The B Modelling Language (second trial!) and maybe . . . Sequential Algorithms

Dominique Méry

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

September 26-29 2006, Paris, France

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 in the refinement model!
- 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

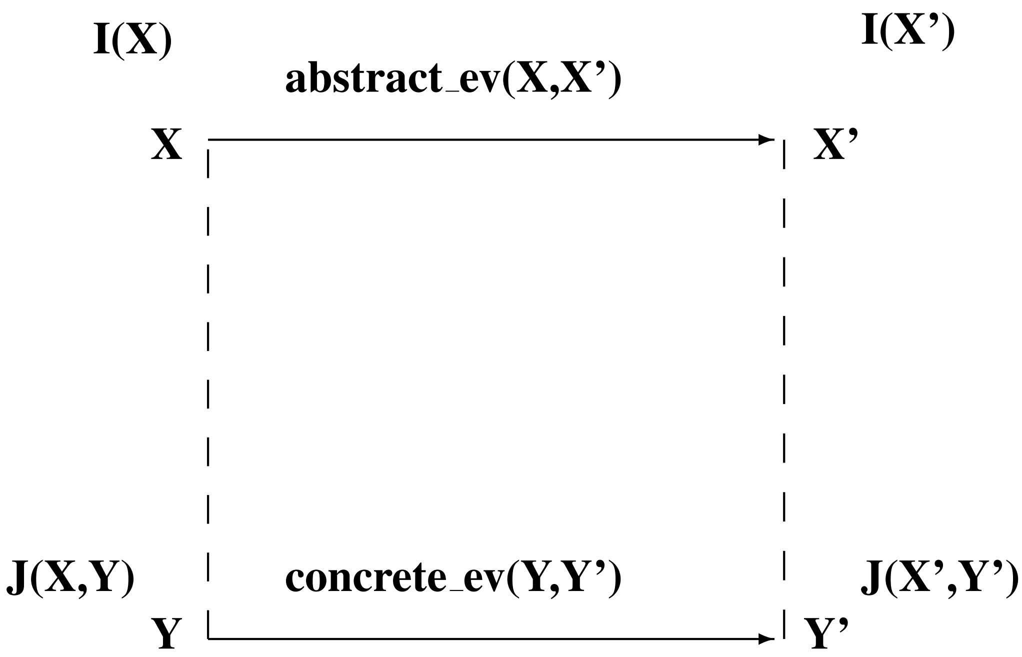
$y : INIT(y)$

EVENTS

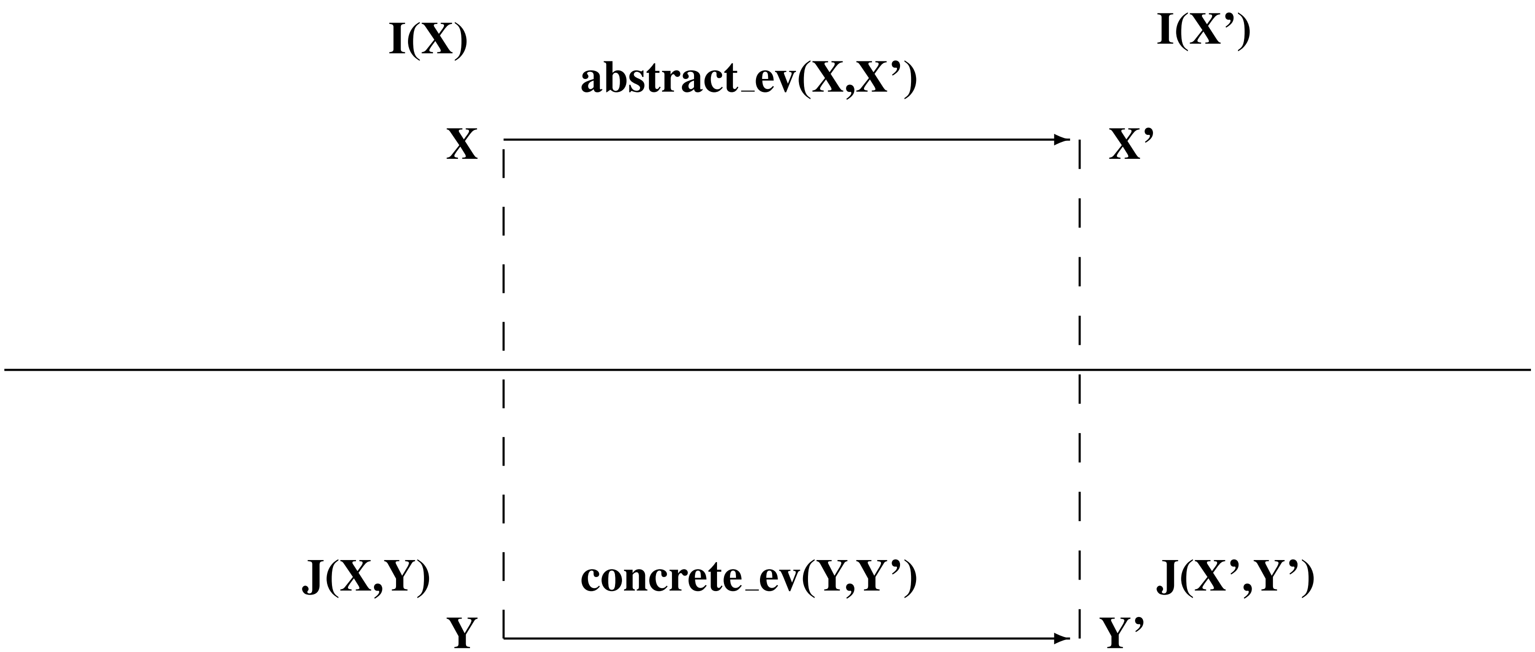
<list of events>

END

Refinement of a model by another one



Refinement of a model by another one



Proof obligations for refinement

(REF1) $\textsc{InitC}(y) \Rightarrow \exists x \cdot (\textsc{Init}(x) \;\wedge\; \mathsf{J}(x,y))$:

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

The factorial model

MODEL
FACTORIAL_EVENTS
CONSTANTS $factorial, m$
PROPERTIES

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

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

Refining the factorial model

REFINEMENT $RFACT$
REFINES
FACTORIAL_EVENTS
VARIABLES
fac
INVARIANT
$fac \in \mathbb{N} \nrightarrow \mathbb{N} \wedge fac \subseteq factorial \wedge dom(fac) \subseteq 0..m \wedge$
$dom(fac) \neq \emptyset$
ASSERTIONS

VARIANT
$m - card(dom(fac))$
INITIALISATION
$fac := 0 \mapsto 1$
