La méthode B

Cours donné à l'Ecole des Jeunes Chercheurs en Programmation
Aussois - 28 juin 2003

Marie-Laure Potet *Didier Bert*

LSR-IMAG, Grenoble, France

LSR-IMAG
- Intro
- Données
- SubsG
- Machine
- Raffin

Plan du cours

1. Introduction à la méthode B
2. Formalisme de modélisation
3. Spécification des opérations : substitutions
4. Les machines abstraites
5. Raffinement et implémentation

Particularités du logiciel

- produit intellectuel
  - coût de fabrication nul
  - conception complexe
- logiciel pour la sécurité
  - pas d'usure
  - duplication à coût nul
  - fonctionnalités complexes
  - rapidité, réactivité

$\Rightarrow$ coût Validation/Vérification élevé

Contraintes

- Fiabilité (transport ferroviaire) :
  - Système : 10^{-9} pannes par rame/heure
  - Logiciel : 10^{-11} pannes par rame/heure

  $\Rightarrow$ non vérifiable par expérimentation
- Coût du développement (aérospaciale) :
  - $\times 3$ les fonctionnalités embarquées
  - $\times 60$ l'effort de production de code

  $\Rightarrow$ maîtrise du processus

Systèmes critiques

- Intérêt limité de la redondance
  - double développement
  - double support matériel
  - absence de mode d'erreur commun
- Pas de principe de sécurité intrinsèque
  - panne $\not\Rightarrow$ état dangereux
  - système discret

$\Rightarrow$ Vers des techniques formelles

Le ferroviaire et B

Logiciel pour les fonctions critiques de sécurité (fin 80)

1) développement non redondé avec validation à l'aide de méthodes formelles

- correction du code vis-a-vis des spécifications fontionnelles

2) utilisation de la technique du Processeur Sécuritaire Codé pour la détection des pannes matérielles

- codage des données et vérification à l'exécution
- etat sûr si non conformité à l'exécution

Le ferroviaire et B ... suite

1. vérification a posteriori :
   - ajout d'assertions dans le code
   - vérification semi-automatique
2. lien avec la spécification :
   - réexpression formelle
   - conformité (manuelle) du code

$\Rightarrow$ Méthode B (J-R Abrial)

- développement correct par construction

Météor : B + PSC $\Rightarrow$ suppression des tests unitaires

Méthode B

- écriture de spécifications à état
  - formalisation rigoureuse mais accessible
- preuve de propriétés sur les données
- raffinement et génération de code
  - représentation informatique des données
  - introduction progressive de la complexité
- construction incrémentale
  - des spécifications, des développements et des preuves
- méthodologie et outillage

Spécification

Une machine abstraite :

- un état
- une initialisation
- des opérations
- des propriétés invariantes

Données	ensembles
initialisation opérations	substitutions généralisées
propriétés	prédicats du premier ordre

Vérification

Obligations de preuve :

- Les propriétés invariantes sont vérifiées par la dynamique
- Les raffinements préservent la correction totale
- Le code est exempt d’erreur à l’exécution

L'atelier B (ClearSy)

- Analyseur
- Générateur d'obligations de preuve
- Démonstrateur automatique
- Démonstrateur interactif
- Générateur de code (C et Ada)
- Gestionnaire de projets

Partie 2 : Modélisation

1. Introduction à la méthode B
2. Formalisme de modélisation
3. Spécification des opérations : substitutions
4. Les machines abstraites
5. Raffinement et implémentation

Prédicats

- Logique du premier ordre :
  $P \wedge Q$, $P \Rightarrow Q$, $\neg\, P$
  $\forall x \cdot P$ quantification
  $[x := E]\, P$ substitution dans un prédicat

- Prédicats de base :
  $x \in S$ appartenance
  $E_1 = E_2$ égalité

- Les autres constructeurs sont dérivés :
  $P \vee Q$, $\exists x \cdot P$, $x \notin S$, etc.

Bases de la modélisation : Expressions et Ensembles

- Les ensembles (typés) :
  $S_1 \times S_2$ produit
  $\mathbb{P}(S)$ ensemble des parties
  $\{\, x \mid P \}$ ensemble en compréhension
  BIG un ensemble infini

- Les expressions :
  x variable
  $[x := E_1]E_2$ substitution dans une expression
  (E_1, E_2) paire d'expressions
  $\mathsf{choice}(S)$ fonction de choix
  S ensemble

Quelques notations

La substitution :

$$[x := E]\, P$$

les occurrences libres de x sont remplacées par E dans P.

Autre notation :

$$x \backslash P$$

qui signifie : x n'est pas libre dans P.

Axiomes de base

	Axiome
SET1	$(E, F) \in s \times t \;\Leftrightarrow\; E \in s \;\wedge\; F \in t$
SET2	$s \in \mathbb{P}(t) \;\Leftrightarrow\; \forall x \cdot (x \in s \Rightarrow x \in t)$
SET3	$E \in \{x \mid x \in s \wedge P\} \;\Leftrightarrow\; (E \in s \wedge [x := E]\, P)$
SET4	$\forall x \cdot (x \in s \Leftrightarrow x \in t) \;\Rightarrow\; s = t$
SET5	$\exists x \cdot (x \in s) \;\Rightarrow\; \mathsf{choice}(s) \in s$
SET6	$\mathsf{infinite}(\mathsf{BIG})$

Constructions dérivées

Les autres opérateurs et notations sur les ensembles sont dérivés du jeu de base donné.
Les propriétés usuelles sur ces opérateurs peuvent être démontrées à partir des axiomes. Exemples :

$s \subseteq t$	$s \in \mathbb{P}(t)$	
$s \cup t$	$\{a \mid a \in u \ \wedge \ (a \in s \ \vee \ a \in t)\}$	$s \subseteq u \wedge t \subseteq u$
$s \cap t$	$\{a \mid a \in u \ \wedge \ (a \in s \ \wedge \ a \in t)\}$	$s \subseteq u \wedge t \subseteq u$
$s - t$	$\{a \mid a \in u \ \wedge \ (a \in s \ \wedge \ a \notin t)\}$	$s \subseteq u \wedge t \subseteq u$
$\{E\}$	$\{a \mid a \in u \ \wedge \ a = E\}$	$E \in u$
$\{E, F\}$	$\{E\} \cup \{F\}$	$E \in u \wedge F \in u$
$\varnothing$	BIG – BIG	

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Construction des relations

Relation entre deux ensembles $\quad s \leftrightarrow t \ \widehat{=} \ \mathbb{P}(s \times t)$

Construction des relations

Relation entre deux ensembles $s \leftrightarrow t \ \widehat{=}\ \mathbb{P}(s \times t)$

Opérateurs classiques sur les relations :

Condition	Expression	Définition
$r \in s \leftrightarrow t$	$\mathsf{dom}(r)$	$\{x \mid x \in s \ \wedge \ \exists y \cdot (y \in t \ \wedge \ (x,y) \in r)\}$
$r \in s \leftrightarrow t$	$\mathsf{ran}(r)$	$\{y \mid y \in t \ \wedge \ \exists x \cdot (x \in s \ \wedge \ (x,y) \in r)\}$
$r \in s \leftrightarrow t \ \wedge \ u \subseteq s$	$r[u]$	$\{y \mid y \in t \ \wedge \ \exists x \cdot (x \in u \ \wedge \ (x,y) \in r)\}$
$r \in s \leftrightarrow t$	r^{-1}	$\{(y,x) \mid (y,x) \in t \times s \ \wedge \ (x,y) \in r\}$

