Développement de systèmes critiques avec la méthode B

La modélisation avec la méthode B

3A Cursus Ingénieurs - Dominante Informatique
CentraleSupelec - Université Paris-Saclay - 2021/2022

Présenté par :

Idir AIT SADOUNE

idir.aitsadoune@centralesupelec.fr

1 Substitutions généralisées

1. Substitutions généralisées
2. Machine abstraite
3. Raffinement
4. La synthèse

CentraleSupélec
université PARIS-SACLAY

La construction des programmes

- **La programmation** permet de transformer nos idées sur les fonctionnalités d'un logiciel en une suite d'instructions pour la machine.

La construction des programmes

- **La programmation** permet de transformer nos idées sur les fonctionnalités d'un logiciel en une suite d'instructions pour la machine.
  - ➜ transformer une connaissance en un programme.

CentraleSupélec
université PARIS-SACLAY

La construction des programmes

- **La programmation** permet de transformer nos idées sur les fonctionnalités d'un logiciel en une suite d'instructions pour la machine.
  - → transformer une connaissance en un programme.
  - → transformer une idée abstraite en une idée concrète.

La construction des programmes

- **La programmation** permet de transformer nos idées sur les fonctionnalités d'un logiciel en une suite d'instructions pour la machine.
  - ➜ transformer une connaissance en un programme.
  - ➜ transformer une idée abstraite en une idée concrète.
- **La preuve de programmes** est le fait de donner une signification à une suite d'instructions.

La logique de Hoare

- Une assertion de la **logique de Hoare** est décrite par la formule $\{P\}\ S\ \{R\}$.

La logique de Hoare

- Une assertion de la **logique de Hoare** est décrite par la formule $\{P\}\ S\ \{R\}$.
  - P représente la pré-condition et R la post-condition de l'instruction S

CentraleSupélec
université PARIS-SACLAY

La logique de Hoare

- Une assertion de la **logique de Hoare** est décrite par la formule $\{P\}\ S\ \{R\}$.
  - P représente la pré-condition et R la post-condition de l'instruction S
- La formule $\{P\}\ S\ \{R\}$ signifient que :
  - **si** P est vraie et S se termine,
  - **alors** R est vraie après l'exécution de S.

La logique de Hoare

- Une assertion de la **logique de Hoare** est décrite par la formule $\{P\}\ S\ \{R\}$.
  - P représente la pré-condition et R la post-condition de l'instruction S
- La formule $\{P\}\ S\ \{R\}$ signifient que :
  - **si** P est vraie et S se termine,
  - **alors** R est vraie après l'exécution de S.
- La post-condition R peut être ce que le programme doit préserver.
  - invariants du programme par exemple.

Axiome de l'affectation

Axiome de l’affectation

- L’affectation est l’instruction $x := E$, associant à la variable x la valeur de l’expression E.

CentraleSupélec
université PARIS-SACLAY

Axiome de l'affectation

- L'affectation est l'instruction $x := E$, associant à la variable x la valeur de l'expression E.
- L'**axiome de l'affectation** :

$$\{R[E/x]\}\; x := E\; \{R\}$$

$R[E/x]$ désigne l'expression R dans laquelle les occurrences de la variable x ont été remplacées par l'expression E

Axiome de l'affectation

- L'affectation est l'instruction $x := E$, associant à la variable x la valeur de l'expression E.
- L'**axiome de l'affectation** :

$$\{R[E/x]\}\ x := E\ \{R\}$$

$R[E/x]$ désigne l'expression R dans laquelle les occurrences de la variable x ont été remplacées par l'expression E

→ signifie que $R[E/x]$ sera vrai si et seulement si R l'est après l'affectation.

Exemple

$$\{x + 1 = 43\}\; y := x + 1\; \{y = 43\}$$

CentraleSupélec
université PARIS-SACLAY

Règle de l'affaiblissement

- Du point de vue de la programmation, on peut renforcer une pré-condition et affaiblir une post-condition

$$\frac{P \Rightarrow P',\ P'\{S\}R',\ R' \Rightarrow R}{P\{S\}R}$$

Règle de l'affaiblissement

- Du point de vue de la programmation, on peut renforcer une pré-condition et affaiblir une post-condition

$$\frac{P \Rightarrow P',\ P'\{S\}R',\ R' \Rightarrow R}{P\{S\}R}$$

- Pour établir le triplet $\{P\}\ S\ \{R\}$, il suffit d'établir à la place le triplet $\{P'\}\ S\ \{R'\}$ où P' est une conséquence de P et R est une conséquence de R'.

Exemple

$? \qquad \{x < N\} \qquad x := x + 1 \qquad \{x \leq N\}$

Exemple

?	$\{x < N\}$	$x := x + 1$	$\{x \leq N\}$
✓	$\{x + 1 \leq N\}$	$x := x + 1$	$\{x \leq N\}$

CentraleSupélec
université PARIS-SACLAY

Exemple

?	$\{x < N\}$	$x := x + 1$	$\{x \leq N\}$
✓	$\{x + 1 \leq N\}$	$x := x + 1$	$\{x \leq N\}$
➜		$(x < N) \Rightarrow (x + 1 \leq N)$	

CentraleSupélec

université PARIS-SACLAY

La plus faible pré-condition de Dijkstra

L'idée de Dijkstra (Weakest Precondition - $wp_S(R)$ ou $wp(S, R)$) :
déterminer, pour une post-condition R et une instruction S, la plus faible pré-condition qui assure que S termine et que R est vraie après S.

