

## TP3 – Des Cartons de Preuves

En Event-B (comme dans toutes les approches formelles par ailleurs), un **modèle n'est rien sans preuve**.

Le modèle que l'on a créé décrit en effet un automate logistique comme prévu, mais tant que l'on n'aura pas réalisé de preuves dessus, on ne pourra rien en tirer d'intéressant.

Fort heureusement, Rodin génère automatiquement les obligations de preuve à décharger sur le modèle pour garantir sa cohérence. La plupart du temps, les obligations de preuve concernent les invariants; autrement dit, il s'agira de prouver que les événements ne violent pas les invariants donnés dans la machine.

### 1 Preuves de Modèle




Dans l'explorateur de projets, dépliez l'élément correspondant à la première machine (**Automate\_1**). Vous devez alors apercevoir un sous-item appelé *Proof Obligations*; dépliez cet élément pour obtenir la liste des obligations de preuve générées par Rodin.



Notez que les obligations de preuves les plus simples (typiquement concernant l'initialisation, somme toute assez simple) ont déjà été déchargées automatiquement : elles sont en vert et ont un petit "A" sur leur icône. A contrario, les obligations de preuve en orange n'ont pas encore été déchargées, et il va donc nous incomber de le faire.



Commencez par basculer en perspective "Proving" (Window > Perspective > Open Perspective > Other..., puis sélectionner "Proving"). Double-cliquez ensuite sur une preuve qu'il reste à faire, par exemple l'obligation de preuve **Prendre/INV/inv3** (une preuve que l'INVARIANT **inv3** n'est pas violé par l'événement **Prendre**). Vous devriez tomber sur une vue similaire à celle donnée dans la figure 1.


La fenêtre est alors coupée en trois grandes zones :

- L'arbre de preuve pour l'obligation de preuve en cours (1), qui retrace toutes les étapes prises dans la preuve
- L'explorateur de projet (2) avec notamment la liste des obligations de preuve
- Le prouveur lui même (3), avec en haut la liste des hypothèses, au milieu le but (ce qui doit être prouvé) et en bas l'assistant

Pour réaliser une preuve, on peut s'y prendre de deux manières : les preuves ou buts très simples peuvent être complétées ou au moins partiellement avancées avec les prouveurs automatiques tels que PP, p0, p1, ml ou encore l'icône  (qui applique automatiquement un certain nombre de tactiques) et  (pour "nettoyer" un peu les hypothèses et la conclusion), ou même  (qui appelle des solveurs SMT). Les autres preuves quant à elles doivent être réalisées "interactivement".

Une preuve interactive consiste principalement à cliquer sur les opérateurs en rouge (ce qui indique qu'il est possible de transformer l'expression), ajouter des hypothèses en l'écrivant dans la zone de texte en bas et en appuyant sur  (pour *add hypothesis*), ou encore en utilisant des règles d'inférence comme "ct" (pour *contradict*). Si des hypothèses semblent manquantes, il est par ailleurs possible d'aller les chercher automatiquement avec le "lasso" .

Enfin, il est possible de remonter dans un arbre de preuve avec , ou même revenir à un noeud précis et détruire tout ce qui se trouve en-dessous avec .

Pour cette preuve très simple, placez vous à la racine et utilisez  (si la preuve était déjà faite). Cliquez sur le signe égal dans le but.

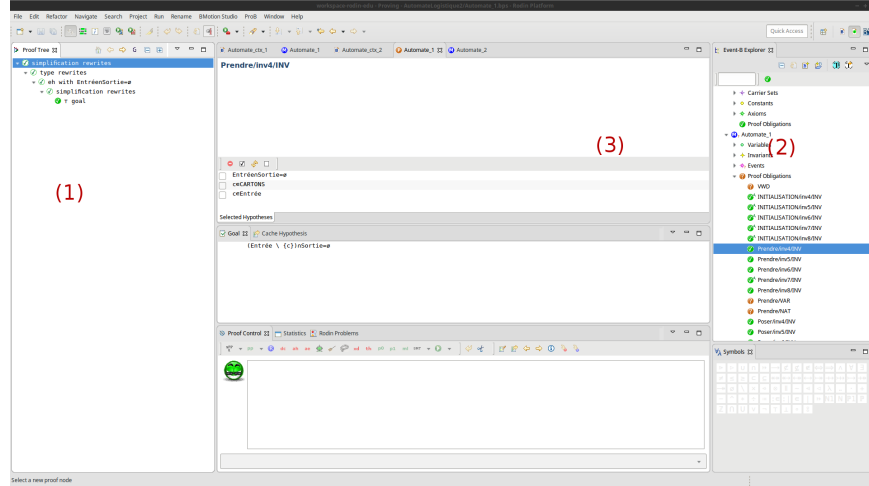


FIGURE 1 – Vue "Proving" de Rodin

Le prouveur transforme le but en deux nouveaux buts, l'un avec à la place le signe  $\subseteq$  et l'autre avec le signe  $\supseteq$  (notez que ce second but a été prouvé automatiquement car il est très simple).

