Raffinements

Notion de raffinement

- L'objectif de la méthode B est le passage progressif de l'abstrait (le besoin, le « quoi ») au concret (le « comment »), précisant progressivement les choses tout en restant dans un cadre formel.
- C'est l'objet du processus de raffinement (ou raffinage « refinement »).
- Une MACHINE (et ses substitutions, abstraites, non déterministes etc.) s'affine progressivement (un ou plusieurs REFINEMENTs) jusqu'à une IMPLEMENTATION (quasiment du code, transcodage automatique en code source).
- Des POs de raffinement ont pour objectif de montrer que les raffinements sont compatibles avec (i.e. ne font que préciser) les machines.

Notion de raffinement

Une substitution généralisée T raffine une substitution S si elle est compatible avec (ne fait pas, ce que ne ferait pas) S en n'apportant que des précisions :

- Réduction du non déterminisme :
  x:=1 raffine x:=1 [] x:=2
- Affaiblissement des préconditions :
  [x>1|x:=x-1] raffine [x>2|x:=x-1]

Notion de raffinement

- L'utilisateur peut utiliser T (substitution dite plus concrète) à la place de S (substitution dite plus abstraite) sans remarquer de différence :
  - Les valeurs de variables obtenues par la substitution concrète font partie de celles possibles avec la substitution abstraite (en revanche il se peut que certaines valeurs abstraites ne correspondent à aucune valeur concrète).
  - Si la précondition abstraite est respectée, la précondition concrète (plus faible) l'est également (en revanche la précondition concrète peut être respectée sans que la précondition abstraite le soit).

Notion de raffinement

5 Formellement, pour tout prédicat P :

- $[x:=1 \; [] \; x:=2]P \Leftrightarrow [x:=1]P \wedge [x:=2]P$ (WP abstraite)
- $[x:=1]P$ (WP concrète)
- $[x:=1]P \wedge [x:=2]P \Rightarrow [x:=1]P$
  (WP abstraite $\Rightarrow$ WP concrète)

Donc tout prédicat établi par la substitution abstraite l'est également par la substitution concrète

- $[x>2 | x:=x-1] \; P \Leftrightarrow x>2 \wedge [x:=x-1] \; P$ (WP abstraite)
- $[x>1 | x:=x-1] \; P \Leftrightarrow x>1 \wedge [x:=x-1] \; P$ (WP concrète)
- $x>2 \wedge [x:=x-1] \; P \Rightarrow x>1 \wedge [x:=x-1] \; P$
  (WP abstraite $\Rightarrow$ WP concrète)

Donc tout prédicat établi par la substitution abstraite l'est également par la substitution concrète

Raffinement de machine

- Les raffinements de machines
  - Manipulent leurs propres variables
  - Ont des opérations qui doivent avoir la même signature (même paramètres, même type de retour) que celles de la machine raffinée
  - Les variables du raffinement sont liées à celles de la machine raffinée, par une partie de l'invariant du raffinement, appelée invariant de liaison (ou invariant de collage)

Les POs de raffinement ont pour objectif de prouver que le comportement du raffinement est compatible avec celui de la machine (initialisation et toutes opérations compatibles compte tenu de la liaison des variables).

Exemple de raffinement

Exemple

```
MACHINE Ex3
VARIABLES xx
INVARIANT xx : 0..4
INITIALISATION
CHOICE
        xx:=1 OR
        xx:=2 OR
        xx:= 3
END /*CHOICE*/
END /*MACHINE*/
```

```
REFINEMENT R_Ex3
REFINES Ex3
VARIABLES yy
INVARIANT yy=xx /*Liaison*/
INITIALISATION
CHOICE
        yy:=2 OR
        yy:=3
END /*CHOICE*/
END /*MACHINE*/
```

