

Introducing formal modelling techniques

Dominique Méry

**26th IFIP WG 6.1 International Conference on Formal Methods for Networked and
Distributed Systems
September 26-29 2006, Paris, France**

Formal modelling: why?

- ◇ Analysing requirements
 - ◇ Providing a view of the system
 - ◇ Justifying design decisions
 - ◇ Integrating mechanized and sound techniques for analysing systems.
 - ◇ Improving the communication among designers
 - ◇ Promoting abstraction and refinement techniques for developing (systems) models
 - ◇ Improving software systems quality
-

What systems?

- ◇ Distributed systems: distributed algorithms, agents-based systems, . . .
 - ◇ Embedded systems at home: mobile phone, wash machine, dish washer, micro-wave. . .
 - ◇ Hardware/software systems: SoC
 - ◇ Manufacturing systems
-

Organisation of lectures

- ◇ To develop models of realistic systems
 - ◇ To introduce step by step concepts and notations
 - ◇ To use tools
 - ◇ To play with abstractions and concretizations over models.
-

Summary of the lectures

- ▶ Introduction and History of B
- ▶ Event-based systems in B
- ▶ Simple case studied
- ▶ Sequential algorithms and Data Modelling
- ▶ Distributed programming
- ▶ Proof based System Engineering

Tools for the lectures

- ◇ Mathematical logic and set theory: B(ourbaki)
 - ◇ Proof assistants and Development assistants: B4free, Atelier B, ...
 - ◇ Model checking: not required, but choose your way!
 - ◇ Events System Models
 - ◇ Induction
 - ◇ Case studies: sequential algorithms, distributed algorithms, control access, ...
-

The History of B

- ◇ **Jean-Raymond Abrial**: Z in the 70s, B in the 80s, event B in the 90s and B[#] in the current millenium.
 - ◇ **Books**: **the B Book by Jean-Raymond Abrial in 1996**, **the B[#] Book by Jean-Raymond Abrial in ????**, others textbooks by K. Lano, H. Habrias, E. Sekerinski and K. Sere, ...
 - ◇ **Conferences**: ZB serie, ...
 - ◇ Success story: Meteor ligne 14 (control system), Smartcards (Gemplus), ...
 - ◇ Case studies: sequential algorithms (Schorr and Waite, ...), distributed algorithms (IEEE 1394 leader election protocol, PCI Bus Producer/Consumer Model,
-

Modelling systems

- ◇ A **system** is **observed**
 - ◇ Observation of things which are changing over the **time**
 - ◇ A system is characterized by a **state**
 - ◇ A state is made up of contextual **constant informations** over the problem theory and of **modifiable flexible informations** over the system.
-

Changing state of system

A **flexible variable** x is observed at different instants:

$$x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} x_2 \xrightarrow{\tau} x_3 \xrightarrow{\tau} \dots \xrightarrow{\tau} x_i \xrightarrow{\tau} x_{i+1} \xrightarrow{\tau} \dots$$

Changing state of system

A **flexible variable** x is observed at different instants:

$$x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} x_2 \xrightarrow{\tau} x_3 \xrightarrow{\tau} \dots \xrightarrow{\tau} x_i \xrightarrow{\tau} x_{i+1} \xrightarrow{\tau} \dots$$

τ hides effective changes of state or actions or events

Changing state of system

A **flexible variable** x is observed at different instants:

$$x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} x_2 \xrightarrow{\tau} x_3 \xrightarrow{\tau} \dots \xrightarrow{\tau} x_i \xrightarrow{\tau} x_{i+1} \xrightarrow{\tau} \dots$$

τ hides effective changes of state or actions or events

$$x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} x_2 \xrightarrow{\alpha_3} x_3 \xrightarrow{\alpha_4} \dots \xrightarrow{\alpha_i} x_i \xrightarrow{\alpha_{i+1}} x_{i+1} \xrightarrow{\alpha_{i+2}} \dots$$

Changing state of system

A **flexible variable** x is observed at different instants:

$$x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} x_2 \xrightarrow{\tau} x_3 \xrightarrow{\tau} \dots \xrightarrow{\tau} x_i \xrightarrow{\tau} x_{i+1} \xrightarrow{\tau} \dots$$