La plus faible pré-condition de Dijkstra

L'idée de Dijkstra (Weakest Precondition - $wp_S(R)$ **ou** $wp(S, R)$**)** :
déterminer, pour une post-condition R et une instruction S, la plus faible pré-condition qui assure que S termine et que R est vraie après S.

$$\{P\}\ S\ \{R\} \ \widehat{=}\ P \Rightarrow wp_S(R)$$

La spécification des instructions

- La spécification de l'instruction $x := E$ des langages de programmation est le transformateur de prédicat noté :

$$wp_{x:=E}$$

La spécification des instructions

- La spécification de l'instruction $x := E$ des langages de programmation est le transformateur de prédicat noté :

$$wp_{x:=E}$$

- Il a été montré que ce transformateur de prédicat est la substitution de x par E dans le prédicat post-condition (**Hoare** et **Dijkstra**).

$$wp_{x:=E}(R) \;\Leftrightarrow\; R[E/x]$$

les occurrences libres de x sont replacées par E dans R

Exemple

? $\{x < N\}$ $x := x + 1$ $\{x \leq N\}$

Exemple

? $\{x < N\}$ $x := x + 1$ $\{x \leq N\}$

✓ $\{wp_{x:=x+1}(x \leq N)\}$ $x := x + 1$ $\{x \leq N\}$

CentraleSupélec
université PARIS-SACLAY

Exemple

? $\{x < N\}$ $x := x + 1$ $\{x \leq N\}$

✓ $\{wp_{x:=x+1}(x \leq N)\}$ $x := x + 1$ $\{x \leq N\}$

✓ $wp_{x:=x+1}(x \leq N) \Leftrightarrow (x + 1 \leq N)$

CentraleSupélec

université PARIS-SACLAY

Exemple

?	$\{x < N\}$	$x := x + 1$	$\{x \leq N\}$
✓	$\{wp_{x:=x+1}(x \leq N)\}$	$x := x + 1$	$\{x \leq N\}$
✓		$wp_{x:=x+1}(x \leq N) \Leftrightarrow (x + 1 \leq N)$	
➜		$(x < N) \Rightarrow (x + 1 \leq N)$	

CentraleSupélec
université PARIS-SACLAY

Conclusion

$$wp_{x:=E}(R) \Leftrightarrow R[E/x]$$

Considérant cette équivalence, la méthode B a choisi d'utiliser un langage proche des notations informatiques dont le sens est donné par la "substitution" de cette notation dans un prédicat.

CentraleSupélec
université PARIS-SACLAY

Liste des substitutions primitives

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

Liste des substitutions primitives

skip	substitution sans effet

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

Liste des substitutions primitives

skip	substitution sans effet
$x := E$	substitution simple

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

CentraleSupélec
université PARIS-SACLAY

Liste des substitutions primitives

skip	substitution sans effet
$x := E$	substitution simple
$x, y := E, F$	substitution multiple simple

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

CentraleSupélec
université PARIS-SACLAY

Liste des substitutions primitives

skip	substitution sans effet
$x := E$	substitution simple
$x, y := E, F$	substitution multiple simple
$P \mid S$	substitution préconditionnée

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

CentraleSupélec
université PARIS-SACLAY

Liste des substitutions primitives

skip	substitution sans effet
$x := E$	substitution simple
$x, y := E, F$	substitution multiple simple
$P \mid S$	substitution préconditionnée
$P \Longrightarrow S$	substitution gardée

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

Liste des substitutions primitives

skip	substitution sans effet
$x := E$	substitution simple
$x, y := E, F$	substitution multiple simple
$P \mid S$	substitution préconditionnée
$P \Longrightarrow S$	substitution gardée
$S \; [] \; T$	substitution de choix borné

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

CentraleSupélec

université PARIS-SACLAY

Liste des substitutions primitives

skip	substitution sans effet
$x := E$	substitution simple
$x, y := E, F$	substitution multiple simple
$P \mid S$	substitution préconditionnée
$P \Longrightarrow S$	substitution gardée
$S \; [] \; T$	substitution de choix borné
$@z \cdot S$	substitution de choix non borné

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

Liste des substitutions primitives

skip	substitution sans effet
$x := E$	substitution simple
$x, y := E, F$	substitution multiple simple
$P \mid S$	substitution préconditionnée
$P \Longrightarrow S$	substitution gardée
$S \mathbin{[]} T$	substitution de choix borné
$@z \cdot S$	substitution de choix non borné
$S \, ; \, T$	séquencement de substitutions

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

Liste des substitutions primitives

skip	substitution sans effet
$x := E$	substitution simple
$x, y := E, F$	substitution multiple simple
$P \mid S$	substitution préconditionnée
$P \Longrightarrow S$	substitution gardée
$S\ [\,]\ T$	substitution de choix borné
$@z \cdot S$	substitution de choix non borné
$S\ ;\ T$	séquencement de substitutions
$W(P, S, J, V)$	substitution d'itération

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

Les WP des substitutions primitives

Le cas de base $[x := E]\, R$ est la substitution uniforme de x par E dans R

Cas de substitution	Réduction	Condition

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

$^{*}x \setminus P$: x n'est pas libre dans P.

Les WP des substitutions primitives

Le cas de base $[x := E]\,R$ est la substitution uniforme de x par E dans R

Cas de substitution	Réduction	Condition
$[\mathrm{skip}]\,R$	R	

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

$^*x \setminus P$: x n'est pas libre dans P.

CentraleSupélec
université PARIS-SACLAY

Les WP des substitutions primitives

Le cas de base $[x := E]\ R$ est la substitution uniforme de x par E dans R

Cas de substitution	Réduction	Condition
$[\mathrm{skip}]\ R$	R	
$[x, y := E, F]\ R$	$[z := F][x := E][y := z]\ R$	$z \setminus E, F, R$

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

$^*x \setminus P$: x n'est pas libre dans P.