Cliquez alors sur le symbole  $\subseteq$  et choisissez "Remove inclusion" pour réécrire le but sans le symbole d'inclusion.

Le prouveur va modifier encore le but (avec quelques étapes intermédiaires) et vous propose maintenant de prouver que  $x = c$  sachant que  $x \in \text{Entrée} \cap \text{Sortie}$  et  $\text{Entrée} \cap \text{Sortie} = \emptyset$ ... Surveillez le signe égal dans les hypothèses et choisissez "Apply equality from left to right".

Cela a pour effet de remplacer la première occurrence du membre de gauche de l'égalité ( $\text{Entrée} \cap \text{Sortie}$ ) avec son membre de droite ( $\emptyset$ ). Autrement dit, le prouveur va transformer  $x \in \text{Entrée} \cap \text{Sortie}$  en  $x \in \emptyset$  et détecter tout seul qu'il y a une contradiction, ce qui fait la preuve !

À titre indicatif, l'arbre de preuve :

$$\begin{array}{c}
 \frac{\cdot}{E \cap S = \emptyset, c \in E, \perp \vdash x = c} \\
 \frac{E \cap S = \emptyset, c \in E, \perp \vdash x = c}{E \cap S = \emptyset, c \in E, x \in \emptyset \vdash x = c} \\
 \frac{E \cap S = \emptyset, c \in E, x \in \emptyset \vdash x = c}{E \cap S = \emptyset, c \in E, x \in E \cap S \vdash x = c} \\
 \frac{E \cap S = \emptyset, c \in E \vdash x \in E \cap S \Rightarrow x = c}{E \cap S = \emptyset, c \in E \vdash \forall x \cdot x \in E \cap S \Rightarrow x = c} \\
 \frac{E \cap S = \emptyset, c \in E \vdash \forall x \cdot x \in E \cap S \Rightarrow x \in \{c\}}{E \cap S = \emptyset, c \in E \vdash (E \setminus \{c\}) \cap S \subseteq \emptyset} \\
 \frac{\cdot}{E \cap S = \emptyset, c \in E \vdash \emptyset \subseteq (E \setminus \{c\}) \cap S} \\
 \frac{E \cap S = \emptyset, c \in E \vdash (E \setminus \{c\}) \cap S \subseteq \emptyset}{E \cap S = \emptyset, c \in E \vdash (E \setminus \{c\}) \cap S = \emptyset}
 \end{array}$$

FIGURE 2 – Arbre de Preuve pour Prendre/inv4/INV

**Question 1 :** Déchargez les autres obligations de preuve de la première machine du mieux que vous pouvez.

**Question 2 :** Déchargez les obligations de preuve de la deuxième machine.

**Note :** les obligations de preuve de la troisième machine sont un peu plus techniques... Ne planchez dessus que si vous avez le temps !

## 2 Autres Propriétés

(Dans cette section, on s'intéresse de nouveau à la première et à la deuxième machine.)

Tout au début du sujet, nous avons énoncé une propriété qui pourrait nous intéresser sur nos automates logistique, la propriété (2).

**Question 3 :** De quel genre de propriété s'agit-il ?

Contrairement à la propriété (1), il est difficile d'exprimer (2) sous forme d'un prédicat. For heureusement, nous disposons d'un outil parfaitement adapté à cela : le *variant*.

**Question 4 :** Qu'est-ce qu'un variant ? Dans quel cas très simple (que vous avez très certainement déjà rencontré) l'utilise-t-on ?

Pour rappel, la propriété disait : « l'automate finit par traiter tous les cartons ». Malheureusement, dans la première machine, il est impossible de qualifier tous les cartons (et notamment les cartons pris mais pas encore posés) ; nous allons donc d'abord nous intéresser à une propriété analogue mais plus simple : « l'automate finit par prendre tous les cartons de l'entrée ».

En supposant que tout carton pris finit par être déposé, on remarquera que ces deux propositions sont relativement équivalentes.

