BoB

Manuel d'utilisation

25 février 2005

Table des matières

Chapitre 1

Installation

1.1 Pré-requis

Avant d'installer la boîte à outil B, vous devez vous assurer que vous disposez bien d'une plate-forme java 2. La version standard (J2SE) est largement suffisante. Si vous n'avez pas Java, allez sur le site http ://java.sun.com et téléchargez la version de Java correspondant à votre système. Pour vérifier que Java est correctement installé, tapez la commande 'java' à l'invite de commande. Si vous voyez apparaître l'aide sur les paramètres de java, c'est que tout est installé correctement.

La BoB étant programmée en Java, il n'y a (a priori) aucune contrainte sur le système d'exploitation utilisé. Que vous soyez sous Windows, Linux, SunOS, vous ne devriez pas rencontrer le moindre problème à l'utilisation.

1.2 Installation

Pour installer la BoB, procéder comme suit :

1. Récupérer l'archive de la BoB, elle s'appelle bobUtil.tar.gz.
2. Créer un répertoire quelque part (à un endroit où vous êtes sûr d'avoir tous les droits d'accès) qui servira de répertoire de base au programme BoB. Par la suite, nous appelons ce répertoire "BOBPATH"
3. Décompresser l'archive bobUtil.tar.gz dans ce répertoire (par exemple, sous Linux ou Unix) :
   - `gunzip bobUtil.tar.gz`
   - `tar xvf bobUtil.tar`
4. Ensuite, il faut s'assurer que les variables d'environnement CLASSPATH et LD_LIBRARY_PATH sont bien initialisées correctement. Pour cela,

si vous utilisez le shell bash, éditez votre fichier .bashrc à la racine de votre compte comme suit :

```
BOBPATH = <le path du répertoire de la BoB>
export CLASSPATH = $CLASSPATH:$BOBPATH/build:$BOBPATH/lib/B.jar:$BOBPATH/lib/libreadline-java.jar
export LD_LIBRARY_PATH = $LD_LIBRARY_PATH:$BOBPATH/lib
```

Bien sûr, la variable BOBPATH doit contenir le chemin vers le répertoire de base choisi à l'étape 2. D'autre part, si vous n'utilisez pas bash, le fichier à modifier n'est pas le même. Pour le shell tcsh il faut modifier .tcshrc.

5. Relancer votre terminal pour que les variables d'environnement soient prises en compte.

1.3 Le contenu de la BoB

Les fichiers et répertoires chargés par l'installation sont :

- **bob.jar** l'archive des classes de la BoB
- **lib** un répertoire de classes auxiliaires
- **manuel.pdf** ce manuel
- **DocBoB** le répertoire de la documentation javadoc de la BoB.

Le chapitre suivant détaille les commandes de l'interface batch. Il permet de se rendre compte des diverses possibilités de la BoB.

Pour construire des applications qui utilisent la BoB, vous devez connaître les classes et les méthodes implémentées. La javadoc de la BoB est disponible dans le répertoire DocBoB généré par l'installation. Il suffit d'y accéder à l'aide d'un navigateur web.

Chapitre 2

Utilisation de l'interface

2.1 Généralités sur l'interface

L'interface de la BoB est un programme qui s'exécute en mode texte. Il ne présente pas d'interface graphique. Pour le lancer, on utilise la commande :

- java -jar bob.jar <paramètres>

Pour que cette commande marche, vous devez être dans le répertoire où bob.jar est installé. Si ce n'est pas le cas, vous pouvez lancer la BoB avec l'adresse absolue :

- java -jar $BOBPATH/bob.jar

ou (préférable) créer un alias de cette commande et l'utiliser.

Syntaxe d'exécution :

java -jar bob.jar <project-file> [<command-file>]

- <project-file> est le nom du fichier contenant les informations sur votre projet B.
- <command-file> est un fichier de commandes de l'interface BoB. Il permet d'entrer de manière automatique des commandes à envoyer à la BoB. Ce fichier se substitue à l'entrée au clavier et les commandes qu'il contient sont exécutées. Ce paramètre est facultatif, s'il n'y est pas, les commandes de la BoB pourront être normalement entrées au clavier. A la fin du fichier de commande, s'il n'y a pas la commande "quit" (cf. paragraphe 2.6.4), on passe en mode interactif d'entrée de commandes.

2.2 Généralités sur les projets BoB

Les informations de projet pour la BoB sont stockées dans des fichiers auxquels on donne l'extension ".prj" pour les reconnaître[1].

Ces fichiers contiennent tout simplement la liste des composants que la BoB doit charger en mémoire. Les projets peuvent être modifiés au moyen de l'interface (cf. paragraphe 2.3).

Au démarrage, si le fichier de projet spécifié en ligne de commande n'existe pas, le programme considère qu'il s'agit d'un nouveau projet et crée donc un projet vide. Les informations du projet sont automatiquement sauvegardées lorsqu'on sort du programme.

