Cours 5

Boulanger Jean-Louis

RATP

Décembre 2000

Vérification d'un modèle

Boulanger Jean-Louis

RATP

Décembre 2000

Vérification d'un modèle

- Vérification interne
- Vérification externe

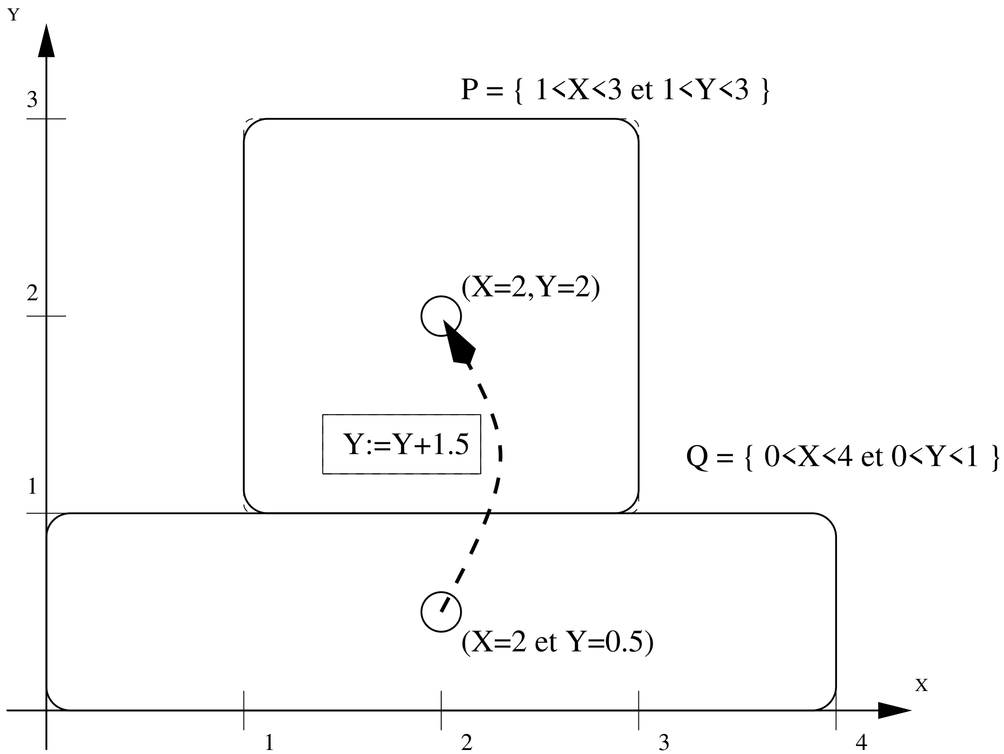
Proof Obligation

Tout lemme permettant de vérifier la correction d'une machine abstraite est appelé "Proof Obligation" (Obligation de Preuve) et noté **PO**.

Les POs sont construites sur l'idiome :

L'opération plus concrète ne fait pas ce qu'elle ne devrait pas faire.

Modèle d’exécution



Caractérisation d’une machine

MACHINE	M1
VARIABLES	V
INVARIANT	I
INITIALISATION	Init
OPERATIONS	OP1 = **PRE** P **THEN** Q **END**
END	

Consistence de la Machine Abstraite

Pour une machine de haut niveau, deux types de PO sont générés par le **GOP**

1. Le modèle n'est pas vide :

$$< Context > \Rightarrow [Init]Invariant \tag{1}$$

2. Les opérations de la machine préservent l'invariant :

$$< Context > \wedge\ Invariant \Rightarrow P \wedge [Q]Invariant \tag{2}$$

Exemple de machine

```
MACHINE         AVIONS
SETS            AVION ; IDENTIFICATEUR
VARIABLES      avion,vol_id
INVARIANT
   avion <: AVION
&  vol_id : avion --> IDENTIFICATEUR

INITIALISATION
   avion, vol_id := {}, {}
END
```