Les WP des substitutions primitives

Le cas de base $[x := E]\,R$ est la substitution uniforme de x par E dans R

Cas de substitution	Réduction	Condition
$[\mathrm{skip}]\,R$	R	
$[x, y := E, F]\,R$	$[z := F][x := E][y := z]\,R$	$z \setminus E, F, R$
$[P \mid S]\,R$	$P \wedge [S]\,R$	

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

$^{*}x \setminus P$: x n'est pas libre dans P.

Les WP des substitutions primitives

Le cas de base $[x := E]\ R$ est la substitution uniforme de x par E dans R

Cas de substitution	Réduction	Condition
$[\mathrm{skip}]\ R$	R	
$[x, y := E, F]\ R$	$[z := F][x := E][y := z]\ R$	$z \setminus E, F, R$
$[P \mid S]\ R$	$P \wedge [S]\ R$	
$[P \Longrightarrow S]\ R$	$P \Rightarrow [S]\ R$	

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

$^*x \setminus P$: x n'est pas libre dans P.

Les WP des substitutions primitives

Le cas de base $[x := E]\, R$ est la substitution uniforme de x par E dans R

Cas de substitution	Réduction	Condition
$[\mathrm{skip}]\, R$	R	
$[x, y := E, F]\, R$	$[z := F][x := E][y := z]\, R$	$z \setminus E, F, R$
$[P \mid S]\, R$	$P \wedge [S]\, R$	
$[P \Longrightarrow S]\, R$	$P \Rightarrow [S]\, R$	
$[S \,[]\, T]\, R$	$[S]\, R \wedge [T]\, R$	

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

$^{*}x \setminus P$: x n'est pas libre dans P.

Les WP des substitutions primitives

Le cas de base $[x := E]\ R$ est la substitution uniforme de x par E dans R

Cas de substitution	Réduction	Condition
$[\mathrm{skip}]\ R$	R	
$[x, y := E, F]\ R$	$[z := F][x := E][y := z]\ R$	$z \setminus E, F, R$
$[P \mid S]\ R$	$P \wedge [S]\ R$	
$[P \Longrightarrow S]\ R$	$P \Rightarrow [S]\ R$	
$[S\ []\ T]\ R$	$[S]\ R \wedge [T]\ R$	
$[@z \cdot S]\ R$	$\forall z \cdot [S]\ R$	$z \setminus R$

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

$^*x \setminus P$: x n'est pas libre dans P.

Les WP des substitutions primitives

Le cas de base $[x := E]\ R$ est la substitution uniforme de x par E dans R

Cas de substitution	Réduction	Condition
$[\mathrm{skip}]\ R$	R	
$[x, y := E, F]\ R$	$[z := F][x := E][y := z]\ R$	$z \setminus E, F, R$
$[P \mid S]\ R$	$P \wedge [S]\ R$	
$[P \Longrightarrow S]\ R$	$P \Rightarrow [S]\ R$	
$[S\ []\ T]\ R$	$[S]\ R \wedge [T]\ R$	
$[@z \cdot S]\ R$	$\forall z \cdot [S]\ R$	$z \setminus R$
$[S\ ;\ T]\ R$	$[S]\ ([T]\ R)$	

x, y, z des variables, E, F, V des expressions, S, T, U, W des substitutions généralisées, I, J, P, Q, R des prédicats.

$^{*}x \setminus P$: x n'est pas libre dans P.

Exemples

$$[x := x - 1 \mathbin{[]} x := x + 1]\,(1 \leq x \leq 10) \quad \Leftrightarrow \quad 2 \leq x \leq 11 \land 0 \leq x \leq 9$$

Exemples

$$
\begin{aligned}
[x := x - 1 \, [] \, x := x + 1] \, (1 \leq x \leq 10) \quad & \Leftrightarrow \quad 2 \leq x \leq 11 \wedge 0 \leq x \leq 9 \\
& \Leftrightarrow \quad 2 \leq x \leq 9
\end{aligned}
$$

CentraleSupélec

université PARIS-SACLAY

Exemples

$$
\begin{aligned}
[x := x - 1 \,[]\, x := x + 1]\,(1 \leq x \leq 10) \quad &\Leftrightarrow \quad 2 \leq x \leq 11 \wedge 0 \leq x \leq 9 \\
&\Leftrightarrow \quad 2 \leq x \leq 9
\end{aligned}
$$

$$
[x > 0 | x := x - 1]\,(-1 \leq x \leq 1) \quad \Leftrightarrow \quad x > 0 \wedge 0 \leq x \leq 2
$$

CentraleSupélec

université PARIS-SACLAY

Exemples

$$
\begin{aligned}
[x := x-1 \;[]\; x := x+1]\,(1 \le x \le 10) \quad &\Leftrightarrow \quad 2 \le x \le 11 \wedge 0 \le x \le 9 \\
&\Leftrightarrow \quad 2 \le x \le 9
\end{aligned}
$$

$$
\begin{aligned}
[x > 0 | x := x-1]\,(-1 \le x \le 1) \quad &\Leftrightarrow \quad x > 0 \wedge 0 \le x \le 2 \\
&\Leftrightarrow \quad 0 < x \le 2
\end{aligned}
$$

CentraleSupélec

université PARIS-SACLAY

Exemples

$$
\begin{aligned}
[x := x - 1 \,[]\, x := x + 1]\,(1 \leq x \leq 10) &\Leftrightarrow 2 \leq x \leq 11 \wedge 0 \leq x \leq 9 \\
&\Leftrightarrow 2 \leq x \leq 9
\end{aligned}
$$

$$
\begin{aligned}
[x > 0 | x := x - 1]\,(-1 \leq x \leq 1) &\Leftrightarrow x > 0 \wedge 0 \leq x \leq 2 \\
&\Leftrightarrow 0 < x \leq 2
\end{aligned}
$$

$$
[@z \cdot (z > 0 \Longrightarrow x := x + z)]\,(x \geq 3) \Leftrightarrow \forall z \cdot (z > 0 \Rightarrow [x := x + z](x \geq 3))
$$