τ hides effective changes of state or actions or events

$$x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} x_2 \xrightarrow{\alpha_3} x_3 \xrightarrow{\alpha_4} \dots \xrightarrow{\alpha_i} x_i \xrightarrow{\alpha_{i+1}} x_{i+1} \xrightarrow{\alpha_{i+2}} \dots$$

Occurrences of τ can be added between two instants ie **stuttering steps**:

Changing state of system

A **flexible variable** x is observed at different instants:

$$x_0 \xrightarrow{\tau} x_1 \xrightarrow{\tau} x_2 \xrightarrow{\tau} x_3 \xrightarrow{\tau} \dots \xrightarrow{\tau} x_i \xrightarrow{\tau} x_{i+1} \xrightarrow{\tau} \dots$$

τ hides effective changes of state or actions or events

$$x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} x_2 \xrightarrow{\alpha_3} x_3 \xrightarrow{\alpha_4} \dots \xrightarrow{\alpha_i} x_i \xrightarrow{\alpha_{i+1}} x_{i+1} \xrightarrow{\alpha_{i+2}} \dots$$

Occurrences of τ can be added between two instants ie **stuttering steps**:

$$x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} x_2 \xrightarrow{\tau} x_2 \xrightarrow{\alpha_3} x_3 \xrightarrow{\alpha_4} \dots \xrightarrow{\alpha_i} x_i \xrightarrow{\tau} x_i \xrightarrow{\alpha_{i+1}} x_{i+1} \xrightarrow{\alpha_{i+2}} \dots$$

Properties of system

A **safety** property S over x states that something bad will not happen: $S(x)$ means that S holds for x

An **invariant** property I over x states a strong safety property

$$x_0 \xrightarrow{\alpha_1} x_1 \xrightarrow{\alpha_2} x_2 \xrightarrow{\tau} x_2 \xrightarrow{\alpha_3} x_3 \xrightarrow{\alpha_4} \dots \xrightarrow{\alpha_i} x_i \xrightarrow{\tau} x_i \xrightarrow{\alpha_{i+1}} x_{i+1} \xrightarrow{\alpha_{i+2}} \dots$$

$$\begin{array}{cccccccccccc} (S(x_0) & \xrightarrow{\alpha_1} & S(x_1) & \xrightarrow{\alpha_2} & S(x_2) & \xrightarrow{\tau} & S(x_2) & \xrightarrow{\alpha_3} & S(x_3) & \xrightarrow{\alpha_4} & \dots & \xrightarrow{\alpha_i} & S(x_i) & \xrightarrow{\tau} \\ S(x_i) & \xrightarrow{\alpha_{i+1}} & S(x_{i+1}) & \xrightarrow{\alpha_{i+2}} & \dots & & & & & & & & & & \end{array}$$

or equivalently $\forall i \in \mathbb{N} : S(x_i)$

Your decision?

- ◇ You can check for every i in \mathbb{N} that $S(x_i)$ is true but it can be long if states are different
 - ◇ You can compute an abstraction of the set of states
 - ◇ You can try to prove and for instance the induction principle may be useful
 - ◇ So be careful and improve your modelling before to run the checker
 - ◇ Use the induction
-

State properties of a system

- ◇ A state property namely $P(x)$ is a first order predicate with free variables x , where x is a flexible variable.
 - ◇ $P(x)$ denotes the set of values of x such that $P(x)$ holds.
 - ◇ $P(x)$ is interpreted over states of flexible variables for a system ($s \in States$)
 - ◇ $s \models P(x)$ means that $P(x)$ holds, when one substitutes occurrences of x by values of x , $s(x)$, in $P(x)$.
-

Examples of state properties

- ◇ Mutual exclusion
 - ◇ Deadlock freedom
 - ◇ Partial correctness
 - ◇ Safety properties
-

Relation/action over states

- ◇ An action α over states is a relation between values of state variables **before** and values of variables **after**

$$\alpha(x, x') \text{ or } x \xrightarrow{\alpha} x'$$

- ◇ Flexible variable x has two values x and x' .
 - ◇ Priming flexible variables is borrowed from TLA (See **lectures of S. Merz**)
 - ◇ **Hypothesis 1: Values of x belongs to a set of values called VALUES**
 - ◇ **Hypothesis 2: Relations over x and x' belong to a set of relations $\{r_0, \dots, r_n\}$**
-