EVENTS
$computation =$ **SELECT** $m \in dom(fac)$ **THEN** $result := fac(m)$ **END**
$prog =$ **SELECT** $m \notin dom(fac)$ **THEN**
ANY x **WHERE**
$x \in \mathbb{N} \wedge x \in dom(fac) \wedge x+1 \notin dom(fac)$
THEN
$fac(x+1) := (x+1) \cdot fac(x)$
END
END

Proof obligations for RFACT

(R1) $fac = \{0 \mapsto 1\} \Rightarrow \exists result.\left(\begin{array}{l} result \in \mathbb{N} \\ \left(\begin{array}{l} fac \in \mathbb{N} \nrightarrow \mathbb{N} \wedge \\ fac \subseteq factorial \wedge \\ dom(fac) \subseteq 0..m \wedge \\ dom(fac) \neq \emptyset \end{array}\right) \end{array}\right)$

(R2)

$$\left(\begin{array}{l} result \in \mathbb{N} \ \wedge \\ \left(\begin{array}{l} fac \in \mathbb{N} \nrightarrow \mathbb{N} \wedge \\ fac \subseteq factorial \wedge \\ dom(fac) \subseteq 0..m \wedge \\ dom(fac) \neq \emptyset \end{array}\right) \wedge \\ result' = fac(m) \wedge \\ fac = fac' \end{array}\right) \Rightarrow \exists result'.\left(\begin{array}{l} result' = factorial(m) \ \wedge \\ \left(\begin{array}{l} fac' \in \mathbb{N} \nrightarrow \mathbb{N} \wedge \\ fac' \subseteq factorial \wedge \\ dom(fac') \subseteq 0..m \wedge \\ dom(fac') \neq \emptyset \end{array}\right) \end{array}\right)$$