Exemples

$$
\begin{aligned}
[x := x - 1 \; [] \; x := x + 1]\,(1 \leq x \leq 10) &\Leftrightarrow 2 \leq x \leq 11 \wedge 0 \leq x \leq 9 \\
&\Leftrightarrow 2 \leq x \leq 9
\end{aligned}
$$

$$
\begin{aligned}
[x > 0 | x := x - 1]\,(-1 \leq x \leq 1) &\Leftrightarrow x > 0 \wedge 0 \leq x \leq 2 \\
&\Leftrightarrow 0 < x \leq 2
\end{aligned}
$$

$$
\begin{aligned}
[@z \cdot (z > 0 \Longrightarrow x := x + z)]\,(x \geq 3) &\Leftrightarrow \forall z \cdot (z > 0 \Rightarrow [x := x + z](x \geq 3)) \\
&\Leftrightarrow \forall z \cdot (z > 0 \Rightarrow x + z \geq 3)
\end{aligned}
$$

CentraleSupélec
université PARIS-SACLAY

La substitution d'itération $W(P, S, J, V)$

➜ tant que P est vrai, faire la substitution S

CentraleSupélec
université PARIS-SACLAY

La substitution d'itération $W(P, S, J, V)$

- tant que P est vrai, faire la substitution S
- la boucle termine quand P devient faux.

La substitution d'itération $W(P, S, J, V)$

- ➜ tant que P est vrai, faire la substitution S
- ➜ la boucle termine quand P devient faux.

- l'expression J est l'invariant de la boucle

La substitution d'itération $W(P, S, J, V)$

- tant que P est vrai, faire la substitution S
- la boucle termine quand P devient faux.

- l'expression J est l'invariant de la boucle
  - doit être vrai à l'entrée de la boucle

La substitution d'itération $W(P, S, J, V)$

- ➜ tant que P est vrai, faire la substitution S
- ➜ la boucle termine quand P devient faux.

- l'expression J est l'invariant de la boucle
  - ➜ doit être vrai à l'entrée de la boucle
  - ➜ se maintenir vrai tant que la boucle s'exécute

La substitution d'itération $W(P, S, J, V)$

➜ tant que P est vrai, faire la substitution S

➜ la boucle termine quand P devient faux.

- l'expression J est l'invariant de la boucle
  - ➜ doit être vrai à l'entrée de la boucle
  - ➜ se maintenir vrai tant que la boucle s'exécute

- l'expression V est le variant de la boucle.

La substitution d'itération $W(P, S, J, V)$

- ➜ tant que P est vrai, faire la substitution S
- ➜ la boucle termine quand P devient faux.

- l'expression J est l'invariant de la boucle
  - ➜ doit être vrai à l'entrée de la boucle
  - ➜ se maintenir vrai tant que la boucle s'exécute

- l'expression V est le variant de la boucle.
  - une expression entière positive qui décroît à chaque itération de la boucle.

La substitution d'itération $W(P, S, J, V)$

- ➜ tant que P est vrai, faire la substitution S
- ➜ la boucle termine quand P devient faux.

- l'expression J est l'invariant de la boucle
  - ➜ doit être vrai à l'entrée de la boucle
  - ➜ se maintenir vrai tant que la boucle s'exécute

- l'expression V est le variant de la boucle.
  - une expression entière positive qui décroît à chaque itération de la boucle.
  - ➜ assure que la boucle termine.

CentraleSupélec
université PARIS-SACLAY

La WP de la substitution d'itération

$$[W(P,S,J,V)]\,R \quad \Leftrightarrow \quad \Big(\qquad \Big)$$

La WP de la substitution d'itération

$$[W(P, S, J, V)]\, R \quad \Leftrightarrow \quad ($$

$J \wedge$ l'invariant est une pré-condition

$)$

La WP de la substitution d'itération

$$
\begin{array}{rcl}
[W(P, S, J, V)]\, R & \Leftrightarrow & (\\
 & & J \wedge \\
\forall x \cdot ((J \wedge P) & \Rightarrow & [S]\, J) \wedge \\
 & &)
\end{array}
$$

l'invariant est une pré-condition
il est conservé dans la boucle

La WP de la substitution d'itération

$$
\begin{array}{rcll}
[W(P, S, J, V)]\, R & \Leftrightarrow & (& \\
 & & J \wedge & \text{l'invariant est une pré-condition} \\
\forall x \cdot ((J \wedge P) & \Rightarrow & [S]\, J) \wedge & \text{il est conservé dans la boucle} \\
\forall x \cdot (J & \Rightarrow & V \in \mathbb{N}) \wedge & \text{le variant est un entier naturel} \\
 & & & \\
 & &) &
\end{array}
$$

CentraleSupélec
université PARIS-SACLAY

La WP de la substitution d'itération

$$
\begin{array}{rcll}
[W(P,S,J,V)]\,R & \Leftrightarrow & (& \\
 & & J \wedge & \text{l'invariant est une pré-condition} \\
\forall x \cdot ((J \wedge P) & \Rightarrow & [S]\,J) \wedge & \text{il est conservé dans la boucle} \\
\forall x \cdot (J & \Rightarrow & V \in \mathbb{N}) \wedge & \text{le variant est un entier naturel} \\
\forall x \cdot ((J \wedge P) & \Rightarrow & [n := V][S](V < n)) \wedge & \text{qui décroit à chaque pas} \\
 & & & \\
 & &) &
\end{array}
$$