Operational model of a system

- ◇ A system \mathcal{S} is observed with respect to flexible variables x .
- ◇ Flexible variables x of \mathcal{S} are modified according to a finite set of relations over the set of values **VALUES**: $\{r_0, \dots, r_n\}$
- ◇ **INIT**(x) denotes the set of possible initial values for x .

$$\mathcal{OMS} = (x, \mathbf{VALUES}, \mathbf{INIT}(x), \{r_0, \dots, r_n\})$$

Safety and invariance of system

◇ **Hypothesis 3:** $\mathcal{O}MS = (x, \text{VALUES}, \text{INIT}(x), \{r_0, \dots, r_n\})$

◇ **Hypothesis 4:** $x \longrightarrow x' \triangleq (x \ r_0 \ x') \vee \dots \vee (x \ r_n \ x')$

◇ $P(x)$ is inductively invariant for a system called \mathcal{S} , if

$$\begin{cases} \forall x \in \text{VALUES} : \text{INIT}(x) \Rightarrow P(x) \\ \forall x, x' \in \text{VALUES} : P(x) \wedge x \longrightarrow x' \Rightarrow P(x') \end{cases}$$

$P(x)$ is called an invariant in \mathcal{B}

◇ $P(x)$ is a safety property for a system called \mathcal{S} , if

$$\forall x, x' \in \text{VALUES} : \text{INIT}(x) \wedge x \xrightarrow{\star} x' \Rightarrow P(x')$$

$P(x)$ is called an assertion in \mathcal{B}

Modelling systems: first attempt

MODEL

m

...

...

...

VARIABLES

x

INVARIANT

$I(x)$

ASSERTIONS

$P(x)$

INITIALISATION

$Init(x)$

RELATIONS

$\{r_0, \dots, r_n\}$

END

- ◇ A model has a name m
- ◇ Flexible variables x are declared
- ◇ $I(x)$ provides information over x
- ◇ $P(x)$ provides information over x

Checking safety properties of the model

- ◇ $\forall x, x' \in \text{VALUES} : \text{INIT}(x) \wedge x \xrightarrow{*} x' \Rightarrow \text{P}(x')$
 - ◇ **Solution 1** Writing a procedure checking $\text{INIT}(x) \wedge x \xrightarrow{*} x' \Rightarrow \text{P}(x')$ for each pair $x, x' \in \text{VALUES}$, when VALUES is finite and small.
 - ◇ **Solution 2** Writing a procedure checking $\text{INIT}(x) \wedge x \xrightarrow{*} x' \Rightarrow \text{P}(x')$ for each pair $x, x' \in \text{VALUES}$, by constructing an abstraction of VALUES .
 - ◇ **Solution 3** Writing a proof for $\forall x, x' \in \text{VALUES} : \text{INIT}(x) \wedge x \xrightarrow{*} x' \Rightarrow \text{P}(x')$.
-

Defining an induction principle for an operational model

$$(I) \forall x, x' \in \mathbf{VALUES} : \mathbf{INIT}(x) \wedge x \xrightarrow{*} x' \Rightarrow \mathbf{P}(x')$$

if, and only if,

(II) there exists a state property $I(x)$ such that:

$$\forall x, x' \in \mathbf{VALUES} : \begin{cases} (1) \mathbf{INIT}(x) \Rightarrow I(x) \\ (2) I(x) \Rightarrow \mathbf{P}(x) \\ (3) I(x) \wedge x \longrightarrow x' \Rightarrow I(x') \end{cases}$$

if, and only if,

(III) there exists a state property $I(x)$ such that:

$$\forall x, x' \in \mathbf{VALUES} : \begin{cases} (1) \mathbf{INIT}(x) \Rightarrow I(x) \\ (2) I(x) \Rightarrow \mathbf{P}(x) \\ (3) \forall i \in \{0, \dots, n\} : I(x) \wedge x r_i x' \Rightarrow I(x') \end{cases}$$

Modelling systems: second attempt

MODEL

m

...

