École nationale supérieure
d'électrotechnique, d'électronique,
d'informatique, d'hydraulique et des
télécommunications

Modélisation d'un Système Ferroviaire avec Event-B

Développement Formel de Systèmes

Année universitaire 2024-2025

Membres du groupe :

Ayoub DIOURI
Khadija AKKAR
Otman FARHAT
Safae BELAHRACH
Wissal ABOUMEJD

2 février 2025

Table des matières

1 Introduction

Les systèmes ferroviaires sont des environnements critiques pour la sécurité avec un potentiel élevé de dommages. Ils nécessitent une coordination précise pour éviter les collisions. C'est là où interviennent les méthodes formelles, qui sont « des techniques permettant de raisonner rigoureusement, à l'aide de logique mathématique, sur un programme informatique ou du matériel électronique numérique, afin de démontrer leur validité par rapport à une certaine spécification» [1].

Dans ce projet, nous allons modéliser un système de gestion ferroviaire à l'aide de Event-B qui est une méthode formelle qui permet de décrire les systèmes d'une manière précise. Nous allons commencer par une structure de base et nous ajouterons au fur et à mesure les détails.

1.1 Objectifs

Modéliser le réseau ferroviaire
Simuler des déplacements des trains
Prouver que le système respecte les exigences de sécurité
Commencer par un modèle simple puis le raffiner

Notre projet est organisé en diverses « couches logiques », chaque couche représentant un ensemble d'exigences propre à une partie du système seulement, ces couches doivent s'articuler pour former un système complet :

Niveau topologique et signalisation : Gestion des trains sur un réseau ferré fixe segmenté en sections (tronçons), signalisation pour éviter les incidents.

Couche matériel roulant : Illustration logique des trains (wagons, emplacements, taille).

Couche physique et dynamique : Modélisation des déplacements physiques des trains (vitesse, accélération, freinage).

Ainsi, nous avons employé *« Event B »* pour le développement formel de notre système ferroviaire, en utilisant l'approche des raffinements successifs.

2 Couche « Topologie et signalisation »

2.1 Exigences extraites

2.1.1 Hypothèses du système

HYP1 : Un arrêt est associé à un seul tronçon sortie.

$$\text{axm7, Ctx\_0} : \forall a \in arrets, \quad \text{card}(reseau[\{a\}]) = 1$$

HYP2 : Un arrêt est associé à un seul tronçon d'entrée.

$$\text{axm8, Ctx\_0} : \forall a \in arrets, \quad \text{card}(reseau^{-1}[\{a\}]) = 1$$

HYP3 : Chaque train doit avoir une position initiale située dans un arrêt .

$$\text{axm3, Ctx\_0} : p0 \in TRAINS \rightarrow arrets$$

HYP4 : Chaque train a une position initiale définie.

$$\text{axm5, Ctx\_0} : \text{card}(\text{dom}(p0)) = \text{card}(TRAINS)$$

2.1.2 Exigences fonctionnelles

REQ1 : Le réseau ferroviaire est défini comme une relation entre les tronçons.

$$\text{axm1, Ctx}_0 : \text{reseau} \in TRONCONS \leftrightarrow TRONCONS$$

REQ2 : Un train peut être localisé par l'arrêt qu'il occupe ou le tronçon.

$$\text{inv1, Mch\_1} : trainPosition \in TRAINS \rightarrow TRONCONS$$
$$\text{axm1, Ctx\_0} : arrets \subset TRONCONS$$

REQ3 : Une destination est un arrêt du réseau.

$$\text{grd2, AssignerDestination} : destination \in arrets$$

REQ4 : Un train doit avoir une destination attribuée.

$$\text{inv3, Mch\_0} : destination \in TRAINS \rightarrow arrets$$

REQ5 : Un train suit une route définie.

inv2, Mch_0 : $trainRoute \in TRAINS \rightarrow \mathcal{P}(TRONCONS)$

REQ6 : Une route attribuée à un train doit contenir au moins deux tronçons.

grd3, AssignerRoute :

$$card(route) > 1$$

REQ7 : Un train ne peut prendre qu'un tronçon adjacent pour avancer.

grd2, Mouvement :

$$troncon\_suivant \in \{trainPosition(train)\} \cup reseau[\{trainPosition(train)\}]$$

REQ8 : Un train ne peut avancer que s'il possède une route.

grd5, Mouvement :

$$train \in dom(trainRoute)$$

REQ9 : Une destination ne peut pas être la position actuelle du train.

grd4, AssignerDestination :

$$dest \neq trainPosition(train)$$

REQ10 : Lorsqu'un train atteint sa destination, il ne doit plus avoir de route active.

grd8, AssignerRoute :

$$destination(train) \in route \Rightarrow (reseau[\{destination(train)\}] \cap route = \emptyset)$$

REQ11 : Les routes des trains sont recalculées lorsqu'un blocage est détecté.

DebloquerTrain : $\exists train1, train2 \in TRAINS, \quad trainPosition(train1) \in reseau^{\sim}[\{trainPosition(train2)\}]$

REQ12 : Lorsqu'un train atteint un tronçon qui permet plusieurs choix de direction, ce tronçon est ajouté à 'routeParcourue'. Ensuite, lors de l'assignation d'une nouvelle route ('AssignerRoute'), seules les routes non encore parcourues sont sélectionnées lorsque plusieurs options sont disponibles.

grd7 : historique_Route, grd 9 : AssignerRoute

2.1.3 Exigences de sûreté

SAF1 : Il ne doit pas y avoir deux trains dans un même tronçon.

$$\text{inv4, Mch\_0 :}$$
$$\forall t \in TRONCONS, \quad \text{card}(\{tr \in TRAINS \mid trainPosition(tr) = t\}) \leq 1$$

SAF2 : Un train ne peut pas retourner sur le tronçons par lequel il est arrivé.

$$\text{grd5, AssignerRoute : } reseau^{\sim}[\{trainPosition(train)\}] \cap route = \emptyset$$

SAF3 : Un tronçon ne peut pas être relié à lui-même.

$$\text{axm6, Ctx\_0 : } \forall t \in TRONCONS, \quad \neg(t \mapsto t \in reseau)$$

SAF4 : Un arrêt ne peut pas être directement connecté à un autre arrêt.

$$\text{axm10, Ctx\_0 : } \forall a \in arrets, \quad (reseau[\{a\}] \cap arrets) = \emptyset$$

SAF5 : Empêche les blocages en identifiant les conflits entre trains.

$$\text{grd5, DebloquerTrain : } trainPosition(train1) \in reseau^{\sim}[\{trainPosition(train2)\}]$$

SAF6 : S'il y'a un bloquage, on assigne une nouvelle route au train qui bloque la circulation.

$$\text{grd12, DebloquerTrain :}$$
$$destination(train2) \in new\_route \Rightarrow (reseau[\{destination(train2)\}] \cap new\_route = \emptyset)$$

$$\text{act1, DebloquerTrain}$$

SAF7 : Une route ne peut pas être affectée deux fois au même train.

$$\text{inv5, Mch\_0 : } \forall tr \in TRAINS, \quad \forall r_1, r_2 \in trainRoute(tr), \quad r_1 \neq r_2$$

SAF8 : Une route doit mener à la destination du train.

$$\text{grd8, AssignerRoute, Mch\_0 :}$$
$$destination(train) \in route \Rightarrow (reseau[\{destination(train)\}] \cap route = \emptyset)$$