CentraleSupélec

université PARIS-SACLAY

La WP de la substitution d'itération

$$
\begin{array}{rcll}
[W(P,S,J,V)]\,R & \Leftrightarrow & (& \\
 & & J \wedge & \text{l'invariant est une pré-condition} \\
\forall x \cdot ((J \wedge P) & \Rightarrow & [S]\,J) \wedge & \text{il est conservé dans la boucle} \\
\forall x \cdot (J & \Rightarrow & V \in \mathbb{N}) \wedge & \text{le variant est un entier naturel} \\
\forall x \cdot ((J \wedge P) & \Rightarrow & [n := V][S](V < n)) \wedge & \text{qui décroit à chaque pas} \\
\forall x \cdot ((J \wedge \neg P) & \Rightarrow & R) & \text{la sortie de boucle implique } R \\
 & &) &
\end{array}
$$

Exemple

$[W(P, S, J, V)]\ R$ avec

Exemple

$[W(P, S, J, V)]\ R$ avec

$P \Leftrightarrow (index < 3)$

Exemple

$[W(P, S, J, V)]\ R$ avec

$P \Leftrightarrow (index < 3)$

$S \Leftrightarrow (v2 := v2 + 1; index := index + 1)$

Exemple

$[W(P, S, J, V)]\, R$ avec

$P \Leftrightarrow (index < 3)$

$S \Leftrightarrow (v2 := v2 + 1; index := index + 1)$

$J \Leftrightarrow (index \in 0..3 \wedge v2 = v1 + index)$

Exemple

$[W(P, S, J, V)]\, R$ avec

$P \Leftrightarrow (index < 3)$

$S \Leftrightarrow (v2 := v2 + 1; index := index + 1)$

$J \Leftrightarrow (index \in 0..3 \wedge v2 = v1 + index)$

$V \Leftrightarrow (3 - index)$

Exemple

$$[W(P,S,J,V)]\,R \quad \text{avec}$$

$$P \Leftrightarrow (index < 3)$$

$$S \Leftrightarrow (v2 := v2 + 1; index := index + 1)$$

$$J \Leftrightarrow (index \in 0..3 \wedge v2 = v1 + index)$$

$$V \Leftrightarrow (3 - index)$$

$$R \Leftrightarrow (index \geq 3)$$

Exemple

$$[W(P, S, J, V)]\, R \quad \text{avec}$$

$$P \Leftrightarrow (index < 3)$$
$$S \Leftrightarrow (v2 := v2 + 1; index := index + 1)$$
$$J \Leftrightarrow (index \in 0..3 \wedge v2 = v1 + index)$$
$$V \Leftrightarrow (3 - index)$$
$$R \Leftrightarrow (index \geq 3)$$

$(index < 3) \wedge (v2 = v1 + index) \wedge (index \in 0..3) \Rightarrow (v2 + 1 = v1 + index + 1) \wedge (index + 1 \in 0..3)$ invariant

Exemple

$$[W(P, S, J, V)]\, R \quad \text{avec}$$

$$\begin{aligned}
&P \Leftrightarrow (index < 3)\\
&S \Leftrightarrow (v2 := v2 + 1; index := index + 1)\\
&J \Leftrightarrow (index \in 0..3 \wedge v2 = v1 + index)\\
&V \Leftrightarrow (3 - index)\\
&R \Leftrightarrow (index \geq 3)
\end{aligned}$$

$(index < 3) \wedge (v2 = v1 + index) \wedge (index \in 0..3) \Rightarrow (v2 + 1 = v1 + index + 1) \wedge (index + 1 \in 0..3)$ invariant

$(v2 = v1 + index) \wedge (index \in 0..3) \Rightarrow (3 - index) \in \mathbb{N}$ variant def

CentraleSupélec

université PARIS-SACLAY

Exemple

$$[W(P,S,J,V)]\ R \quad \text{avec}$$

$$P \Leftrightarrow (index < 3)$$
$$S \Leftrightarrow (v2 := v2 + 1; index := index + 1)$$
$$J \Leftrightarrow (index \in 0..3 \wedge v2 = v1 + index)$$
$$V \Leftrightarrow (3 - index)$$
$$R \Leftrightarrow (index \geq 3)$$

$(index < 3) \wedge (v2 = v1 + index) \wedge (index \in 0..3) \Rightarrow (v2 + 1 = v1 + index + 1) \wedge (index + 1 \in 0..3)$ invariant

$(v2 = v1 + index) \wedge (index \in 0..3) \Rightarrow (3 - index) \in \mathbb{N}$ variant def

$(index < 3) \wedge (v2 = v1 + index) \wedge (index \in 0..3) \Rightarrow (3 - index + 1) < 3 - index$ variant décroit

CentraleSupélec

université PARIS-SACLAY

Exemple

$$[W(P,S,J,V)]\ R \quad \text{avec}$$

$$P \Leftrightarrow (index < 3)$$
$$S \Leftrightarrow (v2 := v2 + 1; index := index + 1)$$
$$J \Leftrightarrow (index \in 0..3 \land v2 = v1 + index)$$
$$V \Leftrightarrow (3 - index)$$
$$R \Leftrightarrow (index \geq 3)$$

$(index < 3) \land (v2 = v1 + index) \land (index \in 0..3) \Rightarrow (v2 + 1 = v1 + index + 1) \land (index + 1 \in 0..3)$	invariant
$(v2 = v1 + index) \land (index \in 0..3) \Rightarrow (3 - index) \in \mathbb{N}$	variant def
$(index < 3) \land (v2 = v1 + index) \land (index \in 0..3) \Rightarrow (3 - index + 1) < 3 - index$	variant décroit
$(v2 = v1 + index) \land (index \in 0..3) \land \neg(index < 3) \Rightarrow (index \geq 3)$	sortie de boucle

Définition des substitutions verbeuses

- **La méthode B** utilise des notations verbeuses qui facilitent la modélisation.