...

...

VARIABLES

x

INVARIANT

$I(x)$

ASSERTIONS

$P(x)$

INITIALISATION

$Init(x)$

RELATIONS

$\{r_0, \dots, r_n\}$

END

✓ $\forall x, x' \in \text{VALUES} : \text{INIT}(x) \Rightarrow I(x)$

✓ $\forall x, x' \in \text{VALUES} : \forall i \in \{0, \dots, n\} :$

$$I(x) \wedge x r_i x' \Rightarrow I(x')$$

✓ $\forall x, x' \in \text{VALUES} : I(x) \Rightarrow P(x)$

Modelling systems: last attempt?

MODEL

m

?
?
?

VARIABLES

x

INVARIANT

$I(x)$

ASSERTIONS

$P(x)$

INITIALISATION

$Init(x)$

RELATIONS

$\{r_0, \dots, r_n\}$

END

- ✓ What are the environment of the proof for properties?
- ✓ What are theories?
- ✓ How are defining the static objects?

Modelling systems: last attempt!

MODEL

m

$\Gamma(m)$

VARIABLES

x

INVARIANT

$I(x)$

ASSERTIONS

$P(x)$

INITIALISATION

$Init(x)$

RELATIONS

$\{r_0, \dots, r_n\}$

END

- ✓ $\Gamma(m)$ defines the static environment for the proofs related to m .
- ✓ $\Gamma(m) \vdash \forall x, x' \in \text{VALUES} : \text{INIT}(x) \Rightarrow I(x)$
- ✓ $\forall i \in \{0, \dots, n\} :$
 $\Gamma(m) \vdash \forall x, x' \in \text{VALUES} : I(x) \wedge x r_i x' \Rightarrow I(x')$
- ✓ $\langle \Gamma(m) \vdash \forall x, x' \in \text{VALUES} : I(x) \Rightarrow P(x)$

Events System Models

An **event system model** is made of

State **constants** and state **variables** constrained by a state **invariant**

A finite set of **events**

Proofs ensures the consistency between the invariant and the events

An event system model can be **refined**

Proofs must ensure the correctness of refinement

Modelling systems: Hello world!

MODEL

FACTORIAL_EVENTS

CONSTANTS *factorial, m*

PROPERTIES

$$\begin{aligned} & m \in \mathbb{N} \wedge \text{factorial} \in \mathbb{N} \leftrightarrow \mathbb{N} \wedge 0 \mapsto 1 \in \text{factorial} \wedge \\ & \forall (n, fn). (n \mapsto fn \in \text{factorial} \Rightarrow n+1 \mapsto (n+1) \cdot fn \in \text{factorial}) \wedge \\ & \forall f \cdot \left(\begin{array}{l} f \in \mathbb{N} \leftrightarrow \mathbb{N} \wedge \\ 0 \mapsto 1 \in f \wedge \\ \forall (n, fn). (n \mapsto fn \in f \Rightarrow n+1 \mapsto (n+1) \times fn \in f) \\ \Rightarrow \\ \text{factorial} \subseteq f \end{array} \right) \end{aligned}$$

VARIABLES

result

INVARIANT

$result \in \mathbb{N}$

ASSERTIONS

$\text{factorial} \in \mathbb{N} \longrightarrow \mathbb{N};$

$\text{factorial}(0) = 1;$

$\forall n. (n \in \mathbb{N} \Rightarrow \text{factorial}(n+1) = (n+1) \times \text{factorial}(n))$

INITIALISATION

$result := \mathbb{N}$

EVENTS

$\text{computation} = \mathbf{begin} \text{ result } := \text{factorial}(m) \mathbf{end}$

END

Modelling systems: relations to events

MODEL

m

SETS

s

CONSTANTS

c

PROPERTIES

$P(s, c)$

VARIABLES

x

INVARIANT

$I(x)$

ASSERTIONS

$P(x)$

INITIALISATION

$Init(x)$

EVENTS

$\{r_0, \dots, r_n\}$

END

- ✓ $\Gamma(m)$ defines the static environment for the proofs related to m from s , c and $P(s, c)$.
- ✓ $\Gamma(m) \vdash \forall x, x' \in \text{VALUES} : \text{INIT}(x) \Rightarrow I(x)$
- ✓ $\forall i \in \{0, \dots, n\} :$
 $\Gamma(m) \vdash \forall x, x' \in \text{VALUES} : I(x) \wedge x r_i x' \Rightarrow I(x')$
- ✓ $\Gamma(m) \vdash \forall x, x' \in \text{VALUES} : I(x) \Rightarrow P(x)$