LSR-IMAG

Autres opérateurs sur les relations

Condition	Expr	Définition
	$\mathsf{id}(s)$	$\{x,y \mid (x,y) \in s \times s \ \wedge \ x = y\}$
$r_1 \in s \leftrightarrow t \ \wedge \ r_2 \in t \leftrightarrow u$	$r_1 \ ; \ r_2$	$\{x,z \mid (x,z) \in s \times u \ \wedge \ \exists y \cdot (y \in t \ \wedge \ (x,y) \in r_1 \ \wedge \ (y,z) \in r_2)\}$
$r \in s \leftrightarrow t \ \wedge \ q \in s \leftrightarrow t$	$r \lhd\!\!+ q$	$\{x,y \mid (x,y) \in s \times t \ \wedge \ (((x,y) \in r \ \wedge \ x \notin \mathsf{dom}(q)) \ \vee \ (x,y) \in q)\}$

Construction des fonctions

Les fonctions sont un cas particulier de relations :

Signification	Notation	Définition
f. partielles	$s \pfun t$	$\{r \mid r \in s \rel t \ \land \ \forall x,y,z \cdot (x,y \in r \ \land \ x,z \in r \Rightarrow y = z)\}$
f. totales	$s \tfun t$	$\{f \mid f \in s \pfun t \ \land \ \mathsf{dom}(f) = s\}$
injectives part.	$s \pinj t$	$\{f \mid f \in s \pfun t \ \land \ f^{-1} \in t \pfun s\}$
injectives tot.	$s \tinj t$	$s \pinj t \ \cap \ s \tfun t$
evaluation	$f(E)$	$\mathsf{choice}(f[\{E\}])$ si $f \in s \pfun t \ \land \ E \in \mathsf{dom}(f)$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Construction des ensembles inductifs

Comment définir les ensembles tels que :

- les entiers naturels $\mathbb{N}$
- les parties finies d'un ensemble $\mathbb{F}(s)$
- la fermeture réflexive transitive d'une relation r^*

. . . tout en restant dans la "syntaxe" B. . .

Définition par récurrence

On dispose d'un schéma d'induction pour caractériser un sous-ensemble E de s :

- un élément de base $a \in E$
- une règle $x \in E \Rightarrow f(x) \in E$
- une clause de fermeture : E est le plus petit sous-ensemble de s finiment engendré par *la règle* à partir de *la base*.

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Définition par récurrence

On dispose d'un schéma d'induction pour caractériser un sous-ensemble E de s :

- un élément de base $a \in E$
- une règle $x \in E \Rightarrow f(x) \in E$
- une clause de fermeture : E est le plus petit sous-ensemble de s finiment engendré par *la règle* à partir de *la base*.

Soit g tel que $g : e \mapsto \{a\} \cup f[e]$

La fonction g est monotone :

$e_1 \subseteq e_2 \Rightarrow g(e_1) \subseteq g(e_2)$

Plus petit point fixe

D'après le théorème de Tarski :

La plus petite solution de $X = g(X)$ est le *plus petit point fixe* de g, qui est défini par :

$$\mathsf{fix}(g) = \mathsf{inter}(\{e \mid e \in \mathbb{P}(s) \ \wedge \ g(e) \subseteq e\})$$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Plus petit point fixe

D'après le théorème de Tarski :

La plus petite solution de $X = g(X)$ est le *plus petit point fixe* de g, qui est défini par :

$$\mathsf{fix}(g) = \mathsf{inter}(\{e \mid e \in \mathbb{P}(s) \ \wedge \ g(e) \subseteq e\})$$

avec : si $u \in \mathbb{P}_1(\mathbb{P}(s))$ alors

$$\mathsf{inter}(u) \ = \ \{x \mid x \in s \ \wedge \ \forall y \cdot (y \in u \Rightarrow x \in y)\}$$

c'est-à-dire l'ensemble $\mathsf{inter}(u)$ est contenu dans tous les sous-ensembles de u.

Exemple : définition de r^*

On a une relation $r \in s \leftrightarrow s$

r^* est réflexive $\quad \mathrm{id}(s) \subseteq r^*$

elle contient r

et est fermée par composition $\quad v \subseteq r^* \Rightarrow (r \,;\, v) \subseteq r^*$

La fonction g de génération de r^* est :

$$g \;:\; e \mapsto \mathrm{id}(s) \cup (r \,;\, e)$$

Les ensembles utilisables directement en B

Notations d'ensembles prédéfinis en B avec, pour chacun, un jeu d'opérateurs usuels. Ce sont :

- les ensembles *donnés* : ce sont des ensembles infinis, non vides
- les ensembles finis *énumérés*
- les entiers relatifs $\mathbb{Z}$ (avec les sous-ensembles $\mathbb{N}$ et $\mathbb{N}_1$)
- les séquences (fonctions de $1..n \rightarrow s$)
- les arbres n-aires (avec le sous-ensemble des arbres binaires)

ensembliste

Soit un ensemble

$personne \subseteq PERSONNE$:

R1 : toute personne est soit un homme, soit une femme

R2 : une personne ne peut être à la fois un homme et une femme

R3 : seules les femmes peuvent avoir un mari qui est un homme

R4 : les femmes ont au plus un mari

R5 : les hommes ne peuvent être mariés qu'à au plus une femme

R6 : les mères d'une personne sont des femmes mariées

Voir la solution

Exercice de modélisation ensembliste (suite)

A l'aide des définitions précédentes, définir les notions de :

R7 : père

R8 : parent

R9 : enfant

R10 : grand-parent et ancêtre

R11 : frère-sœur

R12 : Démontrer $mere = pere \; ; \; mari^{-1}$

Voir la solution

Solution des exercices

Retour Solution :

LSR-IMAG

Solution des exercices

- Intro
- Données
- SubsG
- Machine
- Raffin

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

Solution des exercices

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$$homme \subseteq personne$$
$$femme \subseteq personne$$
$$homme \cup femme = personne$$

LSR-IMAG

Solution des exercices

- Intro
- Données
- SubsG
- Machine
- Raffin

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$$homme \subseteq personne$$
$$femme \subseteq personne$$
$$homme \cup femme = personne$$

R2 : une personne ne peut être à la fois un homme et une femme

Solution des exercices

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$$homme \subseteq personne$$
$$femme \subseteq personne$$
$$homme \cup femme = personne$$

R2 : une personne ne peut être à la fois un homme et une femme

$$homme \cap femme = \varnothing$$

Solution des exercices

- Intro
- Données
- SubsG
- Machine
- Raffin

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$$homme \subseteq personne$$
$$femme \subseteq personne$$
$$homme \cup femme = personne$$

