c is a proposition (e.g. *max, FALSE, TRUE*)
  - Variable v is a proposition (e.g. *status, fault_c1*)
  - *Finite(fault_c2)* is a proposition
- The semantics or interpretation or valuation enables to give to a formula a meaning, a truth value (*{TRUE, FALSE}*)
- Logical operators or connectives
  - $\neg\ (NOT),\ \wedge\ (AND), \vee\ (OR)$
  - $\Longrightarrow\ (IMPLY), \Longleftrightarrow\ (EQUIVALENT)$

Set theory basics

- SET is an abstract collection of objects
- Set theory symbols
  - $\in$, $\subseteq$, $\cap$, $\cup$
  - A\B or A-B : relative complement (elements belonging to A and not to B)
  - AxB : cartesian product (set of all ordered pairs from A and B)
  - $\rightarrow$, $\rightarrowtail$, $\twoheadrightarrow$: respectively total function, total injection, total surjection

Example of formal statements

We would like to express formally relationships between members of the same family (viz. relatives, children, bothers, wife,…). We assume a given abstract set called PERSON, which encompasses all concerned people.

1. *Each person is either a man or a woman*
2. *Nobody is both a man and a woman*
3. *Only women have husbands who are men*
4. *Each woman has one husband at most, who is not the husband of none other woman*
5. *A mother is a married woman*

Example of formal statements

We look into set-based ways to model notions like: *man, woman, husband (or has_as_her_husband), mother (or has_as_mother)*

$$man \subseteq PERSON$$
$$woman = PERSON - man$$
$$husband \in woman \rightarrowtail man$$
$$mother \in PERSON \rightarrow dom(husband)$$

Example of Simpson's family

$$PERSON = \{Homer, Marge, Bart, Lisa, Maggie\}$$
$$man = \{Homer, Bart\}$$
$$woman = \{Marge, Lisa, Maggie\}$$
$$husband = \{Marge \mapsto Homer\}$$
$$mother = \{Bart \mapsto Marge, Lisa \mapsto Marge, Maggie \mapsto Marge\}$$

Example of location access controller

Design of a control system to allow people to access buildings

Functional requirements on authorization:

"Each concerned person is supposed to possess an authorization. This authorization allows him, under the control of the system, to go inside some buildings, and not into others. For example, a person p1 is authorized to enter building b1 and not building b2; however, another person p2 is allowed to enter both buildings. These authorizations are given on a "permanent" basis: in other words, they will not change during a normal functioning of the system."

The system relates to people and buildings	FUN-1

People are permanently assigned the authorization to access some buildings	FUN-2

A person which is in a building must be authorized to be there	FUN-3

First abstraction

We only consider an operational scenario where people move from one building to another

Observation of objects

- **Abstract sets: *PERSON*, *BUILDING***

Observation of function/operation/event

- ***Event* or observable operation, *pass*, corresponding to a *person* going from one given *building* to another different one.**

State, system behaviour

- ***Variable* describing the system behaviour: here, variable *sit*, denoting the *building* where each *person* is.**

Example of traffic controller on a bridge

Design of a traffic control system allowing cars to access an island

Operational and functional requirements:

"This controller controls the traffic of cars between a mainland and an island connected to each other by a narrow bridge. The traffic is two-way, i.e. cars can enter on the island or leave it, but not both at the same time.

Moreover the number of cars on the island and the bridge is limited."

The system is to control cars on a bridge connecting the mainland and an island	FUN-1
The system controls the entrance to the bridge at both ends of it	FUN-2
The number of cars on the bridge and island is limited	OPE-1
The bridge is one-way or the other, but not both at the same time	OPE-2

First abstraction

We only consider an operational scenario where cars go in and go out the island-bridge compound

Observation of objects

- Void… we are not interested in features of cars indeed

Observation of function/operation

- *ML_out* : corresponds to cars going out the mainland
- *ML_in*: corresponds to cars going in the mainland

State, system behaviour

- Natural number *n* representing number of cars on the island-bridge at a given time

