

UV LO22 - P13

TP Méthode B

1 Introduction à l'Atelier B

L'Atelier B est un ensemble d'outils logiciels permettant le développement d'applications suivant la méthode B.

L'Atelier B assiste le concepteur dans la formalisation de son application :

- En exécutant automatiquement sur les spécifications et leurs raffinements, un certain nombre d'actions dictées par la méthode,
- En proposant des services annexes à la méthode, mais néanmoins indispensables à des développements industriels, comme la gestion, l'analyse et la documentation d'un projet.

1.1 Objets manipulés par l'Atelier B

Nous introduisons dans ce paragraphe les principaux objets manipulés par les fonctions de l'Atelier B.

- Composant : Fichier contenant un source écrit en langage B. Il constitue la base d'un développement suivant la méthode B. Un composant est un terme générique et représente :
 - soit une spécification B (machine abstraite),
 - soit un raffinement de cette spécification,
 - soit son implémentation (dernier niveau de raffinement).

La saisie des composants est réalisée grâce à l'éditeur de texte de la station de développement.

- Projet : Ensemble de fichiers (composants, fichiers annexes sources C, C++, makefiles) utilisés ou produits durant le développement d'une application suivant la méthode B, complété des informations nécessaires à la gestion de ces fichiers sous l'Atelier (voir BDP).

- Utilisateur : Il est nécessaire de définir une liste d'utilisateurs autorisés à accéder et modifier le projet. Notez que tous les utilisateurs ont les mêmes pouvoirs, et qu'il est nécessaire qu'au moins un utilisateur soit renseigné.
- Base de Données Projet (BDP) : Toutes les informations internes nécessaires à la bonne exécution des outils de l'Atelier B sont stockés dans un répertoire nommé BDP. Ce répertoire contient aussi des fichiers produits par les outils de documentation de l'Atelier B.

1.2 Modes d'utilisation de l'Atelier B

Nous appelons Outils B les outils liés à l'application de la méthode B, ainsi qu'à l'analyse, la mise au point et la documentation de logiciels écrits en B. L'environnement B propose deux modes d'utilisation des outils B :

- un mode interactif, utilisant une interface graphique à base de fenêtres et de boutons de commande. Nous appellerons ce mode interface utilisateur graphique,
- un mode programmé, reposant sur un langage appelé langage de commande. Nous appellerons cette interface interface utilisateur mode commande.

1.3 Fonctions fournies par l'Atelier B

En résumé, afin de démarrer un développement avec l'Atelier B, vous devez:

1. Créer un projet,
2. Ouvrir ce projet,
3. Ajouter des composants à ce projet.

Une fois ces étapes accomplies vous pouvez commencer à appliquer la méthode B sur vos composants; il est possible de :

1. Effectuer l'analyse syntaxique et le contrôle de types,
2. Générer les obligations de preuve,
3. Démontrer automatiquement une partie de ces obligations de preuve,
4. Afficher les obligations de preuve,
5. Utiliser le prouveur interactif pour démontrer les obligations de preuve restantes.

Après avoir créé les implémentations de votre projet, vous serez en mesure de :

1. Contrôler le langage des implémentations,
2. Traduire le projet en langage cible en utilisant un traducteur.

Lors de ces phases du développement, utilisez les fonctions d'analyse de l'Atelier B, pour :

1. Afficher l'état d'avancement du projet ou d'un composant,
2. Afficher des graphes de dépendances entre les composants,

Vous pouvez également utiliser des fonctions de documentation pour produire automatiquement des documentations au format des traitements de texte LATEX, PDF, ou RTF.

Lorsque vos projets atteignent une taille importante, vous pouvez :

1. Les archiver pour en faire des sauvegardes,
2. découper vos gros projets en plusieurs petits projets en utilisant la notion de bibliothèques.

1.4 Création d'un nouveau projet

1. Commencez par créer un repertoire "Dir" avec trois sous repertoires "bdp", "spec" et "trad". Lancez l'atelier B et créez un projet (Bouton Attach). Nommez ce projet TP1 et indiquez respectivement Dir/bdp et Dir/trad comme repertoires de Base de Données et de Traduction.

A quoi serviront les répertoires "bdp", "spec" et "trad" ?