R2 : une personne ne peut être à la fois un homme et une femme

$$homme \cap femme = \varnothing$$

R3 : seules les femmes peuvent avoir un mari qui est un homme

Solution des exercices

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$homme \subseteq personne$

$femme \subseteq personne$

$homme \cup femme = personne$

R2 : une personne ne peut être à la fois un homme et une femme

$homme \cap femme = \varnothing$

R3 : seules les femmes peuvent avoir un mari qui est un homme

$mari \in femme \leftrightarrow homme$

Solution des exercices

- Intro
- Données
- SubsG
- Machine
- Raffin

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$homme \subseteq personne$

$femme \subseteq personne$

$homme \cup femme = personne$

R2 : une personne ne peut être à la fois un homme et une femme

$homme \cap femme = \varnothing$

R3 : seules les femmes peuvent avoir un mari qui est un homme

$mari \in femme \leftrightarrow homme$

R4 : les femmes ont au plus un mari

Solution des exercices

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$homme \subseteq personne$

$femme \subseteq personne$

$homme \cup femme = personne$

R2 : une personne ne peut être à la fois un homme et une femme

$homme \cap femme = \varnothing$

R3 : seules les femmes peuvent avoir un mari qui est un homme

$mari \in femme \leftrightarrow homme$

R4 : les femmes ont au plus un mari

$mari \in femme \nrightarrow homme$

Solution des exercices

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$homme \subseteq personne$

$femme \subseteq personne$

$homme \cup femme = personne$

R2 : une personne ne peut être à la fois un homme et une femme

$homme \cap femme = \varnothing$

R3 : seules les femmes peuvent avoir un mari qui est un homme

$mari \in femme \leftrightarrow homme$

R4 : les femmes ont au plus un mari

$mari \in femme \nrightarrow homme$

R5 : les hommes ne peuvent être mariés qu'à au plus une femme

Solution des exercices

- Intro
- Données
- SubsG
- Machine
- Raffin

Retour   Solution :

R1 : toute personne est soit un homme, soit une femme

$homme \subseteq personne$

$femme \subseteq personne$

$homme \cup femme = personne$

R2 : une personne ne peut être à la fois un homme et une femme

$homme \cap femme = \varnothing$

R3 : seules les femmes peuvent avoir un mari qui est un homme

$mari \in femme \leftrightarrow homme$

R4 : les femmes ont au plus un mari

$mari \in femme \mathrel{+\!\!\!\rightarrow} homme$

R5 : les hommes ne peuvent être mariés qu'à au plus une femme

$mari^{-1} \in homme \mathrel{+\!\!\!\rightarrow} femme$

Solution des exercices

- Intro
- Données
- SubsG
- Machine
- Raffin

Retour   Solution :

R1 : toute personne est soit un homme, soit une femme

$homme \subseteq personne$

$femme \subseteq personne$

$homme \cup femme = personne$

R2 : une personne ne peut être à la fois un homme et une femme

$homme \cap femme = \varnothing$

R3 : seules les femmes peuvent avoir un mari qui est un homme

$mari \in femme \leftrightarrow homme$

R4 : les femmes ont au plus un mari

$mari \in femme \mathrel{+\!\!\!\rightarrow} homme$

R5 : les hommes ne peuvent être mariés qu'à au plus une femme

$mari^{-1} \in homme \mathrel{+\!\!\!\rightarrow} femme$

$mari \in femme \mathrel{\rightarrowtail\!\!\!\!\!+} homme$

Solution des exercices

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$homme \subseteq personne$

$femme \subseteq personne$

$homme \cup femme = personne$

R2 : une personne ne peut être à la fois un homme et une femme

$homme \cap femme = \varnothing$

R3 : seules les femmes peuvent avoir un mari qui est un homme

$mari \in femme \leftrightarrow homme$

R4 : les femmes ont au plus un mari

$mari \in femme \nrightarrow homme$

R5 : les hommes ne peuvent être mariés qu'à au plus une femme

$mari^{-1} \in homme \nrightarrow femme$

$mari \in femme \rightarrowtail homme$

R6 : les mères d'une personne sont des femmes mariées

Solution des exercices

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Retour Solution :

R1 : toute personne est soit un homme, soit une femme

$homme \subseteq personne$

$femme \subseteq personne$

$homme \cup femme = personne$

R2 : une personne ne peut être à la fois un homme et une femme

$homme \cap femme = \varnothing$

R3 : seules les femmes peuvent avoir un mari qui est un homme

$mari \in femme \leftrightarrow homme$

R4 : les femmes ont au plus un mari

$mari \in femme \nrightarrow homme$

R5 : les hommes ne peuvent être mariés qu'à au plus une femme

$mari^{-1} \in homme \nrightarrow femme$

$mari \in femme \rightarrowtail homme$

R6 : les mères d'une personne sont des femmes mariées

Solution des exercices (suite)

Solution :

Solution des exercices (suite)

- Intro
- Données
- SubsG
- Machine
- Raffin

Solution :

R7 : père

Solution des exercices (suite)

Solution :

R7 : père $pere = mere$; $mari$

LSR-IMAG

Solution des exercices (suite)

- Intro
- Données
- SubsG
- Machine
- Raffin

Solution :

R7 : père $pere = mere$; $mari$

R8 : parent

Solution des exercices (suite)

Solution :

R7 : père $pere = mere \; ; mari$

R8 : parent $parent = mere \cup pere$

Solution des exercices (suite)

- Intro
- Données
- SubsG
- Machine
- Raffin

Solution :

R7 : père $pere = mere \; ; mari$

R8 : parent $parent = mere \cup pere$

R9 : enfant

Solution des exercices (suite)

Solution :

R7 : père $pere = mere\ ;\ mari$

R8 : parent $parent = mere \cup pere$

R9 : enfant $enfant = parent^{-1}$

Solution des exercices (suite)

- Intro
- Données
- SubsG
- Machine
- Raffin

Solution :

R7 : père $pere = mere\ ;\ mari$

R8 : parent $parent = mere \cup pere$

R9 : enfant $enfant = parent^{-1}$

R10 : grand-parent et ancêtre

Solution des exercices (suite)

Solution :

R7 : père $pere = mere\ ;\ mari$

R8 : parent $parent = mere \cup pere$

R9 : enfant $enfant = parent^{-1}$

R10 : grand-parent et ancêtre $grand\_parent = parent\ ;\ parent$

$ancetre = parent\ ;\ parent^*$

Solution des exercices (suite)

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Solution :

R7 : père $pere = mere\ ;\ mari$

R8 : parent $parent = mere \cup pere$

R9 : enfant $enfant = parent^{-1}$

R10 : grand-parent et ancêtre $grand\_parent = parent\ ;\ parent$

$ancetre = parent\ ;\ parent^*$

R11 : frère-sœur

Solution des exercices (suite)

Solution :

R7 : père $pere = mere\ ;\ mari$

R8 : parent $parent = mere \cup pere$

R9 : enfant $enfant = parent^{-1}$

R10 : grand-parent et ancêtre $grand\_parent = parent\ ;\ parent$

$ancetre = parent\ ;\ parent^*$

R11 : frère-sœur $frere\_soeur = (mere\ ;\ mere^{-1}) - \mathsf{id}(personne)$

Solution des exercices (suite)

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Solution :

R7 : père $pere = mere\ ;\ mari$

R8 : parent $parent = mere \cup pere$

R9 : enfant $enfant = parent^{-1}$

R10 : grand-parent et ancêtre $grand\_parent = parent\ ;\ parent$

$ancetre = parent\ ;\ parent^*$

R11 : frère-sœur $frere\_soeur = (mere\ ;\ mere^{-1}) - \mathsf{id}(personne)$

R12 : Démontrer $mere = pere\ ;\ mari^{-1}$

Solution des exercices (suite)

Solution :

R7 : père $pere = mere ; mari$

R8 : parent $parent = mere \cup pere$

R9 : enfant $enfant = parent^{-1}$

R10 : grand-parent et ancêtre $grand\_parent = parent ; parent$

$ancetre = parent ; parent^*$

R11 : frère-sœur $frere\_soeur = (mere ; mere^{-1}) - \mathsf{id}(personne)$

R12 : Démontrer $mere = pere ; mari^{-1}$

$$\begin{aligned} pere ; mari^{-1} &= (mere ; mari) ; mari^{-1} \\ &= mere ; (mari ; mari^{-1}) \\ &= mere ; \mathsf{id}(\mathsf{dom}(mari)) \\ &= mere \end{aligned}$$

Retour

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Partie 3 : Les substitutions généralisées

1. Introduction à la méthode B
2. Formalisme de modélisation
3. Spécification des opérations : substitutions
4. Les machines abstraites
5. Raffinement et implémentation

"substitutions" primitives

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

Spécification des instructions

- Logique des programmes : correction partielle

  $P\{S\}Q$

  Si l'état satisfait P avant S et si S termine,
  alors l'état satisfait Q après.

