UNIVERSITÉ DE NAMUR

Projet individuel

Modélisation en Event-B d'une pile atomique

Martin DANHIER
Bloc 2 - Informatique
INFOB236
8 mars 2020

1 Vue générale

Pour faciliter la représentation d'une pile atomique en Event-B, le raisonnement a été divisé en trois étapes. Chaque étape est un raffinement de la précédente et apporte de la complexité au modèle. Ces étapes sont les suivantes :

1. **Version basique de la pile** : le système est représenté sous sa forme la plus basique. On ne distingue pas encore les éléments, et il n'y a encore ni la notion de monte-charge ni celle de tapis roulant.
2. **Ajout de la notion d'éléments** : les éléments sont désormais différenciés dans le système, ainsi que leur position dans chaque étage.
3. **Ajout de la notion de monte-charge et de tapis roulant** : pour faciliter l'animation, le monte-charge doit désormais être déplacé, et les éléments sont apportés et évacués avec deux tapis roulants.

2 Étape 1 : version basique de la pile

Version basique du système. Le contexte est défini dans le fichier `atomicStackAbstractContext` et la machine est définie dans le fichier `atomicStackAbstractMachine` .

2.1 Contexte

La première version du contexte ne comporte que deux constantes :

— **nbFloors** : entier représentant le nombre d'étages de la pile. (ici, 6)
— **nbElementsPerFloor** : entier représentant le nombre d'éléments pouvant être placés sur un même étage. (ici, 2)

2.2 Variables

Il y a quatre variables :

— **simpleStack** : version simplifiée de la pile. Il s'agit d'une fonction totale associant, pour chaque étage, le nombre d'éléments s'y trouvant. La valeur est donc entre 0 et `nbElementsPerFloor` , et le domaine entre 1 et `nbFloors` .
— **handlingElement** : booléen indiquant si un élément est en train d'être manipulé ou non.

De plus, les deux variables suivantes sont des entiers se trouvant entre 0 et `nbFloors` :

— **barTo** : entier représentant la destination de l'élément manipulé. S'il vaut 0, aucun étage n'est sélectionné.
— **barFrom** : entier représentant la provenance de l'élément manipulé. Il faut 0 si l'élément vient d'être apporté par le tapis roulant d'entrée, sinon il vaut le numéro de l'étage d'où il vient d'être extrait.

2.3 Invariants

Les 4 premiers invariants définissent les types des variables tels que définis à la section précédente. Ensuite, il y a deux invariants assurant la cohérence du système :

— **last_floor_reserved** : cet invariant assure qu'un objet placé au dernier étage provient d'un autre étage de la pile. Cela permet d'éviter de placer des nouveaux éléments sur le dernier étage, celui-ci étant conçu comme un étage temporaire pour pouvoir accéder aux éléments au fond d'un étage.

— **place_elements_in_empty_spots** : vérifie que `barTo` n'est jamais assigné à un étage complet.

2.4 Événements

La première version comporte 6 événements.

— **INITIALISATION** : Initialise les variables. La `simpleStack` est initialisée à une pile vide, les variables numériques sont initialisées à 0 et la variable `handlingElement` est initialisée à `FALSE` . Au début de l'exécution, le système représente donc une pile vide, sans aucun élément en cours de manipulation.

— **Bar_new** : Permet d'importer un nouvel élément. À ce stade, il n'y a pas encore de tapis roulant ni de monte-charge pour le déplacer, et son type n'est pas encore géré.
  — Les trois premières gardes s'assurent qu'aucun élément n'est déjà en train d'être manipulé.
  — La garde **last_floor_must_be_empty** empêche l'importation d'éléments si le dernier étage n'est pas vide. Sans cette garde, on pourrait stocker de manière permanente des éléments au dernier étage.

— **Bar_in** : Permet de définir une destination pour l'élément manipulé. La variable `barTo` sera modifiée en la valeur du paramètre **destinationFloor**, un entier entre 1 et `nbFloors` .
Les gardes s'assurent qu'un élément est en cours de manipulation, que l'étage de destination existe et qu'il y reste de la place. Elles garantissent également qu'un élément ne peut être placé au dernier étage que s'il provient déjà de la pile, et enfin que la destination de l'élément n'est pas déjà définie.

— **Bar_insert** : Insère la barre en cours de manipulation dans l'étage de destination tel qu'indiqué dans la variable `barTo` . Comme la pile est encore basique, la valeur associée à l'étage de destination est simplement incrémentée de 1. Les variables relatives à l'élément en cours de manipulation sont réinitialisées pour permettre la manipulation d'autres éléments.
Les gardes s'assurent qu'il y a un élément en cours de manipulation à insérer, et qu'un étage de destination a été défini.

— **Bar_withdraw** : Extrait un élément de l'étage indiqué par le paramètre **barFrom** (entre 1 et `nbFloors`). Dans la pile basique, la valeur est simplement décrémentée de 1.
Les gardes s'assurent qu'il n'y a pas d'élément manipulé, et que l'étage indiqué n'est pas vide.

— **Bar_out** : Exporte la barre en cours de manipulation en dehors du système. Les variables concernant la manipulation d'élément sont simplement réinitialisées.
Les gardes s'assurent qu'un élément est manipulé, et qu'il n'a pas déjà une destination.

3 Étape 2 : premier raffinement

Ce raffinement ajoute la notion de type d'élément.

Le contexte raffiné est défini dans le fichier `atomicStackContext` et la machine raffinée est définie dans le fichier `atomicStackMachine` .