**Question 5 :** Parmi les actions des événements, qu'est-ce qui modélise en particulier le fait de prendre un carton de l'entrée ? Quel donnée dans le modèle peut alors servir de variant ? (*Rappel : un variant est une variable strictement décroissante.*)

Un seul événement fait décroître le variant : **Prendre**. Pour signifier à Rodin que c'est à cet événement que l'on s'intéresse, on va le marquer *convergent*. Pour cela, cliquez sur *ordinary* puis, dans le menu déroulant, sélectionnez *convergent*. Cela permet d'indiquer à Rodin que l'événement va s'effectuer un nombre fini de fois.

Une fois que c'est fait, ajoutez à la machine le fameux variant. Notez que de nouvelles obligations de preuve apparaissent : **FIN**, qui représente le fait que le variant ne décroît pas indéfiniment, et **Prendre/VAR**, qui vérifie que l'événement convergent fait bien décroître strictement le variant.

**Note :** avant de continuer, remarquez qu'un variant ne peut décroître finiment que si il est effectivement fini. Dans le cas présent, cela revient à dire que l'ensemble des cartons est fini. Afin de pouvoir réaliser la preuve **FIN**, nous allons donc ajouter l'axiome *finite(CARTONS)* dans le contexte de la machine.

**Question 6 :** Réalisez la preuve **FIN**. (*Pour prouver qu'un ensemble est fini, on peut écrire dans la zone de texte un sur-ensemble de cet ensemble dont on sait qu'il est fini puis cliquer sur l'opérateur finite.*)

**Note :** il est possible de prendre pour variant ou bien un ensemble (pour lequel la relation d'ordre strict est  $\subset$ ) ou alors le cardinal de cet ensemble (avec pour relation d'ordre  $<$ ). Notez que, si ces deux représentations sont équivalentes, les preuves pour la seconde sont un peu plus délicates car elles font appel à l'arithmétique.

### Pour aller plus loin

Le variant que nous avons écrit nous assure que l'automate peut finir par prendre tous les cartons ; mais cela ne suffit pas à dire qu'il finira par *traiter* tous les cartons. En effet, pour cela, il va falloir s'intéresser à la deuxième machine, et notamment à la variable **Transit**.

On aimerait écrire un variant pour **Entrée** et un autre pour **Transit** ; mais le problème est qu'une machine ne peut avoir qu'un seul variant. Il va donc falloir trouver un bon variant qui concerne à la fois ces deux variables.

**Question 7 :** Une approche naïve consiste à considérer l'expression **Entrée**  $\cup$  **Transit** ou  $\text{card}(\text{Entrée}) + \text{card}(\text{Transit})$ . Pourquoi ces variants ne conviennent-ils pas ?

**Question 8 :** Proposez un variant pour cette machine.

Avec le nouveau variant, on remarquera que **Prendre** et **Poser** sont convergents.

On remarquera par ailleurs que, parmi les nouvelles obligations de preuve, on a : **VWD** (pour *Variant Well Definedness*) qui vérifie si le variant est correctement défini, autrement dit qu'il utilise les opérateurs qu'il a le droit d'utiliser, ainsi que **Prendre/NAT** (pour *NATural*), qui vérifie que le variant reste bien toujours positif.

**Note :** les preuves pour ce variant sont techniques – en grande partie à cause de l'ergonomie du prouveur. Vous pouvez tenter de faire les parties faciles puis vous convaincre (ou prouver à la main) que les obligations de preuve sont bien correctes.

Remarquez que ce nouveau variant signifie que, éventuellement, l'entrée et l'automate seront vides. En conjonction avec l'invariant  $Entrée = \emptyset \wedge Transit = \emptyset \Rightarrow Sortie = CARTONS$  écrit dans un TP précédent, cela signifie par ailleurs que la sortie finira par contenir tous les cartons que l'automate devait traiter.

### 3 Réflexions et Conclusion

**Question 9 :** La compagnie *Ad Hoc Logistics* décide de réaliser un automate avec 1 ligne d'entrée et un nombre quelconque de lignes de sortie. Comment feriez-vous pour fournir un modèle de cette machine ?

**Question 10 :** La compagnie veut maintenant précisément un automate avec 1 ligne d'entrée et 5 lignes de sortie. Comment feriez-vous pour fournir un modèle de cette machine ?

**Question 11 :** Quel avantage voyez-vous à avoir réalisé des modèles abstraits d'automate ?