Exemple de fichier projet :

```
Systeme.mch
SystemeR1.ref
SystemeI.imp
Capteur.mch
```

Lorsqu'on est entré dans l'interface batch, l'invite de commande est le prompt "**BoB>**".

2.3 Les commandes de gestion de projet

2.3.1 La commande af (add file)

Syntaxe : **af <fichier>**

Cette commande permet d'ajouter un composant au projet.

Si le fichier mentionné en paramètre n'existe pas, un message d'erreur du type "File not found" est affiché, et aucune modification n'est apportée au projet.

S'il y a une erreur de syntaxe dans le fichier B passé en paramètre, un message d'erreur est aussi affiché et le fichier n'est bien sûr pas ajouté au projet.

Si le composant que vous ajoutez dépend d'autres composants, un message d'avertissement vous sera affiché vous indiquant quels sont les composants à charger pour que les dépendances soient satisfaites.

[1] ATTENTION : Ils ne sont en aucune façon compatibles avec les fichiers de l'atelier B.

2.3.2 La commande rc (remove component)

Syntaxe : **rc <nom de composant>**

Le paramètre ici est un nom de composant et non pas le nom du fichier qui contient le source B. Donc le nom ne contient pas d'extension.

Cette commande enlève le composant mentionné en paramètre du projet. Un message d'erreur est affiché si le nom du composant n'existe pas dans le projet. D'autre part, une demande de confirmation est affichée lorsque le composant que l'on veut supprimer est référencé dans d'autres composants du projet. Autrement dit : si le composant qu'on veut supprimer appartient à des dépendances d'autres composants, le programme vous demande confirmation avant de supprimer.

2.3.3 La commande scd (show component dependences)

Syntaxe : **scd [<nom de composant>]**

Cette commande permet d'afficher les dépendances éventuelles des composants. Si aucun nom de composant n'est spécifié en paramètre, le programme affiche les dépendances de tous les composants. Si un nom de composant est donné en paramètre, les dépendances sont affichées seulement pour le composant en question.

2.3.4 La commande scl (show component list)

Syntaxe : **scl**

Cette commande permet d'afficher la liste des composants B du projet chargés en mémoire. La liste donne pour chaque composant 2 informations :

- Le nom du fichier contenant le code B du composant
- L'entête du composant

2.4 Commandes d'information sur les composants

2.4.1 La commande vc (visualize component)

Syntaxe : **vc <nom du composant>**

Cette commande permet de visualiser le composant à partir de sa forme interne. Il est présenté dans un format de sortie, dit format "LSR". C'est un format de sortie avec indentation. Il n'est pas possible actuellement de paramétrer cette présentation.

2.4.2 La commande vv (visible variables)

Syntaxe : **vv <nom du composant>**

Cette commande affiche la liste des variables de l'état du composant passé en paramètre, y compris les variables provenant de machines incluses, étendues, vues ou importées.

2.4.3 Les commandes co et cox (callable operations)

Syntaxe : **co <nom de composant>**
cox <nom de composant>

Les commandes co et cox permettent d'afficher les opérations que l'on a le droit d'appeler dans le composant donné. La commande co, n'affiche que les entêtes tandis que la commande cox affiche l'entête et le corps des opérations.

2.4.4 Les commandes eo et eox (exported operations)

Syntaxe : **eo <nom de composant>**
eox <nom de composant>

Les commandes eo et eox permettent d'afficher la liste des opérations exportées (visibles) d'un composant. La commande eo affiche seulement l'entête des opérations, tandis que la commande eox affiche l'entête et le corps des opérations.

2.5 Commandes de calcul

Ces commandes construisent de nouveaux composants ou calculent des informations (prédicats, substitutions, etc.) à partir des composants donnés en paramètres.

2.5.1 La commande cc (copy component)

Syntaxe : **cc <nom de composant>$_1$ <nom de composant>$_2$**

Cette commande construit un composant et un fichier qui sont des copies de <nom de composant>$_1$ et de son fichier associé, et dont les noms sont remplacés de manière cohérente par <nom de composant>$_2$, avec l'extension appropriée pour le fichier.

2.5.2 La commande ic (include component)

Syntaxe : **ic <nom de composant>$_1$ [<nom de composant>$_2$]**

Cette commande permet de procéder à l'inclusion d'un composant : les composants référencés par le lien INCLUDES ou EXTENDS (directement ou par la transitivité du lien) à partir de <nom de composant>$_1$ sont "inclus" textuellement au sens de l'inclusion B, et les appels aux opérations des composants inclus sont remplacés par leur corps, avec substitution des paramètres.

Le résultat est un nouveau composant qui s'appelle "<nom de composant>$_2$" ou, par défaut, "<nom de composant>$_1$_inc". Le fichier source correspondant au résultat, qui est de même nature (machine ou raffinement) que le fichier initial, est généré dans le répertoire courant.

