LSR-IMAG

- Intro
- Données
- SubsG
- Machine

La méthode B

Cours donné à l'Ensimag
Partie 1

Marie-Laure Potet

LSR-IMAG, Grenoble, France

LSR-IMAG

- Intro
- Données
- SubsG
- Machine

Plan du cours

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

Particularités du logiciel

- Intro
- Données
- SubsG
- Machine

- 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

- Intro
- Données
- SubsG
- Machine

- 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 fonctionnelles

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

- Intro
- Données
- SubsG
- Machine

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

- Intro
- Données
- SubsG
- Machine

- 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

LSR-IMAG
- Intro
- Données
- SubsG
- Machine

Partie 2 : Modélisation

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

Bases du formalisme logique : 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
  choice(S) fonction de choix
  S ensemble

LSR-IMAG

- Intro
- Données
- SubsG
- Machine

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	$\exists x \cdot (x \in s) \;\Rightarrow\; \text{choice}(s) \in s$
SET6	$\text{infinite}(\text{BIG})$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine

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 :

$s \subseteq t$	$s \in \mathbb{P}(t)$	
$s \cup t$	$\{a \mid a \in u \;\wedge\; (a \in s \;\vee\; a \in t)\}$	$s \subseteq u \wedge t \subseteq u$
$s \cap t$	$\{a \mid a \in u \;\wedge\; (a \in s \;\wedge\; a \in t)\}$	$s \subseteq u \wedge t \subseteq u$
$s - t$	$\{a \mid a \in u \;\wedge\; (a \in s \;\wedge\; a \notin t)\}$	$s \subseteq u \wedge t \subseteq u$

LSR-IMAG

- Intro
- Données
- SubsG
- Machine

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

- Intro
- Données
- SubsG
- Machine

Autres opérateurs sur les relations

Condition	Expr	Définition
	$\mathsf{id}(s)$	$\{x,y \mid (x,y) \in s \times s \;\wedge\; x = y\}$
$r_1 \in s \leftrightarrow t \;\wedge\; r_2 \in t \leftrightarrow u$	$r_1 \,;\, r_2$	$\{x,z \mid (x,z) \in s \times u \;\wedge\; \exists y \cdot (y \in t \;\wedge\; (x,y) \in r_1 \;\wedge\; (y,z) \in r_2)\}$

Construction des fonctions

Les fonctions sont un cas particulier de relations :

Signification	Notation	Définition
f. partielles	$s \nrightarrow t$	$\{r \mid r \in s \leftrightarrow t \ \wedge \ \forall x, y, z \cdot (x, y \in r \ \wedge \ x, z \in r \Rightarrow y = z)\}$
f. totales	$s \rightarrow t$	$\{f \mid f \in s \nrightarrow t \ \wedge \ \mathsf{dom}(f) = s\}$
injectives part.	$s \rightarrowtail\!\!\!\!\!\!\nrightarrow t$	$\{f \mid f \in s \nrightarrow t \ \wedge \ f^{-1} \in t \nrightarrow s\}$
injectives tot.	$s \rightarrowtail t$	$s \rightarrowtail\!\!\!\!\!\!\nrightarrow t \ \cap \ s \rightarrow t$
evaluation	$f(E)$	$\mathsf{choice}(f[\{E\}])$ si $f \in s \nrightarrow t \ \wedge \ E \in \mathsf{dom}(f)$

Construction des ensembles inductifs

LSR-IMAG
- Intro
- Données
- SubsG
- Machine

On peut définir les ensembles tels que :

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

. . . tout en restant dans la théorie de base . . .

non détaillé ici ...

Les ensembles utilisables directement en B

- Intro
- Données
- SubsG
- Machine

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

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

Exercice de modélisation ensembliste

- Intro
- Données
- SubsG
- Machine

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

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 $mere = pere \mathbin{;} mari^{-1}$

Partie 3 : Les substitutions généralisées

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

Notation mathématique des “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
$S \talloblong T$	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.

LSR-IMAG

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

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 $[S]\,Q$

Plus faible précondition des substitutions primitives

Cas de substitution	Réduction	Condition
$[x := E]\ R$	$[x := E]\ R$	
$[x, y := E, F]\ R$	$[z := F][x := E][y := z]\ R$	$z \backslash E, F, R$
$[\text{skip}]\ R$	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 de plus faible précondition

- $[x := y + 1]\ x > 0 \quad = \quad y + 1 > 0$

Exemple de calcul de plus faible précondition

- $[x := y + 1]\ x > 0 \quad = \quad y + 1 > 0$
- $[y \neq 0 \mid x := x \texttt{ div } y]\ x > 0 \quad = \quad y \neq 0 \wedge\ (x \texttt{ div } y > 0)$

Exemple de calcul de plus faible précondition

- Intro
- Données
- SubsG
- Machine

- $[x := y + 1]\ x > 0 \quad = \quad y + 1 > 0$
- $[y \neq 0 \mid x := x \texttt{ div } y]\ x > 0 \quad = \quad y \neq 0 \wedge\ (x \texttt{ div } y > 0)$