PO du à l'initialisation

```
go(Initialisation.1)
        "`Check that the invariant (avion <: AVION)
         is established by the initialisation - ref 3.3'"
         => {} <: AVION
go(Initialisation.2)
        "`Check that the invariant (vol_id: avion --> IDENTIFICATEUR)
         is established by the initialisation - ref 3.3'"
         => {}: {} +-> IDENTIFICATEUR
go(Initialisation.3)
        "`Check that the invariant (vol_id: avion --> IDENTIFICATEUR)
         is established by the initialisation - ref 3.3'"
         => dom({}) = {}
```

Exemple de machine

```
MACHINE           ARRAY (INDEX, VALEUR)
VARIABLES         array
INVARIANT         array : INDEX --> VALEUR
INITIALISATION    array :(array : INDEX --> VALEUR)
OPERATIONS
put (xx, valeur) =  PRE xx : INDEX  &  valeur : VALEUR
                    THEN
                     array := array <+ { xx |-> valeur}
                    END;
vv <-- get(xx) = PRE xx : INDEX
                    THEN vv := array(xx) END
END
```

PO du à l'opération "put"

```
 go(put.1)
        "`Check that the invariant (array: INDEX --> VALEUR)
         is preserved by the operation - ref 3.4'"
        => array<+{xx|->valeur}: INDEX +-> VALEUR
go(put.2)
        "`Check that the invariant (array: INDEX --> VALEUR)
         is preserved by the operation - ref 3.4'"
        => dom(array<+{xx|->valeur}) = INDEX
```

Caractérisation d'un Raffinement

REFINEMENT	M_1
REFINES	M
VARIABLES	V_1
INVARIANT	I_1
INITIALISATION	$Init_1$
OPERATIONS	OP1 = **PRE** P_1 **THEN** Q_1 **END**
END	

Caractérisation d’une Implémentation

IMPLEMENTATION	M_1
REFINES	M
VARIABLES	V_1
INVARIANT	I_1
INITIALISATION	$Init_1$
OPERATIONS	OP1 = **BEGIN** Q_1 **END**
END	

Preuves de raffinement

Pour chaque opération d'un raffinement, on a les deux POs suivantes :

$$< Context > \Rightarrow [Init_n] \neg\, [Init_{n-1}] \neg\, I_n \qquad (3)$$

$$< Context > \wedge\, I_1 \wedge I_2 .. \wedge I_n \wedge Q_1 \Rightarrow \qquad (4)$$

$$Q_n \wedge [[u_n := u_{n-1}]S_n] \neg\, [S_{n-1}] \neg\, (I_n \wedge u_n = u_{n-1})$$

Avec $Init_i, I_i, Q_i, S_i, u_i$ qui sont respectivement l'initialisation, l' invariant, la pré-condition de l'opération, l'action de l'opération, le paramètre formel de l'opération pour le i^{th} raffinement.

Exemple de raffinement

```
MACHINE            ensemble
VARIABLES          contenu
INVARIANT          contenu : POW(NAT)
INITIALISATION     contenu :( contenu : POW(NAT))
OPERATIONS
elt <-- choisir =
  IF contenu = {}
    THEN elt :: NAT
    ELSE elt :: contenu
  END
END
```

Exemple de raffinement

```
REFINEMENT        ensemble_1
REFINES           ensemble
VARIABLES         contenu_seq
INVARIANT         contenu_seq : seq(NAT)      &
                  contenu = ran (contenu_seq)
INITIALISATION    contenu_seq :( contenu_seq : seq(NAT))
OPERATIONS
elt <-- choisir =
    IF contenu_seq = <>
      THEN elt := 0
      ELSE elt := first (contenu_seq)
    END
END
```

PO du à l'initialisation

```
"‘Local hypotheses’" &
contenu_seq$2: seq(NAT) &
"‘Check initialisation refinement - ref 4.3, 5.4’" &
=>
ran(contenu_seq$2) <: NAT
```

