Introduction à la Méthode B

Yamine AIT-AMEUR

IRIT/INPT-ENSEEIHT
yamine@n7.fr

04 Février 2014

Plan

Plan

Quels sont les modèles traités ?

Modèles caractérisés par

- un état i.e. ensemble de variables
- ensemble d'actions pouvant modifier l'état i.e. les variables

1. Quelles sont les propriétés satisfaites par ces modèles ?
2. Comment prouver ces propriétés ?
3. Comment obtenir du code à partir de ces modèles ?

Développement de programmes avec B

- Développement formel de programmes impératifs à partir de spécification

Méthode

- Raffinement de la spécification jusqu'au code.
- Succession de modèles liés par une relation de raffinement
- Le dernier modèle est écrit dans un langage proche du code

Développement de programmes avec B : la spécification

Machine Abstraite

1. Décrire les *variables d'état* du système,
2. Exprimer l'*invariant* i.e. propriétés satisfaites par les *variables d'état*
   - à l'initialisation,
   - avant et après toute *opération* manipulant les variables d'état,
3. Spécifier enfin ces opérations

Développement de programmes avec B : le raffinement

1. Décrire (ou réécrire) les *variables d'état* du modèle raffiné
2. Exprimer l'*invariant* i.e. propriétés satisfaites par les (nouvelles) variables d'état + l'invariant de collage
   - à l'initialisation
   - avant et après toute *opération* manipulant les variables d'état
3. Réécrire (préciser) les opérations avec les (nouvelles) variables

Attention, certaines constructions ne sont pas permises au niveau de la machine et des raffinements

$\Longrightarrow$ Besoin d'exprimer, ensembles, propriétés, opérations et transformations d'états, machines et raffinements.

Plan

Développement de programmes avec B : le langage de modélisation

Expression de variables, constantes, opérations, invariants, théorèmes, $\cdots$

1. Théorie des ensembles

$$SIEGES = 1..nb\_max$$
$$occupes \subseteq SIEGES$$

2. Fonctions, relations

$$etat \in SIEGES \rightarrow BOOL$$
$etat^{-1}$ fonction réciproque.

3. Logique des prédicats

$$occupes = etat^{-1}[\{TRUE\}]$$
$$x \in 1..1280 \wedge y \in 1..1024$$

Des opérations sont définies pour manipuler et construire des ensembles, relations, fonctions, et prédicats

Développement de programmes avec B : le langage de modélisation

4. Substitutions généralisées
   - Elles permettent de décrire les changements d'états

ANY
x', y'
WHERE
$x' \in 1..1280 \ \wedge y' \in 1..1024$
THEN
$x := x' \ || \ y := y'$
END ;

PRE
$x + 100 \leq 1280 \ \wedge y + 100 \leq 1024$
THEN
$x := x + 100 \ || \ y := y + 100$
END ;

Substitutions généralisées

- Substitutions permettant la modification d'une variable

Notation	Nom
skip	substitution identité
$x := e$	substitution devient égal simple
$x, y := e, f$	substitution devient égal multiple
$x := e \ \|\| \ y := f$	
$x : (P)$	substitution devient tel que

- Substitutions permettant de structurer une opération

Syntaxe de la substitution	Nom de la substitution
BEGIN S END	bloc
PRE P THEN S END	pré-condition
ANY l WHERE G THEN S END	choix non borné
...	...

Calcul des substitutions généralisées

- Obligation de preuve : théorème à prouver
- Triplet de Hoare $\{P\}\ S\ \{Q\}$
- Weakest Precondition (WP) de Dijkstra $P \equiv [S]Q$
- Plus faible pré-condition P qui établit Q après S

$[S]Q$	WP
$[x := E]Q$	$Q(E \rightarrow x)$
Exemple $[x := y + 5](x > y)$	$y + 5 > y$

$[S]Q$	WP
$[\ \text{PRE}\ P\ \text{THEN}\ S\ \text{END}]Q$	$P \wedge [S]Q$
Exemple $[\ \text{PRE}\ x > 0$ $\text{THEN}\ x := y + 5$ $\text{END}\ ](x > y)$	$(x > 0) \wedge$ $(y + 5 > y)$

- Calcul de plus faibles pré-conditions $\Longleftrightarrow$ Principe d'extraction des obligations de preuve (OP)

Calcul des substitutions généralisées

$[S]Q$	WP
$[x := E]Q$	$Q(E \rightarrow x)$
[PRE P THEN S END]Q	$P \wedge [S]Q$
[ANY l WHERE G THEN S END]Q	$\forall l.(G \Rightarrow [S]Q)$
[IF P THEN $S1$ ELSE $S2$ END]Q	$(P \Rightarrow [S1]Q) \wedge (not\ P \Rightarrow [S2]Q)$
[SELECT P THEN S END]$Q1$	$P \Rightarrow [S]Q$
...	...