A simple model SM

```
MODEL
  SM
VARIABLES
  x
INVARIANT
  x : INTEGER &
  x = -1
THEOREMS
  x <= 0
INITIALISATION
  x := -1
EVENTS
act =
  WHEN x >= 0 THEN
    x := x + 1
  END
END
```

```
MODEL
  SM
VARIABLES
  x
INVARIANT
   $x \in \mathbb{Z}$ 
   $x = -1$ 
THEOREMS
   $x \leq 0$ 
INITIALISATION
   $x := -1$ 
EVENTS
act =
  WHEN  $x \geq 0$  THEN
     $x := x + 1$ 
  END
END
```

Proof obligations for the model

- $\Gamma(SM)$ defines the static environment for the proofs related to arithmetic.
 - $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x = -1 \Rightarrow x \leq 0$
 - $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \Rightarrow x' \leq 0$
-

Proof obligations for the model

✓ $\Gamma(SM)$ defines the static environment for the proofs related to arithmetic.

✓ $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x = -1 \Rightarrow x \leq 0$

□ $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \Rightarrow x' \leq 0$

Proof obligations for the model

✓ $\Gamma(SM)$ defines the static environment for the proofs related to arithmetic.

✓ $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x = -1 \Rightarrow x \leq 0$

□ $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \Rightarrow x' \leq 0$

□ $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x = 0 \wedge x' = x+1 \Rightarrow x' \leq 0$

□ $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x = 0 \Rightarrow x+1 \leq 0$

□ $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x = 0 \Rightarrow 1 \leq 0: !$

Interpreting unprovable proof obligations

- $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x \leq 0 \wedge x \geq 0 \wedge x' = x+1 \Rightarrow x' \leq 0$
 - $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x = 0 \wedge x' = x+1 \Rightarrow x' \leq 0$
 - $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x = 0 \Rightarrow x+1 \leq 0$
 - $\Gamma(SM) \vdash \forall x, x' \in \mathbb{Z} : x = 0 \Rightarrow 1 \leq 0: !$
 - $x \leq 0$ is not (inductively) invariant for the model SM: it is a safety property.
-

A simple model SM'

```
MODEL
  SM'
VARIABLES
  x
INVARIANT
  x : INTEGER &
  x <= 0
INITIALISATION
  x := -1
EVENTS
act =
  WHEN x >= 0 THEN
    x := x + 1
  END
END
```

```
MODEL
  SM'
VARIABLES
  x
INVARIANT
   $x \in \mathbb{Z}$ 
   $x \leq 0$ 
INITIALISATION
   $x := -1$ 
EVENTS
act =
  WHEN  $x \geq 0$  THEN
     $x := x + 1$ 
  END
END
```

Proof obligations for the model SM'

- ✓ $\Gamma(SM')$ defines the static environment for the proofs related to arithmetic.
 - ✓ $\Gamma(SM') \vdash \forall x, x' \in \mathbb{Z} : x = -1 \Rightarrow x \leq 0$
 - ✓ $\Gamma(SM') \vdash \forall x, x' \in \mathbb{Z} : x \in \mathbb{Z} \wedge x = -1 \wedge x \geq 0 \wedge x' = x+1 \Rightarrow x' = -1$
 - ✓ $\Gamma(SM') \vdash x = -1 \Rightarrow x \leq 0$
 - ✓ The invariant is strong enough!
-

Modelling systems

step 1: Understanding the **problem** to solve

step 2: Organizing requirements and extracting properties

step 3: Writing a first very **abstract** system model

step 4: Consulting the requirements and **adding** a new detail in the current model by **refinement**

step 5: Either the model is enough detailed and the process stops, or the model is not yet enough concrete and the step 4 is repeated.