Exemple de raffinement : preuves

8 Le processus de preuve commence par la preuve locale à la machine :

MACHINE Ex3

VARIABLES xx

INVARIANT xx : 0..4

INITIALISATION

CHOICE

xx:=1 OR

xx:=2 OR

xx:= 3

END /*CHOICE*/

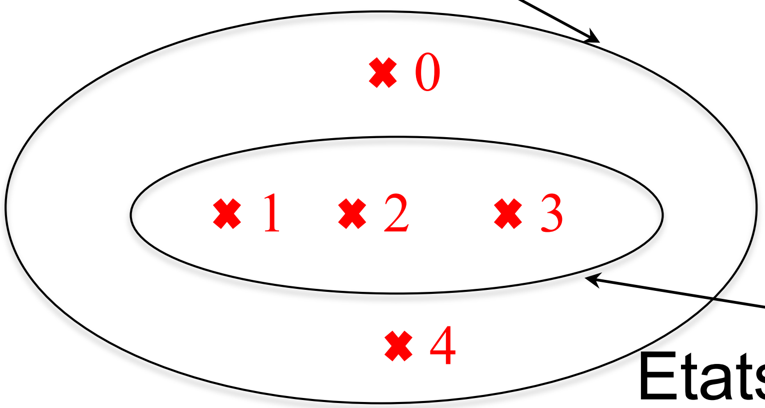
END /*MACHINE*/

PO d'initialisation :

[xx:=1[]xx:=2[]xx:=3](xx : 0..4)

$1 \in 0..4 \wedge 2 \in 0..4 \wedge 3 \in 0..4$

Etats respectant l'invariant



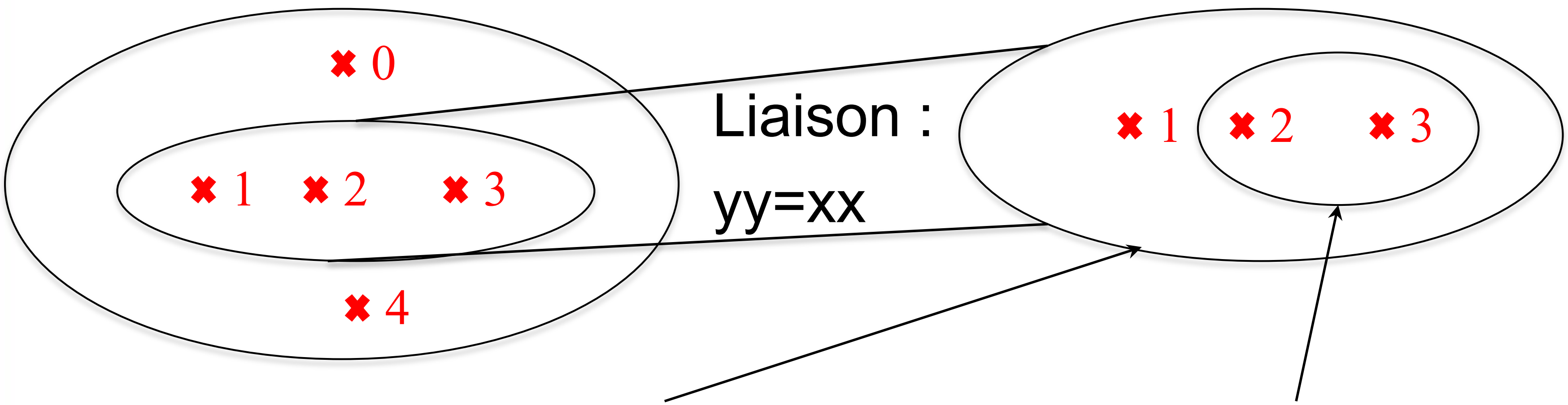
Etats possibles après initialisation

Exemple de raffinement : preuves

9 La preuve de raffinement consiste à prouver que l'initialisation du raffinement est compatible avec l'initialisation abstraite, compte tenu de l'invariant de liaison.

