Cours 4

Boulanger Jean-Louis

RATP

Décembre 2000

De la Spécification à l'Implémentation

Boulanger Jean-Louis

RATP

Décembre 2000

Introduction

En B , on dispose de trois niveaux de détail (de la spécification jusqu'au code),

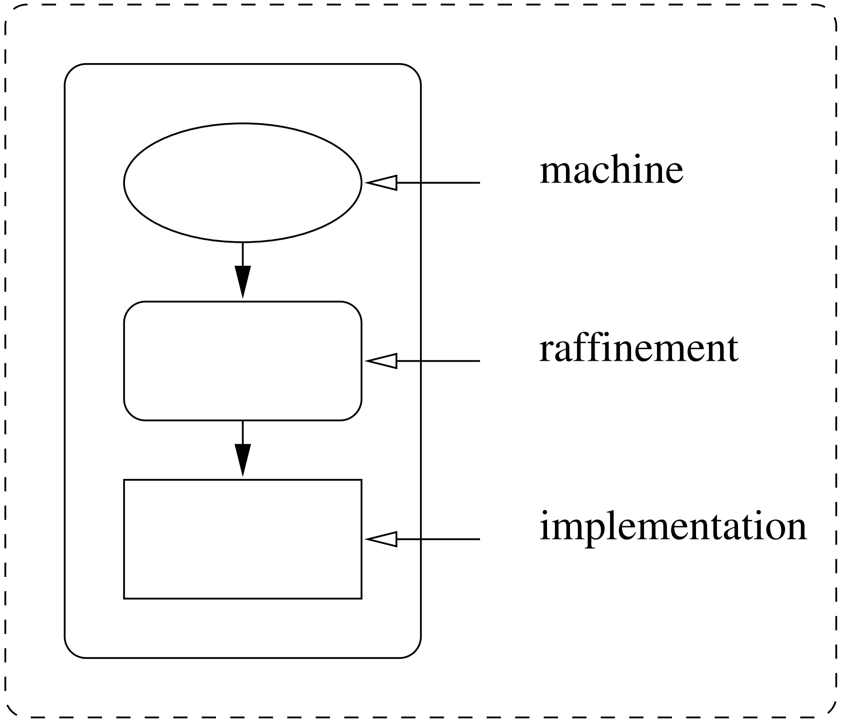
– La `MACHINE`

– Le(s) `REFINEMENT`(s)

– L' `IMPLEMENTATION`

On introduit ainsi la notion de **chaîne** de développement.

Chaîne de développement



Questions

D’où un certain nombre de questions :

– Qu’est-ce qu’une spécification en B ?

– Comment répartir l’information sur les trois niveaux ?

– Qu’est-ce qu’un développement en B ?

Hello World

```
MACHINE
    HelloWorld

OPERATIONS

Hello =
       skip

END
```

Et l’implantation du programme “Hello World”

```
IMPLEMENTATION
    HellWorld_n
REFINES
    HelloWorld
IMPORTS
    BASIC_IO
OPERATIONS
Hello =
        STRING_WRITE("Hello World")
END
```

Machine de haut niveau.

Le contenu d'une machine est en général très sobre :

- Ensemble (`SETS`) et Constante (`CONSTANTS`) globale à la machine et leurs propriétés (`PROPERTIES`),
- Variable(s) (`VARIABLES`) introduisant des `INVARIANT` importants et leurs initialisations (`INITALISATION`),
- Signature(s) (`OPERATIONS`) de toutes les opérations de la machine.

Description abstraite des opérations :

1. non séquentielle,
2. et non-déterministe

Caractérisation d'une machine

```
MACHINE                  Nom de la machine (listes de paramètres)
CONSTRAINTS              contraintes sur les paramètres
SETS                     "types" locaux à la machine
(ABSTRACT_)CONSTANTS     constantes locales
PROPERTIES               définitions d'ensembles et de constantes
(CONCRETE_)VARIABLES     variables locales
INVARIANT                propriétés des variables
INITIALISATION           description de l'état initial
OPERATIONS
 y <- op1(x) =  PRE précondition THEN substitution END;
      op2(z) =  PRE précondition THEN substitution END
END
```

Exemple de déclaration d’ensembles (1)

```
MACHINE
        couleur
SETS
        COULEUR = {ROUGE, VERT, JAUNE, BLEU, BLANC, NOIR }
END
```

Exemple de déclaration d'ensembles (2)

```
MACHINE         EX_SET
SETS            set1 = { val1, val2, val3 }
;               set2 ; set3
PROPERTIES
                card(set2) =5
END

IMPLEMENTATION  EX_SET_n
REFINES         EX_SET
VALUES          set2 = 1..5
;               set3 = { 1, 2 }
END
```

CONSTANTE

`ABSTRACT_CONSTANTS`	Constantes abstraites destinées a disparaitre
`CONSTANTS`	Constantes concrètes