Spécification des instructions

- Logique des programmes : correction partielle

  $P\{S\}Q$

  Si l'état satisfait P avant S et si S termine,
  alors l'état satisfait Q après.

- Plus faible précondition : correction totale

  $wp(S, Q)$

  Si l'état satisfait $wp(S, Q)$ avant S
  alors S termine et l'état satisfait Q après.

Remarque : $P\{S\}Q$ en correction totale est équivalent à $P \Rightarrow wp(S, Q)$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Spécification des instructions

- Logique des programmes : correction partielle

  $P\{S\}Q$

  Si l'état satisfait P avant S et si S termine,
  alors l'état satisfait Q après.

- Plus faible précondition : correction totale

  $wp(S, Q)$

  Si l'état satisfait $wp(S, Q)$ avant S
  alors S termine et l'état satisfait Q après.

Remarque : $P\{S\}Q$ en correction totale est équivalent à $P \Rightarrow wp(S, Q)$

- En B, la plus faible précondition $wp(S, Q)$ est notée sous la forme d'une substitution $[S]Q$

substitutions primitives

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

Exemple de calcul

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Le langage des substitutions généralisées

Dans les programmes B, les substitutions s'écrivent avec des mots-clés :

Substitution	Notation
$P \mid S$	PRE P THEN S END
$P \Longrightarrow S$	SELECT P THEN S END
$S \parallel T$	CHOICE S OR T END
$@z \cdot S$	VAR z IN S END
$\mathcal{W}(P, S, J, V)$	WHILE P DO S INVARIANT J VARIANT V END

généralisées (suite)

$x := E \mid\mid y := F$	$x, y := E, F$
BEGIN S END	(S)
IF P THEN S ELSE T END	$(P \Longrightarrow S) \parallel (\neg\, P \Longrightarrow T)$
CHOICE S OR $T \ldots$ OR U END	$S \parallel T \parallel \cdots \parallel U$
ANY z WHERE P THEN S END	$@z \cdot (P \Longrightarrow S)$
LET $x, \ldots, y$ BE $x = E \ \wedge \ldots \wedge y = F$ IN S END	ANY $x, \ldots, y$ WHERE $x = E \ \wedge \ldots \wedge y = F$ THEN S END

Exemple de calcul

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Le langage des substitutions généralisées (suite)

$x :\in E$	ANY x' WHERE $x' \in E$ THEN $x := x'$ END
$x :=$ bool$((P)$	$x :=$ IF P THEN TRUE ELSE FALSE END
$f(x) := E$	$f := f \Leftarrow \{(x, E)\}$

Exemple de calcul

précondition

- $[\text{ IF } P \text{ THEN } S \text{ ELSE } T \text{ END}]\, R$
  $[(P \Longrightarrow S) \,[\!]\, (\neg\, P \Longrightarrow T)]\, R$ définition IF
  $[P \Longrightarrow S]\, R \;\wedge\; [\neg\, P \Longrightarrow T]\, R$ définition wp
  $(P \Rightarrow [S]\, R) \;\wedge\; (\neg\, P \Rightarrow [T]\, R)$

Exemple de calcul de plus faible précondition

- $[\text{ IF } P \text{ THEN } S \text{ ELSE } T \text{ END}]\, R$
  $[(P \Longrightarrow S) \,[\!]\, (\neg\, P \Longrightarrow T)]\, R$ définition IF
  $[P \Longrightarrow S]\, R \;\wedge\; [\neg\, P \Longrightarrow T]\, R$ définition wp
  $(P \Rightarrow [S]\, R) \;\wedge\; (\neg\, P \Rightarrow [T]\, R)$

- $[x :\in E]\, R$
  $[\text{ ANY } x' \text{ WHERE } x' \in E \text{ THEN } x := x' \text{ END}]\, R$ définition $:\in$
  $\forall x' \cdot (x' \in E \Rightarrow [x := x']\, R)$ définition wp

présentation du problème

Problème :
On veut spécifier une opération qui alloue un mot dans une mémoire et retourne l'adresse de l'emplacement alloué, s'il y a de la place en mémoire.

Quelques préliminaires de modélisation :

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

Exemples de modélisation (1)

Cas 1 : L'opération $allouer$ ne peut agir que s'il reste des adresses libres. Première modélisation : une précondition assure qu'il reste de la place.

$r \longleftarrow allouer =$	entête de l'opération
PRE $libres \neq \varnothing$ THEN	précondition
ANY v WHERE	
$v \in libres$	choix d'une adresse libre
THEN	
$libres := libres - \{v\} \;\|\|$	modification de l'état
$r := v$	retour de l'adresse allouée
END	
END	

Exemples de modélisation (1) suite

Cas 1 : Dans la méthode B, lorsqu'on appelle une opération, il y a une obligation de preuve qui permet d'assurer que la précondition est vérifiée à l'appel.

D'un point de méthode de spécification, il faut, dans ce cas, fournir à l'utilisateur des opérations pour tester *de l'extérieur* qu'une précondition est vérifiée. On aura ici :

$$b \longleftarrow n\_est\_pas\_pleine = b := \mathsf{bool}(libres \neq \varnothing)$$

Exemples de modélisation (2)

Cas 2 : Autre manière de spécifier : l'utilisateur n'a pas à tester la précondition. Si l'adresse de retour est $null$, cela signifie à l'utilisateur que la mémoire est pleine et que l'allocation n'a pas été possible

$r \longleftarrow allouer =$	entête de l'opération
IF $libres \neq \varnothing$ THEN	test dynamique
ANY v WHERE	
$v \in libres$	choix d'une adresse libre
THEN	
$libres := libres - \{v\} \mid\mid$	modification de l'état
$r := v$	retour de l'adresse allouée
END	
ELSE	il n'y a plus d'adresse libre

Exemples de modélisation (3)

Cas 3 : On pourrait simplement spécifier avec les deux cas possibles de retour de valeur de r :

$r \longleftarrow allouer =$	entête de l'opération
CHOICE	choix interne
ANY $v \in libres$ THEN	
$libres := libres - \{v\} \;\|\|$	modification de l'état
$r := v$	retour de l'adresse allouée
END	
OR	autre possibilité
$r := null$	retour de la valeur de non allocation
END	

Comparez cette solution avec la précédente. Que peut-on dire ?