2.2 Modélisation en Event-B

On a débuté avec un modèle abstrait simple qui décrit les spécifications du système. Ce modèle se compose du contexte Ctx_0 et la machine Mch_0.

```
CONTEXT
    Ctx_0   ›
SETS
◦   TRONCONS     ›
◦   TRAINS   ›
CONSTANTS
◦   reseau   not symbolic ›
◦   p0   not symbolic ›
◦   arrets   not symbolic ›
◦   T1   not symbolic ›
◦   T2   not symbolic ›
◦   T3   not symbolic ›
◦   T4   not symbolic ›
◦   Tr1  not symbolic ›
◦   Tr2  not symbolic ›
AXIOMS
◦   axm1:   reseau ∈ TRONCONS ↔ TRONCONS not theorem ›
◦   axm12:  partition(TRAINS, {Tr1}, {Tr2}) not theorem ›
◦   axm13:  partition(TRONCONS, {T1}, {T2}, {T3}, {T4}) not theorem ›
◦   axm14:  partition(arrets,{T1}, {T2}) theorem ›
◦   axm2:   arrets ⊂ TRONCONS not theorem ›
◦   axm3:   p0 ∈ TRAINS ⟶ arrets not theorem ›
◦   axm4:   card(arrets) > 1 not theorem ›
◦   axm15:  p0 = {Tr1 ↦ T1, Tr2 ↦ T2} not theorem ›
◦   axm5:   card(dom(p0)) = card(TRAINS) not theorem ›
◦   axm6:   ∀ t· t ∈ TRONCONS ∧ ¬(t ↦ t ∈ reseau) not theorem ›
◦   axm7:   ∀ a· a ∈ arrets ⇒ card(reseau[{a}]) = 1 not theorem ›
◦   axm8:   ∀ a· a ∈ arrets ⇒ card(reseau~[{a}]) = 1 not theorem ›
◦   axm11:  card(TRONCONS) = 4 not theorem ›
◦   axm10:  ∀ a· a ∈ arrets ⇒ (reseau[{a}] ∩ arrets) = ∅ not theorem ›
◦   axm16:  ∀ t1, t2· ((t1 ∈ TRONCONS ∧ t2 ∈ TRONCONS ∧ t1↦t2 ∈ reseau) ⇒ t2↦t1 ∉ reseau)  not theorem ›
END
```

FIGURE 1 – Contexte Ctx_0

Contexte Ctx_0 **:** Ce contexte introduit les éléments de base du réseau ferroviaire, tels que les ensembles de données comme les tronçons (TRONCONS) et les trains (TRAINS). Il définit également des constantes comme les arrêts (arrets) qui sont des tronçons particuliers du réseau.

On a aussi défini la position initiale du train (p0) qui est une fonction totale qui associe chaque train (TRAINS) à un arrêt initial (arrets) vu qu'au début de la simulation, chaque train est positionné initialement sur un arrêt.

On a aussi défini la structure du réseau à travers une relation binaire (reseau) qui décrit comment les tronçons sont connectés entre eux. ce réseau qui est un graphe de tronçons.

On a défini dans ce contexte des axiomes garantissant des propriétés importantes, comme le fait que chaque arrêt est connecté à un seul tronçon d'entrée et de sortie (AXIOME 7 et AXIOME 8, et aussi qu'il ne soit pas connecté à un autre arrêt (AXIOME 10) et que le réseau est cohérent (un tronçon ne peut pas être relié à lui-même qui est défini par AXIOME 6 (*relation non-irréflexive*)), et aussi qu'il n'est pas symétrique, ce qui signifie que deux tronçons peuvent être adjacents dans un sens mais pas nécessairement dans l'autre (garder une seule direction) défini par AXIOME 16 afin de limiter de modéliser les voies bidirectionnelles.

Machine initiale Mch_0 **:** Cette machine décrit le comportement dynamique du système ferroviaire en utilisant le contexte Ctx_0. Elle modélise :

— La position des trains sur le réseau (trainPosition).
— Les routes assignées aux trains (trainRoute).
— La destination des trains (destination).

— L'historique des tronçons parcourus (routeParcourue).

Les invariants garantissent :

— Que chaque train est associé à un tronçon (trainPosition est une fonction totale).
— Qu'une route est une suite de tronçons à parcourir (trainRoute est une fonction partielle qui associe certains trains à des séquences de tronçons, avec un codomaine $\mathcal{P}$(Troncons)).
— Qu'une destination, si elle est assignée, est bien un arrêt du réseau (destination est une fonction partielle des trains vers les arrêts).
— Que l'historique des tronçons parcourus est maintenu (routeParcourue est une fonction partielle des trains vers des ensembles de tronçons).

Événements principaux de Mch_0

— **INITIALISATION** : Cet événement initialise l'état du système en positionnant les trains sur leurs arrêts initiaux (position p0), en assignant des routes vides (trainRoute := $\emptyset$), en supprimant toute destination (destination := $\emptyset$), et en vidant l'historique des trajets (routeParcourue := $\emptyset$).

— **Mouvement** : Cet événement modélise le déplacement d'un train d'un tronçon à un autre en respectant plusieurs contraintes :

  — Le train doit appartenir à l'ensemble des TRAINS (GRD1) et doit avoir une route assignée pour pouvoir avancer (GRD5).
  — Le tronçon suivant doit être adjacent à la position actuelle du train pour garantir un déplacement valide (GRD2).
  — Le tronçon suivant ne doit pas être occupé par un autre train (GRD4).

  Après satisfaction de ces gardes, l'événement met à jour la position du train et ajuste sa route en supprimant le tronçon actuel, illustrant ainsi sa progression.

— **AssignerRoute** : Cet événement assigne une nouvelle route à un train. Il s'assure que :

  — La route est un ensemble valide de tronçons (GRD2).
  — La route contient au moins deux tronçons (GRD3).
  — Le tronçon actuel du train appartient bien à la route (GRD6).
  — La route suit la topologie du réseau sans introduire d'incohérences (GRD4).
  — La route assignée ne doit pas être totalement incluse dans l'historique des tronçons déjà parcourus (GRD9).

$$\neg(train \in dom(routeParcourue)) \lor (route \not\subseteq routeParcourue(train))$$

  Cette condition garantit que :

  - Si un train n'a encore parcouru aucun tronçon, il peut recevoir n'importe quelle route.
  - Si un train a déjà parcouru certains tronçons, la nouvelle route doit contenir au moins un tronçon qu'il n'a pas encore visité.
  - Cela empêche de réassigner un trajet **déjà entièrement parcouru**, évitant ainsi des boucles et assurant une progression du train.

  Une fois validée, la route est assignée à la variable trainRoute.

- **AssignerDestination** : Cet événement attribue une destination à un train. Il s'assure que :

  - La destination est bien un arrêt du réseau (GRD2).
  - La destination est différente de la position actuelle du train (GRD4).
  - Un train ayant déjà une destination doit l'atteindre avant d'en recevoir une nouvelle (GRD5).

  Une fois validée, la destination est associée au train.

- **DebloquerTrain** : Cet événement permet de rediriger un train si un blocage est détecté. Il vérifie que :

  - Deux trains sont bloqués sur le même tronçon (GRD5).
  - Une nouvelle route alternative peut être définie (GRD6).
  - La nouvelle route ne doit pas contenir les tronçons précédemment parcourus pour éviter les boucles (GRD9).

  Après validation, le train reçoit une nouvelle route, lui permettant de contourner l'obstacle.

- **Historique des routes parcourues** : Cet événement met à jour la variable routeParcourue pour enregistrer les tronçons empruntés par chaque train. Il s'assure que :

  - Le train est en mouvement (GRD1).
  - Le tronçon suivant est bien connecté à la position actuelle du train (GRD2).
  - Les bifurcations sont enregistrées pour éviter de les reprendre inutilement (GRD7).

  Une fois validé, le tronçon parcouru est ajouté à l'historique.

2.3 Validation et Vérification

2.3.1 Animation avec ProB

Paramètres de simulation La simulation a été effectuée avec les paramètres suivants :

— Quatre troncons dans le réseau dont deux arrêts (T_1 et T_2) (reseau).
— Deux trains placés sur des positions initiales. Train1 sur T_1 et Train2 sur T_2. (p0).

Déroulement de la simulation

1. **Initialisation** : Le réseau suivant à été généré, on vérifie bien que c'est un réseau connexe. Les trains sont placés à leurs positions initiales et n'ont pas encore de route assignée.
2. **Assignation de destinations** : Chaque train reçoit une destination qui est un arrêt du réseau.
3. **Calcul des routes** : L'événement **AssignerRoute** attribue des trajets aux trains en tenant compte des bifurcations et en évitant les tronçons déjà parcourus.
4. **Déplacements** : Les trains avancent sur leur route en respectant les règles de connexion du réseau (Mouvement).
5. **Mise à jour des trajets parcourus** : L'historique des tronçons visités est enregistré dans routeParcourue.
6. **Gestion des blocages** : Si un train est bloqué, un itinéraire alternatif est recalculé avec DebloquerTrain.