Exemple

```
MACHINE
            new_nat
ABSTRACT_CONSTANTS
            nat_plus
,           nat
PROPERTIES
            nat      = { nn | (nn : INTEGER) & (nn >= 0)}
&           nat_plus = { nn | (nn : NAT)     & (nn >  0)}
END
```

Exemple d’une relation de tri

```
MACHINE        Sorting
CONSTANTS      sort_of
PROPERTIES
  sort_of : FIN(NAT) --> seq(NAT)
& !aa.(aa: FIN(NAT) =>
      (  ran(sort_of(aa)) = aa
       & !(ii,jj).((  ii : dom(sort_of(aa))
                    & jj : dom(sort_of(aa))
                    & ii<jj )=>
        (sort_of(aa))(ii)<(sort_of(aa))(jj)
      )   )            )
END
```

Exemple d'utilisation de variables

```
MACHINE     EX_VAR
VARIABLES   aa , bb, cc, set,ff
INVARIANT
            aa, bb : NAT* NAT
&           cc     : NAT
&           cc     = aa + bb
&           set    : POW(NAT)
&           aa     : set
&           ff     : INTEGER --> INTEGER
INITIALISATION
            aa, bb, cc , set := 1, 2, 3, {1,2,3}
||          ff := %xx.((xx: INTEGER)| (2*xx) )
END
```

Entête des opérations

		nom_operation		=	....
		nom_operation	(p1, p2, ...)	=	....
r1,r2,..	<–	nom_operation		=	....
r1,r2,..	<–	nom_operation	(p1, p2, ...)	=	....

Exemple d'une machine

```
MACHINE     OUTILS
OPERATIONS
uu,vv,ww<-- Ordonne_3_NAT(xx,yy,zz) =
PRE     xx,yy,zz :  NAT * NAT* NAT THEN
  ANY ua,va,wa
  WHERE
    ua : NAT          & va : NAT          & wa : NAT &
    ua : {xx,yy,zz} & va : {xx,yy,zz} & wa : {xx,yy,zz} &
    ua <= va          & va <= wa
  THEN uu,vv,ww := ua,va,wa
  END
END
END
```

Exemple simple de machine abstraite

MACHINE LittleExample

VARIABLES y

INVARIANT $y \in \mathbb{F}(\mathbb{N}_1)$

INITIALISATION $y := \varnothing$

OPERATIONS

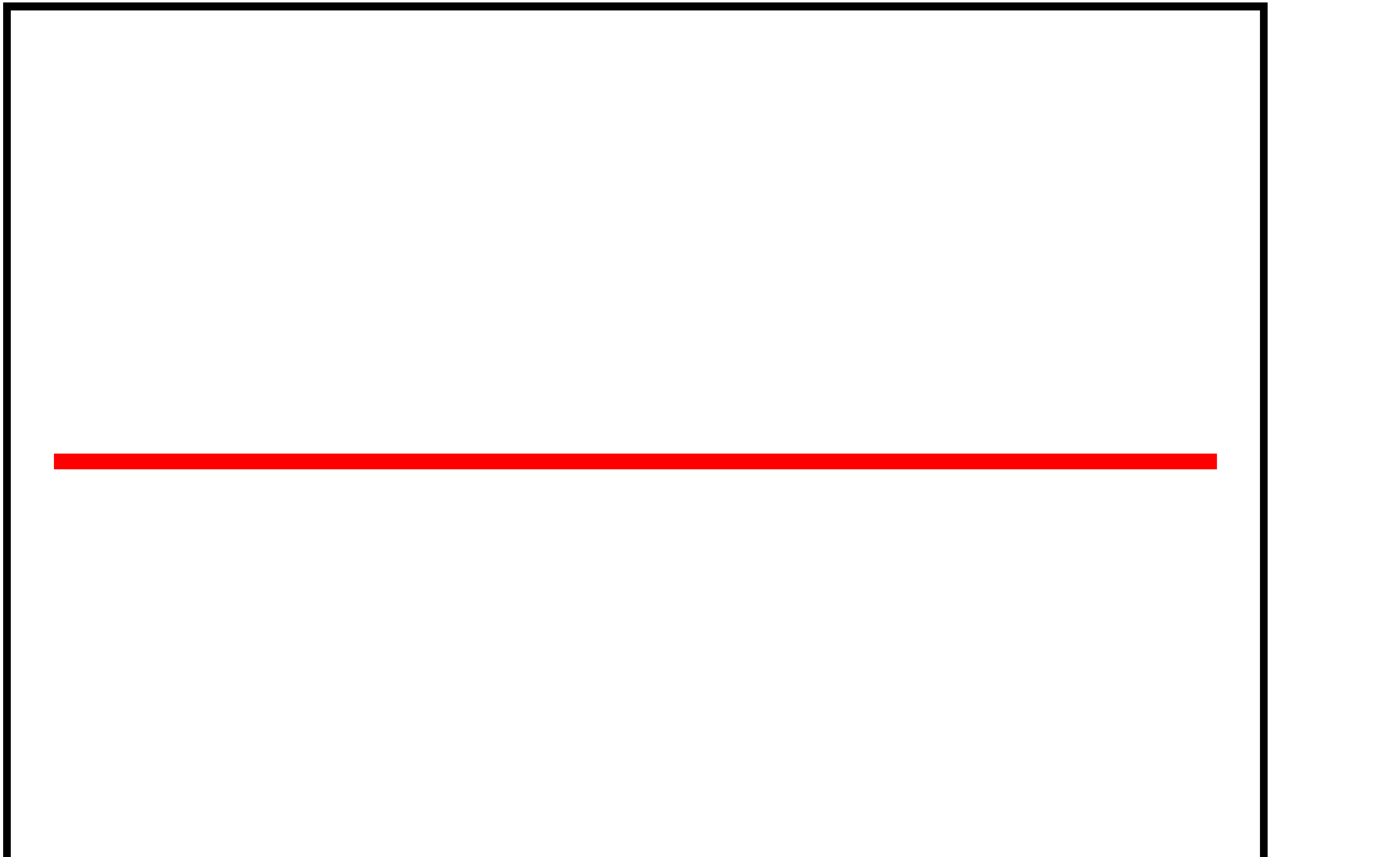
enter (n) = **PRE** $n \in \mathbb{N}_1$ **THEN** $y := y \cup \{n\}$ **END** ;