Machine : variable xx

Raffinement : variable yy



Etats du raffinement compatibles avec la liaison

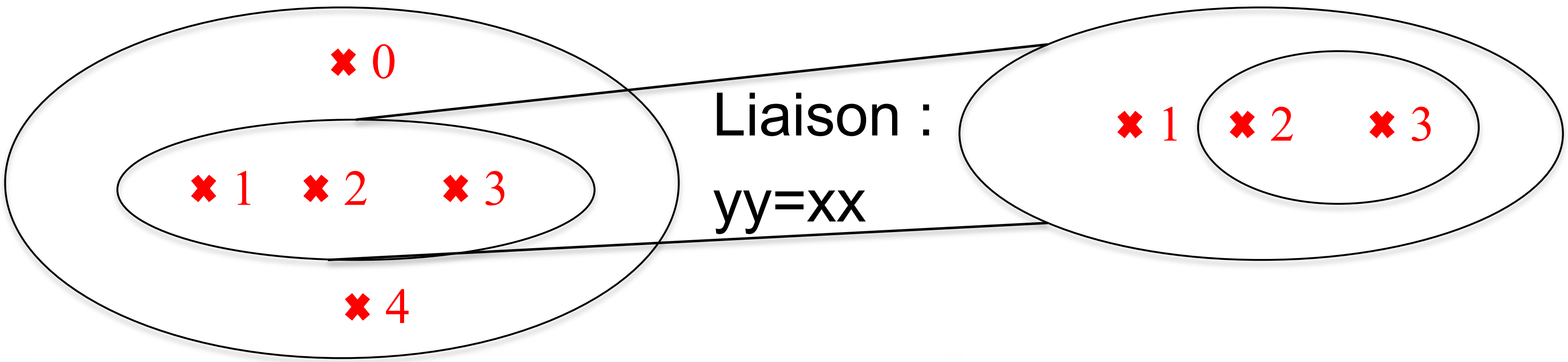
Etats possibles après initialisation du raffinement

Exemple de raffinement : preuves

10 La preuve d'initialisation de raffinement exprime formellement le fait le raffinement ne fait pas ce que ne ferait pas la machine, à savoir que l'initialisation concrète (du raffinement) ne viole pas le lien avec l'initialisation abstraite (de la machine), soit :

L'initialisation concrète établit qu'il est faux que l'initialisation abstraite établisse que l'invariant de liaison est faux. Ce qui s'écrit :

[yy:=2[]yy:=3]¬([xx:=1[]xx:=2[]xx:=3]¬yy=xx)



Exemple de raffinement : preuves

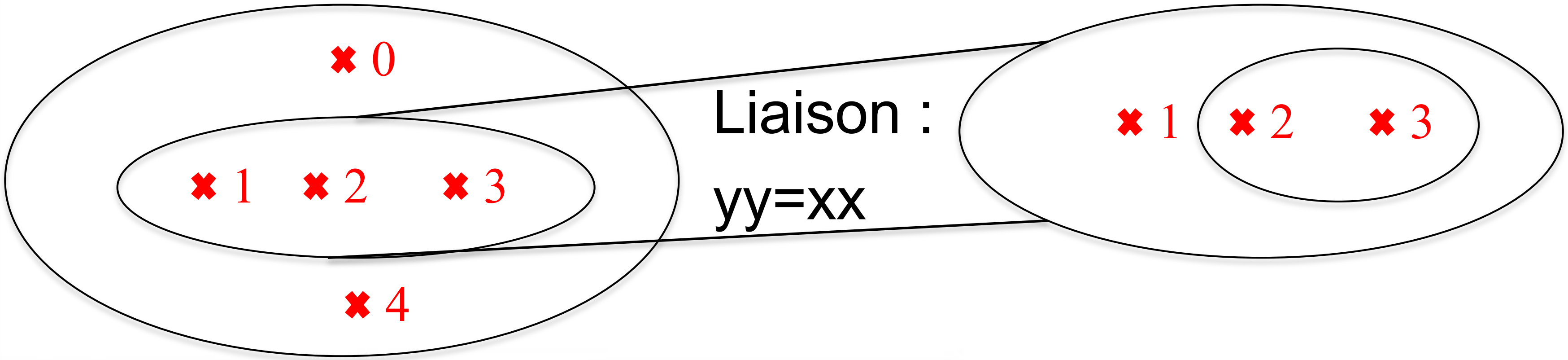
11 L'expression formelle précédente traduit bien le fait que l'ensemble des valeurs après initialisation concrète est inclus dans l'ensemble des valeurs liées à l'initialisation abstraite. En effet :