Event-B method

Basic concepts

Principles of formal language

Formal language

Correctness for computer programming *[Hoare, 1969]*

- Correctness of sequential program by adding assertions for each instruction: « Hoare » logic
- Reasoning framework based on axioms and rules of inference to prove a transformation of properties by using Hoare triplet $\{pre\}P\{post\}$

Instruction P is correct if before its execution {pre} is true and after its execution {post} is true too. {pre}P{post} is said the most efficient and accurate if to satisfy {pre} there is the largest number of values for each variable that will satisfy P{post}, {pre} is then the weakest pre-condition, called ***wp(P,post)***

For instance, in the case of a substitution (or assignment) S ≡ $x{:=}E$, we get

$wp(A,q)=\{[x\backslash E]q\}$

NB: $[x\backslash E]$ or $[x{:=}E]$ relates a formula E to a variable x, i.e. "x becomes E". $[x\backslash E]q$ is the result of simultaneous substitution of all occurrence of x in property q

Formal language

Correctness for computer programming *[Hoare, 1969]*

- Correctness of sequential program by adding assertions for each instruction: « Hoare » logic
- Reasoning framework based on axioms and rules of inference to prove a transformation of properties by using Hoare triplet $\{pre\}P\{post\}$

Example

$$\{n+1<5\}n \coloneqq n+1\{n<5\}$$

{post} property: we want to get this set of natural numbers {0, 1, 2, 3, 4} after executing P.

For that, we need to define a set of preliminary values, the largest possible while satisfying P{post}. This set of natural numbers is {0, 1, 2, 3}.

The sets {0, 1} or {0} (Boolean domain for n) could also be a solution, but it is a restriction for my instruction by excluding/omitting some values.

Rules for theorem proving

Proof Obligation rules

- Invariant preservation

INV	$P(c) \wedge I(c,v) \wedge G(c,v) \wedge A(c,v,v') \Rightarrow I(c,v')$

Despite the state variable change by an event, each invariant must remain true

- Feasibility

FIS	$P(c) \wedge I(c,v) \wedge G(c,v) \Rightarrow \exists v'. A(c,v,v')$

For any non-deterministic event, we must prove its action is feasible

Proof Obligation rules

- Invariant preservation for initialization event

INI_INV	$P(c) \wedge I(c,v) \wedge K(v) \Rightarrow I(c,v')$

- Feasibility for initialization

INI_FIS	$P(c) \wedge I(c,v) \Rightarrow \exists v. K(v)$

Optional rule for permanent run

Deadlock Freedom rules

- Permanent availability of the system

DLF	$P(c) \land I(c,v) \Rightarrow G_1(c,v) \lor \cdots \lor G_m(c,v)$

A risk of dead-lock occurs when the guards of all events are continually false.

This theorem means that under the conditions where the axioms and invariants of the model are fulfilled, one of the guards is obligatorily true.

Rules for theorem proving

Refinement (Abrial, 1996)

- Let's consider events ae et re, the refinement of ae by re noted $ae \sqsubseteq re$ denotes
  - ✦ Reinforcement of the abstract guard

GRD_REF	$P(c) \wedge I(c,v) \wedge J(c,v,w) \wedge G_{re}(c,w) \Rightarrow G_{ae}(c,v)$

  - ✦ Reinforcement of the abstract action and reduction of the non determinism

INV_REF	$P(c) \wedge I(c,v) \wedge J(c,v,w) \wedge G_{re}(c,w) \wedge A_{re}(c,w,w') \Rightarrow \exists v'.(A_{ae}(c,v,v') \wedge J(c,v',w'))$

Abstract & refined models

- Abstract model

CONSTANTS : c

AXIOMES : P(c)

VARIABLES : v

INVARIANTS : I(c,v)

INITIALISATION
v := K(c)

EVENTS : ae
WHEN G(c,v)
THEN v := E(c,v)

- Refined model

INITIALISATION
w := N(c)

VARIABLES : w

INVARIANTS : J(c,v,w)

EVENTS : re
WHEN H(c,w)
THEN w := F(c,w)