m <– **maximum** = **PRE** $y \neq \varnothing$ **THEN** $m := max(y)$ **END**

END

Exemple de machine

```
MACHINE          SquareRoot_0
DEFINITIONS      square (x)        ==        x*x
OPERATIONS
sqrt <-- SquareRoot (xx) =
PRE          xx : NAT
THEN
      ANY
              yy
      WHERE
              (yy : NAT) & (square(yy) <= xx) & (yy <= square(xx+1))
      THEN
              sqrt := yy
      END
END
```



Raffinement

Le raffinement a été défini par C. Morgan

C. Morgan

Deriving programs from specifications

Prentice Hall International, 1990

Idée : Le raffinement consiste à tranformer un modéle *abstrait* en un autre fonctionnellement équivalent mais plus concret.

Raffinement

Une machine abstraite de haut niveau (`MACHINE`) peut être raffinée autant de fois que voulu (0,1,...).

Les éventuels raffinement doivent introduire plus de détails tout en restant le plus possible

1. non séquentiel,
2. et non-déterministe

Raffinement : les principes

Le « raffinement » d'une spécification (abstraite) s'obtient par :

1. Réduction du non-déterminisme
2. Concrétisation des données
   (par changement de variables)

Techniques de raffinement

Il y a trois chemins à emprunter pour raffiner :

- éliminer le pseudo-code non exécutable (précondition, choix,...)
- introduire des structures de contrôle des langages de programmation (séquence, boucle, ...)
- transformer les structures (de données) mathématiques (fonctions relations, ensembles, suites et arbres) en des structures qui peuvent être programmables (variables, tableaux, fichiers, ...).

Caractérisation d'un Raffinement

```
REFINEMENT               Nom du raffinement (listes de paramètres)
REFINES                  Nom de la machine
SETS                     "types" locaux à la machine
(ABSTRACT_)CONSTANTS     constantes locales
PROPERTIES               définitions d'ensembles et de constantes
(CONCRETE_)VARIABLES     variables locales
INVARIANT                propriétés des variables
INITIALISATION           description de l'état initial
OPERATIONS
 y <- op1(x) =  PRE précondition THEN substitution END;
      op2(z) =  PRE précondition THEN substitution END
END
```

Exemple simple de machine abstraite

MACHINE LittleExample

VARIABLES y

INVARIANT $y \in \mathbb{F}(\mathbb{N}_1)$

INITIALISATION $y := \varnothing$

OPERATIONS

enter (n) = **PRE** $n \in \mathbb{N}_1$ **THEN** $y := y \cup \{n\}$ **END** ;

m <– **maximum** = **PRE** $y \neq \varnothing$ **THEN** $m := max(y)$ **END**

END

Et son raffinement

REFINEMENT LittleExample1

REFINES LittleExample

VARIABLES z

INVARIANT $z \in NAT \wedge z = max(y \cup \{0\})$

INITIALISATION z := 0

OPERATIONS

enter(n) = **PRE** $n \in \mathbb{N}_1$ **THEN** z :=$max(\{z, n\})$ **END** ;

m<– **maximum** = **PRE** $z \neq 0$ **THEN** m :=z **END**

END

Second exemple de raffinement (1)

```
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
```

Second exemple de raffinement (2)

```
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
```

Troisième exemples de raffinement (1)

```
MACHINE            Search_0
DEFINITIONS        NN == (10)
VARIABLES          AA
INVARIANT          AA : 1..NN --> NAT
INITIALISATION     AA :( AA       : 1.. NN --> NAT )
OPERATIONS
yy <-- Search (nn) =
PRE
    nn : NAT
THEN
    yy :(  yy : 0 .. NN
          &  (((yy = 0     )  & (nn /: ran(AA)) ) or ((nn = AA(yy)
        )
END
END
```