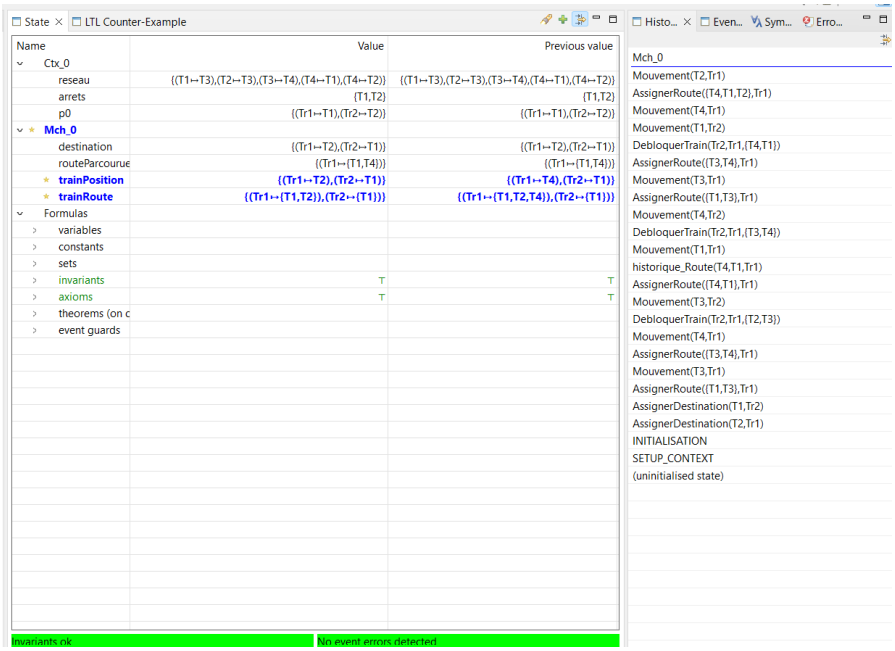


FIGURE 2 – Simulation ProbB

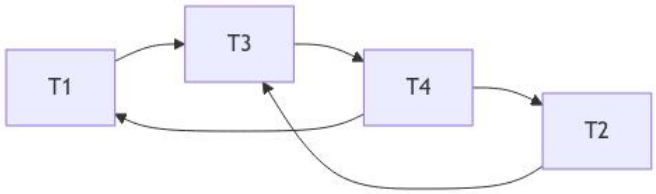


FIGURE 3 – Le réseau généré

Validation et résultats Pour cet exemple, on remarque que les trains ont bien atteints leurs destinations, quand il y'a eu bloquage, le système a débloqué la circulation. Les invariants sont tous respectés et aucune erreur d'événement n'a été détéctée.

Après plusieurs cycles de simulation, nous avons validé que :
— Chaque train atteint bien sa destination.
— Les trains suivent des routes cohérentes et évitent les tronçons déjà parcourus.
— Aucune boucle infinie ou incohérence dans les déplacements n'est détectée.
— La gestion des bifurcations et des blocages fonctionne correctement.

Conclusion de la simulation Grâce à cette animation, nous avons pu vérifier que le modèle fonctionne correctement dans un réseau simplifié avec 4 tronçons.

2.3.2 Preuves et obligations de preuve

Voir la section 3.3.2 Preuves et obligations de preuve, Propriété LTL pour Mch_2

3 Couche « Matériel Roulant »

Dans un second temps, nous introduisons des notions de systèmes physiques à noter les wagons, la taille du train ainsi que l'emplacement de ces wagons par rapport au train. Pour cela, on a défini le contexte Ctx_1 et la machine Mch_2.

3.1 Exigences extraites

3.1.1 Hypothèses du système

— **HYP-1** : Un train est composé d'une succession ordonnée de wagons.

axm1, ctx$_1$: $\quad$ **wagons** $\in TRAINS \rightarrow (1..taille \rightarrow WAGONS)$

— **HYP-2** :Le nombre total de wagons est déterminé par la taille et le nombre de trains .

axm6, ctx$_1$:

card(WAGONS) = **taille** $\times$ **card(TRANS)**

3.1.2 Exigences fonctionnelles

— **REQ-1** : Les trains et leurs wagons doivent se déplacer sur les tronçons du réseau.

inv1, inv2, Mch_2 : $\quad trainPosition \in TRAINS \rightarrow TRONCONS$ et $wagonPosition \in WAGONS \rightarrow TRONCONS$

— **REQ-2** : Lorsqu'un train avance, son wagon de tête suit immédiatement.

act3, mouvementTete, Mch_2

— **REQ-3** : Un wagon doit toujours appartenir à un train.

grd3, attachWagon, Mch_2 : $\quad wagon \in ran(wagons(train))$

— **REQ-4** : Un train peut attacher un wagon uniquement s'il n'est pas déjà attaché à un autre train.

grd2, attachWagon, Mch_2 :
$$i \in dom(wagons(train)) \Rightarrow wagons(train)(i) \notin dom(wagonPosition)$$

— **REQ-5** : Un tronçon est considéré comme libre lorsqu'il ne contient aucun wagon.

grd8,mouvementTete : troncon_suivant $\notin$ ran(wagonPosition)

— **REQ-6** : La taille d'un train correspond au nombre de ses wagons (wagon moteur inclus).

— **REQ-5** : Un train occupe un tronçon dès lors que le wagon tête se trouve dans ce tronçon.

— **REQ-6** : Pour chaque train, chacun de ses wagons passe le tronçon/arrêt dans le même ordre.

— **REQ-7** : Un train est considéré comme arrivé lorsque tous ses wagons sont arrivés.

3.1.3 Exigences de sûreté

— **SAF-1** : Un wagon ne peut appartenir qu'à un seul train à la fois.

axm8, ctx$_1$: $\forall t_1, t_2 \cdot w \in \mathbf{WAGONS} \land t_1 \in \mathbf{TRAINS} \land t_2 \in \mathbf{TRAINS} \Rightarrow (t_1 \neq t_2 \land w \in (\mathbf{wagons}(t_1))) \Rightarrow w \notin (\mathbf{wagons}(t_2))$

— **SAF-2** : Un wagon ne peut pas se trouver sur un tronçon où se trouve déjà un autre wagon.

grd8, mouvementTete, Mch_2 : $troncon_suivant \notin ran(wagonPosition)$

— **SAF-3** : Un train ne peut pas avancer si ses wagons bloquent la voie.

grd10, mouvementTete, Mch_2 :
$$\forall i, \quad i \in dom(wagons(train)) \Rightarrow wagons(train)(i) \notin dom(wagonPosition)$$

— **SAF-4** : Un wagon ne peut pas précéder son train.

grd3, mouvementWagon, Mch_2

3.2 Modélisation en Event-B

Contexte Ctx_1 **:** On a étendu Ctx_0 pour définir Ctx_1 en introduisant un nouveau ensemble "les wagons" (**WAGONS**) et en définissant leur relation avec les trains.

On précise également dans ce contexte Ctx_1 la taille des trains, une constante entière naturelle (AXIOME 2).

Dans l'objectif d'assurer que chaque wagon appartient à un seul train et que les wagons sont uniques dans le système, on a défini des axiomes dans ce contexte.

CONTEXT `Ctx_1`
EXTENDS `Ctx_0`
SETS `WAGONS, TRAINS, TRONCONS`
CONSTANTS `wagons, taille, w0`
AXIOMS
- **axm1** : $wagons \in TRAINS \rightarrow (1..taille \rightarrow WAGONS)$
- **axm2** : $taille \in \mathbb{N}$
- **axm3** : $\forall w, t_1, t_2 \cdot w \in WAGONS \land t_1 \in TRAINS \land t_2 \in TRAINS \Rightarrow ((t_1 \neq t_2 \land w \in \text{ran}(wagons(t_1))) \Rightarrow w \notin \text{ran}(wagons(t_2)))$
- **axm4** : $\{w \mid \exists t \cdot t \in TRAINS \land w \in \text{ran}(wagons(t))\} = WAGONS$
- **axm5** : $taille = 2$
- **axm6** : $\text{card(WAGONS)} = \text{taille} \times \text{card(TRANS)}$
- **axm7** : $w0 \in WAGONS \rightarrow TRONCONS$

Étant donné qu'un wagon est considéré comme un élément insécable d'un train. Par conséquent, un train est une succession de wagons qui se touchent, ce qui nous a amenés à définir une constante **wagons** comme étant une fonction totale.

