La méthode B
Machines Abstraites

- Motivation : modulariser et décomposer les programmes.
- Forme : une structure avec plusieurs composantes.
- Propriétés : bonne construction.
- Composition de machines.

Substitutions de base

Substitution	Utilisation	Comportement
$x := E$	$[x := E]P$	$[x := E]P$
$x, y := E, F$	$[x, y := E, F]P$	$[z := F][x := E][y := z]P$ (z fraîch
$S_1; S_2$	$[S_1; S_2]P$	$[S_1][S_2]P$
`skip`	$[\texttt{skip}]P$	P
$Q \Longrightarrow S$	$[Q \Longrightarrow S]P$	$Q \Rightarrow [S]P$
$Q \mid S$	$[Q \mid S]P$	$Q \wedge [S]P$
$S_1 \ [] \ S_2$	$[S_1 \ [] \ S_2]P$	$[S_1]P \wedge [S_2]P$
$@z.S$	$[@z.S]P$	$\forall z.[S]P$

Quelques différences...

Attitude défensive :

$$Q \Longrightarrow S$$

La propriété Q est vérifiée avant d'exécuter le programme S.

Attitude offensive :

$$Q \mid S$$

La propriété Q n'est pas vérifiée avant d'exécuter le programme S, on fait confiance au programmeur.

Construction de programmes

SELECT Q THEN S END	$Q \Longrightarrow S$
PRE Q THEN S END	$Q \mid S$
CHOICE S_1 OR $S_2 \ldots S_n$ END	$S_1 [] \ldots [] S_n$
IF Q THEN S_1 ELSE S_2 END	$(Q \Longrightarrow S_1) [] (\mathtt{not}(Q) \Longrightarrow S_2)$
IF Q THEN S END	$(Q \Longrightarrow S) [] (\mathtt{not}(Q) \Longrightarrow \mathtt{skip})$

L'exemple de la tondeuse

Un enrouleur de câble électrique porte une étiquette Puissance max. 2000 W avec câble déroulé, 1000 W avec câble enroulé.

La tondeuse électrique de 1200 W est utilisée avec le câble enroulé. Le câble a fondu. Il ne disposait pas d'un composant testant si le câble était déroulé/enroulé.

Lorsque le fabricant a conçu l'enrouleur il a **démontré** qu'il fonctionne correctement si l'on respecte les conditions d'emploi sur l'étiquette. Cette démonstration a été faite **une fois**.

C'est ensuite à l'utilisateur de ne faire appel à l'enrouleur qu'en respectant les conditions d'emploi. Ces conditions d'emploi ne peuvent être vérifiées dans le *code* de l'opération.

Construction de programmes (suite)

VAR x IN S END	$@x.S$
ANY x WHERE Q THEN S END	$@x.(Q \Longrightarrow S)$
$x :\in E$	ANY y WHERE $y \in E$ THEN $x := y$ END
	n'importe quel objet de E
$x : P$	$@y.([x := y]P \Rightarrow x := y)$
	n'importe quel objet qui vérifie P
$x := \mathtt{bool}(P)$	IF P THEN true ELSE false END

Exemples de calcul

$$
\begin{array}{ll}
[x := z+1; y := x+z](y \in 0..5) & = \\
[z' := x+z][x := z+1][y := z'](y \in 0..5) & = \\
[z' := x+z][x := z+1](z' \in 0..5) = & \\
[z' := x+z](z' \in 0..5) & = \\
(x+z \in 0..5) &
\end{array}
$$

Exemples de calcul

$$
\begin{array}{ll}
[x := z+1][y := x+z](y \in 0..5) & = \\
[x := z+1]([y := x+z](y \in 0..5)) & = \\
[x := z+1](x+z \in 0..5) & = \\
z+1+z \in 0..5 &
\end{array}
$$

Exemples de calcul

$$
\begin{array}{ll}
[\texttt{IF } Q \texttt{ THEN } S \texttt{ END}]R & = \\
[(Q \Longrightarrow S) \ [] \ (\texttt{not}(Q) \Longrightarrow \texttt{skip})]R & = \\
[Q \Longrightarrow S]R \wedge [\texttt{not}(Q) \Longrightarrow \texttt{skip}]R & = \\
(Q \Rightarrow [S]R) \wedge (\texttt{not}(Q) \Rightarrow [\texttt{skip}]R) & = \\
(Q \Rightarrow [S]R) \wedge (\texttt{not}(Q) \Rightarrow R) &
\end{array}
$$

Exemples de calcul

$$
\begin{array}{ll}
[\texttt{IF } Q \texttt{ THEN } S_1 \texttt{ ELSE } S_2 \texttt{ END}]R & = \\
[(Q \Longrightarrow S_1) \ [] \ (\texttt{not}(Q) \Longrightarrow S_2)]R & = \\
[Q \Longrightarrow S_1]R \wedge \ [\texttt{not}(Q) \Longrightarrow S_2]R & = \\
(Q \Rightarrow [S_1]R) \wedge \ (\texttt{not}(Q) \Rightarrow [S_2]R) &
\end{array}
$$