Troisième exemples de raffinement (2)

```
REFINEMENT          Search_1
REFINES             Search_0
VARIABLES           AA
INVARIANT           AA : 1..NN --> NAT
INITIALISATION      AA :( AA : 1.. NN --> NAT )
DEFINITIONS         NN      == (10)
OPERATIONS
yy <-- Search (nn) =
        SELECT      (nn /: ran(AA))
        THEN         yy := 0
        ELSE         yy :( yy : 1..NN & nn = AA(yy) )
        END
END
```

Caractérisation d'un Raffinement

```
IMPLEMENTATION      Nom de l'implantaton (listes de paramètres)
REFINES             Nom de la machine
SETS                "types" locaux à la machine
CONSTANTS           constantes locales
PROPERTIES          définitions d'ensembles et de constantes
VALUES              Valuation des ensembles et des constantes
CONCRETE_VARIABLES  variables locales
INVARIANT           propriétés des variables
INITIALISATION      description de l'état initial
OPERATIONS
 y <- op1(x) =  PRE précondition THEN substitution END;
      op2(z) =  PRE précondition THEN substitution END
END
```

Implantation

L'`IMPLEMENTATION` est le dernier raffinement et doit être purement algorithmique.

Une implantation doit faire disparaître

- toute Simultanéité (||) au profit du séquentiel (`;`),
- tout non déterminisme au profit du déterminisme,
- toute affectation (`:=`) au profit du retour d'opérations (`<--`),
- tout opérateur au profit d'opération de base,
- toute variable locale au profit d'une variable encapsulée.

Exemple d'imlantation

IMPLANTATION LittleExample2

REFINES LittleExample1

VARIABLES z

INVARIANT $z : NAT$

INITIALISATION z := 0

OPERATIONS

enter(n) = **IF** $n >= z$ **THEN** $z := n$ **END** ;

m<– **maximum** = $m := z$

END

```
IMPLEMENTATION    outils_n
REFINES           outils_0
OPERATIONS
uu,vv,ww<-- Ordonne_3_NAT(xx,yy,zz) =
BEGIN
  IF (xx <= yy)
    THEN IF (yy <= zz)
         THEN BEGIN uu := xx; vv := yy; ww := zz  END
         ELSIF (zz <= xx)
         THEN BEGIN uu := zz; vv := xx; ww := yy  END
         ELSE BEGIN uu := xx; vv := zz; ww := yy  END
         END
         ELSIF (zz <= yy)
         THEN BEGIN uu := zz; vv := yy; ww := xx  END
         ELSIF (xx <= zz)
         THEN BEGIN uu := yy; vv := xx; ww := zz  END
         ELSE BEGIN uu := yy; vv := zz; ww := xx  END
```

```
        END
    END
END
```

Principes de la modularisation

La modularisation consiste a permettre le partage d’information entre *machines abstraites*.

Ce partage d’information est réalisé au travers de liens de composition de *machines abstraites*.

Ces liens sont répartis en trois familles :
- des liens de spécification : **USES** et **INCLUDES**,
- des liens de modélisation : **SEES** et **IMPORTS**,
- des liens d’extension : **EXTENDS** et **PROMOTES**.

On n’oublit pas le lien de *raffinement* : **REFINE**

Le lien de “Raffinement”

Exemple d'utilisation

M1

$\uparrow$ REFINES

M2

Une seule machine abstraite est raffinée par une autre.

Accessibilité de la clause “REFINES”

	M.PROPERTIES	M.INVARIANTS	M.OPERATIONS
M1.PARAMETERS			
M1.SETS	Oui	Oui	Oui
M1.CONSTANTS	Oui	Oui	Oui
M1.VARIABLES		Oui	
M1.OPERATIONS			

Il faut le lire : Les **PROPERTIES** de M peuvent contenir des constantes de M1 et des types (**SETS**) de $M1$, ...

Les liens de modélisation

Description

La clause `SEES`

- peut être utilisée à tous les niveaux,
- permet un accès en lecture a des données,
- doit être reportée dans tous les raffinements de la machine en cours.

Rq : Une machine ne doit être importée qu'une seule fois.

Exemple d'utilisation

$$\begin{array}{cccccc}
\mathrm{M1} & & \mathrm{M2} & & \mathrm{M3} & \\
 & \nwarrow & \uparrow & \nearrow & & \text{SEES} \\
 & & \mathrm{M} & & &
\end{array}$$

Les variables des machines $M1$, $M2$ et $M3$ doivent être distinctes.

Accessibilité

	M.PROPERTIES	M.INVARIANTS	M.OPERATIONS
M1.PARAMETERS			
M1.SETS	Oui	Oui	Oui
M1.CONSTANTS	Oui	Oui	Oui
M1.VARIABLES			Lecture seule
M1.OPERATIONS			Oui

Il faut le lire : Les **PROPERTIES** de M peuvent contenir des constantes de M1 et des types (**SETS**) de $M1$, ...

