Introduction aux obligations de preuves

Rappels sur les substitutions

- Effectuer une substitution (exemple [x:=x+1]) dans un prédicat consiste à remplacer toutes les occurrences libres (non concernées par la portée d'un quantificateur universel ou existentiel), de la variable à gauche du := par l'expression figurant à droite du :=
- Ainsi $(\exists x.(x \in NAT \wedge x \neq 0)) \Rightarrow y=x$ est un prédicat (très mal écrit !) qui contient des occurrences libres de x (concernées par une substitution) et d'autres non libres.
- [x:=x+1] $((\exists x.(x \in NAT \wedge x \neq 0)) \Rightarrow y=x)$ donne $(\exists x.(x \in NAT \wedge x \neq 0)) \Rightarrow y=x+1$
- Les variables liées ne peuvent donc être substituées, mais afin d'éviter toute ambiguïté on peut les renommer $(\exists z.(z \in NAT \wedge z \neq 0)) \Rightarrow y=x$ est tout de suite plus clair !

Obligation de preuve d'initialisation

- L'initialisation d'une machine (clause INITIALISATION) contient des substitutions type variables_machine:=valeurs initiales
- L'état initial de la machine doit respecter l'invariant, ce qui constitue une première propriété à prouver.
- D'une manière générale ces propriétés à prouver pour une machine B s'appellent obligations de preuve (Proof Obligations : POs)
- Elles doivent être prouvées à partir d'éléments de base (axiomes) en utilisant des règles d'inférence.
- L'obligation de preuve d'initialisation consiste donc à prouver que [S]I est vrai où S est la substitution initiale et I l'invariant.

Exemple

Si comme dans un exemple déjà vu on a :

INVARIANT nbTrains:NATURAL

INITIALISATION nbTrains:=0

L'obligation de preuve d'initialisation va être

- [nbTrains:=0]nbTrains:NATURAL
- Soit 0:NATURAL

Des outils comme l'atelier B savent prouver tous seuls que 0 est un entier naturel, à partir de leur bases d'axiomes et de règles d'inférence.

Obligation de preuve d'opération

- Les opérations d'une machine (clause OPERATIONS) contiennent des substitutions faisant évoluer les variables (exemple x:=x-1) où :
  - Les identifiants à gauche de := réfèrent aux valeurs de variables à l'issue de l'opération
  - Les identifiants à droite de := réfèrent aux valeurs de variables avant de l'opération
- Obligation de preuve d'opération : doit exprimer le fait que si l'invariant est vrai avant l'opération alors il le sera également après l'opération

Notion de plus faible précondition

- P étant un prédicat et S une substitution, [S]P désigne le prédicat obtenu en appliquant la substitution *S* sur le prédicat P.
- Si [S]P est vrai, on dit que S établit P
- [S]P peut être vu comme la plus faible précondition (Weakest Precondition : WP, notion introduite par E.W. Dijkstra) devant être vérifiée avant l'action de S, pour que P soit vraie après l'action de S.
- Dijkstra notait wp(S,P) ce que le B note [S]P
- En logique de Hoare, où l'on note {Pré-condition} Instruction {Post-Condition} on noterait donc {[S]P} S {P}

Notion de plus faible précondition

Exemple :

- P : x>0 (Postcondition devant être vraie après S)
- S : [x:=x-1] (Substitution)
- [S]P : x-1>0 soit x>1
- Donc si x>1 (avant x:=x-1) alors x>0 (après x:=x-1)
- Plus faible précondition car pour toute condition plus forte (i.e. impliquant [S]P, la postcondition est vérifiée)
- Ainsi x>2 ⇒x > 1 donc si x > 2 (avant x:=x-1)

Obligation de preuve d'opération

L'obligation de preuve d'une opération de substitution S consiste tout simplement à écrire que l'invariant est plus fort (implique) que la plus faible précondition établissant I après S soit :

$$I \Rightarrow [S]I$$

- Ce qui se lit : si l'invariant I est vrai alors la substitution S établit I
- Cette écriture est un peu simplifiée : le prédicat correct est que quelles que soient les variables de machines vérifiant I, alors S établit I, soit :

$$\forall(\text{variables de machine}).(I \Rightarrow [S]I)$$

Exemple

Si comme dans un exemple déjà vu on a :

INVARIANT nbTrains:NATURAL

OPERATIONS

AddTrain=nbTrains:=nbTrains+1

L'obligation de preuve de l'opération va être

- nbTrains:NATURAL⇒ [nbTrains:=nbTrains+1]nbTrains:NATURAL
- nbTrains:NATURAL⇒nbTrains+1:NATURAL

Des outils comme l'atelier B savent prouver tous seuls que le successeur d'un entier naturel est un entier naturel (pour mémoire c'est un des axiomes de Péano).

Mais l'entier naturel est un concept mathématique abstrait qui n'a pas cours en informatique…

Exemple

En implantation donc (où il faut des entiers concrets), on pourrait donc avoir :

INVARIANT nbTrains:NAT

OPERATIONS

AddTrain=nbTrains:=nbTrains+1

L'obligation de preuve devient

- nbTrains:NAT⇒nbTrains+1:NAT

Que l'atelier B génère bien mais ne parvient pas à prouver car elle est fausse ! voici une partie de la PO générée automatiquement (le grand entier représente MAXINT sur la machine utilisée dans l'exemple, soit 2^31-1) :

nbTrains <=2147483647=> nbTrains+1<=2147483647

La méthode B inclut donc par construction, une protection contre les débordements de variable !