Exemples de calcul

$$
\begin{array}{ll}
[x :\in E]R & = \\
[\texttt{ANY } y \texttt{ WHERE } y \in E \texttt{ THEN } x := y \texttt{ END}]R & = \\
[@y.(y \in E \Longrightarrow x := y)]R & = \\
\forall\, y.[y \in E \Longrightarrow x := y]R & = \\
\forall\, y.(y \in E \Rightarrow [x := y]R) &
\end{array}
$$

Opérations

- Routine : $opn =_{def} S$
- Procédure : $opn(var) =_{def} S$
- Fonction sans paramètre : $var \leftarrow opn =_{def} S$
- Fonction avec paramètre : $var_1 \leftarrow opn(var_2) =_{def} S$

Exemple de modélisation

Allouer un mot dans une mémoire et retourner l'adresse de l'emplacement alloué s'il y a de la place en mémoire.

ADRESSES	ensemble abstrait d'adresses
$memoire \subseteq$ ADRESSES	les adresses de la mémoire
$libres \subseteq memoire$	les adresses libres de la mémoire
$null \in$ ADRESSES	une adresse particulière
$null \notin memoire$	null n'est pas une adresse de la mémoire

Exemple d'utilisation

Calculer la moyenne de 4 entiers.

On se donne d'abord :

$$m \leftarrow moyenne(a, b) =_{def} m := (a + b)/2$$

Ensuite

$$\begin{aligned} m \leftarrow moyenne4(a, b, c, d) =_{def} \quad & \texttt{VAR } tmp1, tmp2 \texttt{ IN} \\ & tmp1 \leftarrow moyenne(a, b); \\ & tmp2 \leftarrow moyenne(c, d); \\ & m \leftarrow moyenne(tmp1, tmp2); \end{aligned}$$

Solution 1

$$\begin{aligned} & r \leftarrow allouer = \\ & \texttt{PRE } libres \neq \{\} \texttt{ THEN} \\ & \quad \texttt{ANY } v \texttt{ where } v \in libres \texttt{ THEN} \\ & \quad libres := libres - \{v\} \mid\mid r := v \\ & \quad \texttt{END} \\ & \texttt{END} \end{aligned}$$

Avant d'allouer un mot il faudra donc vérifier la précondition :

$$b \leftarrow paspleine = (b := \texttt{bool}(libres \neq \{\}))$$

Solution 2

$r \leftarrow allouer =$

IF $libres \neq \{\}$ THEN

ANY v where $v \in libres$ THEN

$libres := libres - \{v\} \parallel r := v$

END

ELSE $r := null$

END

Pas de vérification dans ce cas, si l'adresse de retour est $null$, alors la mémoire est pleine.

Machine de base

MACHINE nom	nom de la machine
VARIABLES $x_1, ..., x_n$	noms des variables
INVARIANT I	propriété
INITIALISATION S_0	substitution
OPERATIONS	substitutions

$var1 \leftarrow op(var2) =_{def} S;$

$\vdots$

END

Exemple de machine

MACHINE point

VARIABLES x, y

INVARIANT $x \in (0 \ldots 1024) \ \wedge \ y \in (0 \ldots 768)$

INITIALISATION $x, y := 0, 0$

OPERATIONS

$v1 \leftarrow$ abs $=_{def} v1 := x; v2 \leftarrow$ ord $=_{def} v2 := y;$

move$(dx, dy) =_{def}$

PRE $x + dx \leq 1024 \ \wedge \ y + dy \leq 768$

THEN $(x, y) := (x + dx, y + dy)$ END

END

Conditions

- Assurer l'appartenance des variables
- Variables distinctes
- Invariant satisfait dans l'état initial
- Invariant préservé par chaque opération (sous contraintes d'une pré-condition)

Dans notre exemple

- cas initial : $0 \in (0 \ldots 1024) \ \wedge \ 0 \in (0 \ldots 768)$
- abs : $x \in (0 \ldots 1024) \ \wedge \ y \in (0 \ldots 768) \Rightarrow$
  $\forall\, v1.[v1 := x]\ x \in (0 \ldots 1024) \ \wedge \ y \in (0 \ldots 768).$
- ord : $x \in (0 \ldots 1024) \ \wedge \ y \in (0 \ldots 768) \Rightarrow$
  $x \in (0 \ldots 1024) \ \wedge \ \forall\, v2.[v2 := y]\ y \in (0 \ldots 768).$
- move : $x \in (0 \ldots 1024) \ \wedge \ y \in (0 \ldots 768) \Rightarrow$
  $\forall\, (dx, dy).x + dx \leq 1024 \ \wedge \ y + dy \leq 768 \Rightarrow$
  $[x, y := x + dx, y + dy]\ (x \in (0 \ldots 1024) \ \wedge \ y \in (0 \ldots 768))$

Un autre exemple de machine

Implémenter deux feux de couleur rouge,jaune,vert tels qu'ils sont tout les deux jaune simultanément ou bien l'un d'entre eux est rouge et pas l'autre.