Définition des substitutions verbeuses

- **La méthode B** utilise des notations verbeuses qui facilitent la modélisation.

Substitution	Notation
S	**BEGIN** S **END**
$P \mid S$	**PRE** P **THEN** S **END**
$P \Longrightarrow S$	**SELECT** P **THEN** S **END**
$S \, [] \, T$	**CHOICE** S **OR** T **END**
$@z \cdot S$	**VAR** z **IN** S **END**
$W(P, S, J, V)$	**WHILE** P **DO** S **INVARIANT** J **VARIANT** V **END**

Exemple

$W(P, S, J, V)$ avec

$P \Leftrightarrow (index < 3)$

$S \Leftrightarrow (v2 := v2 + 1; index := index + 1)$

$J \Leftrightarrow (index \in 0..3 \wedge v2 = v1 + index)$

$V \Leftrightarrow (3 - index)$

```
WHILE (index < 3) DO
   v2 := v2 + 1;
   index := index + 1
INVARIANT
   index ∈ 0..3  ∧
   v2 = v1 + index
VARIANT
   3 − index
END
```

CentraleSupélec

université PARIS-SACLAY

Substitutions dérivées

Notation	Définition

la notation $x\$0$ fait référence à la nouvelle valeur x avant la substitution.

Substitutions dérivées

Notation	**Définition**
ASSERT P **THEN** S **END**	$P \mid P \Longrightarrow S$

la notation $x\$0$ fait référence à la nouvelle valeur x avant la substitution.

Substitutions dérivées

Notation	Définition
ASSERT P **THEN** S **END**	$P \mid P \Longrightarrow S$
IF P **THEN** S **ELSE** T **END**	$P \Longrightarrow S \; [] \; \neg P \Longrightarrow S$

la notation $x\$0$ fait référence à la nouvelle valeur x avant la substitution.

Substitutions dérivées

Notation	Définition
ASSERT P **THEN** S **END**	$P \mid P \Longrightarrow S$
IF P **THEN** S **ELSE** T **END**	$P \Longrightarrow S \, [] \, \neg P \Longrightarrow S$
CHOICE S **OR** T ... **OR** U **END**	$S \, [] \, T \, [] \, \ldots \, [] \, U$

la notation $x\$0$ fait référence à la nouvelle valeur x avant la substitution.

Substitutions dérivées

Notation	Définition
ASSERT P **THEN** S **END**	$P \mid P \Longrightarrow S$
IF P **THEN** S **ELSE** T **END**	$P \Longrightarrow S \; [] \; \neg P \Longrightarrow S$
CHOICE S **OR** $T \ldots$ **OR** U **END**	$S \; [] \; T \; [] \; \ldots \; [] \; U$
ANY z **WHERE** P **THEN** S **END**	$@z \cdot (P \Longrightarrow S)$

la notation $x\$0$ fait référence à la nouvelle valeur x avant la substitution.

Substitutions dérivées

Notation	Définition
ASSERT P **THEN** S **END**	$P \mid P \Longrightarrow S$
IF P **THEN** S **ELSE** T **END**	$P \Longrightarrow S \; [] \; \neg P \Longrightarrow S$
CHOICE S **OR** T ... **OR** U **END**	$S \; [] \; T \; [] \; \ldots \; [] \; U$
ANY z **WHERE** P **THEN** S **END**	$@z \cdot (P \Longrightarrow S)$
$x \; :\in \; E$	**ANY** x' **WHERE** $x' \; :\in \; E$ **THEN** $x \; := \; x'$ **END**

la notation $x\$0$ fait référence à la nouvelle valeur x avant la substitution.

Substitutions dérivées

Notation	Définition
ASSERT P **THEN** S **END**	$P \mid P \Longrightarrow S$
IF P **THEN** S **ELSE** T **END**	$P \Longrightarrow S \, [] \, \neg P \Longrightarrow S$
CHOICE S **OR** $T \ldots$ **OR** U **END**	$S \, [] \, T \, [] \, \ldots \, [] \, U$
ANY z **WHERE** P **THEN** S **END**	$@z \cdot (P \Longrightarrow S)$
$x \ :\in \ E$	**ANY** x' **WHERE** $x' \ :\in \ E$ **THEN** $x \ := \ x'$ **END**
$x \ : \ P$	**ANY** x' **WHERE** $[x\$0, x \ := \ x, x']P$ **THEN** $x \ := \ x'$ **END**

la notation $x\$0$ fait référence à la nouvelle valeur x avant la substitution.

Substitutions dérivées

Notation	Définition
ASSERT P **THEN** S **END**	$P \mid P \Longrightarrow S$
IF P **THEN** S **ELSE** T **END**	$P \Longrightarrow S \, [] \, \neg P \Longrightarrow S$
CHOICE S **OR** $T \ldots$ **OR** U **END**	$S \, [] \, T \, [] \, \ldots \, [] \, U$
ANY z **WHERE** P **THEN** S **END**	$@z \cdot (P \Longrightarrow S)$
$x \; :\in \; E$	**ANY** x' **WHERE** $x' \; :\in \; E$ **THEN** $x \; := \; x'$ **END**
$x \; : \; P$	**ANY** x' **WHERE** $[x\$0, x \; := \; x, x']P$ **THEN** $x \; := \; x'$ **END**
$x := \mathrm{bool}(P)$	**IF** P **THEN** $x := \mathrm{TRUE}$ **ELSE** $x := \mathrm{FALSE}$ **END**

la notation $x\$0$ fait référence à la nouvelle valeur x avant la substitution.