On a aussi précisé qu'un wagon ne peut appartenir qu'à un seul train à la fois. En d'autres termes, les wagons sont uniques et ne peuvent pas être partagés entre plusieurs trains dans axiome 3 (AXM3) comme montré ci-dessus, et que tous les wagons du système sont associés à au moins un train afin qu'il n'y a pas de wagons seuls séparés dans notre modèle par l'axiome 4 (AXM4).

Tout comme on a établi la position de départ du train dans le contexte Ctx_0 (**p0**), on fait de même pour les wagons **w0** en définissant une fonction totale qui associe chaque wagon (**WAGONS**) à un tronçon initial (**TRONCONS**).

Afin de garantir une pertinence et une cohérence à notre modélisation entre les trains et les wagons en termes de cardinal, on a bien précisé par l'axiome 5 (AXM5) que

$$\text{card(WAGONS)} = \text{taille} \times \text{card(TRANS)}$$

Pour l'animation et afin de simplifier notre modélisation, on a fixé la *taille* des trains à 2 wagons, on a également défini une partition de l'ensemble des wagons, leurs positions initiales et la manière dont ils sont répartis parmi les trains.

Machine Mch_2 **:** Cette Machine 2 raffine la machine Mch_0 et utilise le contexte Ctx_1 en introduisant des variables et des événements de gestion de la position des wagons et les contraintes physiques des trains.

On a préservé les variables des machines précédentes et on a ajouté une nouvelle variable wagonPosition qui représente la position de chaque wagon sur le réseau.

MACHINE `Mch_2`

REFINES `Mch_0`

VARIABLES `trainPosition, trainRoute, destination, wagonPosition`

INVARIANTS
- **inv1** : $wagonPosition \in WAGONS \rightarrow TRONCONS$

ÉVÉNEMENTS
- **INITIALISATION** : Initialise les positions des trains et des wagons, la route et la destination du train.
- `AssignerRoute` : même définition que dans la machine 0.
- `AssignerDestination` : même définition que dans la machine 0.
- `DebloquerRoute` : même définition que dans la machine 0.
- `historique_Route` : même définition que dans la machine 0.
- **mouvementTete** : Vérifie les conditions de déplacement vers un tronçon suivant et met à jour la position du train et sa motrice (premier wagon).
- `mouvementWagon` : Déplace les autres wagons vers la position de la tête du train.
- `attachWagon` : Attache un wagon à un train.

Puisqu'il est noté que si un train occupe un tronçon dès lors qu'au moins un wagon s'y trouve, on a défini l'invariant INV1 (comme montré ci-dessus) afin de garantir cette condition que chaque wagon est associé à un tronçon du réseau.

A l'**INITIALISATION** raffinée, la variable wagonPosition est initialisée à w0, où les wagons sont positionnés sur des tronçons initiaux (position initiale de wagon w0).

Par notre raffinement, on a préservé également les événements de la machine précédente Mch_0 à savoir **AssignerRoute**, **AssignerDestination**, **historique_Route** et

DebloquerTrain.

Lorsqu'un train se déplace, il en va de même pour tous ses wagons, de manière simultanée et uniforme.

Afin de garantir cela, nous avons raffiné l'événement **Mouvement** en deux sous-événements :

— **MouvementTete** : On déplace le train et sa tête (sa motrice, le premier wagon) vers un nouveau tronçon.

— **MouvementWagon** : On réajuste chaque wagon pour se positionner sur le même tronçon que celui du train après son déplacement, afin qu'ils suivent la motrice.

Grâce à cette décomposition, nous assurons un déplacement synchronisé de l'ensemble du train.

3.3 Validation et Vérification

3.3.1 Simulation et Animation de la machine Mch_2 avec ProB

Nous avons utilisé **ProB** pour vérifier les invariants et exécuter le modèle.

Pour illustrer notre animation, nous avons simulé le déplacement de deux trains, **Tr1 et Tr2**, qui se trouvent initialement sur deux arrêts distincts, **T1 et T2**, chacun ayant une taille maximale de deux wagons. L'objectif est de les acheminer vers leurs destinations respectives : **Tr1** doit atteindre **T2**, tandis que **Tr2** doit se rendre à **T1**.

Nous avons ainsi défini une partition de l'ensemble des wagons :

$$\text{partition}(WAGONS, \{wagon1\}, \{wagon2\}, \{wagon3\}, \{wagon4\}) \tag{1}$$

Ainsi, les wagons ont été répartis parmi les trains comme suit :

$$wagons = \{(Tr1 \mapsto \{(1 \mapsto wagon1), (2 \mapsto wagon2)\}), (Tr2 \mapsto \{(1 \mapsto wagon3), (2 \mapsto wagon4)\})\} \tag{2}$$

Et les positions initiales des wagons sont définies par :

$$w0 = \{wagon1 \mapsto T1, wagon2 \mapsto T1, wagon3 \mapsto T2, wagon4 \mapsto T2\} \tag{3}$$

Enfin, la taille des trains a été fixée à :

$$taille = 2 \tag{4}$$

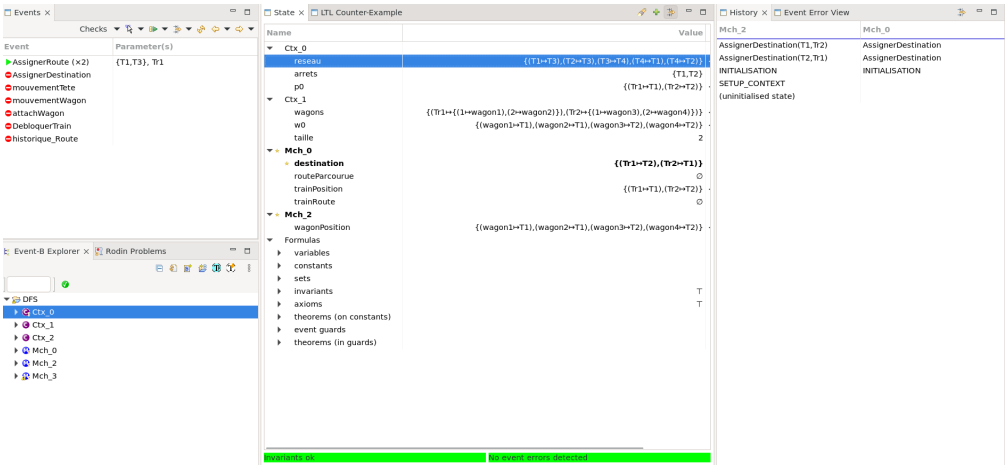
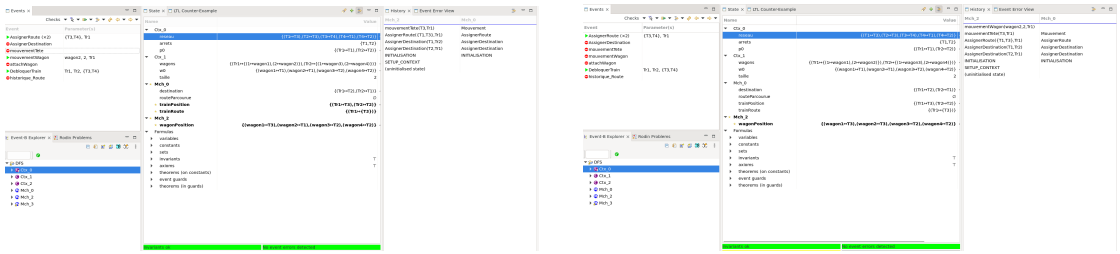


FIGURE 4 – Phase préliminaire d'exécution (Initialisation et Assignation des destinations des trains **Tr1 et Tr2** - Mch_2 -)