- $$\begin{aligned} &[y := x\ ;\ y \neq 0 \mid x := x \texttt{ div } y]\ x > 0 \\ &\quad = \quad [y := x]\ (y \neq 0 \wedge\ x \texttt{ div } y > 0) \\ &\quad = \quad x \neq 0 \wedge\ x \texttt{ div } x > 0 \\ &\quad = \quad x \neq 0 \end{aligned}$$

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

Le langage des substitutions 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	$@z \cdot (P \Longrightarrow S)$
$x :\in E$	ANY x' WHERE $x' \in E$ THEN $x := x'$ END
$x := \text{bool}(P)$	$x :=$ IF P THEN TRUE ELSE FALSE END
$f(x) := E$	$f := f - \{(x, f(x))\} \cup \{(x, E)\}$

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
  $(P \Rightarrow [S]\ R)\ \wedge\ (\neg\ P \Rightarrow [T]\ R)$

Exemple de calcul de plus faible précondition

LSR-IMAG

- Intro
- Données
- SubsG
- Machine

- $[\ \text{IF}\ P\ \text{THEN}\ S\ \text{ELSE}\ T\ \text{END}]\ R$
  $[(P \Longrightarrow S)\ [\!]\ (\neg\ P \Longrightarrow T)]\ R$ 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 :∈
  $\forall x' \cdot (x' \in E \Rightarrow [x := x']\ R)$ définition wp

Exemples de modélisation : 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)

- Intro
- Données
- SubsG
- Machine

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	
$libres := libres - \{v\} \parallel$	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* si une précondition est vérifiée. On aura ici :

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

LSR-IMAG
- Intro
- Données
- SubsG
- Machine

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	
$libres := libres - \{v\} \ \|\|$	modification de l'état
$r := v$	retour de l'adresse allouée
END	
ELSE	il n'y a plus d'adresse libre
$r := null$	retour de la valeur de non allocation
END	

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 WHERE	
$v \in libres$	
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 ?

Plus faible précondition de l'itération

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

$[$ WHILE P DO S
INVARIANT J
VARIANT V END $]\, R$

$\Leftrightarrow$

$J \;\wedge$	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

- Intro
- Données
- SubsG
- Machine

```
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

- Intro
- Données
- SubsG
- Machine

```
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 */
      END
END
```

LSR-IMAG

- Intro
- Données
- SubsG
- Machine

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

LSR-IMAG

- Intro
- Données
- SubsG
- Machine

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

END

Rubriques d'une machine

MACHINE M	
SETS S;	/* ensembles donnés */
$T = \{a, b\}$	/* ensembles énumérés */
CONSTANTS c	/* liste de constantes */
PROPERTIES C	/* spécification des constantes */
VARIABLES x	/* liste de variables */
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	

Obligations de preuves d'une machine

Hypothèses sur les SETS :

$$B \;\equiv\; S \subseteq INT \wedge S \neq \emptyset \wedge T \subseteq INT \wedge T = \{a, b\} \wedge a \neq b$$

- L'initialisation établit l'invariant :
  $$B \;\wedge\; C \;\Rightarrow\; [U]I$$
- Chaque opération préserve l'invariant :
  $$B \;\wedge\; C \;\wedge\; I \;\wedge\; P \;\Rightarrow\; [K]I$$

Spécification et vérification

- Intro
- Données
- SubsG
- Machine

Ecrire la spécification (dynamique et invariant), produire les obligations de preuve et essayer la preuve automatique

- Obligation de preuve prouvée automatiquement : OK
- Echec de la preuve automatique : 2 cas
  1. incomplétude de la preuve automatique
  2. la formule n'est pas un théorème

Rappel : semi-décidabilité de la logique du premier ordre

Une propriété intéressante pour la preuve :
$[S]\ (I_1 \wedge\ I_2) = [S]\ I_1 \wedge\ [S]\ I_2$

Correction

- Intro
- Données
- SubsG
- Machine

1. Preuve interactive
   (a) trouver la preuve (invention)
   (b) vérifier la preuve (décidable)

2. corriger la spécification :
   (a) renforcer l'invariant
   (b) mettre en accord l'invariant et les opérations

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)

- Intro
- Données
- SubsG
- Machine

- Restrictions de typage :
  - les ensembles de valeurs sont finis (ex: entiers représentables)

$$
\begin{array}{ll}
NAT = 0..MAXINT & avec\ NAT \subseteq NATURAL \\
INT = MININT..MAXINT & avec\ INT \subseteq INTEGER
\end{array}
$$

  - 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

Substitutions dans les machines

- Intro
- Données
- SubsG
- Machine

Les substitutions généralisées suivantes ne sont pas autorisées dans les machines abstraites :

- le séquencement (spécificable par une affectation parallèle)
- la substitution var qui est un cas particulier de la substitution any
- la substitution itération

La modélisation ensembliste permet de ne pas avoir besoin d'introduire des itérations dans un modèle abstrait. Par exemple soit tab un tableau modélisé par : $tab \in 0..5 \rightarrow INT$. Rechercher un élément i dans le tableau peut être décrit par la substitution : $bool(i \in ran(tab))$.