[yy:=2[]yy:=3]¬([xx:=1[]xx:=2[]xx:=3]¬yy=xx)

[yy:=2[]yy:=3]¬(¬yy=1∧¬yy=2∧¬yy=3)

[yy:=2[]yy:=3](yy=1∨yy=2∨yy=3)

(2=1∨2=2∨2=3)∧(3=1∨3=2∨3=3)



Exemple de raffinement : preuves

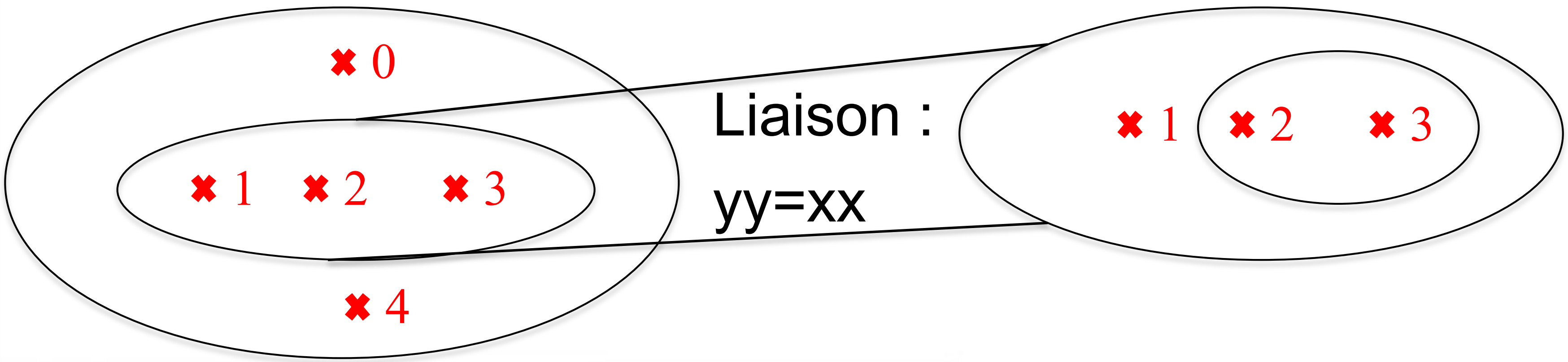
12 Attention, dire « ne fait pas ce que ne ferait pas » n'est pas la même chose que dire « fait ce que ferait » : on ne peut pas simplifier la double négation, sauf en cas de substitutions simples :