Proof obligations for RFACT

(R3)

$$\left(\begin{array}{l} result \in \mathbb{N} \ \wedge \\ \left(\begin{array}{l} fac \in \mathbb{N} \nrightarrow \mathbb{N} \wedge \\ fac \subseteq factorial \wedge \\ dom(fac) \subseteq 0..m \wedge \\ dom(fac) \neq \emptyset \end{array}\right) \wedge \\ m \notin dom(fac) \wedge \\ x \in \mathbb{N} \wedge \\ x \in dom(fac) \in x+1 \notin dom(fac) \wedge \\ fac' = fac \cup \{x+1 \mapsto fac(x).(x+1)\} \end{array}\right) \Rightarrow \left(\left(\begin{array}{l} fac' \in \mathbb{N} \nrightarrow \mathbb{N} \wedge \\ fac' \subseteq factorial \wedge \\ dom(fac') \subseteq 0..m \wedge \\ dom(fac') \neq \emptyset \end{array}\right)\right)$$

(R4):

$$\left(\begin{array}{l} result \in \mathbb{N} \ \wedge \\ \left(\begin{array}{l} fac \in \mathbb{N} \nrightarrow \mathbb{N} \wedge \\ fac \subseteq factorial \wedge \\ dom(fac) \subseteq 0..m \wedge \\ dom(fac) \neq \emptyset \end{array}\right) \wedge \\ true \end{array}\right) \Rightarrow \left(m \in dom(fac) \vee m \notin dom(fac) \right)$$

Proof obligations for RFACT

(R5):

$$\left(\begin{array}{l} result \in \mathbb{N} \ \wedge \\ \left(\begin{array}{l} fac \in \mathbb{N} \nrightarrow \mathbb{N} \wedge \\ fac \subseteq factorial \wedge \\ dom(fac) \subseteq 0..m \wedge \\ dom(fac) \neq \emptyset \end{array}\right) \ \wedge \\ result' = factorial(m) \end{array}\right) \Rightarrow \ m - card(dom(fac)) \in \mathbb{N}$$

(R6):

$$\left(\begin{array}{l} result \in \mathbb{N} \ \wedge \\ \left(\begin{array}{l} fac \in \mathbb{N} \nrightarrow \mathbb{N} \wedge \\ fac \subseteq factorial \wedge \\ dom(fac) \subseteq 0..m \wedge \\ dom(fac) \neq \emptyset \end{array}\right) \ \wedge \\ m \notin dom(fac) \wedge \\ x \in \mathbb{N} \wedge \\ x \in dom(fac) \in x+1 \notin dom(fac) \wedge \\ fac' = fac \cup \{x+1 \mapsto fac(x).(x+1)\} \end{array}\right) \Rightarrow \ m - card(dom(fac')) < m - card(d$$

Proof obligations for RFACT

(R7):

$$\left(\begin{array}{l} result \in \mathbb{N} \ \wedge \\ \left(\begin{array}{l} fac \in \mathbb{N} \mathrel{+\!\!\!\to} \mathbb{N} \wedge \\ fac \subseteq factorial \wedge \\ dom(fac) \subseteq 0..m \wedge \\ dom(fac) \neq \emptyset \end{array}\right) \ \wedge \\ \left(\begin{array}{l} m \notin dom(fac) \wedge \\ x \in \mathbb{N} \wedge \\ x \in dom(fac) \ \wedge \ x+1 \notin dom(fac) \end{array}\right) \\ \end{array}\right)$$

$$\Rightarrow \ \exists fac', x. \left(\begin{array}{l} m \notin dom(fac) \wedge \\ x \in \mathbb{N} \wedge \\ x \in dom(fac) \ \wedge \ x+1 \notin dom(fac) \wedge \\ fac' = fac \cup \{x+1 \mapsto fac(x).(x+1)\} \end{array}\right)$$

Tools for the B modelling language

- B tool: tool for proving properties over abstract machines
- B ToolKit: tool which integrates proof assistant, POG, animator, ...
- Atelier B: tool which integrates proof assistant, POG, animator, pp, ...
- B4free: tool which integrates proof assistant, pp in an Xemacs environment

Automating proofs obligations checking

- B tools provide automatic proof procedure for sequent calculus
- pr
- pp (predicate prover)
- Atelier B (ClearSy): `http://www.atelierb.societe.com/`
- Click'n'Prove: `http://www.loria.fr/~cansell/cnp.html`
- B4free (ClearSy): `http://www.b4free.com/`

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
- **Now case studies**

Summary of case studies

- The factorial function
- Finding an index in a table
- Third case study
- Exercices

First example

Assume that a function $factorial$ is mathematically defined.