Exemple

```
MACHINE
        couleur
SETS
        COULEUR = {ROUGE, VERT, JAUNE, BLEU, BLANC, NOIR }
VARIABLES
        une_couleur
INVARIANT
        une_couleur : COULEUR
INITIALISATION
        une_couleur :( une_couleur : COULEUR)
OPERATIONS
PRINT (cc) = PRE cc : COULEUR THEN skip END
END
```

Exemple

```
MACHINE       use_couleur
SEES          couleur
OPERATIONS
        New_Print(cc) =
        PRE
                cc : COULEUR
        THEN
                PRINT(cc)
        END
END
```

Exemple

```
MACHINE
fact

CONSTANTS
        facto

PROPERTIES
    facto : NAT --> NAT
&   facto(0) = 1
&   (!nn.(nn:NAT & nn>0  => facto(nn) = nn * facto(nn-1)))

END
```

Exemple

```
MACHINE
        use_fact
SEES
        fact
OPERATIONS

ff <-- Print_fact(nn) =
   PRE
       nn : NAT
   THEN
       ff := facto(nn)
   END
END
```

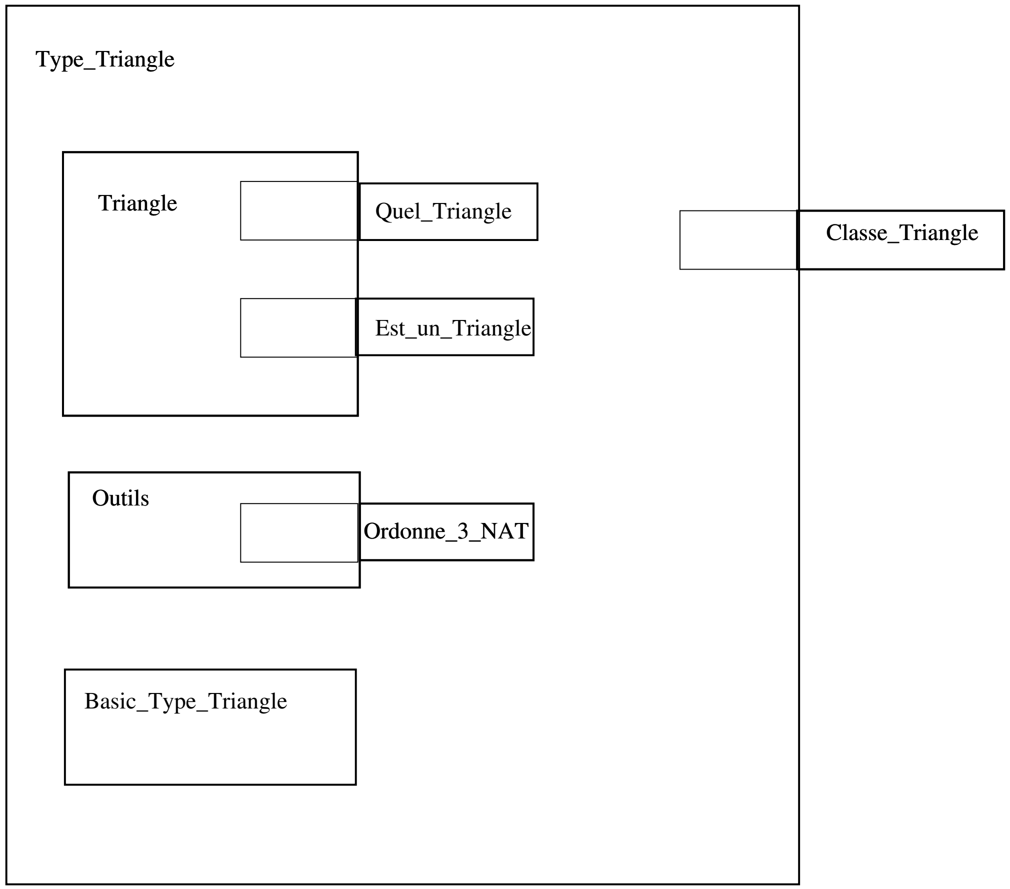
Description

La clause `IMPORTS`

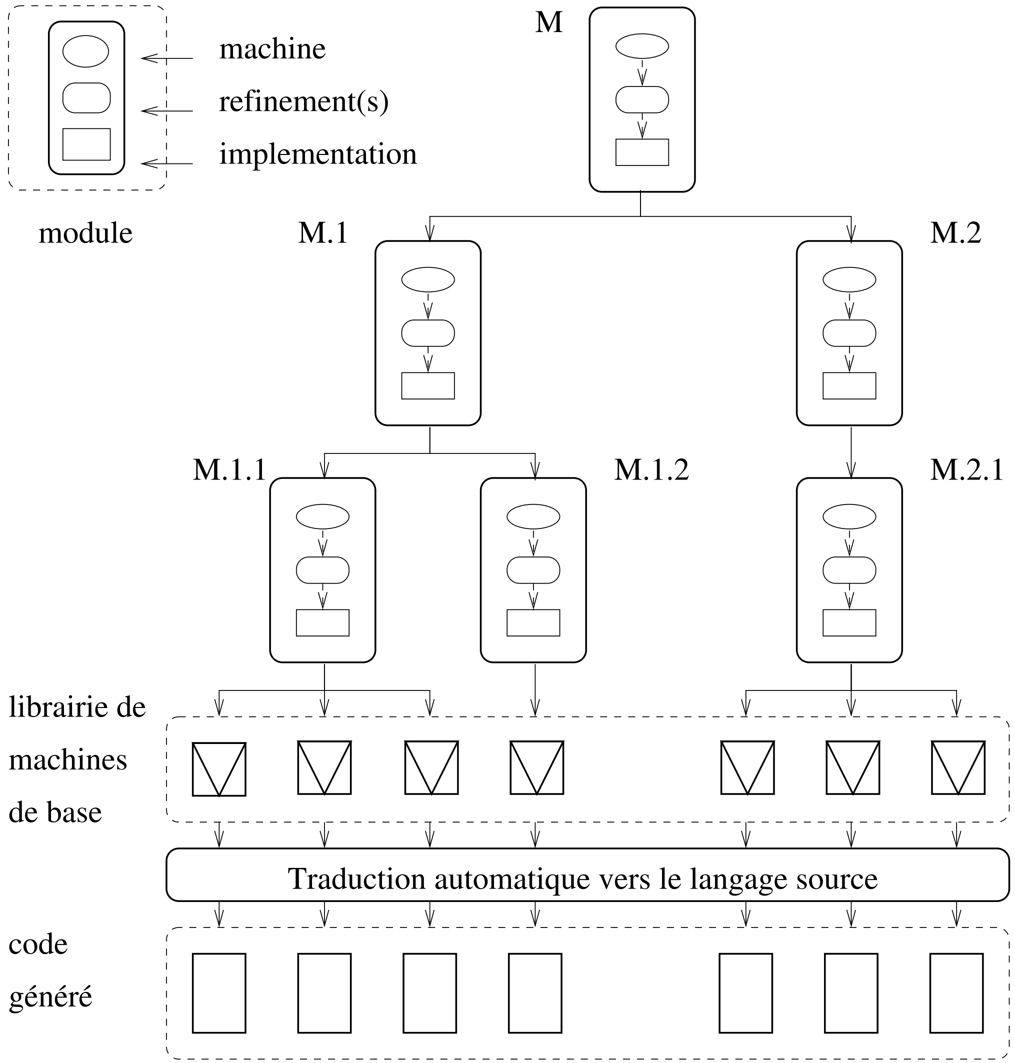
- est réservées aux implantation,
- est la seule clause qui permet de créer une instance,
- permet de structurer le projet (couche).

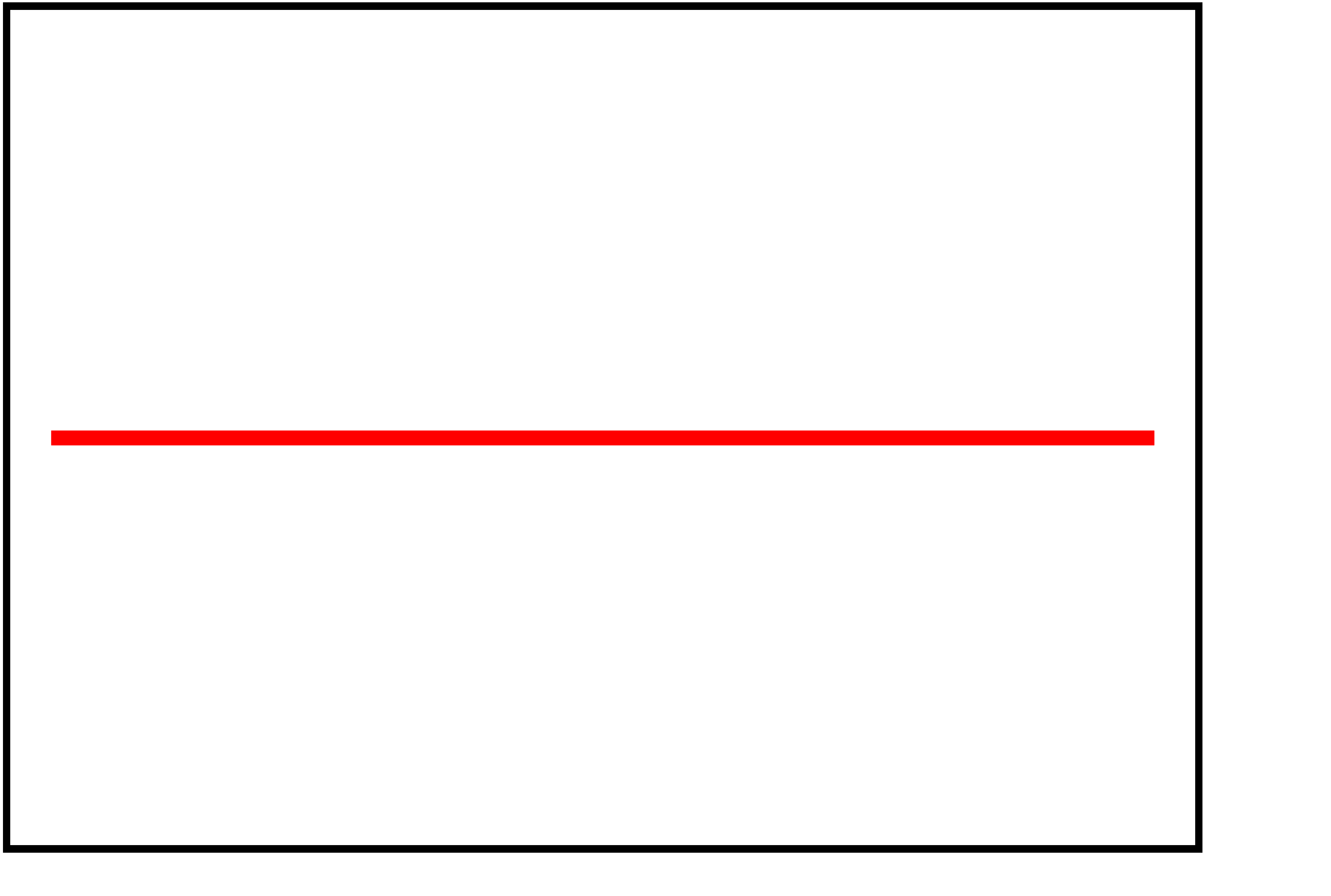
Rq : Une machine ne doit être importée qu'une seule fois.