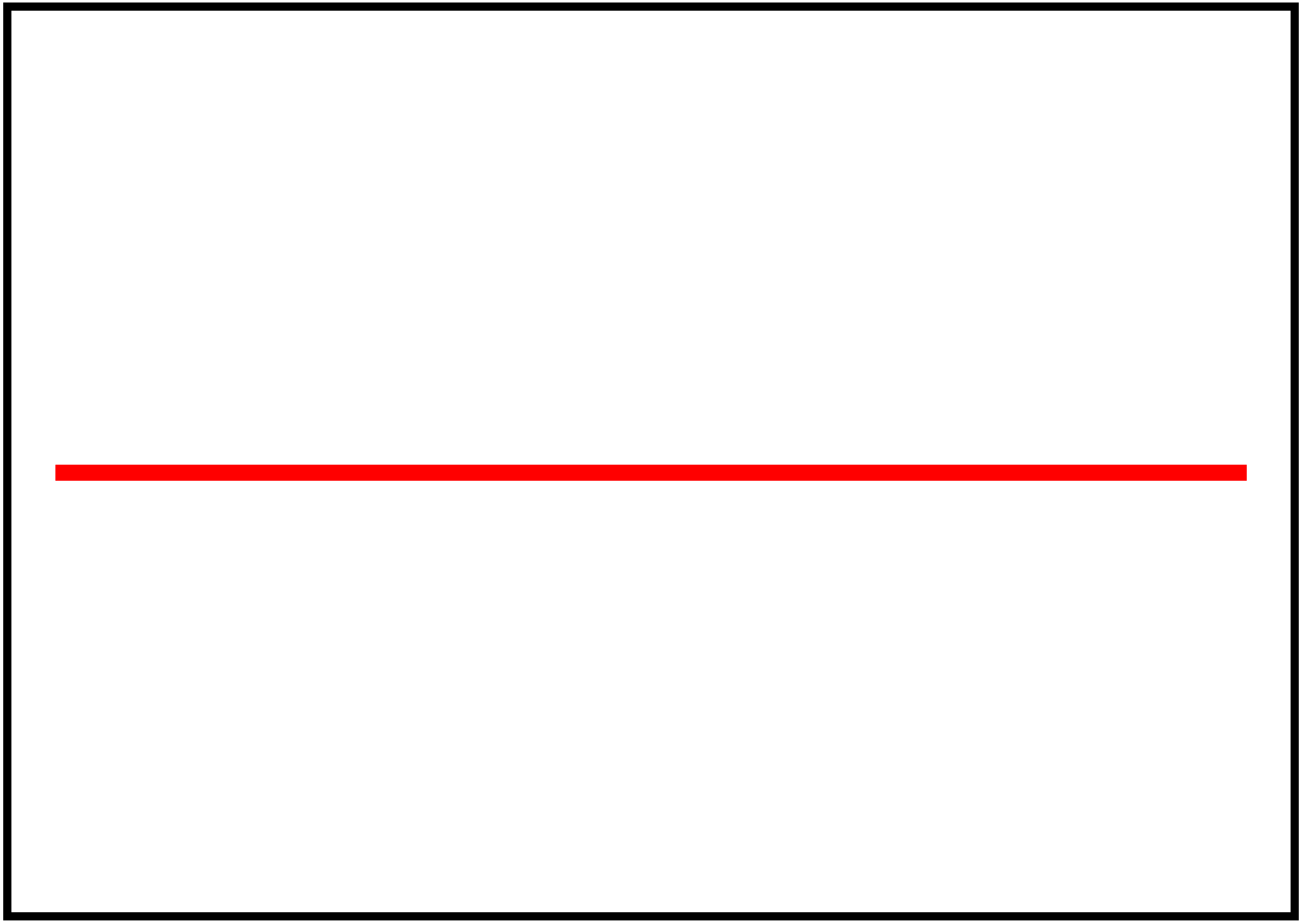
PO du à l'opération (1 ere branche du IF)

```
choisir.1
  "`Previous components invariants'" &
  contenu <: NAT &
  "`Component invariant'" &
  contenu_seq$1: seq(NAT) &
  contenu = ran(contenu_seq$1) &
  "`Local hypotheses'" &
  contenu_seq$1 = {} &
  not(contenu = {}) &
  "`Check operation refinement - ref 4.4, 5.5'" &
  =>
  0: contenu
```