Assume that m is a natural number.

Problem: compute $result$ such that $result = factorial(m)$

The factorial function: first model, problem

CONSTANTS

factorial, m

PROPERTIES

$$\mathit{factorial} \in \mathbb{N} \longrightarrow \mathbb{N} \land$$

$$\mathit{factorial}(0) = 1 \land$$

$$\forall n.(n \in \mathbb{N}^{\star} \Rightarrow \mathit{factorial}(n) = n \times \mathit{factorial}(n-1)) \land$$

$$m \in \mathbb{N}$$

The factorial function: first model, magical event

computation $\widehat{=}$

BEGIN

$$result := factorial(m)$$

END

The one-shot computation

The factorial function: second model, the recipe

$$\mathit{fac} = \{\, 0 \mapsto 1 \,\}$$

The factorial function: second model, the recipe

$$fac = \{0 \mapsto 1\}$$

$$fac = \{0 \mapsto 1, 1 \mapsto 1\}$$

The factorial function: second model, the recipe

$$fac = \{0 \mapsto 1\}$$

$$fac = \{0 \mapsto 1, 1 \mapsto 1\}$$

$$fac = \{0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2\}$$

The factorial function: second model, the recipe

$$\mathit{fac} = \{0 \mapsto 1\}$$

$$\mathit{fac} = \{0 \mapsto 1, 1 \mapsto 1\}$$

$$\mathit{fac} = \{0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2\}$$

$$\ldots$$

$$\mathit{fac} = \{0 \mapsto 1, 1 \mapsto 1, 2 \mapsto 2, ..., m \mapsto m!\}$$

The factorial function: second model, progress event

prog $\widehat{=}$

SELECT

$m \notin \mathrm{dom}(fac)$

THEN

ANY x **WHERE**

$x \in \mathbb{N}$

$x \in \mathrm{dom}(fac)$

$x+1 \notin \mathrm{dom}(fac)$

THEN

$fac(x+1) := (x+1) \cdot fac(x)$

END

END

The factorial function: second model, computation event

computation $\widehat{=}$
BEGIN
$result := factorial(m)$
END

computation $\widehat{=}$
SELECT
$m \in \mathrm{dom}(fac)$
THEN
$result := fac(m)$
END

The factorial function: second model, the invariant

$$
\begin{array}{l}
fac \in \mathbb{N} \nrightarrow \mathbb{N} \wedge \\
fac \subseteq factorial \\
\mathsf{dom}(fac) \subseteq 0..m \wedge \\
\mathsf{dom}(fac) \neq \emptyset
\end{array}
$$

The factorial function: last model

$$i \mapsto \mathit{fn}\ \{\boxed{0 \mapsto 1}\}$$

$$i \mapsto \mathit{fn}\ \{0 \mapsto 1,\ \boxed{1 \mapsto 1}\}$$

$$i \mapsto \mathit{fn}\ \{0 \mapsto 1,\ 1 \mapsto 1,\ \boxed{2 \mapsto 2}\}$$

$$\ldots$$

$$i \mapsto \mathit{fn}\ \{0 \mapsto 1,\ 1 \mapsto 1,\ 2 \mapsto 2,\ \ldots.,\ \boxed{m \mapsto m!}\}$$

The factorial function: last model

prog $\widehat{=}$

SELECT

$i \neq m$

THEN

$fq := (i+1) \cdot fq \,||$

$i := i+1$

END

The factorial function: last model

computation $\widehat{=}$

SELECT

$i = m$

THEN

$result := fq$

END

The factorial function: getting a final algorithm

$$\boxed{\begin{array}{l} i \in 0..m \\ fq = fac(i) \end{array}}$$

$$\boxed{\begin{array}{l} i := 0 \,||\, fq := 1 \\ \textbf{WHILE } i \neq m \textbf{ DO} \\ \quad fq := (i+1) \cdot fq \,|| \\ \quad i := i+1 \\ \textbf{END} \\ result := fq \end{array}}$$

Second example

Assume that a table f is defined.