(a) Déplacement de Tr1 de T1 à T3 (b) Synchronisation des wagons de Tr1

FIGURE 5 – Phase intermédiaire d'exécution (Mouvement du train **Tr1** et ses wagons **wagon1 et wagon2** en synchronisation - Mch_2 -)

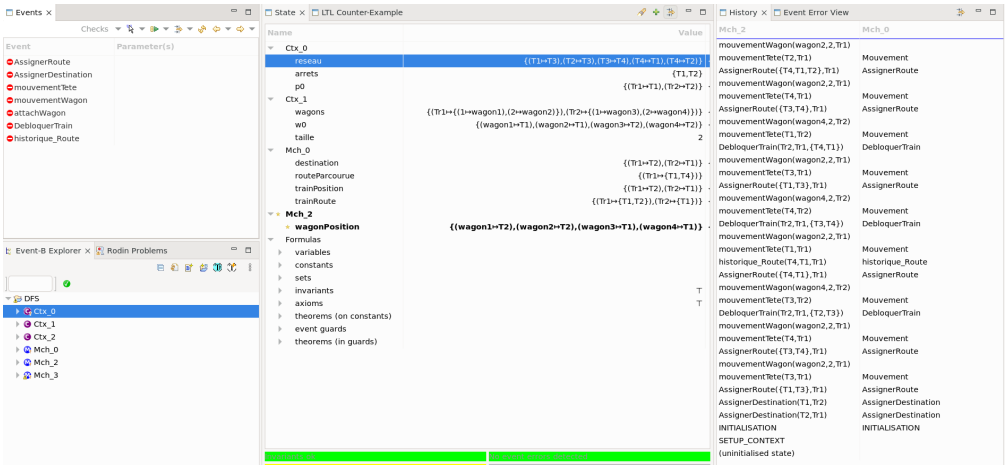


FIGURE 6 – Phase finale d'exécution (Arrivée des deux trains - Mch_2 -)

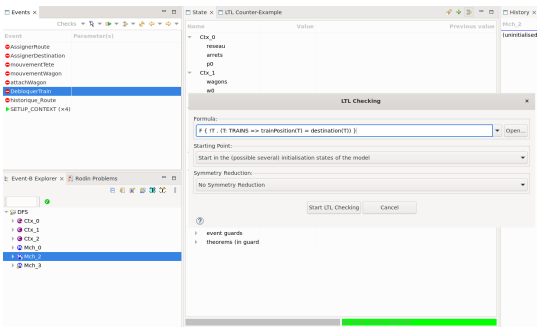
Les figures ci-dessus indiquent que les trains parviennent à leurs destinations dès que tous leurs wagons ont atteint l'arrêt final. Ceci respecte l'exigence **REQ-9**, qui indique qu'un train est arrivé lorsque tous ses wagons ont atteint leur destination. En plus, un tronçon est libre lorsqu'il ne renferme plus aucun train, conformément à **REQ-10**.

Notre animation se termine dès que tous les trains arrivent à leurs destinations respectives et que tous les segments sont dégagés. Cette conclusion de simulation signifie que tous les événements planifiés ont été réalisés avec succès et que les trains ont complété leur trajet conformément aux règles établies.

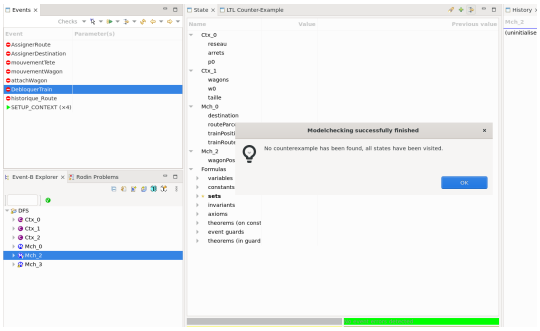
3.3.2 Preuves et obligations de preuve, Propriété LTL pour Mch_2

1. Un train finira par atteindre sa destination :

$$\diamond(\forall t \in \text{TRAINS} \implies (\text{trainPosition}(t) = \text{destination}(t)))$$



(a) Propriété LTL 1

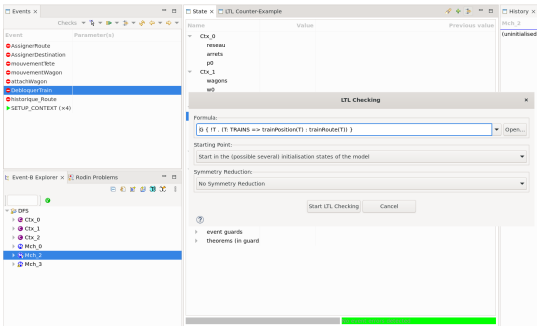


(b) Validation de Propriété LTL 1

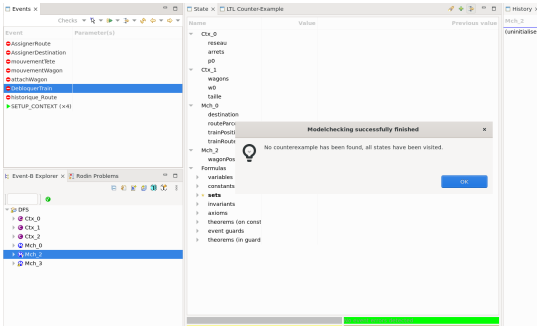
FIGURE 7 – Propriété LTL et Validation (- Mch_2 -)

2. La position du train reste toujours dans son itinéraire (propriété de continuité du train sur la route) :

$$\square\,(\forall t \in \text{TRAINS} \implies (\text{trainPosition}(t) = \text{trainRoute}(t)))$$



(a) Propriété LTL 2

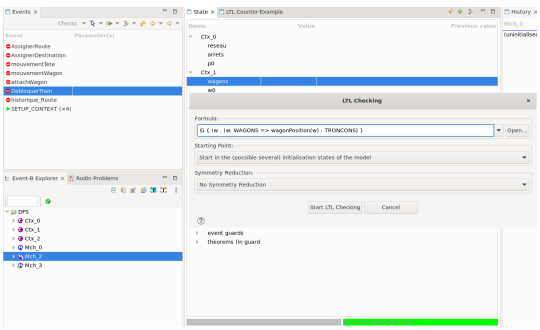


(b) Validation de Propriété LTL 2

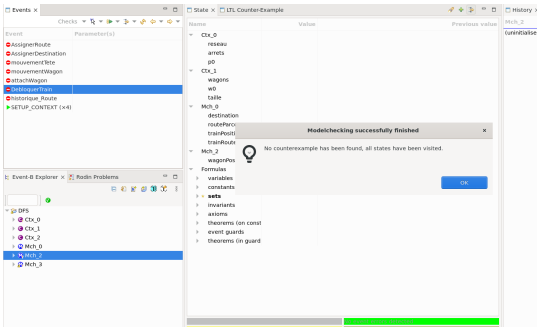
FIGURE 8 – Propriété LTL et Validation (- Mch_2 -)

3. On a également vérifier que la position de chaque wagon est toujours correcte :

$$\square\,(\forall w \in \text{WAGONS}\,(\text{wagonPosition}(w) \in \text{TRONCONS}))$$



(a) Propriété LTL 3

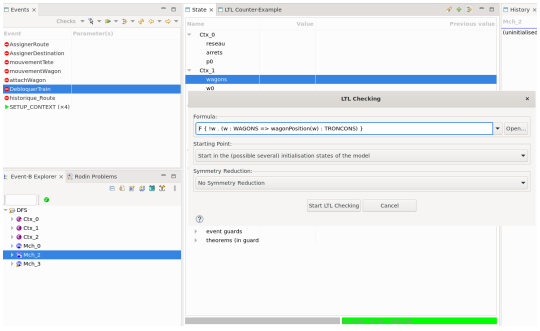


(b) Validation de Propriété LTL 3

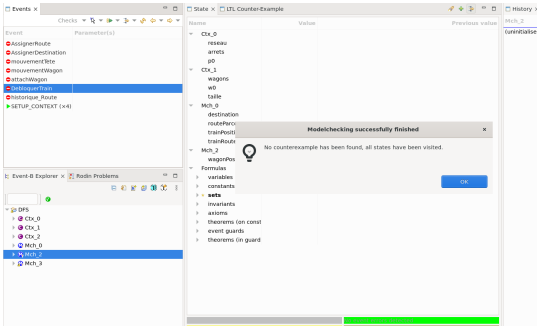
FIGURE 9 – Propriété LTL et Validation (- Mch_2 -)

