# The B Modelling Language

Dominique Méry

26th IFIP WG 6.1 International Conference on Formal Methods for Networked and Distributed Systems

September 26-29 2006, Paris, France

# B is a modelling language

- ◇ B is used for modelling closed systems
- ◇ B is not a programming language but a language for developing models of systems
- ◇ A B model is defined by a set of events
- ◇ A B model is characterized by an invariant
- ◇ A B model states safety properties of the current system
- ◇ A B model is internally consistent with respect to a list of proof obligations

# 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

# B models

**MODEL**
  $m$
**SETS**
  $s$
**CONSTANTS**
  $c$
**PROPERTIES**
  $P(s, c)$
**VARIABLES**
  $x$
**INVARIANT**
  $I(x)$
**ASSERTIONS**
  $A(x)$
**INITIALISATION**
  $<$substitution$>$
**EVENTS**
  $<$list of events$>$
**END**

- ☑ A model has a name $m$
- ☑ the clause **SETS**, **CONSTANTS** and the clause **PROPERTIES** introduce information related to the mathematical structure of the problem to solve
- ☑ The invariant $I(x)$ types the variable $x$, which is assumed to be initialized with respect to the initial conditions and which is preserved by events (or transitions) of the list of events.

# Meaning of the B model

- $s$, $c$ and $P(s,c)$ define the mathematical structure of the problem: $\Gamma(s,c)$.
- Each computation starts by a state satisfying $Init(x)$.
- The list of possible events is $\{e_1,\ldots,e_n\}$ and any event $e$ is characterized by a binary relation $BA(e)(x,x')$ over possible values of $x$.
- For each event $e$, there is a condition called a guard which is true, when the event is observed.

# Events

| Event : $E$ | Before-After Predicate |
|---|---|
| **BEGIN** $x : P(x_0, x)$ **END** | $P(x, x')$ |
| **SELECT** $G(x)$ **THEN** $x : P(x_0, x)$ **END** | $G(x) \quad \wedge \quad P(x, x')$ |
| **ANY** $t$ **WHERE** $G(t, x)$ **THEN** $x : P(x_0, x, t)$ **END** | $\exists\, t \cdot (\, G(t, x) \quad \wedge \quad P(x, x', t)\, )$ |

# Guards of event

| Event : $E$ | Guard: grd(E) |
|---|---|
| **BEGIN** $S$ **END** | $TRUE$ |
| **SELECT** $G(x)$ **THEN** $T$ **END** | $G(x)$ |
| **ANY** $t$ **WHERE** $G(t,x)$ **THEN** $T$ **END** | $\exists\, t \cdot G(t,x)$ |

# Proof obligations for a B model

| | Proof obligation |
|---|---|
| (INV1) | $\Gamma(s,c) \ \vdash \ Init(x) \ \Rightarrow \ I(x)$ |
| (INV2) | $\Gamma(s,c) \ \vdash \ I(x) \ \wedge \ BA(e)(x,x') \ \Rightarrow \ I(x')$ |
| (DEAD) | $\Gamma(s,c) \ \vdash \ I(x) \ \Rightarrow \ (\mathsf{grd}(e_1) \ \vee \ \ldots \ \mathsf{grd}(e_n))$ |
| (SAFE) | $\Gamma(s,c) \ \vdash \ I(x) \ \Rightarrow \ A(x)$ |
| (FIS) | $\Gamma(s,c) \ \vdash \ I(x) \ \wedge \ \mathsf{grd}\,(E) \ \Rightarrow \ \exists x' \cdot P(x,x')$ |

# Modelling systems: Hello world!

**MODEL**

*FACTORIAL_EVENTS*

**CONSTANTS** $factorial, p$

**PROPERTIES**

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

**VARIABLES**

$result$

**INVARIANT**

$result \in \mathbb{N}$

**ASSERTIONS**

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

$factorial(0) = 1$ ;