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Caractérisation des substitutions

- Le langage des substitutions généralisées est conçu pour décrire des changement d'états.
- Il y a une grande variété de substitutions.
- Que peut-on dire de commun à toutes les substitutions ?
- Peut-on "représenter" les substitutions par l'effet qu'elle produisent comme une relation entre les états ?

Terminaison d'une substitution

La terminaison est un prédicat $\mathsf{trm}(S)$ qui caractérise la terminaison de la substitution S. Définition :

$$\mathsf{trm}(S) \;\Rightarrow\; [S]\,\mathsf{btrue}$$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Terminaison d'une substitution

La terminaison est un prédicat $\mathsf{trm}(S)$ qui caractérise la terminaison de la substitution S. Définition :

$$\mathsf{trm}(S) \;\Rightarrow\; [S]\,\mathsf{btrue}$$

Quelques résultats :

$$\begin{array}{lcl}
\mathsf{trm}(x := E) & \Leftrightarrow & \mathsf{btrue} \\
\mathsf{trm}(\mathsf{skip}) & \Leftrightarrow & \mathsf{btrue} \\
\mathsf{trm}(P \mid S) & \Leftrightarrow & P \;\wedge\; \mathsf{trm}(S) \\
\mathsf{trm}(P \Longrightarrow S) & \Leftrightarrow & P \Rightarrow \mathsf{trm}(S) \\
\mathsf{trm}(S \parallel T) & \Leftrightarrow & \mathsf{trm}(S) \;\wedge\; \mathsf{trm}(T) \\
\mathsf{trm}(@z \cdot S) & \Leftrightarrow & \forall z \cdot \mathsf{trm}(S)
\end{array}$$

Prédicat avant-après

Le prédicat avant-après $\mathsf{prd}_x(S)$ donne la relation entre les valeurs avant et après la substitution S pour les variables x. Définition :

$$\mathsf{prd}_x(S) \;\Rightarrow\; \neg\,[S]\,(x' \neq x)$$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Prédicat avant-après

Le prédicat avant-après $\mathsf{prd}_x(S)$ donne la relation entre les valeurs avant et après la substitution S pour les variables x. Définition :

$$\mathsf{prd}_x(S) \;\Rightarrow\; \neg\,[S]\,(x' \neq x)$$

$$
\begin{array}{lcll}
\mathsf{prd}_x(x := E) & \Leftrightarrow & x' = E & \\
\mathsf{prd}_{x,y}(x := E) & \Leftrightarrow & x', y' = E, y & \\
\mathsf{prd}_x(\mathsf{skip}) & \Leftrightarrow & x' = x & \\
\mathsf{prd}_x(P \mid S) & \Leftrightarrow & P \Rightarrow \mathsf{prd}_x(S) & \\
\mathsf{prd}_x(P \Longrightarrow S) & \Leftrightarrow & P \;\wedge\; \mathsf{prd}_x(S) & \\
\mathsf{prd}_x(S \,[\!]\, T) & \Leftrightarrow & \mathsf{prd}_x(S) \;\vee\; \mathsf{prd}_x(T) & \\
\mathsf{prd}_x(@z \cdot S) & \Leftrightarrow & \exists z \cdot \mathsf{prd}_x(S) & \text{si } z \backslash x'
\end{array}
$$

Forme normalisée

Toute substitution peut se mettre sous la forme :

$$S \;=\; \mathsf{trm}(S) \mid @x' \cdot (\mathsf{prd}_x(S) \Longrightarrow x := x')$$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Forme normalisée

Toute substitution peut se mettre sous la forme :

$$S \;=\; \mathsf{trm}(S) \mid @x' \cdot (\mathsf{prd}_x(S) \Longrightarrow x := x')$$

Deux substitutions sont égales si elles ont le même effet sur tout prédicat :

$$S = T \;\;\widehat{=}\;\; [S]\,R \Leftrightarrow [T]\,R \qquad \text{pour tout prédicat } R$$

Forme normalisée

Toute substitution peut se mettre sous la forme :

$$S \;=\; \mathsf{trm}(S) \mid @x' \cdot (\mathsf{prd}_x(S) \Longrightarrow x := x')$$

Deux substitutions sont égales si elles ont le même effet sur tout prédicat :

$$S = T \;\widehat{=}\; [S]\,R \Leftrightarrow [T]\,R \qquad \text{pour tout prédicat } R$$

Les substitutions généralisées satisfont les propriétés :

$[S]\,(R \wedge Q) \;\Leftrightarrow\; [S]\,R \wedge [S]\,Q$	Distributivité
$\forall x \cdot (R \Rightarrow Q) \;\Rightarrow\; ([S]\,R \Rightarrow [S]Q)$	Monotonie

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Quelques autres propriétés

Pour toute substitution généralisée S, on a les propriétés suivantes :

Totalité	$\neg\,\mathsf{trm}(S) \;\Rightarrow\; \mathsf{prd}_x(S)$
Terminaison	$[S]\,R \;\Rightarrow\; \mathsf{trm}(S)$

Quelques autres propriétés

Pour toute substitution généralisée S, on a les propriétés suivantes :

Totalité	$\neg\, \mathsf{trm}(S) \;\Rightarrow\; \mathsf{prd}_x(S)$
Terminaison	$[S]\, R \;\Rightarrow\; \mathsf{trm}(S)$

Une substitution generalisée est *faisable* s'il existe une valeur après associée à une valeur avant :

$$\mathsf{fis}(S) \quad \Leftrightarrow \quad \exists x' \cdot \mathsf{prd}_x(S)$$

Une autre définition de la faisabilité est :

$$\mathsf{fis}(S) \quad \Leftrightarrow \quad \neg\, [S]\, \mathsf{bfalse}$$

Substitutions généralisées vs prédicats

On a vu que l'on peut passer des substitutions généralisées aux prédicats avant-après et terminaison et vice-versa. Pourquoi choisir les SG pour spécifier, plutôt que les prédicats comme en Z ?

- le style d'écriture est plus proche de la programmation
- par défaut, les variables ne sont pas modifiées ($y' = y$)
- l'utilisation des substitutions est plus efficace pour les preuves :
  $[x := 1\ ;\ x := x + 1]\,(x > 0)$ —> $2 > 0$
  avec les prédicats : $\exists x_2 \cdot (x_2 = 1 \wedge x' = x_2 + 1) \Rightarrow x' > 0$
- Il y a un continuum entre les spécifications et les programmes à l'aide du raffinement.