Encapsulation



Structuration de projet


machine
refinement(s)
implementation
module
M
M.1
M.2
M.1.1
M.1.2
M.2.1
librairie de
machines
de base
Traduction automatique vers le langage source
code
généré



Exemple d'utilisation

$$
\begin{array}{ccccccc}
\mathrm{M1} & & \mathrm{M2} & & \mathrm{M3} & \quad & \text{Machines} \\
 & \nwarrow & \uparrow & \nearrow & & & \text{IMPORTS} \\
 & & \mathrm{M} & & & & \text{Implantation}
\end{array}
$$

Les variables des machines $M1$, $M2$ et $M3$ doivent être distinctes.

Accessibilité

	M.VALUE	M.PROPERTIES	M.INVARIANTS	M.OPERATIONS
M1.PARAMETERS				
M1.SETS	Oui	Oui	Oui	Oui
M1.CONSTANTS	Oui	Oui	Oui	Oui
M1.VARIABLES			Oui	Lecture seule
M1.OPERATIONS				Oui

Il faut le lire : Les **PROPERTIES** de M peuvent contenir des constantes de M1 et des types (**SETS**) de $M1$, ...

Les liens de spécification

Exemple d'utilisation

$$\begin{array}{cccccc} \mathrm{M1} & & \mathrm{M2} & & \mathrm{M3} & \\ & \nwarrow & \uparrow & \nearrow & & \text{INCLUDES} \\ & & \mathrm{M} & & & \end{array}$$

Les variables des machines $M1$, $M2$ et $M3$ doivent être distinctes.

Accessibilité

	M.PROPERTIES	M.INVARIANTS	M.OPERATIONS
M1.PARAMETERS			
M1.SETS	Oui	Oui	Oui
M1.CONSTANTS	Oui	Oui	Oui
M1.VARIABLES		Oui	Lecture seule
M1.OPERATIONS			Oui

Il faut le lire : Les **PROPERTIES** de M peuvent contenir des constantes de M1 et des types (**SETS**) de $M1$, ...

Quelque remarques

- L'inclusion est transitive.
- Quand on inclut plusieures fois une même machine ; on doit renommer en faisant précéder la machine d'un nom et d'un point.

Exemple d'utilisation

$$
\begin{array}{ccccccc}
\mathrm{M1} & \xrightarrow{\mathit{USES}} & \mathrm{M2} & \xleftarrow{\mathit{USES}} & \mathrm{M3} & & \\
 & \nwarrow & \uparrow & \nearrow & & & \mathrm{INCLUDES} \\
 & & \mathrm{M} & & & &
\end{array}
$$

Accessibilité

	M.PROPERTIES	M.INVARIANT	M.OPERATIONS
M1.PARAMETRES		Oui	Oui
M1.SETS	Oui	Oui	Oui
M1.CONSTANTS	Oui	Oui	Oui
M1.VARIABLES		Oui	Lecture seule