$[yy:=2 [] yy:=3]([xx:=1 [] xx:=2 [] xx:=3] yy=xx)$

$[yy:=2 [] yy:=3](yy=1 \wedge yy=2 \wedge yy=3)$

$(2=1 \wedge 2=2 \wedge 2=3) \wedge (3=1 \wedge 3=2 \wedge 3=3)$

A l'évidence faux…



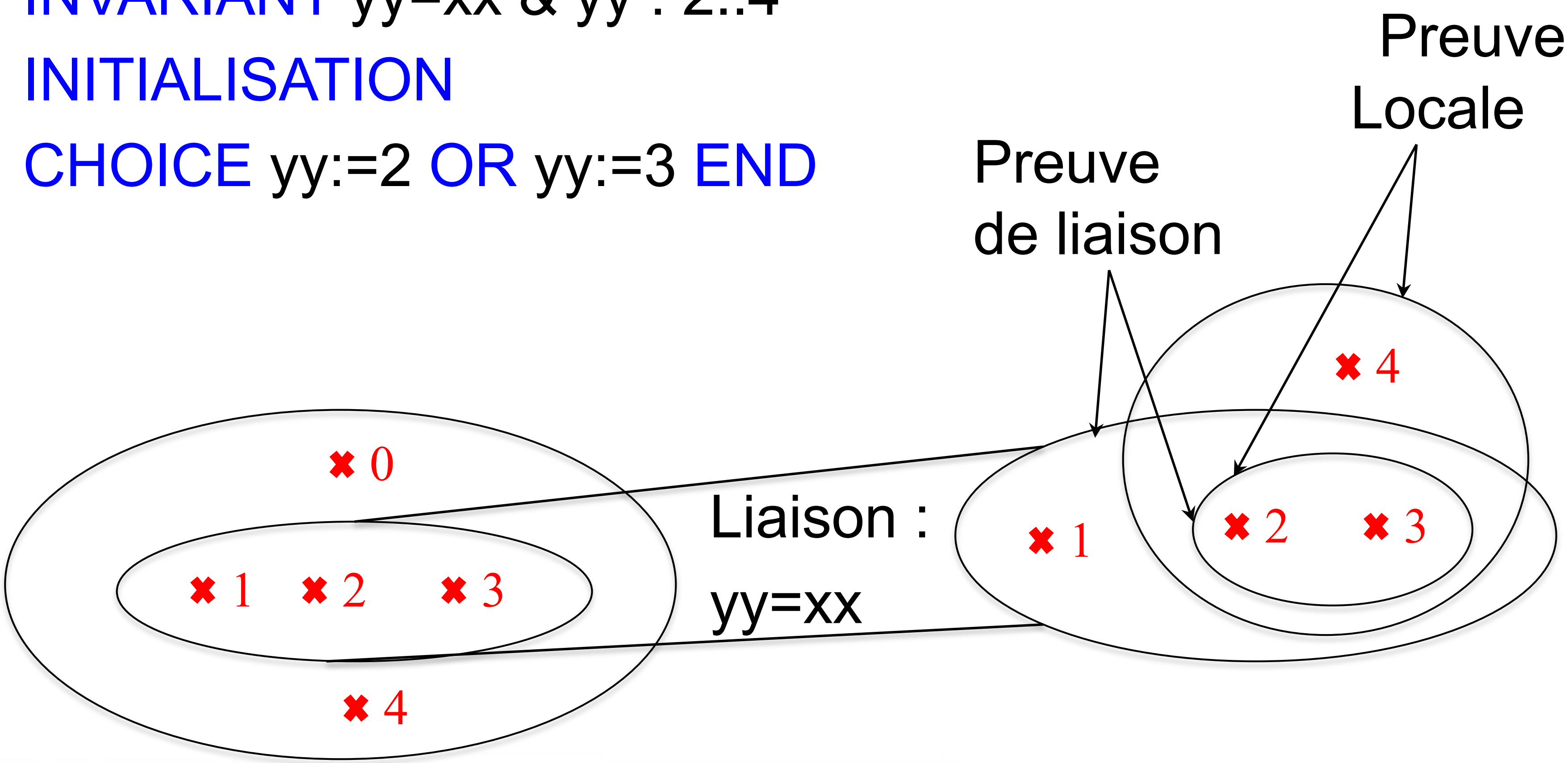
Exemple de raffinement : preuves

Il peut aussi y avoir des preuves locales à un raffinement dans le cas où l'invariant raffiné ne se limite pas à l'invariant de liaison.

INVARIANT yy=xx & yy : 2..4

INITIALISATION

CHOICE yy:=2 OR yy:=3 END



Obligations de preuve de raffinement

Considérons

- Une machine
  - D'invariant INV_{Mch}
  - De substitution d'initialisation $INIT_{Mch}$
- Un raffinement
  - D'invariant INV_{Ref} (dont invariant de liaison)
  - De substitution d'initialisation $INIT_{Ref}$

Alors l'obligation de preuve de raffinement d'initialisation est :

$$[INIT_{Ref}]\neg([INIT_{Mch}]\neg(INV_{Ref}))$$

Raffinement des types de variables

- Le raffinement peut aussi être l'occasion de raffiner les types de variables depuis des types abstraits (ensembles…) vers des types plus concrets pour se rapprocher de l'implantation.
- L'invariant de liaison doit alors rendre explicite le lien entre variables abstraites et concrètes.
- Lorsque (comme dans l'exemple précédent) la variable raffinée est identique à la variable de la machine la liaison peut être effectuée implicitement par homonymie (déclaration d'une variable de même nom dans le raffinement).