Partie 4 : Les machines abstraites

1. Introduction à la méthode B
2. Formalisme de modélisation
3. Spécification des opérations : substitutions
4. Composants B : les machines abstraites
5. Raffinement et implémentation

Composant machine

MACHINE

Partie entête :

nom de la machine et paramètres

contraintes sur les paramètres

Partie statique :

déclaration d'ensembles et de constantes

propriétés des constantes

variables (état)

invariant (caractérisation de l'état)

Partie dynamique :

initialisation de l'état

opérations

Rubriques d'une machine

MACHINE $M(X,u)$
CONSTRAINTS C /* *spécification des paramètres* */
SETS S; /* *ensembles donnés* */
$T = \{a,b\}$ /* *ensembles énumérés* */
CONSTANTS c /* *liste de constantes (concrètes)* */
PROPERTIES R /* *spécification des constantes* */
VARIABLES x /* *liste de variables (abstraites)* */
INVARIANT I /* *spécification des variables* */
INITIALISATION U /* *substitution d'initialisation* */
OPERATIONS /* *liste des opérations* */
$r \longleftarrow nom\_op(p)$ = PRE P THEN K END; ...
END

LSR-IMAG
- Intro
- Données
- SubsG
- Machine
- Raffin

Déclaration d'opération

- Une déclaration d'opération est de la forme :

$$r \longleftarrow op(p) \;=\; \text{PRE } P \text{ THEN } K \text{ END}$$

avec $P \Rightarrow \mathsf{trm}(K)$

Voir aussi l'appel

Déclaration d'opération

- Une déclaration d'opération est de la forme :

$$r \longleftarrow op(p) \;=\; \text{PRE } P \text{ THEN } K \text{ END}$$

avec $P \Rightarrow \mathsf{trm}(K)$

Voir aussi l'appel

- La définition d'opération $r \longleftarrow op(p) \;=\; S$ doit satisfaire :

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Déclaration d'opération

- Une déclaration d'opération est de la forme :

$r \longleftarrow op(p) \;=\; \text{PRE } P \text{ THEN } K \text{ END}$

avec $P \Rightarrow \mathsf{trm}(K)$

Voir aussi l'appel

- La définition d'opération $r \longleftarrow op(p) \;=\; S$ doit satisfaire :
  - p (liste de noms de paramètres) ne sont pas modifiés dans S

Déclaration d'opération

- Une déclaration d'opération est de la forme :

$$r \longleftarrow op(p) \;=\; \text{PRE } P \text{ THEN } K \text{ END}$$

avec $P \Rightarrow \mathsf{trm}(K)$

Voir aussi l'appel

- La définition d'opération $r \longleftarrow op(p) \;=\; S$ doit satisfaire :
  - p (liste de noms de paramètres) ne sont pas modifiés dans S
  - r et p sont disjoints et distincts des variables x

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Déclaration d'opération

- Une déclaration d'opération est de la forme :

$r \longleftarrow op(p) \;=\; \text{PRE } P \text{ THEN } K \text{ END}$

avec $P \Rightarrow \mathsf{trm}(K)$

Voir aussi l'appel

- La définition d'opération $r \longleftarrow op(p) \;=\; S$ doit satisfaire :
  - p (liste de noms de paramètres) ne sont pas modifiés dans S
  - r et p sont disjoints et distincts des variables x
  - r (liste de noms de résultats) sont affectés dans S
    ou, plus formellement, $r \backslash \mathsf{prd}_{x,r}(S)$

- L'initialisation établit l'invariant :
  B représente les conditions sur les ensembles déclarés.

$$B \ \wedge \ C \ \wedge \ R \ \Rightarrow \ [U]I$$

Obligations de preuves d'une machine

- L'initialisation établit l'invariant :
  B représente les conditions sur les ensembles déclarés.

$$B \ \wedge \ C \ \wedge \ R \ \Rightarrow \ [U]I$$

- Chaque opération préserve l'invariant :

$$B \ \wedge \ C \ \wedge \ R \ \wedge \ I \ \wedge \ P \ \Rightarrow \ [K]I$$

machine

- L'initialisation établit l'invariant :
  B représente les conditions sur les ensembles déclarés.

$$B \ \wedge \ C \ \wedge \ R \ \Rightarrow \ [U]I$$

- Chaque opération préserve l'invariant :

$$B \ \wedge \ C \ \wedge \ R \ \wedge \ I \ \wedge \ P \ \Rightarrow \ [K]I$$

- Par la propriété de terminaison, on assure que K termine :

$$B \ \wedge \ C \ \wedge \ R \ \wedge \ I \ \wedge \ P \ \Rightarrow \ \mathsf{trm}(K)$$

Ascenseur

On souhaite spécifier le fonctionnement simplifié d'un ascenseur.

- une porte à chaque étage
- l'appel intérieur et l'appel extérieur ne sont pas distingués
- il n'y a pas de panne
- une constante donne le nombre d'étages : max_etage (> 0)

Les opérations sont :

- ouvrir, fermer une porte
- appeler l'ascenseur
- déplacement de l'ascenseur

Propriétés de l'ascenseur

- l'ascenseur reste dans la limite des étages
- si une porte est ouverte l'ascenseur est arrêté à l'étage correspondant
- chaque appel est traité en un temps raisonnable
- si l'ascenseur est arrêté à un étage, l'appel à cet étage est considéré comme traité
- ...

Modélisation de l'ascenseur

MACHINE $ASCENSEUR$

SETS $MODE = \{arret, mouv\}$

CONSTANTS $max\_etage, ETAGES$

PROPERTIES $max\_etage \in \mathsf{NAT}_1 \wedge ETAGES = 0..max\_etage$

VARIABLES $appels, ouvertes, pos, mode$

INVARIANT

$$ouvertes \subseteq ETAGES \ \wedge \ appels \subseteq ETAGES$$
$$\wedge \ pos \in ETAGES \ \wedge \ mode \in MODE$$
$$\wedge \ (ouvertes \neq \varnothing \Rightarrow ouvertes = \{pos\} \wedge mode = arret)$$
$$\wedge \ (mode = arret \Rightarrow pos \notin appels)$$

1. Introduction à la méthode B
2. Formalisme de modélisation
3. Spécification des opérations : substitutions
4. Les machines abstraites
5. Raffinement et implémentation

Raffinement : principe

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

- Le raffinement est le fait de transformer une spécification abstraite en un texte plus proche de la programmation, pour finalement obtenir un programme
- L'effet des appels d'opérations de la machine abstraite doit être préservé, vu de l'utilisateur
- Le raffinement de machine se fait opération par opération
- Il y a (éventuellement) raffinement de l'état
- Pour chaque opération :
  - reformulation en fonction du changement d'état
  - affaiblissement des préconditions

Exemple : une machine à raffiner

MACHINE $RAFF\_EX1$

VARIABLES yy

INVARIANT $yy \subseteq \mathsf{NAT}_1$

INITIALISATION $yy := \varnothing$

OPERATIONS

$ajouter(nn)$ = PRE $nn \in \mathsf{NAT}_1$

THEN $yy := yy \cup \{nn\}$

END;

$vv \longleftarrow choix$ = PRE $yy \neq \varnothing$

THEN $vv :\in yy$

END

END

Une machine qui “fait presque la même chose”

MACHINE $RAFF\_EX2$

VARIABLES zz

INVARIANT $zz \in \mathsf{NAT}$

INITIALISATION $zz := 0$

OPERATIONS

$ajouter(nn)$ = PRE $nn \in \mathsf{NAT}_1$

THEN $zz := nn$

END;

$vv \longleftarrow choix$ = $vv := zz$

END

le même état

Définition du raffinement de S par T :

$S \sqsubseteq T$	$\forall R \cdot ([S]\, R \;\Rightarrow\; [T]\, R)$

Si S préserve l'invariant, alors le raffinement le préserve.