- Autres substitutions CHOICE ,LET , WHILE ...
- Utilisées en fonction du niveau de développement

- Synthèse automatique des obligations de preuve (OP)
- Les OP sont soumises à un démonstrateur de théorème (theorem prover)
- Preuves semi-automatiques, automatiques

Plan

Structure globale d'une machine abstraite

MACHINE *nom de la machine abstraite*

Partie statique

déclaration d'ensembles
déclaration de constantes
variables (état)
invariant (caractérisation de l'état)
assertions (théorèmes)

Partie dynamique

initialisation de l'état
opérations sur l'état

END

Machine Abstraite : partie statique

MACHINE *nom de la machine*

Partie en-tête

SETS

déclaration d'ensembles

CONSTANTS

déclaration de constantes

PROPERTIES

définition de propriétés sur les constantes

VARIABLES

déclaration de variables (état)

INVARIANT

définition de propriétés invariantes (caractérisation de l'état)

ASSERTIONS

Lemmes portant sur les variables

...

Partie dynamique

...

END

Machine Abstraite : partie Dynamique

- Utilisation de substitutions généralisées
- Structuration d'une opération

Syntaxe de la substitution	Nom de la substitution
BEGIN S END	bloc
PRE P THEN S END	pré-condtion
ANY I WHERE G THEN S END	choix non borné

- Modification de variables

Notation	Nom
skip	substitution identité
$x := e$	substitution devient égal simple
$x, y := e, f$	substitution devient égal multiple
$x : (P)$	substitution devient tel que

Machine Abstraite : partie Dynamique

- Composition de substitutions généralisées
- Exemples d'opérations

```
nom_op1 =
BEGIN
    s1 ||
    s2 ||
    ...
    sn
END
```

```
nom_op2 (w) =
PRE P
THEN
    s1 ||
    s2 ||
    ...
    sn
END
```

```
nom_op3 =
ANY l
WHERE G
THEN
    s1 ||
    s2 ||
    ...
    sn
END
```

```
nom_op4 (w) =
PRE P THEN
    ANY l
    WHERE G
    THEN
        s1 ||
        s2 ||
        ...
        sn
    END
END
```

Machine Abstraite : forme globale

MACHINE *M (paramètre_1, ..., paramètre_n)*
CONSTRAINTS
  X
SETS
  E
CONSTANTS
  c
PROPERTIES
  C
VARIABLES
  v
INVARIANT
  I
ASSERTIONS
  A de la forme $A_1 \wedge A_2 \wedge \cdots \wedge A_n$
DEFINITIONS
  D
INITIALISATION
  i
OPERATIONS
  $u \leftarrow \text{nom\_op}_1\ (w) = \text{PRE}\ P\ \text{THEN}\ S\ \text{END}$;
  (...)
END

- On pose $B = X \wedge E \wedge C$

Machine Abstraite : un exemple

MACHINE Pixel

VARIABLES x, y

INVARIANT
$x \in 1..1280 \land y \in 1..1024$

INITIALISATION
$x, y := 1, 1$

OPERATIONS

$posX \leftarrow \text{abs} = posX := x$;

$posY \leftarrow \text{ord} = posY := y$;

random =
ANY x', y'
WHERE $x' \in 1..1280 \land y' \in 1..1024$
THEN
$x := x' \;||\; y := y'$
END ;

move (dx, dy) =
PRE $dx \in \mathbb{Z} \land dy \in \mathbb{Z} \land (x + dx) \in 1..1280 \land (y + dy) \in 1..1024$
THEN
$x, y := (x + dx), (y + dy)$
END
END

Machine Abstraite : cohérence

1. OP : *assertions*
   $$B \wedge I \wedge A_1 \wedge \ldots \wedge A_{i-1} \Rightarrow A_i$$
   Ordre de définition des assertions significatif
2. OP : *initialisation* $A \wedge B \Rightarrow [i]I$
3. OP : *opération* $\text{OPER} = \text{S}$ OP associée $A \wedge B \wedge I \Rightarrow [S]I$
   - $\text{OPER} = \text{S BEGIN } S \text{ END};$
     $$A \wedge B \wedge I \Rightarrow [S]I$$
   - $\text{OPER} = \text{S PRE } P \text{ THEN } S \text{ END};$
     $$A \wedge B \wedge I \wedge P \Rightarrow [S]I$$
   - $\text{OPER} = \text{S ANY } I \text{ WHERE } G \text{ THEN } S \text{ END};$
     $$A \wedge B \wedge I \Rightarrow (\forall I.(G \Rightarrow [S]I))$$
   - $\ldots$

Machine Abstraite : exemple de cohérence

1. A, et B non renseignés
2. Initialisation. $[i]I$

   - $I \iff x \in 1..1280 \ \wedge y \in 1..1024$
   - $[i]I$
     $\iff [x, y := 1, 1](x \in 1..1280 \ \wedge y \in 1..1024)$
     $\iff 1 \in 1..1280 \ \wedge 1 \in 1..1024$