2. Ajouter les deux fichiers HelloWorldMain.mch et HelloWorldMain.imp à votre projet :

Fichier HelloWorldMain.mch :

```
MACHINE
  HelloWorldMain
OPERATIONS
  afficher ==
  BEGIN
    skip
  END;
END
```

Fichier HelloWorldMain_Imp.imp :

```
IMPLEMENTATION
  HelloWorldMain_Imp
REFINES ::
  HelloWorldMain
IMPORTS BASIC_IO
OPERATION
  afficher==
  BEGIN
    STRING_WRITE("Hello world\n")
  END;
END
```

3. Que est la différence entre un fichier .mch et un fichier .imp ?

1.5 Rappel

Machine abstraite

Une machine abstraite est un composant qui définit à l'aide de différentes clauses, des données et leurs propriétés ainsi que des opérations. Une machine abstraite constitue la spécification d'un module B. Elle se compose d'un en-tête et d'un certain nombre de clauses. L'ordre des clauses dans un composant n'est pas imposé.

```
MACHINE
  Exemple1

VARIABLES
  a,b,c

INVARIANT
  a : NATURAL, b : NATURAL, c : NATURAL

INITIALISATION
  a,b,c :=0,0,0

OPERATIONS
  ....

END
```

Description des principales clauses

SETS : ensembles de base manipulables par la machine. Ces ensembles peuvent

être abstraits (aucune indication sur le type des éléments) ou énumérés (définis par la liste de leurs éléments).

DEFINITIONS : abréviations utilisables dans le texte des autres classes.

VARIABLES : variables de la machine, représentant l'état de la machine. Les variables sont les seules entités qui peuvent être modifiées dans l'étape d'initialisation et dans les opérations de la machine. Toute clause VARIABLES doit être accompagnée des clauses INVARIANT (typage des variables) et INITIALISATION.

INVARIANT : ensemble de prédicats (formules logiques) permettant de typer les variables de la machine, de définir des propriétés mathématiques sur les variables, mais aussi de définir certaines contraintes que doit vérifier l'état de la machine à tout moment.

INITIALISATION : valeurs initiales des variables de la machine. Toute variable doit être initialisée en satisfaisant l'invariant.

OPERATIONS : déclaration et définition des opérations. Les opérations peuvent être paramétrées en entrée et en sortie. Le corps d'une opération définit un comportement, et peut être accompagné d'une pré-condition (PRE) qui fixe les conditions sous lesquelles un appel de l'opération garantit ce comportement.

Implementation d'une machine

Une fois la machine abstraite écrite, il est possible de la raffiner. Le raffinement d'un composant est l'étape qui permet de le rendre moins abstrait, en remplaçant ce qui était un ensemble par un tableau, par exemple. Il est alors possible de raffiner à nouveau ce raffinement, jusqu'à l'implémentation, composant qui doit répondre à des contraintes précises (pas d'opérations non-déterministes, ...)

4. Tester (et corriger si nécessaire) le typage et la syntaxe de vos deux composants.
5. Quels sont les rôles de la machine importé BASIC IO et de l'instruction SKIP?
6. Générer les obligations de preuves (PO) de chaque composant.

Rappel :

À chaque étape du développement, des obligations de preuve sont imposées par la méthode B. Une obligation de preuve est une propriété qui doit être satisfaite pour que le système construit soit correct.

Au niveau des machines abstraites, la preuve consiste à vérifier l'établissement et la conservation de l'invariant :

1. l'initialisation U doit établir l'invariant I : $[U]I$.
2. si l'invariant I est établi, alors toute opération de la forme PRE P THEN G END (substitution P|G) doit le garder établi, en vérifiant pour cela le prédicat:

$$I \wedge P \Rightarrow [G]I$$

Au niveau d'un raffinement, les obligations de preuve consistent à montrer que le modèle raffiné est cohérent avec le modèle abstrait :