Abstract & refined models

- Reinforcement of the abstract guard

$P(c)$ $I(c,v)$ $J(c,v,w)$ $H(c,w)$ $\vdash$ $G(c,v)$	GRD_REF

Abstract & refined models

- Invariant preservation for refined model

$P(c)$ $I(c,v)$ $J(c,v,w)$ $H(c,w)$ $\vdash$ $J_i(c,E(c,v), F(c,w))$	INV_REF

$P(c)$ $\vdash$ $J_i(c,K(c), N(c))$	INI_INV_REF

Abstract & refined models

- Relative deadlock freedom rule for refined model

$P(c)$ $I(c,v)$ $J(c,v,w)$ $G_1(c,v) \vee \ldots \vee G_m(c,v)$ $\vdash$ $H_1(c,w) \vee \ldots \vee H_n(c,w)$	DLF_REF

Example: abstract & refined models

- Abstract model of “primary/secondary” redundancy mechanism
  - Weak property: primary component active and failed first, then secondary component could fail
  - Invariant inv3: must not get secondary component failed when primary component ok
  - Invariant inv4: must not lose both components

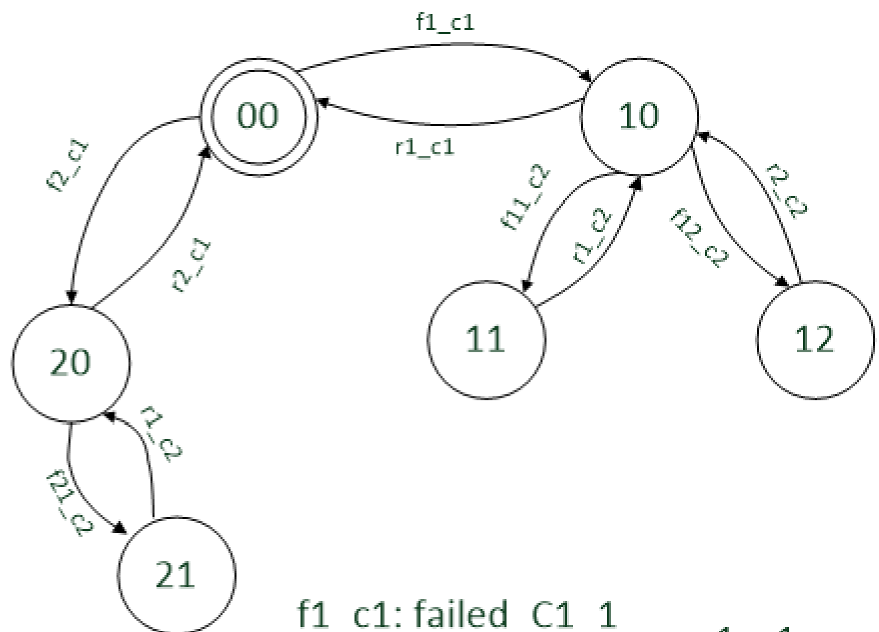
```
MACHINE
  CoSSAP_0
VARIABLES
  s_C1
  s_C2
```

invariants ?

Example: abstract & refined models

- Refined model of “primary/secondary” redundancy mechanism
  - Strong property: less than 2 consecutive failures without recovery

State-transition graph model ?



f1_c1: failed_C1_1
f2_c1: failed_C1_2
f11_c2: failed_C2_1
f21_c2: failed_C2_2
f12_c2: failed_C2_12

r1_c1: recovered_C1(1)
r2_c1: recovered_C1(2)
r1_c2: recovered_C2(1)
r2_c2: recovered_C2(2)

APPENDIX

Sequent

A sequent is a mathematical concept, useful to express our proof obligation rules.

By convention, the form of a sequent is:

$$\boldsymbol{H} \;\vdash\; \boldsymbol{G}$$

meaning the goal/conclusion *G* is provable under the set of hypotheses/assumptions *H*

The symbol $\vdash$ stands for "implies", "entails", "yields"