Substitutions dérivées

Notation	Définition
ASSERT P **THEN** S **END**	$P \mid P \Longrightarrow S$
IF P **THEN** S **ELSE** T **END**	$P \Longrightarrow S \, [] \, \neg P \Longrightarrow S$
CHOICE S **OR** T ... **OR** U **END**	$S \, [] \, T \, [] \, \ldots \, [] \, U$
ANY z **WHERE** P **THEN** S **END**	$@z \cdot (P \Longrightarrow S)$
$x \; :\in \; E$	**ANY** x' **WHERE** $x' \; :\in \; E$ **THEN** $x \; := \; x'$ **END**
$x \; : \; P$	**ANY** x' **WHERE** $[x\$0, x \; := \; x, x']P$ **THEN** $x \; := \; x'$ **END**
$x := \mathrm{bool}(P)$	**IF** P **THEN** $x := \mathrm{TRUE}$ **ELSE** $x := \mathrm{FALSE}$ **END**
$f(x) := E$	$f := f \lhd\!\!\!- \{x \mapsto E\}$

la notation $x\$0$ fait référence à la nouvelle valeur x avant la substitution.

Exemple

$$@xx.((xx \in \mathbb{N} \wedge xx + yy < MAXINT) \Longrightarrow (yy := yy + xx))$$

```
ANY xx WHERE
  xx ∈ ℕ ∧
  xx + yy < MAXINT
THEN
  yy := yy + xx
END
```

CentraleSupélec
université PARIS-SACLAY

Exemple

$$@xx.((xx \in \mathbb{N} \wedge xx + yy < MAXINT) \Longrightarrow (yy := yy + xx))$$

```
ANY xx WHERE
  xx ∈ ℕ ∧
  xx + yy < MAXINT
THEN
  yy := yy + xx
END
```

$$xx \in \mathbb{Z} \,|\, (xx > 0 \Longrightarrow xx := xx-1) \,[]\, (xx \leq 0 \Longrightarrow xx := xx+1)$$

```
PRE
  xx ∈ ℤ
THEN
  IF xx > 0 THEN
    xx := xx − 1
  ELSE
    xx := xx + 1
  END
END
```

Généralités

```
MACHINE
  /* Partie entete : */
    nom de la machine
    parametres de la machine
    contraintes sur les parametres
  /* Partie statique : */
    declaration d ensembles
    declaration de constantes
    proprietes des constantes
    variables (etat)
    invariant (caracterisation de l etat)
  /* Partie dynamique : */
    initialisation de l etat
    operations
END
```

Partie statique

```
MACHINE
  /* Partie entete */
SETS
  declaration d ensembles
CONSTANTS
  declaration de constantes
PROPERTIES
  proprietes des constantes
VARIABLES
  variables (etat)
INVARIANT
  caracterisation de l etat
ASSERTIONS
  assertions supplementaires
  /* Partie dynamique */
END
```

Exemple

```
MACHINE
  RESERVATION
SETS
  SIEGES
CONSTANTS
  nb_max
PROPERTIES
  nb_max ∈ 1..MAXINT ∧ /* nombre max de sieges */
  card(SIEGE) = nb_max ∧
VARIABLES
  occupes, nb_libre
INVARIANT
  occupes ⊆ SIEGES ∧ /* ensemble des sieges occupes */
  nb_libre ∈ 0..nb_max ∧ /* nombre de sieges libres */
  nb_libre = nb_max − card(occupes)
ASSERTIONS
  card(occupes) = nb_max − nb_libre
```

Partie dynamique

```
MACHINE
  /* Partie entete : */
  /* Partie statique : */
INITIALISATION
  initialisation des variables
OPERATIONS
  liste d operations
END
```

CentraleSupélec
université PARIS-SACLAY

Partie dynamique

```
MACHINE
  /* Partie entete : */
  /* Partie statique : */
INITIALISATION
  initialisation des variables
OPERATIONS
  liste d operations
END
```

Les opérations ont la forme générale :

```
R ⟵ nom_op(p) = PRE P THEN S END
```

CentraleSupélec
université PARIS-SACLAY

Exemple (1/2)

```
INITIALISATION
  occupes, nb_libre := ∅, nb_max

OPERATIONS
  place ⟵ reserver =
    PRE
      nb_libre ≠ 0
    THEN
      ANY pp WHERE
        pp ∈ SIEGES − occupes
      THEN
        place := pp ||
        occupes := occupes ∪ {pp} ||
        nb_libre := nb_libre − 1
      END
    END;
```

Exemple (2/2)

```
rr ⟵ liberer(place) =
  PRE
    place ∈ SIEGES
  THEN
    rr, occupes, nb_libre :(
      (rr=TRUE ∧
      place ∈ occupes$0 ∧
      occupes = occupes$0 − {place} ∧
      nb_libre = nb_libre$0 + 1)
      ∨
      (rr=FALSE ∧
      place ∉ occupes$0 ∧
      occupes = occupes$0 ∧
      nb_libre = nb_libre$0)
    )
  END
END
```

Remarque

Les substitutions généralisées séquencement et itération
ne sont pas autorisées dans les composants **MACHINE**.

Raffinement de substitutions

Raffinement de substitutions

- **Raffiner** une substitution S signifie trouver une substitution T dont l'effet sur l'état est un des effets possibles de la substitution S

CentraleSupélec
université PARIS-SACLAY

Raffinement de substitutions

- **Raffiner** une substitution S signifie trouver une substitution T dont l'effet sur l'état est un des effets possibles de la substitution S
- On note $S \sqsubseteq T$ la relation "S est raffiné par T".

$$S \sqsubseteq T \ \widehat{=} \ \forall R \cdot ([S]\, R \Rightarrow [T]\, R)$$

CentraleSupélec
université PARIS-SACLAY

Raffinement de substitutions

- **Raffiner** une substitution S signifie trouver une substitution T dont l'effet sur l'état est un des effets possibles de la substitution S
- On note $S \sqsubseteq T$ la relation "S est raffiné par T".

$$S \sqsubseteq T \;\widehat{=}\; \forall R \cdot ([S]\, R \Rightarrow [T]\, R)$$

- Si S préserve l'invariant R, alors le raffinement T le préserve.