MACHINE feuxcouleur

SETS coul = { rouge,jaune,vert }

ABSTRACT_CONSTANTS suiv

PROPERTIES

suiv : coul → coul

suiv = { (rouge , vert), (vert, jaune), (jaune, rouge) }

VARIABLES $feuA, feuB$

INVARIANT

feuA ∈ coul ∧ feuB ∈ coul ∧

(feuA=jaune ∧ feuB=jaune) or (feuA=rouge ∧ feuB ≠rouge) or

(feuA ≠rouge ∧ feuB=rouge)

INITIALISATION

feuA, feuB := jaune, jaune

OPERATIONS

MiseEnService =

PRE

feuA = jaune ∧ feuB = jaune

THEN

ANY fa, fb WHERE

fa ∈ coul ∧ fb ∈ coul ∧

((fa = rouge ∧ fb ≠ rouge) or (fa ≠ rouge ∧ fb = rouge))

THEN feuA, feuB := fa, fb END

END ;

ChangerFeux=

PRE

(feuA = rouge $\wedge$ feuB $\neq$ rouge) or (feuA $\neq$ rouge $\wedge$ feuB = rouge)

THEN

ANY fa, fb WHERE

fa $\in$ coul $\wedge$ fb $\in$ coul $\wedge$

((fa, fb) = (suiv(feuA) , feuB) or (fa, fb) = (feuA , suiv(feuB)) or

(fa, fb) = (suiv(feuA), suiv(feuB))) $\wedge$

((fa = rouge $\wedge$ fb $\neq$ rouge) or (fa $\neq$ rouge $\wedge$ fb = rouge))

THEN feuA,feuB := fa,fb END

END ;

Spécification vs implantation

On peut ne pas implanter complètement une opération, les machines servent alors à décrire des spécifications. Par exemple, on spécifie une opération qui renvoie l'abscisse la plus petite d'un ensemble fini de points.

$px \leftarrow$ ppabs$(E) =_{def}$

PRE $E : \mathbb{P}((0 \ldots 1024) \times (0 \ldots 768)) \ \wedge \ E \ \neq \ \{\}$

THEN ANY x

WHERE $\exists\, y.(x, y) \in E \ \wedge \ \forall\, (x', y').(x', y') \in E \Rightarrow x \leq x'$

THEN $px := x$ END

END

Ensembles, constantes et propriétés

Les modélisations manipulent souvent des ensembles qui restent non spécifiés jusqu'à la phase d'implémentation.

Les constantes sont des valeurs particulières qui restent symboliques jusqu'à la phase d'implémentation.

MACHINE Mariages

SETS

PERSS ;

STATUT = {marie, celibataire};

SEXE= {homme, femme}

CONSTANTS max_pers

PROPERTIES

max_pers $\in$ NAT - $\{0\} \wedge$ card(PERSS) = max_pers

VARIABLES $pers, sexe, statut$

INVARIANT

$pers \subseteq$ PERSS $\wedge$ $sexe : pers \ \rightarrow$ SEXE $\wedge$ $statut : pers \ \rightarrow$ STATU

INITIALISATION

$pers, sexe, statut := \{\}, \{\}, \{\}$

OPERATIONS

$ajout(s, m) =_{def}$

PRE $s \in$ SEXE $\wedge\ m \in$ STATUT $\wedge$ PERSS $- pers\ \neq \{\}$

THEN

ANY x WHERE $x \in$ PERSS $- pers$

THEN $pers := pers \cup \{x\}\ ||\ sexe(x) := s\ ||\ statut(x) := m$ END

END

Initialisation possible

On peut vouloir spécifier que l'ensemble de personnes est un ensemble quelconque de femmes célibataires :

INITIALISATION

$pers :\in \mathbb{P}(PERSS)\ ||$

$sexe := pers \rightarrow \{$ femme $\}\ ||$

$statut := pers \rightarrow \{$ celibataire $\}$

Machine paramétrée

MACHINE Array(INDEX,VAL)

VARIABLES $table$

INVARIANT $table$: INDEX $\nrightarrow$ VAL

INITIALISATION $table = \{\}$

OPERATIONS

$enter(i, v) =_{def}$ PRE $i \in$ INDEX $\wedge\ v \in$ VAL

THEN $table(i) := v$ END;

$v \leftarrow access(i) =_{def}$ PRE $i \in$ INDEX

THEN $v := table(i)$ END;

$i \leftarrow search(v) =_{def}$ PRE $v \in \mathtt{ran}(table)$

THEN $i \in table^{-1}[v]$ END;

$b \leftarrow test(v) =_{def}$ PRE $v \in$ VAL

THEN $b := bool(v \in \mathtt{ran}(table))$ END;

Combiner des machines

- `INCLUDES` $M_1, \ldots, M_k$ (dynamique)
- `USES` $M_1, \ldots, M_k$ (statique)
- ...