- First Peano (P1) axiom is: $\vdash\ \mathbf{0} \in \mathbb{N}$
- Second Peano (P2) axiom is: $\boldsymbol{n} \in \mathbb{N} \vdash \boldsymbol{n} + \mathbf{1} \in \mathbb{N}$
- and a derived second Peano axiom (P2') is:
  $\boldsymbol{n} \in \mathbb{N},\quad 0 < \boldsymbol{n} \vdash \boldsymbol{n} - \mathbf{1} \in \mathbb{N}$
- Third Peano (P3) axiom is: $\boldsymbol{n} \in \mathbb{N} \vdash \mathbf{0} \leq \boldsymbol{n}$
- INC axiom is: $\boldsymbol{n} \in \mathbb{N}, \boldsymbol{m} \in \mathbb{N},\ \boldsymbol{n} < \boldsymbol{m} \vdash \boldsymbol{n} + \mathbf{1} \leq \boldsymbol{m}$
- DEC axiom is: $\boldsymbol{n} \in \mathbb{N}, \boldsymbol{m} \in \mathbb{N},\ \boldsymbol{n} \leq \boldsymbol{m} \vdash \boldsymbol{n} - \mathbf{1} \leq \boldsymbol{m}$

Rules of inference

Mathematical rules enable to justify formally the transformation of sequents. These are *rules of inference*.

For instance the below rule called *monotonicity* of hypotheses is stated as follows:

$$\frac{H1 \ \vdash G}{H1, H2 \ \vdash G} \quad MON$$

where the upper sequent is called *antecedent* of the rule, whereas the lower is the *consequent*.

Rules of inference

More rules of inference:

$$\frac{H \vdash \mathrm{P}}{H \vdash P \vee Q}\ OR\_R$$	$$\frac{H, P \vdash R \qquad H, Q \vdash R}{H, P \vee Q \vdash R}\ OR\_L$$
$$\frac{}{P \vdash P}\ HYP$$	$$\frac{}{\perp \vdash P}\ CNTR$$
$$\frac{H(F), E = F \vdash \mathrm{P(F)}}{H(E), E = F \vdash P(E)}\ EQ\_LR$$ where P(E) is a predicate depending on an expression E (idem for H(E) and H(F))	$$\frac{}{\vdash E = E}\ EQL$$

Rules of inference

More rules of inference:

$\dfrac{H, \neg P \vdash \mathrm{Q}}{H \vdash P \lor Q}\ NEG$	
$\dfrac{H, P, Q \vdash \mathrm{R}}{H, P \land Q \vdash R}\ AND\_L$	$\dfrac{H \vdash \mathrm{P} \quad H \vdash \mathrm{Q}}{H \vdash P \land Q}\ AND\_R$
$\dfrac{}{H, P, \neg P \vdash Q}\ NOT\_L$	$\dfrac{H, P \vdash Q \quad H, P \vdash \neg Q}{H \vdash \neg P}\ NOT\_R$
$\dfrac{H, P, Q \vdash \mathrm{R}}{H, P, P \Rightarrow Q \vdash R}\ IMP\_L$	$\dfrac{H, P \vdash \mathrm{Q}}{H \vdash P \Rightarrow Q}\ IMP\_R$

Exercise

- A system enables a flight booking on an A350. The passenger capacity is 350. We can book/cancel several seats simultaneously. Q1: How many invariants are to be considered? Give them

Knowing,

```
CONTEXT
  BK_ct
CONSTANTS
  max_seat
AXIOMS
  axm1   :   max_seat ∈ ℕ
  axm2   :   max_seat = 350
END
```

```
MACHINE
  BK_mc
SEES
  BK_ct
VARIABLES
  free_seat
INVARIANTS
  inv1   :
```

Exercise

- Q2: Find out the correct Book/inv2/INV Proof Obligation rule (requirement on limitation)? What about inv1?

```
Book ≙
STATUS
 ordinary
ANY
 n
WHERE
 grd1  :  n ∈ ℕ
THEN
 act1  :  free_seat ≔ free_seat − n
END

END
```

Exercise

- Q3: What is the sequence of validated inference rules for the proof of Book/inv2/INV?

$a)\quad OR\_R$

b) MON

c) $DEC*$

d) $P3$

e) $P2$

```
Book ≙
STATUS
 ordinary
ANY
 n
WHERE
 grd1  :  n ∈ ℕ
THEN
 act1  :  free_seat ≔ free_seat − n
END

END
```

Example: « a; b; e » stands for $OR\_R; MON; P2$