1. il s'agit d'abord de montrer que l'initialisation Ur du raffinement ne fait rien qui soit contraire à l'initialisation U du composant abstrait :

$$[Ur] \neg [U] \neg Ir$$

2. il s'agit ensuite de montrer que l'exécution d'une opération du raffinement qui établit l'invariant correspond à une exécution de l'opération respective du niveau abstrait:

$$I \wedge Ir \wedge P \Rightarrow Pr \wedge [Gr] \neg [G] \neg Ir$$

7. Prouver les obligations de preuves générées.
8. Générer le code C correspondant à votre projet. Dans quelle répertoire est généré ce code ? Compiler ce code et vérifier que "Hello World" est affichée.
9. Générer le statut global de votre projet (pourcentage d'obligation de preuves prouvées, non prouvées...). Que signifie : TC, POG, Obv, nPO, nUv, Pr, BOC.
10. Que peut on conclure?

1.6 Introduction au prouveur interactif

1.7 Exemple1

On considère la machine Exemple1 suivante :

```
MACHINE
  Exemple1
VARIABLES
  few, many
INVARIANT
```

```

few <: NATURAL &
many <: NATURAL &
few <: many
INITIALISATION
few,many := {1,2,3},{2,3,4}
END

```

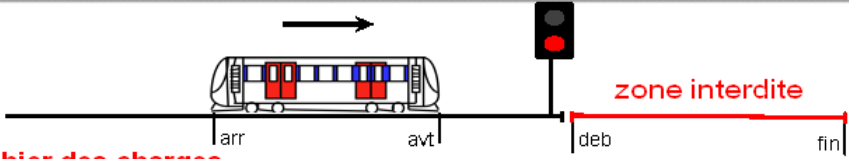
1. Tester le typage et la syntaxe de ce composant.
2. Générer les obligations des preuves.
3. A l'aide du prouveur automatique en force 0, prouvez ces obligations. Que remarquez vous?
4. Lancer le prouveur interactif et conclure.

2 Partie II : Application

2.1 Application (a)

>Cours//NouveauCoursFormel//0-IntroMethodesFormelles.pdf//SLIDE 15//Exemple simplifié
Raffinement et IMPLEMENTATION

5



Cahier des charges

« Si le train pénètre dans la zone interdite
alors un freinage d'urgence doit être déclenché »

Spécification formelle

```

MACHINE ZI
SETS POS
VARIABLES position_train, zone_interdite, freinage_urgence
INVARIANT position_train <: POS & zone_interdite <: POS & freinage_urgence
: BOOL &
(zone_interdite ∧ position_train /= {} => freinage_urgence = TRUE)
INITIALISATION position_train, zone_interdite, freinage_urgence
:(position_train <: POS & zone_interdite <: POS & freinage_urgence : BOOL &
(zone_interdite ∧ position_train /= {} => freinage_urgence=TRUE))
END

```

2.2 Application (b)

Un passage à niveau est un lieu de croisement entre un flux de circulations routières et un flux de circulation ferroviaire. La principale règle de sécurité, pour ce genre de situation, consiste à éviter l'entrée simultanée du trafic routier et du trafic ferroviaire dans la zone définie par le passage à niveau. Nous considérons en premier lieu une ligne à voie ferrée unique qui croise une route au même niveau. La zone d'intersection est appelée zone d'annonce. Les passages à niveau peuvent être gardés ou à signalisation automatique lumineuse et sonore (noté SAL). La sécurité des passages à niveau repose généralement sur la fermeture des barrières. Les feux routiers seront composés d'un seul feu bicolore (éteint, orange, rouge). Lorsque ce dernier est éteint, les utilisateurs de la route (voitures, piétons, ...) ont la possibilité de traverser le passage à niveau. Dans le cas contraire, le passage à niveau sera fermé et le trafic ferroviaire sera prioritaire. Comme leurs noms l'indiquent, les passages à niveaux gardés sont gérés par des gardes. Ces derniers doivent assurer la sécurité de ces passages soit en fermant les barrières dès l'approche d'un train au passage ou en demandant l'arrêt du ou des trains en cas de problème dans le passage en question. Ce type de passage à niveau a tendance à disparaître. Notre étude se focalisera sur le cas des passages à niveau français à signalisation automatique lumineuse et sonore pour une voie de circulation ferroviaire. Le fonctionnement en est le suivant. La détection de l'approche d'une circulation ferroviaire dans un passage à niveau à signalisation automatique lumineuse et sonore se fait par l'intermédiaire de détecteurs (capteurs) placés sur la voie. Cette approche est signalée, aux usagers de la route, simultanément par l'allumage des feux routiers, le tintement des sonneries et l'abaissement des barrières. Le signal sonore peut être atténué ou même supprimé en cas de gêne notable pour les riverains. Ces indications seront maintenues jusqu'au dégagement du passage à niveau. Après, elles disparaîtront progressivement, en commençant par relèvement des barrières, puis par l'extinction des feux et enfin, si nécessaire, l'arrêt du signal sonore. Ce système de contrôle sera activé dès le déclenchement des informations d'annonce. Ce déclenchement est provoqué par les circulations ferroviaires elles-mêmes. Le point à partir duquel l'annonce sera déclenchée s'appelle origine d'annonce. Un délai est nécessaire entre le déclenchement de l'annonce et l'arrivée du train le plus rapide au passage à niveau. Ce délai, appelé délai d'annonce, est fonction de la vitesse maximale des trains au passage à niveau.