4. Chaque wagon finit toujours dans un tronçon :

$$\Diamond(\forall w \in \text{WAGONS} \implies (\text{wagonPosition}(w) \in \text{TRONCONS}))$$



(a) Propriété LTL 4

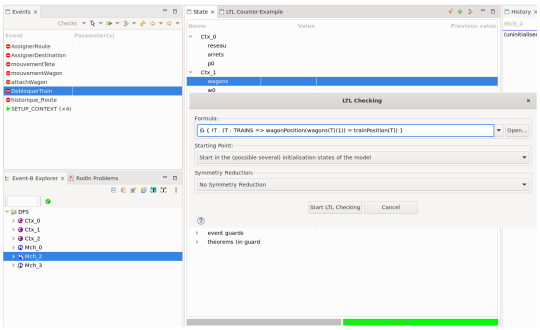


(b) Validation de Propriété LTL 4

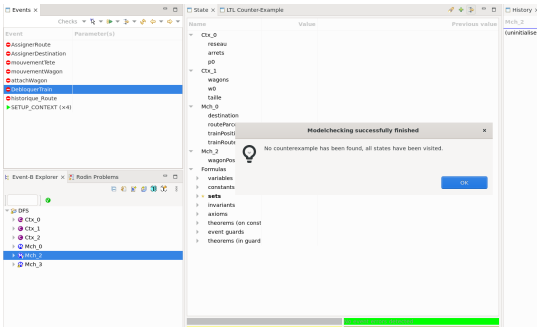
FIGURE 10 – Propriété LTL et Validation (- Mch_2 -)

5. Un wagon de tête doit toujours être dans le même tronçon que le train :

$$\Box(\forall T \in \text{TRAINS} \implies (\text{wagonPosition}(wagons(T)(1)) = \text{trainPosition}(T)))$$



(a) Propriété LTL 5



(b) Validation de Propriété LTL 5

FIGURE 11 – Propriété LTL et Validation (- Mch_2 -)

4 Couche « Physique et Dynamique »

4.1 Exigences extraites

4.1.1 Hypothèses du système

HYP-1 : La vitesse maximale $VMAX$ et l'accélération maximale $AMAX$ sont des entiers naturels ($\mathbb{N}$).

$$\textbf{axm1, axm2} : \quad VMAX \in \mathbb{N}, \quad AMAX \in \mathbb{N}$$

HYP-2 : La force maximale $FMAX$ est un entier négatif ($\mathbb{Z}$) et $FMAX < 0$.

$$\textbf{axm3} : \quad FMAX \in \mathbb{Z} \land FMAX < 0$$

HYP-3 : La vitesse initiale d'un train est comprise entre 0 et une vitesse maximale $VMAX$.

$$\textbf{axm11, Ctx\_2} : \quad \forall tr \in TRAINS, \quad 0 \leq v0(tr) \leq VMAX$$

HYP-4 : L'accélération initiale d'un train est comprise entre une accélération minimale et une accélération maximale.

$$\textbf{axm12, Ctx\_2} : \quad \forall tr \in TRAINS, \quad FMAX \leq a0(tr) \leq AMAX$$

4.1.2 Exigences fonctionnelles

REQ-1 : Chaque train commence avec son premier wagon à une position donnée $p0(tr)$.

$$\textbf{axm4} : \quad \forall tr \in TRAINS \Rightarrow w0(wagons(tr)(1)) = p0(tr)$$

REQ-2 : L'accélération d'un train est toujours comprise entre $FMAX$ et $AMAX$.

$$\textbf{inv4} : \quad \forall tr \in TRAINS \Rightarrow FMAX \leq trainAcceleration(tr) \leq AMAX$$

4.1.3 Exigences de sûreté

SAF-1 : Un train ne peut pas accélérer au-delà des limites.

grd2, modifierAcceleration : $FMAX \leq a \leq AMAX$

SAF-2 : Un train ne peut pas freiner s'il est déjà à l'arrêt.

grd2, freinerTrain : $trainSpeed(train) > 0$

4.2 Modélisation en Event-B

La troisième couche est appelé **Physique et Dynamique**. Elle modélise le comportement des trains en tant qu'objets physiques dans le temps. Cette couche modélise la position du train, sa vitesse, son accélération, l'autorité de mouvement et autres. Un train se définit par sa position avant, sa position après et sa longueur, qui reste constante dans tous les cas. Il se déplace selon un certain modèle : *Mouvement libre*, *Freinage* ou *Attente*, en fonction de la circulation et de la sécurité du chemin de fer.

Contexte Ctx_2 **:** Cette machine repose sur le contexte Ctx_2, qui étend le contexte `Ctx_1` en introduisant des constantes liées aux limites physiques des trains. Ces constantes sont essentielles pour garantir que les trains respectent les contraintes de vitesse et d'accélération.

CONTEXT `Ctx_2`
EXTENDS `Ctx_1`
CONSTANTS `VMAX, FMAX, AMAX` **AXIOMS**
- **axm1** : $VMAX \in \mathbb{N}_1$
- **axm2** : $AMAX \in \mathbb{N}_1$
- **axm3** : $FMAX \in \mathbb{Z} \ \wedge \ FMAX < 0$

— `VMAX` : La vitesse maximale que peut atteindre un train.
— `AMAX` : L'accélération maximale que peut atteindre un train.
— `FMAX` : La force de freinage maximale (accélération négative) que peut appliquer un train.

Les axiomes associés à ces constantes sont les suivants :
— `axm1` : `VMAX` est un entier naturel strictement positif.
— `axm2` : `AMAX` est un entier naturel strictement positif.
— `axm3` : `FMAX` est un entier strictement négatif.

Machine *Mch_3* : La machine `Mch_3` raffine la machine précédente `Mch_2` en introduisant des variables et des invariants supplémentaires pour gérer la position des trains, leur vitesse, leur accélération, et leur route.

MACHINE `Mch_3`

SETS `trainSpeed, trainAcceleration, endOfRoute`

VARIABLES
- **trainSpeed**
- **trainAcceleration**
- **modes**
- **endOfRoute**

Variables Les variables de la machine sont les suivantes :
- `trainPosition` : La position actuelle de chaque train sur le réseau.
- `trainRoute` : La route assignée à chaque train, représentée par une séquence de tronçons.
- `destination` : La destination finale de chaque train.
- `wagonPosition` : La position de chaque wagon sur le réseau.
- `trainSpeed` : La vitesse actuelle de chaque train.
- `modes` : Un indicateur du mode du train (Arret :0, libre :1, freinage :2).
- `trainAcceleration` : L'accélération actuelle de chaque train.
- `endOfRoute` : Un indicateur de la fin de la route pour chaque train.

Invariants Les invariants de la machine garantissent que les trains respectent les contraintes physiques et logiques du système :
- `inv1` : La position du train correspond à la position du wagon de tête.
- `inv2` : La vitesse de chaque train est un entier.
- `inv3` : La vitesse de chaque train ne dépasse pas `VMAX`.
- `inv4` : L'accélération de chaque train est comprise entre `FMAX` et `AMAX`.
- `inv5` : L'indicateur de fin de route est un entier positif.
- `inv6` : La vitesse des trains est toujours positive ou nulle.

Événements Les événements de la machine permettent de modéliser les actions possibles des trains :
- `AssignerRoute` : Assigner une nouvelle route à un train.
- `AssignerDestination` : Assigner une destination à un train.
- `mouvementTete` : Déplacer la tête du train vers un tronçon suivant.
- `mouvementWagon` : Déplacer un wagon vers la position de la tête du train.
- `attachWagon` : Attacher un wagon à un train.
- `modifierAcceleration` : Modifier l'accélération d'un train.
- `freinerTrain` : Freiner un train en appliquant la force de freinage maximale.