PO du à l'opération (2 ème branche du IF)

```
choisir.5
  "`Previous components invariants'" &
  contenu <: NAT &
  "`Component invariant'" &
  contenu_seq$1: seq(NAT) &
  contenu = ran(contenu_seq$1) &

  "`Local hypotheses'" &
  not(contenu_seq$1 = {}) &
  not(contenu = {}) &
  "`Check that the invariant (contenu = ran(contenu_seq))
    is preserved by the operation - ref 4.4, 5.5'" &
  =>
  first(contenu_seq$1): contenu
```



Autres vérifications

Existence de

- Paramètres effectifs validant les `CONSTRAINTS` d'une machine,
- Ensembles et constantes locales validant les `PROPERTIES`,
- Valeur pour les variables qui valident l'`INVARIANT` d'une machine,
- Paramètres effectifs validant les pré-conditions d'une opération.
- Validité des boucles (Terminaison et Vérification de l'`INVARIANT`)

Validité de l'instanciation

La validation de l'existence de valeur effective pour les machines paramétrées se fait à l'instanciation. On a alors la PO 5.

$$\begin{array}{c} < Context > \\ \Rightarrow \\ [Formal := Actual](A \wedge Constraints) \end{array} \tag{5}$$

Le prédicat A introduit le fait que les ensembles passés en paramètres sont finis et non vides. `Constraints` est la clause de la partie Déclarative qui définit certaines contraintes sur les paramètres formels.

Exemple d'instanciation

```
MACHINE
        Avion(AVION, max_avion)
CONSTRAINTS
        max_avion : NAT
&       max_avion > 0
&       card(AVION) = max_avion
END
```

Exemple d'instanciation (suite)

```
MACHINE
        Instanciation_Avion
SETS
        AA = {aa1, aa2, aa3}
INCLUDES
        Avion(AA, 3)
END
```

Instanciation incorrecte

```
     Avion(AA, 8)
```

PO générée

```
AA = {aa,bb,cc}                                              &
btrue                                                        &
"`Previous components properties'"                           &
not(ran(AA) = {AA.enum}) => ran(AA) = {}                     &
AA <: NATURAL*{AA.enum}  => AA: FIN(NATURAL*{AA.enum})       &
not(AA = {})
`Check that the constraint (card(AA) = 3)
 is preserved by the parameters instanciation - ref 3.1, 4.1, 5.1'"
=> card(AA) = 3
```

Exemple d'instanciation (suite)

```
IMPLEMENTATION    Ex_Avion_n
REFINES           Ex_Avion
IMPORTS           Avion(AA,3)
OPERATIONS
main =
skip
END
```

Validité des ensembles et des constantes

La valuation d'un ensemble ou d'une constante peut être faite au niveau de la `MACHINE` ou de l'implémentation (cas des ensembles et des constantes deffërés). La PO 6 est générée au niveau implémentation afin de vérifier que les `PROPERTIES` sont respectées par les valeurs choisies :

$$< Subset\_of\_Context > \Rightarrow$$
$$[DefSET, DefVAR := Values]Properties$$

$< Subset\_of\_Context >$ est une partie du contexte qui est utilisé pour construire les `Values`.

Exemple de déclaration d'ensembles

```
MACHINE         EX_SET
SETS            set1 = { val1, val2, val3 }
;               set2
PROPERTIES
                card(set2) =5
END

IMPLEMENTATION  EX_SET_n
REFINES         EX_SET_2
VALUES          set2 = 1..5
END
```

PO générée pour la déclaration d'ensemble

```
"`Check that the property (card(set2) = 5)
  is preserved by the valuations - ref 5.2'"
=> card(1..5) = 5

"`Check that the property (set2: FIN(INTEGER))
  is preserved by the valuations - ref 5.2'"
=> 1..5: FIN(INTEGER)
```

Introduction

La validation externe porte sur deux points :

- La validation de la spécification,
- La validation des lemmes utilisateurs,
- La validation des démonstrations manuelles.