Exemple

$$S \mathrel{\widehat{=}} (x : (x > x\$0))$$

Exemple

$$S \;\widehat{=}\; (x : (x > x\$0))$$

$$T \;\widehat{=}\; (x := x + 1)$$

Exemple

$$S \;\widehat{=}\; (x : (x > x\$0))$$

$$T \;\widehat{=}\; (x := x + 1)$$

$$S \sqsubseteq T$$

Raffinement avec changement de représentation

- Le raffinement en B permet également le changement de l'espace des variables.

Raffinement avec changement de représentation

- Le raffinement en B permet également le changement de l'espace des variables.
    - on part d'une machine B
    - → des variables abstraites (ensembles, relations, etc.)

Raffinement avec changement de représentation

- Le raffinement en B permet également le changement de l'espace des variables.
    - on part d'une machine B

  ➜ des variables abstraites (ensembles, relations, etc.)
    - on aboutit par raffinements successifs à une implémentation

  ➜ des variables proches des structures informatiques (tableaux, scalaires, etc.).

Exemple

$$couleur_{abs} \in COULEURS$$

Exemple

$$couleur_{abs} \in COULEURS$$

$$def\_couleur \in COULEURS \rightarrow (0..255 \times 0..255 \times 0..255)$$

$$couleur_{conc} \in (0..255 \times 0..255 \times 0..255)$$

Raffinement des machines

Raffinement des machines

- Le raffinement est le fait de transformer une spécification abstraite en un modèle proche d'un programme, pour enfin obtenir un programme.

CentraleSupélec
université PARIS-SACLAY

Raffinement des machines

- Le raffinement est le fait de transformer une spécification abstraite en un modèle proche d'un programme, pour enfin obtenir un programme.
- Il y a éventuellement raffinement de l'état.

Raffinement des machines

- Le raffinement est le fait de transformer une spécification abstraite en un modèle proche d'un programme, pour enfin obtenir un programme.
- Il y a éventuellement raffinement de l'état.
- L'effet des opérations de la machine abstraite doit être préservé

CentraleSupélec
université PARIS-SACLAY

Raffinement des machines

- Le raffinement est le fait de transformer une spécification abstraite en un modèle proche d'un programme, pour enfin obtenir un programme.
- Il y a éventuellement raffinement de l'état.
- L'effet des opérations de la machine abstraite doit être préservé
  - Pour chaque opération :
    - → reformulation en fonction du changement d'état
    - → affaiblissement des pré-conditions
    - → réduction du non-déterminisme

Syntaxe

```
MACHINE M1
SETS
  declaration d ensembles
CONSTANTS
  declaration de constantes
PROPERTIES
  proprietes des constantes
VARIABLES
  variables (etat)
INVARIANT
  caracterisation de l etat
ASSERTIONS
  assertions supplementaires
INITIALISATION
  initialisation des variables
OPERATIONS
  liste d operations
END
```

```
REFINEMENT M2
REFINES M1
SETS
  declaration d ensembles
CONSTANTS
  declaration de constantes
PROPERTIES
  proprietes des constantes
VARIABLES
  variables (etat)
INVARIANT
  caracterisation de l etat
ASSERTIONS
  assertions supplementaires
INITIALISATION
  initialisation des variables
OPERATIONS
  liste d operations
END
```

Exemple

```
REFINEMENT RESERVATION_REF
REFINES RESERVATION
VARIABLES
  etat, nb_libre
INVARIANT
  etat ∈ SIEGES → BOOL ∧ /* chaque siege a une valeur booleenne */
  occupes = etat^-1[{TRUE}] /* invariant de liaison */
INITIALISATION
  nb_libre := nb_max ||
  etat := SIEGES × {FALSE}
```

Exemple

```
OPERATIONS
  place ⟵ reserver =
    ANY pp1 WHERE
      pp1 ∈ etat^-1[{FALSE}]
    THEN
      place := pp1 ||
      etat(pp1) := TRUE ||
      nb_libre := nb_libre − 1
    END;
```

Exemple

```
rr ⟵ liberer(place) =
  PRE
    place ∈ SIEGES
  THEN
    rr, etat, nb_libre :(
      (rr=TRUE ∧
      etat$0(place) = TRUE ∧
      etat(place) = FALSE ∧
      nb_libre = nb_libre$0 + 1)
      ∨
      (rr=FALSE ∧
      etat$0(place) = FALSE ∧
      etat = etat$0 ∧
      nb_libre = nb_libre$0)
    )
  END
END
```

La synthèse

- La méthode B est une méthode formelle basée sur la logique de Hoare et la plus faible pré-condition de Dijkstra.

CentraleSupélec
université PARIS-SACLAY

La synthèse

- La méthode B est une méthode formelle basée sur la logique de Hoare et la plus faible pré-condition de Dijkstra.
- Le développement avec la méthode B permet de transformer une idée abstraite vers un programme informatique.

La synthèse

- La méthode B est une méthode formelle basée sur la logique de Hoare et la plus faible pré-condition de Dijkstra.
- Le développement avec la méthode B permet de transformer une idée abstraite vers un programme informatique.
- La machine est le composant principal d'une spécification B.

La synthèse

- La méthode B est une méthode formelle basée sur la logique de Hoare et la plus faible pré-condition de Dijkstra.
- Le développement avec la méthode B permet de transformer une idée abstraite vers un programme informatique.
- La machine est le composant principal d'une spécification B.
- Le raffinement est l'opération qui permet de transformer une spécification en un modèle proche d'un programme informatique.

⌛ Fin