=> Ces événements de la machine permettent de modéliser les actions possibles des trains. Parmi ces événements, trois nouveaux ont été ajoutés pour gérer les aspects

liés aux wagons et à la dynamique du train : modifierAcceleration et freinerTrain. Les événements existants (AssignerRoute, AssignerDestination, mouvementTete, et mouvementWagon, attachWagon) ont également été modifiés pour intégrer les informations sur les wagons et les lier au mouvement du train.

4.3 Validation et Vérification

4.3.1 Simulation et Animation de la machine Mch_3 avec ProB

La validation de la machine `Mch_3` a été réalisée à l'aide de l'outil ProB.

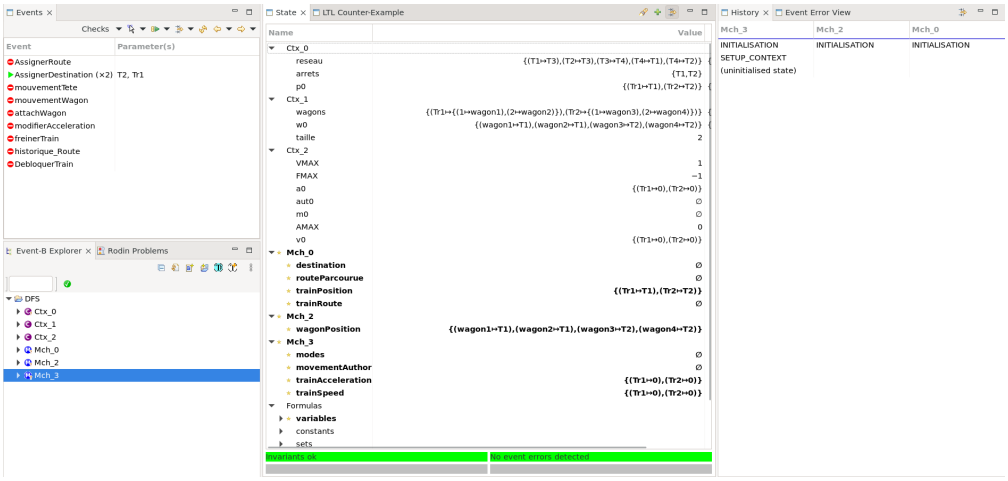
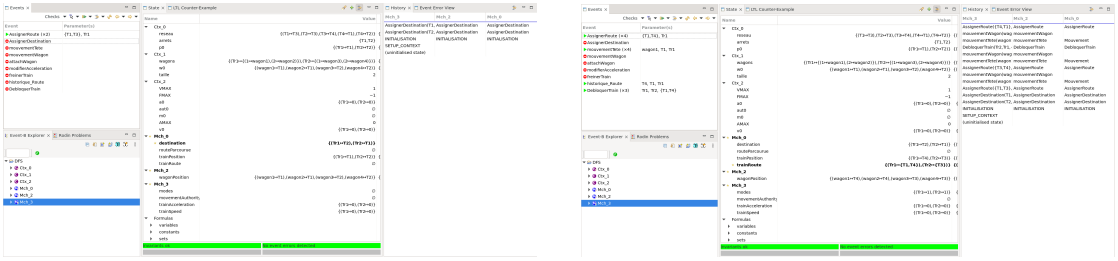


FIGURE 12 – Phase préliminaire d'exécution (Initialisation)



(a) Assigner destinations aux deux trains (b) Mouvement des trains

FIGURE 13 – Phase intermédiaire d'exécution

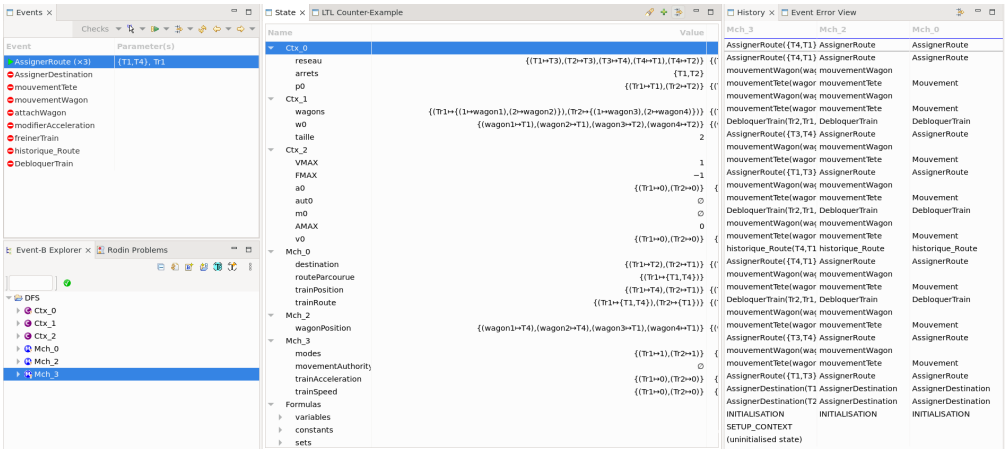


FIGURE 14 – Phase finale d'exécution

5 Bilan Fonctionnel

Tout au long de la réalisation de ce projet, nous avons dû faire des choix de conception qui ont influencé la structuration et la modélisation du système. L'un des principaux défis a été de définir avec précision les concepts fondamentaux, tels que les tronçons, les arrêts et la gestion des itinéraires des trains. Le sujet nous a laissé une certaine liberté d'interprétation, ce qui nous a permis d'explorer différentes approches, mais a également engendré des hésitations et des ajustements successifs.

Un autre défi fonctionnel majeur a été la gestion des itinéraires et des contraintes de sécurité. Nous avons dû assurer que chaque train suivait un itinéraire valide, tout en garantissant l'absence de conflits sur les tronçons et en respectant les contraintes de connexité du réseau. La répartition des responsabilités entre les différentes couches du modèle a nécessité plusieurs itérations afin d'assurer une cohérence globale.

Malgré ces difficultés, ce projet a été particulièrement enrichissant sur le plan fonctionnel. Il nous a permis d'approfondir notre compréhension des systèmes complexes et de leur formalisation, en nous confrontant à des choix critiques et en nous poussant à structurer notre raisonnement de manière rigoureuse. La nécessité de justifier chaque décision et de garantir la cohérence globale du modèle nous a permis d'acquérir une approche plus méthodique et plus rigoureuse de la modélisation formelle.

6 Bilan Technique

Sur le plan technique, nous avons travaillé avec l'outil Rodin et le langage Event-B pour formaliser notre système.

L'un des principaux défis techniques a été le débogage des erreurs de modélisation. Certaines erreurs étaient liées à la syntaxe, tandis que d'autres provenaient d'incohérences dans la définition des contraintes et des événements. Un des aspects les plus complexes a été la gestion des obligations de preuve : bien que notre logique nous paraisse correcte, certaines preuves ne se validaient pas automatiquement, nous obligeant à revoir nos formulations et à repenser certains invariants.

Malgré ces défis, l'utilisation de Rodin nous a permis d'exploiter pleinement les capacités de vérification formelle offertes par Event-B. La validation des propriétés du système, l'analyse des obligations de preuve et les tests via la simulation nous ont permis d'assurer une certaine robustesse du modèle. Ce projet nous a également permis de mieux comprendre l'importance des méthodes formelles dans la conception et la validation des systèmes critiques.

7 Bilan Personnel

7.1 Contribution de chaque membre

7.1.1 Otman FARHAT

En tant que membre du groupe, j'ai activement participé à la conception et au développement du modèle Event-B. Mes principales contributions ont été :

- Conception des couches du modèle responsable de la modélisation des trains, wagons et de leurs interactions.
- Implémentation des contextes et des machines ainsi que les évènements, invariants,etc... liés à cette couche, en veillant à la cohérence avec les autres parties du modèle.
-Participation aux séances de validation avec ProB et à la résolution des obligations de preuve.

Les points positifs de ce projet sont une bonne organisation du groupe et une approche collaborative efficace dans la conception du modèle. Nous avons su répartir les responsabilités de manière équilibrée et travailler de façon cohérente.

