Introduction aux machines abstraites

Objectif des méthodes formelles

Le but des méthodes formelles comme la méthode B est de permettre la spécification et la conception de programmes :

- En utilisant un langage suffisamment formel pour permettre l'analyse et la preuve de propriétés.
- Ce langage ressemble aux langages de programmations usuels (on peut trouver x:=x+1)
- Ce langage contient aussi des choses plus inhabituelles (on peut trouver x devient tel qu'il respecte une certaine propriété sans se demander s'il existe un tel x, ou s'il en existe plusieurs)

Objectif des méthodes formelles

- Le but est de décrire le plus abstraitement possible le quoi et non pas le comment
- Le comment est ensuite dérivé du quoi par un processus de raffinements successifs : disparition progressive de l'abstraction, passage progressif au concret pour expliquer comment on trouve la solution (unique choisie) du quoi.
- Le dernier raffinement (appelé implantation) ne contient plus du tout ce qui fait l'abstraction (et la richesse) des machines. C'est (pratiquement) du code, mais compatible avec ce qui a été dit dans la machine.

Machines abstraites

- L'entité de base en méthode B est la machine abstraite, caractérisée par :
  - Un état défini par les valeurs de ses variables
  - Des opérations qui modifient cet état
- La spécification de la machine comprend :
  - La statique (définition de l'état de la machine et des propriétés qu'il doit satisfaire) : utilise la logique des prédicats et les notions ensemblistes.
  - La dynamique (définition des opérations) : utilise les substitutions qui peuvent contenir des opérations ensemblistes.
- Les machines font ensuite l'objet de raffinements, le dernier raffinement étant l'implantation.

Structure d'une machine abstraite

- En pratique une machine non vide contient au moins
  - Un nom de MACHINE (obligatoire)
  - Une liste de VARIABLES (sauf en cas de machine, pas très utile, ne manipulant pas de variable donc ne changeant pas d'état)
  - Un INVARIANT (propriété, exprimée sous forme de prédicat, devant rester toujours vraie, concernant les variables) : au minimum le « typage » des variables
  - Une INITIALISATION (état initial de la machine sous forme de substitutions portant sur les variables)
  - Des OPERATIONS (services de la machine, qui font évoluer son état, sous forme de substitutions)

Structure d'une machine abstraite

- Le B est fondé sur un socle mathématique d'une grande rigueur utilisant :
  - La théorie des ensembles : notions d'appartenance et d'inclusion, relations et fonctions et leurs cas particuliers (injections, surjections, bijections…)
  - La logique mathématique et plus particulièrement la logique des prédicats (affirmations logiques pouvant dépendre de variables quantifiées par ∀ : quel que soit et ∃ : il existe).
  - Des types de base numériques (entiers, pas de réels), booléens, chaînes de caractères, mais aussi suites, arbres, records, ainsi que des types abstraits (ensembles abstraits) dont l'implantation sera précisée en temps utile.

Notations dans les machines B

Le B se note en utilisant les symboles mathématiques mais peut aussi se noter sous forme de fichiers texte :

MATH	ASCII	Signification
$x \in S$	x : S	Appartenance
$x \notin S$	x /: S	Non appartenance
$S \subseteq T$	S <: T	Inclusion
$S \not\subset T$	S /< T	Non inclusion
$S \subset T$	S <<: T	Inclusion stricte
$S \cup T$	S \/ T	Union
$S \cap T$	S /\ T	Intersection

Voir Liste des mots réservés et des opérateurs du langage B pour liste complète

Il faut s'habituer à la notation ASCII utilisée dans les fichiers source des outils (dont l'atelier B utilisé en TP)

Notations dans les machines B

D'autres notations permettent de manipuler tous les types possibles de relations ensemblistes

ASCII	Libellé
S<->T	Ensemble des relations de S dans T
S+->T	Ensemble des fonctions partielles de S dans T
S-->T	Ensemble des fonctions totales de S dans T
S>+>T	Ensemble des injections partielles de S dans T
S>->T	Ensemble des injections totales de S dans T
S+->>T	Ensemble des surjections partielles de S dans T
S-->>T	Ensemble des surjections totales de S dans T
S>+>>T	Ensemble des bijections partielles de S dans T
S>->>T	Ensemble des bijections totales de S dans T