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Raffinement d'une substitution dans le même état

Définition du raffinement de S par T :

$S \sqsubseteq T$	$\forall R \cdot ([S]\, R \;\Rightarrow\; [T]\, R)$

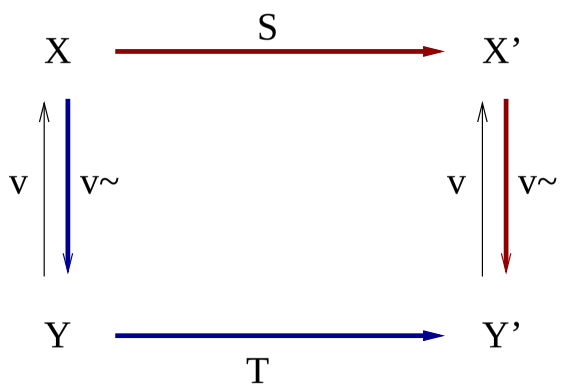
Si S préserve l'invariant, alors le raffinement le préserve.

Autre définition :

$S \sqsubseteq T$	$\mathsf{trm}(S) \;\Rightarrow\; \mathsf{trm}(T)$ $\mathsf{trm}(S) \;\wedge\; \mathsf{prd}_x(T) \;\Rightarrow\; \mathsf{prd}_x(S)$

représentation : principe

Diagramme du raffinement entre un état X et un état Y :



v~ ; rel(T) <: rel(S) ; v~

- La relation v est totale sur le domaine concret Y.
- Relation par prédicat L : $v = \{y, x \mid L\}$

Raffinement avec changement de représentation : formules

$S \sqsubseteq_L T$	$L \ \wedge\ \mathsf{trm}(S) \ \Rightarrow\ \mathsf{trm}(T)$ $L \ \wedge\ \mathsf{trm}(S) \ \wedge\ \mathsf{prd}_y(T) \ \Rightarrow$ $\exists x' \cdot (\mathsf{prd}_x(S) \ \wedge\ [x, y := x', y']\, L)$

représentation : formules

$S \sqsubseteq_L T$	$L \wedge \mathsf{trm}(S) \Rightarrow \mathsf{trm}(T)$ $L \wedge \mathsf{trm}(S) \wedge \mathsf{prd}_y(T) \Rightarrow$ $\exists x' \cdot (\mathsf{prd}_x(S) \wedge [x, y := x', y'] L)$

Autre formulation :

$S \sqsubseteq_L T$	$L \wedge \mathsf{trm}(S) \Rightarrow [T] \neg [S] \neg L$

Raffinement de machines : Syntaxe

MACHINE M
VARIABLES x
INVARIANT I
INITIALISATION U
OPERATIONS
$r \longleftarrow nom\_op(w)$ =
PRE P THEN K END
END

REFINEMENT N REFINES M
VARIABLES y
INVARIANT J
INITIALISATION V
OPERATIONS
$r \longleftarrow nom\_op(w)$ =
PRE Q THEN L END
END

Machine associée au raffinement

Initialisation :

$$[V]\ \neg[U]\ \neg J$$

Obligation de preuve pour chaque opération :

$$I \wedge J \wedge P \Rightarrow Q$$

$$I \wedge J \wedge P \Rightarrow [L]\ \neg[K]\ \neg J$$

$$I \wedge J \wedge P \Rightarrow [[r := r']L]\ \neg[K]\ \neg(J \wedge r = r')$$

Exemple de début en raffinement

REFINEMENT $RAFF\_EX1\_R$ REFINES $RAFF\_EX1$

VARIABLES zz

INVARIANT $zz \in \mathsf{NAT}$

$\wedge\ (zz \in yy\ \vee\ (zz = 0\ \wedge\ yy = \varnothing))$

INITIALISATION $zz := 0$

OPERATIONS

$ajouter(nn)$ = PRE $nn \in \mathsf{NAT}_1$

THEN $zz := nn$

END;

$vv \longleftarrow choix$ = $vv := zz$

END

Application à l'exemple : $ajouter$ (1)

Elements de la preuve de $ajouter$: (voir l'exemple)

I	$yy \subseteq \mathsf{NAT}_1$
J	$zz \in \mathsf{NAT} \ \wedge \ (zz \in yy \ \vee \ (zz = 0 \ \wedge \ yy = \varnothing))$
$ajouterA$	$nn \in \mathsf{NAT}_1 \mid yy := yy \cup \{nn\}$
$ajouterC$	$nn \in \mathsf{NAT}_1 \mid zz := nn$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Application à l'exemple : $ajouter$ (1)

Elements de la preuve de $ajouter$: (voir l'exemple)

I	$yy \subseteq \mathsf{NAT}_1$
J	$zz \in \mathsf{NAT} \ \wedge \ (zz \in yy \ \vee \ (zz = 0 \ \wedge \ yy = \varnothing))$
$ajouterA$	$nn \in \mathsf{NAT}_1 \mid yy := yy \cup \{nn\}$
$ajouterC$	$nn \in \mathsf{NAT}_1 \mid zz := nn$

Obligation de preuve de $ajouter$:

$yy \subseteq \mathsf{NAT}_1 \ \wedge$	I
$zz \in \mathsf{NAT} \ \wedge \ (zz \in yy \ \vee \ (zz = 0 \ \wedge \ yy = \varnothing)) \ \wedge$	J
$nn \in \mathsf{NAT}_1$	P

$\Rightarrow$

Application à l'exemple : $ajouter$ (2)

$$I \wedge J \wedge nn \in \mathsf{NAT}_1 \Rightarrow$$
$$[zz := nn] \neg [yy := yy \cup \{nn\}]$$
$$\neg (zz \in \mathsf{NAT} \wedge (zz \in yy \vee (zz = 0 \wedge yy = \varnothing)))$$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Application à l'exemple : $ajouter$ (2)

$$I \wedge J \wedge nn \in \mathsf{NAT}_1 \Rightarrow$$
$$[zz := nn] \neg [yy := yy \cup \{nn\}]$$
$$\neg (zz \in \mathsf{NAT} \wedge (zz \in yy \vee (zz = 0 \wedge yy = \varnothing)))$$

$$I \wedge J \wedge nn \in \mathsf{NAT}_1 \Rightarrow$$
$$[zz := nn] \neg$$
$$(\neg (zz \in \mathsf{NAT} \wedge (zz \in yy \cup \{nn\} \vee (zz = 0 \wedge yy \cup \{nn\} = \varnothing))))$$

Application à l'exemple : $ajouter$ (2)

$$I \wedge J \wedge nn \in \mathsf{NAT}_1 \Rightarrow$$
$$[zz := nn]\neg[yy := yy \cup \{nn\}]$$
$$\neg(zz \in \mathsf{NAT} \wedge (zz \in yy \vee (zz = 0 \wedge yy = \varnothing)))$$

$$I \wedge J \wedge nn \in \mathsf{NAT}_1 \Rightarrow$$
$$[zz := nn]\neg$$
$$(\neg(zz \in \mathsf{NAT} \wedge (zz \in yy \cup \{nn\} \vee (zz = 0 \wedge yy \cup \{nn\} = \varnothing))))$$

$$I \wedge J \wedge nn \in \mathsf{NAT}_1 \Rightarrow$$
$$nn \in \mathsf{NAT} \wedge (nn \in yy \cup \{nn\})$$

Application à l'exemple : $choix$ (1)

Elements de la preuve de $choix$ (voir l'exemple) :

I	$yy \subseteq \mathsf{NAT}_1$
J	$zz \in \mathsf{NAT} \wedge (zz \in yy \vee (zz = 0 \wedge yy = \varnothing))$
$choixA$	$yy \neq \varnothing \mid vv :\in yy$
$choixC$	$vv' := zz$

Application à l'exemple : $choix$ (1)

Elements de la preuve de $choix$ (voir l'exemple) :

I	$yy \subseteq \mathsf{NAT}_1$
J	$zz \in \mathsf{NAT} \ \wedge \ (zz \in yy \ \vee \ (zz = 0 \ \wedge \ yy = \varnothing))$
$choixA$	$yy \neq \varnothing \mid vv :\in yy$
$choixC$	$vv' := zz$

$I \ \wedge \ J \ \wedge \ P \Rightarrow [choixC] \neg [choixA] \neg (J \ \wedge \ (vv = vv'))$

Mais l'état n'est pas modifié.