Exemple de raffinement (d'après le B Book)

Cahier des charges informel :

- L'application devra manipuler un ensemble d'entiers naturels non nuls.
- Une opération devra permettre de rajouter un élément de cet ensemble.
- Une autre opération retournera la valeur maximale de cet ensemble.

Machine, traduite directement du CdC


```
MACHINE Ex4
VARIABLES xx
INVARIANT xx:FIN(NAT1) /*ensemble des parties finies*/
INITIALISATION xx:={}
OPERATIONS
Enter(nn) = PRE nn:NAT1 THEN xx:=xx\/{nn}
END; /*Enter*/
mm <-- maximum = PRE xx/={} THEN mm:=max(xx)
END /*maximum*/
END /*machine*/
```

POs de machine

INVARIANT xx:FIN(NAT1)

INITIALISATION xx:={}

- PO d'initialisation : [xx:={}] xx:FIN(NAT1)
- {}:FIN(NAT1) (vraie)

PRE nn:NAT1 THEN xx:=xx\/{nn}

- PO d'opération (∀ implicites)
- xx:FIN(NAT1) & nn:NAT1⇒
  nn:NAT1 & [xx:=xx\/{nn}]xx:FIN(NAT1)
- xx:FIN(NAT1) & nn:NAT1⇒ xx\/{nn}:FIN(NAT1)
- Vraie (l'union d'un ensemble fini de naturels non nuls et d'un singleton entier naturel non nul est un ensemble fini d'entiers naturels non nuls)

POs de machine

INVARIANT xx:FIN(NAT1)

mm <-- maximum = PRE xx/={} THEN mm:=max(xx)

- PO d'opération (∀ implicites)
- xx:FIN(NAT1) & xx/={} ⇒
  xx/={} & [mm:=max(xx)]xx:FIN(NAT1)
- xx:FIN(NAT1) & xx/={} ⇒ xx:FIN(NAT1)
- PO triviale (obvious) : l'opération ne fait pas évoluer l'état de machine (simple opération de sortie)

Idée de raffinement

- Au vu du cahier des charges, simplissime, conserver l'ensemble n'est pas utile (de plus il n'est pas directement implantable);
- Il suffit de conserver le maximum;
- Ce maximum sera initialisé arbitrairement à zéro (correspondant à l'initialisation à {} de l'ensemble);
- Lorsqu'un nouvel entier est entré, il suffit de vérifier s'il est plus grand que le maximum courant (qu'il vient remplacer dans ce cas).

Exemple de raffinement


```
REFINEMENT R_ex4
REFINES Ex4
VARIABLES yy
INVARIANT yy = max(xx\/{0}) /*Invariant de liaison*/
INITIALISATION yy:=0
OPERATIONS
Enter (nn) = PRE nn : NAT1 THEN yy := max({yy,nn})
END; /* Enter*/
mm <-- maximum = PRE yy /= 0 THEN mm := yy
END /*maximum*/
END /* refinement*/
```

POs de raffinement

INVARIANT yy = max(xx\/{0}) /*raffinement*/

(simple invariant de liaison, pas de PO « locale »)

INITIALISATION yy:=0 /*raffinement*/

INITIALISATION xx:={} /*machine*/

- PO de raffinement d'initialisation :
- $[yy:=0]\neg[xx:=\{\}]\neg(yy = \max(xx \cup \{0\}))$
- $[yy:=0]\neg\neg(yy = \max(\{\} \cup \{0\}))$
- $[yy:=0](yy = \max(\{\} \cup \{0\}))$ /*pas de non déterminisme*/
- $[yy:=0](yy = \max(\{0\}))$
- $0 = \max(\{0\})$
- Vraie

POs de raffinement d’opérations

- Les POs de raffinement d’opérations ont pour but de montrer que l’opération raffinée est compatible avec l’opération de machine. Ceci se traduit par :
- SI invariant de machine ET invariant de raffinement (dont liaison) ET précondition de machine
- ALORS précondition de raffinement ET l’opération de raffinement établit qu’il est faux que l’opération de machine viole l’invariant de raffinement (dont liaison)