Exemples de machines abstraites

9 Pour typer les variables le B connaît les entiers naturels (NATURAL) ou relatifs (INTEGER) ainsi que les booléens (BOOL) : TRUE ou FALSE et les chaînes de caractères (STRING)

Pas de réels en méthodes formelles (où le but est d'effectuer des preuves)

```
MACHINE
        GestTrains
VARIABLES
        nbTrains
INVARIANT
        nbTrains:NATURAL /* Typage de la variable
INITIALISATION
        nbTrains:=0
END
```

Entiers abstraits, entiers concrets

Le B sait que les entiers représentables en machine (les entiers dits concrets) sont forcément bornés et définit donc :

MAXINT : le plus grand entier représentable (dépend de la machine cible mais sera toujours défini par MAXINT)

MININT : le plus petit entier représentable (dépend de la machine cible mais sera toujours défini par MININT)

Les types mathématiques restent utilisables mais limités aux machines et raffinements (interdits dans les implantations, dernière étape jusqu'au code)

Les types concrets (bornés) sont obligatoires dans les implantations.

Types entiers manipulables en B

INTEGER : entiers relatifs (machines, raffinements, interdits dans les implantations)

NATURAL : entiers naturels (machines, raffinements, interdits dans les implantations)

NATURAL1 : entiers naturels non nuls (machines, raffinements, interdits dans les implantations)

NAT : entiers naturels concrets [0..MAXINT] (machines, raffinements, implantations)

NAT1 : entiers naturels concrets non nuls [1..MAXINT] (machines, raffinements, implantations)

INT : entiers relatifs concrets [MININT..MAXINT] (machines, raffinements, implantations)

Exemples de machines abstraites

Le B permet également de manipuler des ensembles abstraits (clause SETS) ou « différés » (deffered), qui sont juste supposés finis non vides, et de typer des variables comme éléments de ces ensembles.

Très utile pour différer à plus tard l'aspect implémentation

MACHINE GestSignal

SETS AIGUILLE ; SIGNAL

VARIABLES Aiguille , Signal , Protege

INVARIANT Aiguille <: AIGUILLE & Signal <: SIGNAL & Protege : Signal >->> Aiguille /*Toute aiguille est protégée par un signal

INITIALISATION Aiguille,Signal,Protege:={},{},{}

END

Exemples de machines abstraites

L'invariant est l'endroit de choix pour exprimer les propriétés de sécurité en utilisant la puissance expressive de la théorie des ensembles :

Ici : Tout train est dans un canton (fonction totale) et un canton contient au maximum un train (injection).

MACHINE GestTraffic

SETS CANTONS ; TRAINS

VARIABLES Cantons , Trains , PositionTrain

INVARIANT Cantons<:CANTONS & Trains<:TRAINS & PositionTrain : Trains >-> Cantons

INITIALISATION

Cantons,Trains,PositionTrain:={},{},{}

END

Exemples de machines abstraites

Les services de la machine sont spécifiés par les OPERATIONS, exprimées comme l'initialisation par des substitutions. Les opérations font évoluer l'état de la machine et doivent donc préserver l'invariant.

```
MACHINE    GestTrains2
VARIABLES    nbTrains
INVARIANT    nbTrains:NATURAL
INITIALISATION    nbTrains:=0
OPERATIONS
AddTrain=
nbTrains:=nbTrains+1 /*reste bien un NATURAL*/
END
```

Processus de preuve

L'intérêt de la spécification et de la conception formelle est de permettre de prouver (mathématiquement) que les propriétés exprimées restent vraies (quelle que soit l'évolution de la machine). Dans ce qui suit on évoquera :

- Preuve d'initialisation : la substitution initiale place la machine dans un état où l'invariant est vrai.
- Preuve d'opération : si l'invariant est vrai avant l'opération alors il le sera après.
- Preuve de raffinement : le raffinement (qui raffine une machine) est compatible avec la machine (i.e. ne fait pas ce que la machine ne ferait pas).

Donc toute propriété exprimée (dans l'invariant d'une machine) entre dans un processus de preuve qui va jusqu'à l'implantation (i.e. pratiquement jusqu'au code).