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

**INITIALISATION**

$result :\in \mathbb{N}$

**EVENTS**

$computation =$ **BEGIN** $result := factorial(p)$ **END**

**END**

# Modelling systems: communications

**SETS**

$MESSAGES; AGENTS$

**PROPERTIES**

$MESSAGES \neq \emptyset \wedge$

$AGENTS \neq \emptyset$

**VARIABLES**

$sent, got, lost$

# Modelling systems: communications

**INVARIANT**

$$sent \subseteq AGENTS \times AGENTS \times MESSAGES \wedge$$

$$got \subseteq AGENTS \times AGENTS \times MESSAGES \wedge$$

$$lost \subseteq AGENTS \times AGENTS \times MESSAGES \wedge$$

$$(got \cup lost) \subseteq sent \wedge$$

$$lost = \emptyset$$

**INITIALISATION**

$$sent := \emptyset \,||$$

$$got := \emptyset \,||$$

$$lost := \emptyset$$

# Modelling systems: communications

$SENDING$ =

**ANY** $agent1, agent2, message$ **WHERE**

$agent1 \in AGENTS \wedge$

$agent2 \in AGENTS \wedge$

$message \in MESSAGES \wedge$

$agent1 \mapsto agent2 \mapsto message \notin sent$

**THEN**

$sent := sent \cup \{agent1 \mapsto agent2 \mapsto message\}$

**END**;

$RECEIVING$ = **ANY** $agent1, agent2, message$ **WHERE**

$agent1 \in AGENTS \wedge$

$agent2 \in AGENTS \wedge$

$message \in MESSAGES \wedge$

$agent1 \mapsto agent2 \mapsto message \in sent \wedge$

$agent1 \mapsto agent2 \mapsto message \notin got$

**THEN**

$got := got \cup \{agent1 \mapsto agent2 \mapsto message\}$

**END**;

$LOOSING$ = **BEGIN** SKIP **END**

# Modelling systems: problems?

- Systems are generally very complex

- Invariant should be strong enough for proving safety properties

- Problems for modelling: finding suitable mathematical structures, listing events or actions of the system, proving proof obligations, …

# Solution: refining models

- ☺ To understand more and more the system
- ☺ To distribute the complexity of the system
- ☺ To distribute the difficulties of the proof
- ☺ To improve explanations
- ☺ Validation (step by step)
- ☺ Refinement (invariant & behavior)

# Refinement of models

- ◇ we can add more details (like superposition),
- ◇ we can add new events (we can observe more transformations),
- ◇ we prove that the concrete behaviors are abstract ones

  $\leadsto$ we got the abstract invariant for free.

- ◇ each new event refines **SKIP**
- ◇ no deadlock
- ◇ abstract events occur (new events decrease something)