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Application à l'exemple : $choix$ (1)

Elements de la preuve de $choix$ (voir l'exemple) :

I	$yy \subseteq \mathsf{NAT}_1$
J	$zz \in \mathsf{NAT} \ \wedge \ (zz \in yy \ \vee \ (zz = 0 \ \wedge \ yy = \varnothing))$
$choixA$	$yy \neq \varnothing \mid vv :\in yy$
$choixC$	$vv' := zz$

$I \ \wedge \ J \ \wedge \ P \Rightarrow [choixC] \neg [choixA] \neg (J \ \wedge \ (vv = vv'))$

Mais l'état n'est pas modifié.

$I \ \wedge \ J \ \wedge \ yy \neq \varnothing$

$\Rightarrow$

$[vv' := zz] \neg [vv :\in yy]$

$\neg (zz \in \mathsf{NAT} \ \wedge \ (zz \in yy \ \vee \ (zz = 0 \ \wedge \ yy = \varnothing)) \ \wedge \ vv = vv')$

Application à l'exemple : $choix$ (2)

$$I \ \wedge \ J \ \wedge \ yy \neq \varnothing \Rightarrow$$
$$[vv' := zz] \neg [@vv_1 \cdot (vv_1 \in yy \Longrightarrow vv := vv_1)] \neg (vv = vv')$$

LSR-IMAG

Application à l'exemple : $choix$ (2)

- Intro
- Données
- SubsG
- Machine
- Raffin

$$I \ \wedge \ J \ \wedge \ yy \neq \varnothing \Rightarrow$$
$$[vv' := zz] \neg [@vv_1 \cdot (vv_1 \in yy \Longrightarrow vv := vv_1)] \neg (vv = vv')$$

$$I \ \wedge \ J \ \wedge \ yy \neq \varnothing \Rightarrow$$
$$[vv' := zz] \neg \forall vv_1 \cdot (vv_1 \in yy \Rightarrow \neg (vv_1 = vv'))$$

Application à l'exemple : $choix$ (2)

$I \ \wedge \ J \ \wedge \ yy \neq \varnothing \Rightarrow$
$[vv' := zz] \neg [@vv_1 \cdot (vv_1 \in yy \Longrightarrow vv := vv_1)] \neg (vv = vv')$

$I \ \wedge \ J \ \wedge \ yy \neq \varnothing \Rightarrow$
$[vv' := zz] \neg \forall vv_1 \cdot (vv_1 \in yy \Rightarrow \neg (vv_1 = vv'))$

$I \ \wedge \ J \ \wedge \ yy \neq \varnothing \Rightarrow \exists vv_1 \cdot (vv_1 \in yy \ \wedge \ vv_1 = zz))$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine
- Raffin

Application à l'exemple : $choix$ (2)

$I \ \wedge \ J \ \wedge \ yy \neq \varnothing \Rightarrow$
$[vv' := zz] \neg [@vv_1 \cdot (vv_1 \in yy \Longrightarrow vv := vv_1)] \neg (vv = vv')$

$I \ \wedge \ J \ \wedge \ yy \neq \varnothing \Rightarrow$
$[vv' := zz] \neg \forall vv_1 \cdot (vv_1 \in yy \Rightarrow \neg (vv_1 = vv'))$

$I \ \wedge \ J \ \wedge \ yy \neq \varnothing \Rightarrow \exists vv_1 \cdot (vv_1 \in yy \ \wedge \ vv_1 = zz))$

$yy \subseteq \mathsf{NAT}_1 \ \wedge \ zz \in \mathsf{NAT} \ \wedge$
$(zz \in yy \ \vee \ (zz = 0 \ \wedge \ yy = \varnothing)) \ \wedge \ yy \neq \varnothing \Rightarrow$
$zz \in yy$

Machine abstraite d'un raffinement

MACHINE N

VARIABLES y

INVARIANT $\exists x \cdot (I \wedge J)$

INITIALISATION V

OPERATIONS

$r \longleftarrow nom\_op(w) =$

PRE $Q \wedge \exists x \cdot (I \wedge J \wedge P)$ THEN L END

END

Machine et raffinement

Implémentation (1)

- Dernier raffinement d'un développement
- Langage de programmation séquentielle
- Restriction sur les substitutions
  - substitutions déterministes (:=, IF, CASE, skip, ";")
  - substitution VAR
  - substitution d'itération
- Restriction sur les prédicats : pas de quantificateurs

Implémentation (2)

- Restrictions de typage :
  - les ensembles de valeurs sont finis (ex: entiers représentables)
  - constantes et variables sont de type “concret”: entiers, énumérés, ensembles donnés, tableaux (fonctions totales à domaine fini)
- Les ensembles donnés et les constantes sont valués.
- Les opérations arithmétiques ne doivent pas déborder : obligations de preuves de bonne définition

Plus faible précondition de l’itération

En correction totale, il faut assurer que la boucle se termine dans l’état de la postcondition :

$$[\,\mathcal{W}(P,S,J,V)\,]\,R$$

$$\Leftrightarrow$$

$J \ \wedge$	invariant
$\forall x \cdot ((J \ \wedge \ P) \Rightarrow [S]\,J) \ \wedge$	préservation de l’invariant
$\forall x \cdot (J \Rightarrow V \in \mathbb{N}) \ \wedge$	variant
$\forall x \cdot ((J \ \wedge \ P) \Rightarrow [n := V][S](V < n)) \ \wedge$	décroissance du variant
$\forall x \cdot ((J \ \wedge \ \neg\, P) \Rightarrow R)$	sortie de boucle

Programme de la division entière

```
MACHINE
    DIVISION
OPERATIONS                                  /* qq et rr sont le quotient et le reste */
    qq, rr ⟵ divi(aa, bb) =                 /* de la division entière de aa par bb */
        PRE
            aa ∈ NAT ∧ bb ∈ NAT1
        THEN
            ANY ss, tt WHERE
                ss ∈ NAT ∧ tt ∈ NAT ∧
                aa = bb ∗ ss + tt ∧ tt < bb
            THEN
                qq, rr := ss, tt
            END
        END
END
```


Implémentation de la division entière

```
IMPLEMENTATION DIVISION_I REFINES DIVISION
OPERATIONS
    qq, rr ⟵ divi(aa, bb) =
        VAR ss, tt IN                       /* variables locales auxiliaires */
            ss := 0 ;                       /* initialisations */
            tt := aa ;
            WHILE tt ≥ bb DO
                ss := ss + 1 ;              /* corps de la boucle */
                tt := tt − bb
            INVARIANT                       /* conditions invariantes */
                ...
            VARIANT                         /* valeur entière qui décroît */
                ...
            END ;
            qq := ss ; rr := tt             /* retour du résultat */
```