3. Opérations move . Pré-condition $I \wedge P \Rightarrow [S]I$.

   - $I \iff x \in 1..1280 \ \wedge y \in 1..1024$
   - $P \iff dx \in \mathbb{Z} \ \wedge dy \in \mathbb{Z} \ \wedge (x + dx) \in 1..1280 \ \wedge (y + dy) \in 1..1024$,
   - $I \wedge P \Rightarrow [S]I$
     $\iff$
     $I \wedge P \Rightarrow [x, y := (x + dx), (y + dy)](x \in 1..1280 \ \wedge y \in 1..1024)$
     $\iff I \wedge P \Rightarrow (x + dx) \in 1..1280 \ \wedge (y + dy) \in 1..1024$

Le Raffinement

- Plusieurs raffinements peuvent être proposés pour une même spécification
- Denier niveau = *implantation*
- Traduction automatique dans un langage de programmation

Le raffinement : forme globale

MACHINE $M\ (par\_1, \ldots, par\_n)$
CONSTRAINTS
X
SETS
E
CONSTANTS
c
PROPERTIES
C
VARIABLES
v
INVARIANT
I
ASSERTIONS
A
DEFINITIONS
D
INITIALISATION
i
OPERATIONS
$u \leftarrow \mathrm{op}_1\ (w) = \mathrm{PRE}\ P\ \mathrm{THEN}\ S\ \mathrm{END}$;
(...)
END

REFINEMENT $N\ (par\_1, \ldots, par\_n)$
REFINES
M
SETS
K
CONSTANTS
d
PROPERTIES
O
VARIABLES
v'
INVARIANT
J
ASSERTIONS
A'
INITIALISATION
i'
OPERATIONS
$u \leftarrow \mathrm{op}_1\ (w) = \mathrm{PRE}\ Q\ \mathrm{THEN}\ T\ \mathrm{END}$;
(...)
END

Le raffinement : cohérence

- J contient l'invariant de collage *"gluing invariant"*
- Assertions conséquences des déclarations de l'invariant

$$A \wedge B \wedge B' \wedge I \wedge J \wedge A'_1 \wedge ... \wedge A'_{n-1} \Rightarrow A'_n$$

- Initialisation concrète raffinement initialisation abstraite

$$A \wedge B \wedge B' \Rightarrow [i']\neg[i]\neg J$$

- Chaque raffinement d'opération est correct.

$$B \wedge B' \wedge A \wedge A' \wedge I \wedge P \Rightarrow Q \wedge [T]\neg[S]\neg J$$

Plan

Une machine et son raffinement : la machine RESERVATION

```
MACHINE      RESERVATION
CONSTANTS    nb_max, SIEGES
PROPERTIES
    nb_max ∈ NAT1 ∧      SIEGES = 1..nb_max
VARIABLES    occupes, nb_libre
INVARIANT
    occupes ⊆ SIEGES ∧      nb_libre ∈ 0..nb_max ∧      nb_libre = nb_max − card(occupes)
INITIALISATION
    occupes, nb_libre := ∅, nb_max
OPERATIONS
    nb ← place_libre =
            BEGIN
                nb := nb_libre
            END ;
    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 ;
    rr ← liberer(place)  =
            PRE place ∈ SIEGES
            THEN
                IF place ∈ occupes
                THEN rr, occupes, nb_libre := TRUE, occupes − {place}, nb_libre + 1
                ELSE rr := FALSE
                END
            END
END
```

Une machine et son raffinement : le raffinement RESERVATION1

REFINEMENT
  *RESERVATION*1
REFINES
  RESERVATION
VARIABLES
  etat, *nb_libre*
INVARIANT
  $etat \in SIEGES \rightarrow BOOL \wedge$
  $occupes = etat^{-1}[\{TRUE\}]$
INITIALISATION
  $nb\_libre := nb\_max \;||$
  $etat := SIEGES * FALSE$
OPERATIONS
  $place \leftarrow$ reserver $=$
    ANY $pp1$ WHERE $pp1 \in etat^{-1}[\{FALSE\}]$
    THEN
      $place := pp1 \;||$
      $etat(pp1) := TRUE \;||$
      $nb\_libre := nb\_libre - 1$
    END ;
  $rr \leftarrow$ liberer $(place) =$
    BEGIN
      IF etat(place) = TRUE
      THEN rr, $nb\_libre := TRUE, nb\_libre + 1$
      ELSE $rr := FALSE$
      END
      || etat(place) := FALSE
    END
END

Plan

B

- Mécanismes de modularité pour structurer
  - les développements
  - les preuves
- Traduction vers du code
- Nombreux outils disponibles : animateurs, model checkers, provers, visualisation, etc.
- AtelierB

Event B

- Description de systèmes à base d'évènements
- Définition de systèmes de transitions
  - raffinements de systèmes de transitions
  - expression et preuve de propriétés temporelles
- Nombreux outils disponibles

- B et Event B
  - B pour la production de code
  - Event B pour la description des systèmes

FIN

Conférence ABZ 2014 à Toulouse
2 au 6 Juin 2014
www.irit.fr/ABZ2014/
Etude de cas du Landing Gear