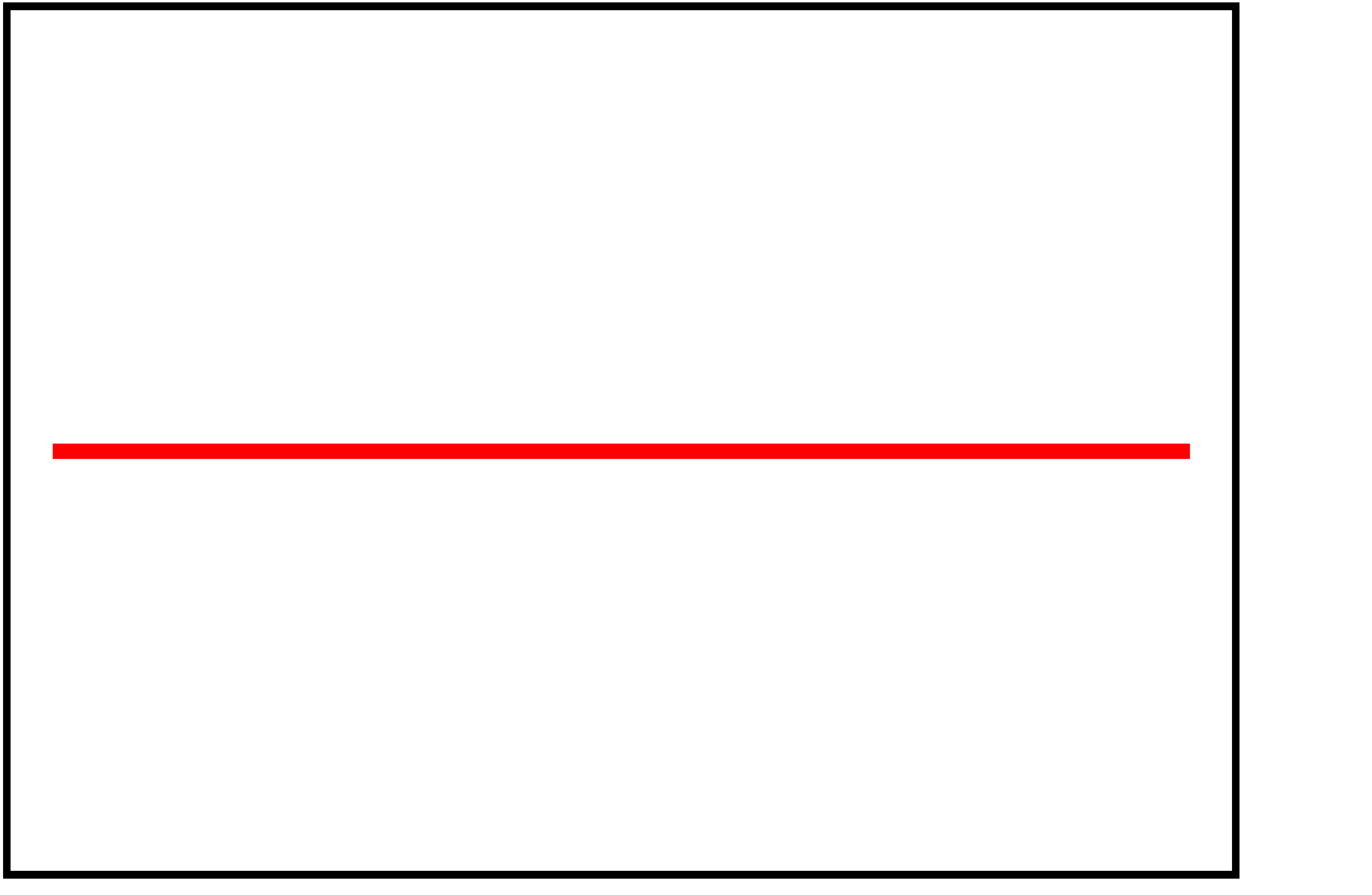
Il faut le lire : Les **PROPERTIES** de M peuvent contenir des constantes de M1 et des types (**SETS**) de $M1$, ...

Exemple

```
MACHINE
        couleur
SETS
        COULEUR = {ROUGE, VERT, JAUNE, BLEU, BLANC, NOIR }
VARIABLES
        une_couleur
INVARIANT
        une_couleur : COULEUR
INITIALISATION
        une_couleur :( une_couleur : COULEUR)
OPERATIONS
PRINT (cc) = PRE cc : COULEUR THEN skip END
END
```

Exemple

```
MACHINE
        use_couleur_bis
USES
        couleur
VARIABLES
        c1
INVARIANT
        c1 : COULEUR
INITIALISATION
        c1 := une_couleur
OPERATIONS
cc <-- Get =
             cc := une_couleur
END
```



Exemple

```
MACHINE           Scalaire(VALEUR)
CONSTRAINTS       VALEUR : NAT
VARIABLES         var
INVARIANT         var : VALEUR
INITIALIZATION    var := 0
OPERATIONS
  set(vv) =
     PRE vv : VALEUR THEN var := vv END;
  vv <-- get =
     vv := var
END
```

Exemple

```
MACHINE
  Swap(VALEUR)
EXTEND
  xx.Scalaire(VALEUR), yy.Scalaire(VALEUR)
OPERATIONS
  swap = BEGIN
           xx.set(yy.var) || yy.set(xx.var)
         END
END
```