3.1 Contexte

En plus des constantes de la première version, le raffinement apporte un ensemble **ELEMENTS** constitué de constantes représentant les types d'éléments : `uranium` et `cadmium` . Enfin, la constante `null` indique l'absence d'élément.

3.2 Variables

— Les variables `barTo` et `barFrom` sont conservées et gardent le même comportement.
— La variable booléenne `handlingElement` est remplacée par la variable `workingElement` . Cette nouvelle variable est de type `ELEMENTS` , elle ajoute donc une précision sur le type de l'élément en cours de manipulation. La constante `null` est utilisée s'il n'y a aucun élément en cours de manipulation.
— La version primitive de la pile, `simpleStack` , est remplacée par une version plus avancée : `stack` . La nouvelle pile est une fonction qui associe, pour chaque étage, une autre fonction associant une position sur l'étage (entre 1 et `nbElementsPerFloor`) à une barre de type `ELEMENTS` .

3.3 Invariants

En plus des invariants de types, il y a deux *gluing invariants* reliant les nouvelles variables à celles qui disparaissent. Enfin, un nouvel invariant, **elements_always_on_the_right**, vérifie que les barres sont toujours situées le plus à droite possible (c'est-à-dire avec le plus grand indice possible) sur leur étage.

3.4 Événements

Les événements restent les mêmes et conservent leur comportement, en s'adaptant aux nouvelles variables. Voici les nouveautés notables :

— **Bar_new** : au lieu de changer `handlingElement` , l'événement permet maintenant de changer `workingElement` à la valeur d'un paramètre, **element**, de type `ELEMENTS` (sauf la valeur `null`). Les gardes sont légèrement modifiées pour supporter les nouvelles variables, mais gardent la même signification.

— **Bar_insert** : au lieu d'incrémenter le nombre de barres dans l'étage indiqué par `barTo` , la valeur du `workingElement` est placée dans l'emplacement situé le plus à droite possible de l'étage.

— **Bar_withdraw** : au lieu de décrémenter le nombre de barres dans l'étage indiqué, la première valeur non `null` rencontrée dans l'étage en partant de la gauche est placée dans `workingElement` , et l'emplacement d'où il provenait est modifié à `null` .

4 Étape 3 : deuxième raffinement

Pour faciliter l'animation, ce raffinement ajoute la notion de monte-charge et de tapis roulants. Le contexte utilisé est le même, mais la machine est définie dans le fichier `atomicStackMachine2` . Ce raffinement n'est pas indispensable pour le système, il est principalement conçu afin de pouvoir améliorer l'animation B-Motion.

4.1 Variables

Toutes les variables précédemment définies sont conservées telles quelles. Ce raffinement ajoute les variables suivantes :

— **elementLift** : Le monte-charge. Sa valeur, un entier entre 0 et `nbFloors` , représente l'étage auquel il se trouve.
— **beltIn** : Le tapis roulant permettant d'importer des barres dans le système. Il s'agit d'une fonction associant deux positions (gauche et droite, représentées par les chiffres 1 et 2) à une barre de type `ELEMENTS` . Ces positions pourront ensuite être représentées sur l'animation.

— **beltOut** : Le tapis roulant permettant d'exporter des barres hors du système. Le principe est le même que pour `beltIn` .
— **elementOnLift** : L'élément, de type `ELEMENTS` , situé sur le monte-charge.

4.2 Invariants

Il n'y a comme nouveaux invariants que ceux indiquant les types des nouvelles variables.

4.3 Événements

Pour faciliter l'animation, il y a 5 nouveaux événements :

— **Lift_up** : fait monter le monte-charge d'un étage. Requiert que le monte-charge ne soit pas déjà tout en haut. L'événement se bloque si `barTo` est supérieur à 0, c'est-à-dire entre le choix d'une destination et l'insertion, ce qui empêche le monte-charge de bouger.
— **Lift_down** : même principe que `Lift_up` , mais pour faire descendre le monte-charge. Il doit donc ne pas être déjà au rez-de-chaussée.
— **BeltIn_roll** : fait rouler le tapis roulant d'entrée. Requiert qu'un élément se trouve dessus. Si l'élément se trouve en deuxième position du tapis roulant, il faut également que le monte-charge se trouve au rez-de-chaussée.
— **Send_out** : déplace l'élément du monte-charge vers le tapis roulant de sortie. Il faut donc que le monte-charge se trouve au rez-de-chaussée, et qu'il y ait un élément dessus.
— **BeltOut_roll** : fait rouler le tapis roulant de sortie. Il faut qu'un élément se trouve en première position de `beltOut` .

Par ailleurs, certains événements sont étendus pour supporter les nouvelles variables :

— **Bar_new** : Le nouvel élément est placé en première position du tapis roulant d'entrée.
— **Bar_in** : Il y a deux nouvelles gardes, assurant d'une part qu'un élément se trouve sur le monte-charge, et limitant d'autre part le paramètre `destinationFloor` à l'étage où se trouve le monte-charge.
— **Bar_insert** : Il faut que le monte-charge se trouve au même étage que `barTo` . Après l'insertion, la valeur de `elementOnLift` est également réinitialisée.
— **Bar_withdraw** : La valeur du paramètre `fromFloor` est maintenant limitée à l'étage où se trouve le monte-charge. `elementOnLift` est également modifié en le type d'élément de la barre extraite.
— **Bar_out** : Il faut désormais que la barre se trouve en deuxième position du tapis roulant de sortie avant de pouvoir déclencher l'événement.