PO de raffinement d'opérations

Considérons

- Une machine
  - D'invariant INV_{Mch}
  - D'opération OP_{Mch} avec la précondition PRE_{Mch}
- Un raffinement
  - D'invariant INV_{Ref} (dont invariant de liaison)
  - D'opération OP_{Ref} avec la précontion PRE_{Ref}

Alors l'obligation de preuve de raffinement d'opération est (∀ variables de machine et de raffinement implicites)

$$INV_{Mch} \wedge INV_{Ref} \wedge PRE_{Mch} \Rightarrow PRE_{Ref} \wedge [OP_{Ref}]\neg[OP_{Mch}]\neg INV_{Ref}$$

Si les opérations rendent des résultats la PO doit être modifiée (voir ci-après) pour montrer que les résultats sont compatibles.

POs de raffinement (Enter)

	Machine	Raffinement
INVARIANT	$xx:FIN(NAT1)$	$yy = max(xx \cup \{0\})$
PRE	$nn : NAT1$	$nn : NAT1$
THEN	$xx:=xx \cup \{nn\}$	$yy := max(\{yy,nn\})$

$$INV_{Mch} \wedge INV_{Ref} \wedge PRE_{Mch} \Rightarrow PRE_{Ref} \wedge [OP_{Ref}] \neg [OP_{Mch}] \neg INV_{Ref}$$

$$[OP_{Ref}] \neg [OP_{Mch}] \neg INV_{Ref}$$

$$[yy:=max(\{yy,nn\})] \neg [xx:=xx \cup \{nn\}] \neg (yy=max(xx \cup \{0\}))$$

$$[yy:=max(\{yy,nn\})](yy=max(xx \cup \{nn\} \cup \{0\}))$$

$$max(\{yy,nn\})=max(xx \cup \{nn\} \cup \{0\})$$

Les PRE étant les mêmes la PO est

$$xx:FIN(NAT1) \wedge yy=max(xx \cup \{0\}) \wedge nn : NAT1 \Rightarrow$$
$$max(\{yy,nn\})=max(xx \cup \{nn\} \cup \{0\}) \text{ : vrai}$$

POs de raffinement d’opérations (suite)

- Dans le cas d’une opération rendant un résultat, il faut rajouter la PO suivante :
- SI invariant de machine ET invariant de raffinement (dont liaison) ET précondition de machine
- ALORS précondition de raffinement ET l’opération de raffinement établit qu’il est faux que l’opération de machine établisse que les valeurs retournées par l’opération de machine et l’opération raffinée soient différentes.
- Il faut donc renommer l’un des deux résultats pour écrire ce prédicat de PO.

PO de raffinement d'opérations

27 Avec les notations précédentes, en ajoutant que l'opération OP rend un résultat r (on rappelle que la signature de OP_{Mch} et de OP_{Ref} doit être la même), s'écrit (en utilisant une substitution dans une substitution pour renommer le résultat de l'opération raffinée en r') :

$$INV_{Mch} \wedge INV_{Ref} \wedge PRE_{Mch} \Rightarrow$$
$$PRE_{Ref} \wedge [[r:=r']OP_{Ref}]\neg[OP_{Mch}]\neg(r=r')$$

POs de raffinement (maximum)

	Machine	Raffinement
INVARIANT	xx : FIN(NAT1)	yy = max(xx\/{0})
PRE	xx /= {}	yy /= 0
THEN	mm := max(xx)	mm := yy

$$INV_{Mch} \wedge INV_{Ref} \wedge PRE_{Mch} \Rightarrow$$
$$PRE_{Ref} \wedge [[mm:=mm']OP_{Ref}]\neg[OP_{Mch}]\neg(mm=mm')$$

$$[[mm:=mm']OP_{Ref}]\neg[OP_{Mch}]\neg(mm=mm')$$

[mm’:=yy]¬[mm:=max(xx)]¬(mm=mm’) soit max(xx)=yy

La PO est donc :

$$xx:FIN(NAT1) \wedge yy=max(xx\backslash/\{0\}) \wedge xx/=\{\} \Rightarrow$$

yy /= $0 \wedge max(xx)=yy$: vrai (bien que l’atelier B peine à le prouver…)