Spécifications formelles avec B : TP1

1 Premiers pas avec l'atelier B

L'atelier B est un atelier de génie logiciel basé sur la méthode B. Initialement développé par GEC Alsthom Transport, il est actuellement maintenu et commercialisé par la société ClearSys.

1.1 Lancement et aide en ligne

L'atelier B est disponible sur la machine `marine`. On lance l'interface graphique de l'atelier en tapant: **startAB &**.

Lancez l'interface, examinez les différents boutons et accedez à l'aide en ligne (une fenêtre netscape). En utilisant `acroread`, visualisez les liens suivants :

1. `Le langage B - Manuel de référence` ;
2. `Liste des mots réservés et opérateurs du langage B` ;

Ils vous seront utiles pour la suite.

1.2 Gestion des projets

L'interface graphique s'ouvre sur la fenêtre des *projets*. Un projet développé avec l'atelier B comporte des sources B (machines abstraites, raffinement et implémentations), des obligations de preuve (générées automatiquement), des preuves (réalisées de façon semi-automatique) et, éventuellement, de la documentation et des traductions en `C`,`C++`,`Ada`. Un projet peut aussi faire appel à des librairies standard comme `BASIC_IO` qui gère les entrées-sorties.

Un projet se définit par : son nom, un repertoire *Base de Données Projet* (`bdp`) contenant les fichiers propres à l'atelier B, un repertoire `trad` pour stocker les *traductions* vers des langages classiques, et un repertoire `spec` dans lequel reposent les spécifications des machines B.

2 Un projet minimum

Nous vous demandons de spécifier et d'implémenter une machine qui calcule le minimum de deux entiers naturels. Commencez par créer un repertoire `Minimum` avec trois sous repertoires `bdp`, `spec` et `trad` (`mkdir Minimum`, `cd Minimum`, `mkdir pdb`, `mkdir spec`, `mkdir trad`, `cd ..`, `ls`, `pwd`).

Lancez l'atelier B et créez un projet (Bouton `Attach`). Nommez ce projet de façon unique (par exemple `Minimum-votre_login`) et indiquez respectivement `Minimum/bdp` et `Minimum/trad` comme repertoires de Base de Donnée et de Traduction. Avant d'ouvrir le projet ainsi créé, attachez lui la librairie standard `LIBRARY` (`Libraries/Add`) pour qu'il puisse accéder à l'implémentation de la machine `BASIC_IO`.

Exercice 1 *Dans le repertoire* `Minimum/spec`*, lancez* `emacs` *et créez un fichier* `CalMin.mch`*, dans lequel vous spécifierez une machine B qui fournie une fonction* `Minimum`*. Ajoutez ce composant au projet* `Minimum` *(* `Components/Add` *).*

Vous pouvez compléter le code suivant :

```
MACHINE CalMin

OPERATIONS
bb <-- Minimum(mm,nn) = PRE
    ???
  THEN
    bb:(???)
  END
END
```

Exercice 2 *Testez (et corrigez si necessaire) la syntaxe de* `CalMin`. *(* `Type Check` *).*

Exercice 3 *Générez les obligations de preuves (* `PO Generate` *).*
Si nécessaire prouvez les (`Prove/Automatic (force 0)` *puis* `BO Check` *).*

Exercice 4 *Dans un fichier que vous appellerez* `CalMin_Imp.imp`*, implémentez la machine* `CalMin`*. Ajoutez ce composant au projet* `Minimum` *et prouvez le.*

Vous pouvez compléter le code ci-dessous :

```
IMPLEMENTATION CalMin_Imp
REFINES CalMin

OPERATIONS

bb <-- Minimum(mm,nn) =
  BEGIN
    ???
  END
END
```

Exercice 5 *Dans le repertoire* `/users/linfg/urvoy/B/Tp/tp1` *vous trouverez les trois fichiers* `Minimum_Int.mch` `Minimum_Int_Imp.imp` *et* `BASIC_IO.mch`*. Copiez les dans votre propre repertoire* `Minimum/spec` *(*`man cp`*), ajoutez les au projet et prouvez les.*

Exercice 6 *Generez du code* `c++` **natif** *à partir du composant* `Minimum_Int_Imp` *(* `Translator` *).*
Compilez (commande `make`*) et executez ce code (repertoire* `Minimum/trad/cpp`*).*

3 Addition bornée

Créez un projet `Addition` et une machine `ZCent`. On consière ici des calculs limités à l'ensemble **0..100** des entiers entre zéro et cent.

Exercice 7 *Spécifiez et implementez une fonction Add qui effectue l'addition de deux entiers entre zéro et cent et qui retourne un entier entre zéro et cent.*

Exercice 8 *Spécifiez et implementez une fonction Mul2 qui effectue la multiplication par deux d'un entier entre zéro et 100 et qui retourne un entier entre zéro et cent.*

4 Un exemple de projet B

Exercice 9 *Ouvrez le projet* `DAB`*, affichez graphe de dépendance du projet et son status. Combien reste t-il d'obligations de preuve non démontrées ? Executez le code* `c++` *généré.*