2.5.3 La commande op (operation predicates)

Syntaxe : **op <nom du composant> [<operation>]**

Pour chaque opération du composant, s'il n'y a pas de deuxième paramètre, ou pour l'opération passée en paramètre, cette commande affiche le prédicat "avant-après" (prd), la terminaison (prédicat trm) et la faisabilité (prédicat fis).

Le format de sortie est le suivant :

- La substitution
- PRD : <prédicat avant après>
- TRM : <terminaison>
- FIS : <faisabilité>

2.5.4 La commande po (proof obligation)

Syntaxe : **po <nom du composant>$_1$ [<nom de composant>$_2$]**

Cette commande génère un composant contenant toutes les obligations de preuves des opérations du composant <nom du composant>$_1$. Ces obli-

gations de preuve, qui sont des prédicats à prouver sous les hypothèses des déclarations et de l'invariant, sont mises dans la clause ASSERTIONS.

Le résultat est un nouveau composant qui s'appelle "<nom de composant>$_2$" ou par défaut "<nom de composant>$_1$_po". Le fichier source correspondant au résultat est généré dans le répertoire courant.

ATTENTION : cette méthode est expérimentale et ne traite pour l'instant que des machines.

2.5.5 La commande ps (primitive substitution)

Syntaxe : **ps <nom du composant> [<operation>]**

Pour chaque opération du composant, s'il n'y a pas de deuxième paramètre, ou pour l'opération passée en paramètre, cette commande affiche la conversion en substitution primitive de l'opération. Les substitutions primitives sont les substitutions mathématiques du B-Book. Les appels d'opération sont remplacés par leur définition et ensuite convertis.

2.5.6 La commande ss (simplify substitution)

Syntaxe : **ss <nom du composant> [<operation>]**

Pour chaque opération du composant, s'il n'y a pas de deuxième paramètre, ou pour l'opération passée en paramètre, cette commande affiche la substitution du corps, avec remplacement des appels d'opérations, en tentant de simplifier la substitution résultat.

NOTE : Notez que cette méthode est expérimentale dans la BoB et qu'elle ne prend en charge que très peu de simplifications. Sa principale tâche est de faire remonter les préconditions le plus à l'extérieur de la substitution.

2.5.7 La commande wp (weakest precondition)

Syntaxe : **wp <nom du composant> [<operation>]**

Cette commande affiche la plus faible précondition pour chaque opération. Le format de sortie est le suivant :

- la substitution
- son prédicat de plus faible précondition par rapport à l'invariant du composant.

2.6 Commandes diverses

2.6.1 La commande ab (Atelier B compatibility)

Syntaxe : **ab**

Cette commande permet de choisir un mode d'affichage normal ou un mode compatible avec l'entrée de l'analyseur de l'atelier B. Cette commande est une bascule : l'appeler 2 fois revient à ne rien faire. Lorsqu'on l'appelle, elle retourne l'indication du mode :

Atelier B display-style is off
Atelier B display-style is on

Cette commande résoud deux problèmes particuliers :

- Pour les variables renommées dans les composants : l'Atelier B n'accepte pas de déclarer une variable renommée de la forme "aa.bb" (qui est pourtant admis par la syntaxe du manuel de référence de l'Atelier B ! !). Le mode de compatibilité avec l'atelier B permet de ne pas afficher le '.' du renommage (voir les variables affichées par le prouveur interactif).
- Pour les prédicats constants. L'Atelier B n'accepte pas que l'on écrire "btrue" ou "bfalse" dans un texte source. Dans le mode de compatibilité avec l'Atelier B, ces prédicats seront écrits : "$0 = 0$" et "$0 = 1$".

Souvent, pour un simple affichage, on préférera utiliser le format normal (non compatible avec l'Atelier B) pour des raisons de lisibilité, en particulier pour la commande "op".

2.6.2 La commande ts (terminal set)

Syntaxe : **ts [<terminal> | -]**

Cette commande permet de rediriger la sortie des résultats de la BoB dans un fichier. Après cette commande, la sortie de toutes les autres commandes sera redirigée dans le fichier spécifié par le paramètre "<terminal>".

Pour restituer un affichage sur la sortie standard, il suffit d'exécuter la commande : "ts -"

2.6.3 La commande help

Syntaxe : **help [<nom de commande>]**

Sans paramètre, cette commande affiche la liste des commandes disponibles avec une mini description et la syntaxe à utiliser. Avec paramètre, elle affiche une description détaillée de la commande.

2.6.4 La commande quit

Syntaxe : **quit**

Cette commande fait sortir de l'interpréteur de commandes de l'interface BoB, sans demander de confirmation. Cependant, les informations du projet en cours sont sauvegardées dans le fichier du projet.