Assume that x is a value of the table f

Problem: find i such that $f(i) = x$.

Second example, problem domain

CONSTANTS

$$n, f, x$$

PROPERTIES

$$
\begin{aligned}
& n \in \mathbb{N}_1 \\
\wedge\ & f \in 1..n \longrightarrow S \\
\wedge\ & x \in \mathbf{ran}(f)
\end{aligned}
$$

Second example, magical event

VARIABLES

i

INVARIANT

$i \in \mathbf{dom}(f)$

computation $\widehat{=}$
  BEGIN
    $i :(i \in dom(f) \wedge f(i) = x)$
  END

The one-shot computation

Second example, refining the first model

INITIALISATION $j := 1$

computation $=$ **SELECT**
$f(j) = x$
THEN
$i := j$
END;

progress $=$ **SELECT**
$f(j) \neq x$
THEN
$j := j+1$
END

INVARIANT $\forall k.(k \in 1..j-1 \Rightarrow x \neq f(k))$

Second example, producing an algorithm

$$j := 1;$$

$$\textbf{WHILE}\ f(j) \neq x\ \textbf{DO}\ j := j+1;\ i := j\ \textbf{END}$$

```
Project status
+------------+----+-----+-----+-----+-----+-----+
| COMPONENT  | TC | POG | Obv | nPO | nUn | %Pr |
+------------+----+-----+-----+-----+-----+-----+
| mm0        | OK | OK  |   5 |   0 |   0 | 100 |
| mm1        | OK | OK  |   8 |   5 |   0 | 100 |
+------------+----+-----+-----+-----+-----+-----+
| TOTAL      | OK | OK  |  13 |   5 |   0 | 100 |
+------------+----+-----+-----+-----+-----+-----+
```

Third case study

CONSTANTS n, f

PROPERTIES $n \in \mathbb{N}_1 \wedge f \in 1..n \longrightarrow \mathbb{N}$

VARIABLES m

INVARIANT $m \in \mathbf{ran}(f)$

Third case study

INITIALISATION

$m :\in \mathbb{N}$

EVENTS

computation $=$ **BEGIN**

$m : (m \in \textbf{ran}(f) \wedge \forall i.(i \in 1..n \Rightarrow f(i) \leq m))$

END

Third case study Getting the maximum of a table

computation $\widehat{=}$

BEGIN

$m :\, (m \in \mathbf{ran}(f) \wedge \forall i.(i \in 1..n \Rightarrow f(i) \leq m))$

END

Third case study refining the first model

$$\textit{Init} = k, M := 1, f(1)$$

$\textit{computation} =$ **SELECT**
$k = n$
THEN
$m := M$
END;

$\textit{test}_1 =$ **SELECT**
$k \neq n \wedge$
$f(k+1) \leq M$
THEN
$k := k+1$
END;

$\textit{test}_2 =$ **SELECT**
$k \neq n \wedge$
$f(k+1) > M$
THEN
$k, M := k+1, f(k+1)$
END

Third case study Getting the invariant

- ☺ ?
- ☺ ?
- ☺ ?

Third case study Getting the invariant

- ☺ $M \in \mathbb{N}$
- ☺
- ☺

Third case study Getting the invariant

- ☺ $M \in \mathbb{N}$

- ☺ $\forall i.(i \in 1..k \Rightarrow f(i) \leq M)$

- ☺

Third case study Getting the invariant

- ☺ $M \in \mathbb{N}$

- ☺ $\forall i.(i \in 1..k \Rightarrow f(i) \leq M)$

- ☺ $M \in \textbf{ran}(f)$

Second example, producing an algorithm

$k, M := 1, f(1);$

WHILE $k \neq n$ **DO**

  IF $f(k+1) \leq M$ **THEN**

    $k := k+1$

  ELSE

    $k, M := k+1, f(k+1)$

  END

END

$m := M$

Exercices

- Develop the addition function
- Develop the multiplication function
- Develop a primitive recursive function
- Develop a sorting algorithm