# Refinement

**REFINEMENT** $r$

**REFINES** $m$

**SETS** $t$

**CONSTANTS** $d$

**PROPERTIES** $Q(t, d)$

**VARIABLES** $y$

**INVARIANT**

$J(x, y)$

**VARIANT**

$V(y)$

**ASSERTIONS**

$B(y)$

**INITIALISATION**

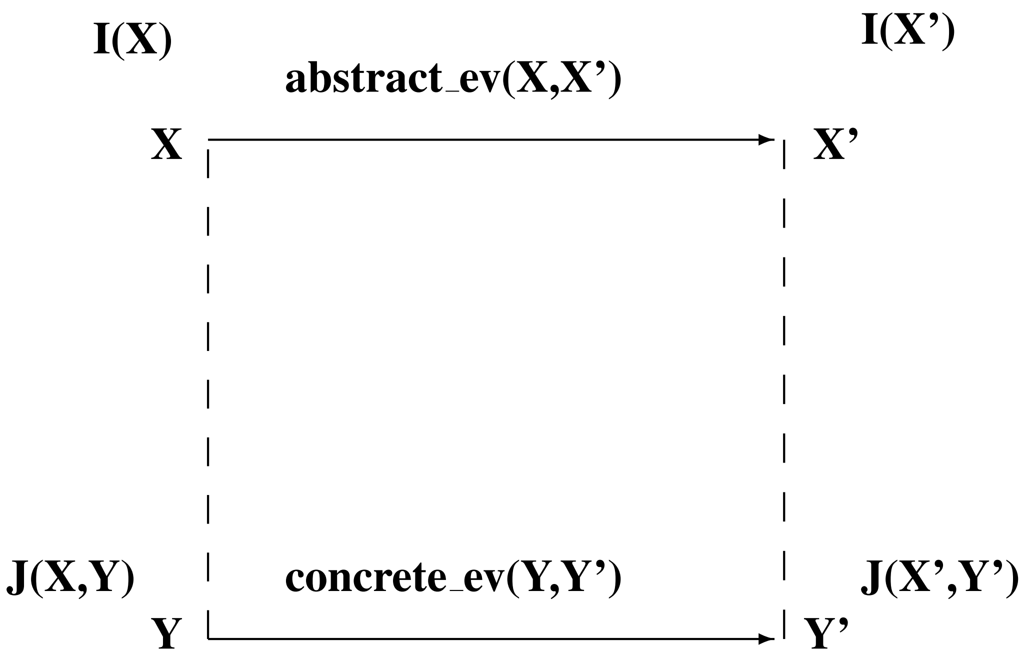
$y : INIT(y)$

**EVENTS**

<list of events>

**END**

# Refinement of a model by another one



# Proof obligations for refinement

(REF1) $\text{INITC}(y) \Rightarrow \exists x \cdot (\text{INIT}(x) \wedge \mathsf{J}(x,y))$ :

The initial condition of the refinement model imply that there exists an abstract value in the abstract model such that that value satisfies the initial conditions of the abstract one and implies the new invariant of the refinement model.

(REF2) $\mathsf{I}(x) \wedge \mathsf{J}(x,y) \wedge BAC(y,y') \Rightarrow \exists x'.(BAA(x,x') \wedge \mathsf{J}(x',y'))$ :

The invariant in the refinement model is preserved by the refined event and the activation of the refined event triggers the corresponding abstract event.

(REF3) $\mathsf{I}(x) \wedge \mathsf{J}(x,y) \wedge BAC(y,y') \Rightarrow \mathsf{J}(x,y')$ :

The invariant in the refinement model is preserved by the refined event but the event of the refinement model is a new event which was not visible in the abstract model; the new event refines $skip$.

# Proof obligations for refinement

(REF4): $\mathsf{I}(x) \;\wedge\; \mathsf{J}(x,y) \;\wedge\; (G_1(x) \vee \ldots \vee G_n(x)) \;\Rightarrow\; H_1(y) \vee \ldots \vee H_k(y)$ :

The guards of events in the refinement model are strengthened and we have to prove that the refinement model is not more blocked than the abstract.

(REF5): $\mathsf{I}(x) \;\wedge\; \mathsf{J}(x,y)) \;\Rightarrow\; V(y) \in \mathbb{N}$

(REF6): $\mathsf{I}(x) \;\wedge\; \mathsf{J}(x,y) \;\wedge\; BAC(y,y') \;\Rightarrow\; V(y') < V(y)$ :

New events should not block forever abstract ones.

(REF7): $\Gamma(s,c) \vdash I(x) \;\wedge\; J(x,y) \;\wedge\; \mathsf{grd}\,(E) \;\Rightarrow\; \exists y' \cdot P(y,y')$

# Refining the factorial model

```
REFINEMENT RFACT
REFINES FACTORIAL_EVENTS
VARIABLES  fac,result,m
INVARIANT
 fac : NATURAL +-> NATURAL & fac <: factorial &
 dom(fac)<: 0..m & dom(fac) /= {}
VARIANT n-x
INITIALISATION x:=0 || fac:={0|->1}
EVENTS
prog = SELECT p /: dom(fac) THEN
          ANY x WHERE
              x:NATURAL & x : dom(fac) & x+1 /: dom(fac)
          THEN
             fac(x+1):=(x+1)*fac(x)
          END
        END;
computation = SELECT p : dom(fac) THEN  result:=fac(p)  END
END
```

# The factorial model

**MODEL**
*FACTORIAL_EVENTS*
**CONSTANTS** $factorial, m$
**PROPERTIES**

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

**VARIABLES**
$result$
**INVARIANT**
$result \in \mathbb{N}$
**ASSERTIONS**
$factorial \in \mathbb{N} \longrightarrow \mathbb{N}$ ;
$factorial(0) = 1$ ;
$\forall n.(n \in \mathbb{N} \Rightarrow factorial(n+1) = (n+1) \times factorial(n))$
**INITIALISATION**
$result :\in \mathbb{N}$
**EVENTS**
$computation =$ **begin** $result := factorial(m)$ **end**
**END**

# Summary on the B modelling language

- A language for expressing mathematical structures: sets, relations, functions, ...
- A language for expressing transitions over states: events
- A language for expressing safety properties
- A language of system models
- And more ... like modalities
- Next session: case studies