Cependant, le principal défi a été le manque de temps, lié aux nombreux autres examens et projets à cette période. Cela nous a parfois obligés à nous précipiter dans certaines décisions de modélisation, au détriment d'une réflexion plus approfondie.

Dans l'ensemble, j'ai trouvé ce projet très intéressant et enrichissant. Bien que complexe, la modélisation formelle d'un système ferroviaire m'a permis d'approfondir mes compétences en méthodes formelles et en raisonnement logique. C'est une expérience qui me sera très utile pour la suite de ma formation et de ma carrière.

7.1.2 AKKAR Khadija

De manière générale, la réalisation d'un projet de développement formel a représenté une étape très inspirante et formatrice pour moi. La démarche qui consiste à partir d'un cahier des charges, passer plusieurs heures pour l'analyser et en extraire les informations nécessaires, pour ensuite dessiner le document comme un système formel, est un défi passionnant. Je suis contente de son résultat et je chercherai certainement une solution plus approfondie si le temps m'est accordé. Dans l'ensemble, ce projet m'a appris à modéliser des systèmes complexes, m'a appris l'importance de la rigueur et de la précision.

Cette phase de modélisation et les premières implémentations ont été les plus critiques pour le succès du projet. Travailler en équipe a été une bonne expérience car elle nous a permis de combiner nos compétences et de partager nos idées pour une solution plus solide. À la fin, j'ai travaillé en collaboration avec tous les membres pour rédiger ce rapport technique. J'ai écrit en détail la contribution, et en particulier pour la couche 3, les différents modèles que nous avons créés, les exigences auxquelles nous avons abouti, les invariants que nous avons imposés, et les différents événements que nous avons suscités. Le but de cette écriture est de garder une trace des efforts accomplis, en décrivant le mieux possible le modèle pour permettre à quiconque souhaitant l'étudier et l'améliorer de le comprendre.

Ce fut un grand plaisir pour moi de travailler avec une équipe très motivée et ambitieuse. Tout le monde est venu avec une ou plusieurs compétences et visions et c'est ce qui nous a aidés à vaincre les difficultés pour mener à bien ce projet. Au début de cette expérience, je ne savais presque rien sur le développement formel. Cependant, cette expérience m'a non seulement aidé à savoir sur ce dernier, mais aussi à me renforcer dans ma capacité de travail en équipe.

7.1.3 Safae BELAHRACH

J'ai eu la chance de faire partie de ce projet en équipe, ce qui a exigé de réfléchir ensemble et de constamment trouver des compromis. La phase de modélisation a été longue et complexe, mais cruciale pour la suite. Après la phase de modélisation, qui a pris beaucoup de temps, j'ai collaboré avec l'équipe pour finaliser le développement et les vérifications. En effet, malgré une modélisation initiale, nous avons souvent dû ajuster notre modèle en ajoutant des spécifications oubliées ou en modifiant celles qui ne correspondaient pas à nos attentes.
J'ai aussi participé à la rédaction du rapport, qui résume tout le travail accompli.
Ce projet m'a permis de voir la modélisation sous un autre angle. Extraire des informations d'un cahier des charges peu clair en le transformant en un système formel était un défi, mais grâce à l'entraide de l'équipe, nous avons réussi à surmonter ce défi et à mettre en place une solution.
À la fin du projet, j'ai travaillé avec Otman sur la correction, la validation, la vérification, la simulation de la machine 2 en ProB et sur la rédaction du rapport, où on a expliqué

en détail notre modélisation, les exigences, les invariants et les événements.
Ce travail de longue haleine était essentiel pour garder une trace de notre effort et rendre notre modèle compréhensible sans oublier le travail avec mon équipe motivée qui a été un réel plaisir, et une expérience très enrichissante en termes de connaissances et compétences.
La phase la plus difficile a été de corriger les erreurs dans le modèle et d'ajuster les animations pour qu'elles correspondent à nos attentes.
J'ai aidé à repérer ces erreurs. J'ai aussi pris part à la vérification du rapport final.
Cette expérience m'a non seulement aiguisé mon esprit critique, mais elle m'a également donné une vision plus claire de ce que sont les métiers fonctionnels.

7.1.4 Ayoub Diouri

Au cours de ce projet, j'ai pu aborder la modélisation et les spécifications sous une nouvelle perspective. J'ai participé à l'extraction d'informations d'un cahier des charges peu explicite et très vague, et nous avons réussi, en collaboration, à concrétiser un travail minutieux répondant aux exigences spécifiées.
Suite à la phase de modélisation, la tâche la plus complexe a consisté à effectuer les ajustements et rectifier les anomalies identifiées ainsi qu'à suivre les animations pour voir où ça bloquait pour tout remettre en ordre. Pour cela, j'ai été accompagné de ma camarade Wissal, on a travaillé tous les deux sur l'identification des dysfonctionnements et l'implémentation des solutions appropriées dans la couche métier. J'ai également contribué à la validation des informations consignées dans le compte-rendu. Je suis satisfait d'avoir participé à un tel projet qui a développé mon esprit critique et m'a offert une nouvelle perspective sur les métiers fonctionnels.

7.1.5 Wissal ABOUMEJD

Mon travail s'est articulé autour de plusieurs aspects du projet, allant de la conception initiale à la correction et au débogage, ainsi qu'à la rédaction du rapport technique.

— **Développement du contexte 0 et la Machine 0** : J'ai participé à l'élaboration du contexte et la machine principale du projet en définissant les variables, invariants et événements essentiels au bon fonctionnement du modèle.

— **Correction des erreurs et validation du modèle** : Une partie importante de mon travail a consisté à identifier et résoudre, en collaboration avec Ayoub, les erreurs détectées par Rodin. Cela a impliqué l'ajustement des gardes des événements, la reformulation des invariants et la vérification de la cohérence globale du modèle.

— **Débogage et simulation avec ProB** : J'ai testé, en collaboration avec Ayoub, plusieurs scénarios pour m'assurer que le modèle répond aux exigences spécifiées

et respecte les contraintes définies.

— **Contribution à la rédaction du rapport** : J'ai participé à la rédaction du rapport final en documentant les différentes étapes du projet, notamment la description des exigences, des modèles développés et des solutions apportées. J'ai aussi structuré et clarifié les résultats obtenus.

Bilan du projet

L'un des points positifs du projet est l'organisation efficace du groupe et la collaboration étroite entre les membres de l'équipe. Chacun a pu contribuer à sa manière pour améliorer le modèle.

Cependant, nous avons été confrontés à plusieurs défis, notamment le manque de temps pour explorer toutes les alternatives possibles et approfondir certaines décisions de modélisation. Le processus de correction et de validation des preuves n'a pas été évident.

Dans l'ensemble, ce projet m'a permis d'améliorer mes compétences en modélisation formelle et en raisonnement logique. Il m'a offert une meilleure compréhension des méthodes formelles appliquées aux systèmes critiques, une expérience précieuse pour mes futurs projets.

8 Conclusion

Ce projet nous a permis d'explorer en profondeur la modélisation formelle à travers l'outil Rodin et le langage Event-B. En structurant un système complexe de gestion ferroviaire, nous avons été confrontés à des défis techniques et conceptuels qui nous ont amenés à affiner notre compréhension des méthodes formelles et à développer une approche rigoureuse de la conception logicielle.

Grâce à une collaboration efficace et une répartition équilibrée des tâches, nous avons pu modéliser un réseau de trains fonctionnel tout en respectant les exigences du cahier des charges. Les différentes étapes de validation, la gestion des obligations de preuve et l'analyse des comportements possibles du système nous ont offert une expérience enrichissante et formatrice.

Malgré certaines difficultés, notamment liées aux contraintes de temps et à la complexité de l'outil, ce projet nous a donné une vision plus claire des avantages et des limites des méthodes formelles dans le développement de systèmes critiques. En somme, il s'agit d'une expérience précieuse qui nous sera bénéfique pour nos futurs projets et notre parcours professionnel.

Bibliographie

Références

[1] https://fr.wikipedia.org/wiki/M%C3%A9thode_formelle_(informatique)

[2] https://prob.hhu.de/w/index.php?title=LTL_Model_Checking