

UV LO19 - TP 2 - Méthode B

Objectif : Spécifier un logiciel et le mettre en œuvre en utilisant l'atelier B.

Exercice 1 :

L'aiguillage est un appareil de voie servant à faire changer de voie un train. On considère dans cette étude un système de contrôle commande composé de 3 capteurs identiques redondants M1, M2 et M3 chargés de renseigner l'aiguilleur sur la direction effective la position de l'aiguillage. Les valeurs possibles de chaque capteur sont:

- Normale (voie B)
- Renversée (voie C)
- Nulle (position intermédiaire quand on change de voie)

On désire spécifier une fonction critique qui permet l'estimation de la position de l'aiguillage, en considérant que:

- Si au moins, une valeur normale et une valeur renversée sont mesurées, alors le résultat est nul.
- Si toutes les valeurs sont nulles, alors le résultat est nul.
- Dans tous les autres cas, elle doit retourner normale ou inversée.

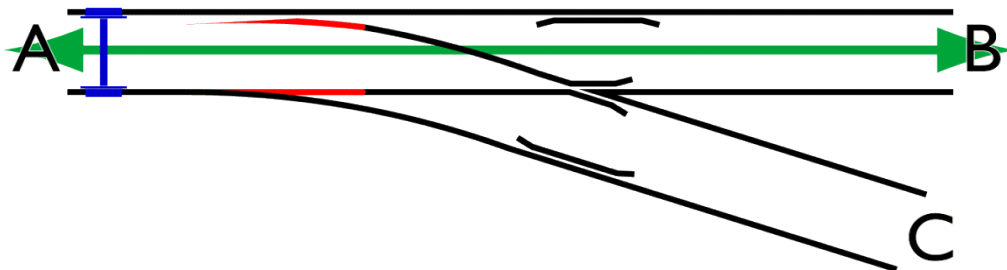


Figure 1

1. Création de la machine abstraite

Lancez l'atelier B et créez un projet Aiguillage en indiquant respectivement vos répertoires de Base de Données et de Traduction. Ajoutez le composant switch qui représente votre machine abstraite. On définira en particulier une opération « estimate » qui a comme paramètres m1, m2 et m3 qui représentent respectivement les valeurs des capteurs M1, M2 et M3. L'opération « estimate » retourne la position estimée « pos » à partir des trois valeurs des capteurs. Cette opération est non déterministe. Le non déterminisme est décrit par la construction SELECT ... THEN ... WHEN, qui permet de représenter un choix borné gardé.

2. Implémentation

Ajoutez le composant `switch_1` qui représente l'implémentation de votre machine abstraite. Les variables doivent correspondre à des structures de données du langage de programmation (entiers bornés, booléens, tableaux, etc.). Les substitutions généralisées autorisées doivent correspondre à des instructions classiques des langages de programmations. Les instructions B admises sont l'affectation, différentes formes de conditionnelle (CASE, IF, etc.), le séquençement, l'itération et la déclaration de variables locales. L'opération `estimate` devient une opération déterministe en utilisant par exemple l'instruction CASE au lieu de SELECT utilisée dans la machine abstraite.

3. Obligations de preuves

Générez les obligations de preuves (Components->Generate POs) pour les deux composants `switch` et `switch_1`. Prouvez-les.

4. Résultats des preuves

Est-ce que votre programme est prouvé à 100% ? Sinon l'améliorer pour prouver le maximum possible de Pos.

5. Conversion en langage C

Générez du code c natif à partir du composant `switch_1` (Component->Generate code). Compilez et exécutez ce code (répertoire `projet/langc`).

Exercice 2 :

Notre objectif est de décrire un service de réservation de places dans un cinéma. Les places sont numérotées de 1 à `nb_max`, cette dernière valeur étant un paramètre de la spécification. Les opérations offertes permettent de tester s'il reste des places (opération `place_libre`), de réserver une place (opération `réserver`) et de libérer une place déjà réservée (opération `libérer`).

1. Décrivez la machine abstraite du service de réservation.

Le paramètre `nb_max` est le nombre maximum de places. Ce paramètre vérifie certaines contraintes (clause `CONSTRAINTS`) : il est supérieur ou égale à 1 et inférieur ou égale à `MAXINT`. On définit aussi l'ensemble `SIEGES` comme l'intervalle `1..nb_max` (clause `DEFINITIONS`). L'état du système est modélisé par un ensemble, `occupes`, qui contient les places déjà allouées. On ajoute au système une variable, `nb_libre`, qui permet d'accéder directement au nombre de places libres. Initialement, l'ensemble des sièges occupés est vide et le nombre de sièges libres est `nb_max`. Les opérations offertes sont :

- Opération `place_libre` renvoie la valeur de la variable `nb_libre`.
- Opération `réserver` a une précondition : elle ne peut pas être appelée que s'il reste des sièges libres. Dans ce cas elle attribue une place parmi celle disponibles. Cette opération est non déterministe. Le non déterminisme est décrit par la construction

ANY z WHERE P THEN S END, qui permet de paramétrer la substitution S pour n'importe quelle valeur z vérifiant le prédicat P.

- Opération libérer a une précondition qui vérifie que la place à libérer est effectivement occupée.
2. Tester votre machine abstraite et générer les POs. Prouvez votre machine abstraite.
 3. Donnez un raffinement du service de réservation. Dans ce raffinement l'ensemble abstrait occupés est représenté de manière concrète par une fonction totale qui à chaque siège associe le booléen TRUE si le siège est occupé et FALSE s'il est libre. L'opération réserver est rendue déterministe : on choisit systématiquement la place libre ayant le plus petit numéro. Le raffinement hérite de certaines informations de la machine réservation, en particulier des contraintes sur les paramètres, de l'invariant sur les variables abstraites et des préconditions abstraites.
 4. Tester votre raffinement et générer les POs. Prouvez votre raffinement.
 5. Proposez une implémentation à votre raffinement. Les variables doivent correspondre à des structures de données du langage de programmation (entiers bornés, booléens, tableaux, etc.). les variables admises en B sont qualifiées de concrètes (clause CONCRETE_VARIABLES). Les substitutions généralisées autorisées doivent correspondre à des instructions classiques des langages de programmations. Les instructions B admises sont l'affectation, différentes formes de conditionnelle, le séquençement, l'itération et la déclaration de variables locales. Dans l'opération libérer l'affectation simultanée est remplacée par un séquençement
 6. Testez votre implémentation et générer les Pos. Prouvez votre implémentation.