On vous demande dans ce TP de donner le modèle B (la machine) obtenu correspondant au fonctionnement du passage à niveaux ainsi que de générer les obligations de preuves PO et le pourcentage s des PO prouvées. Voici un exemple d'une description possible de votre modèle B :

MACHINE

Passage_a_Niveaux

SETS

STATES = {Idle, SafeState, ActivateCrossingLevel, StandStill, AbnormalTimingSafeState, AbnormalCrossingLevel}

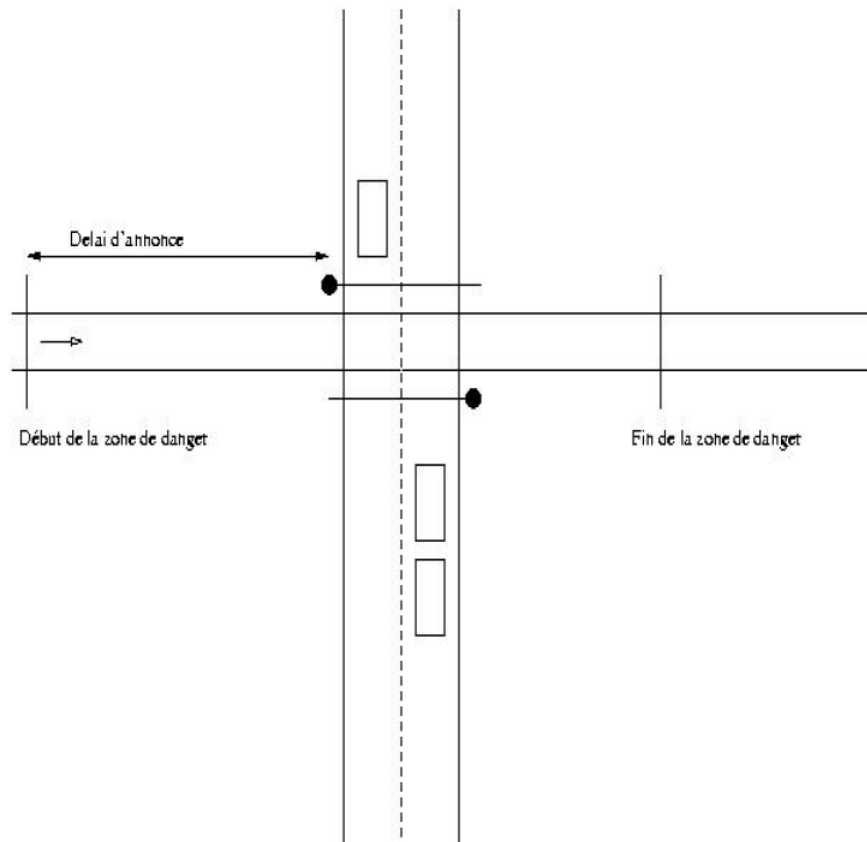


Figure 1: Passage à niveau pour une voie ferrée

```
CONCRETE_VARIABLES
current_state
INVARIANT
current_state : STATES
INITIALISATION
current_state := Idle
OPERATIONS
change_state =
BEGIN
...
END
END
```