



Méthode formelle de conception : Système de contrôle du trafic aéroportuaire

Jules HELLER - Abdelkader BELDJILALI



Table des matières

Introduction	2
1 Généralités	2
2 Expression textuelle des exigences	3
2.1 Description du système et de son environnement	3
2.2 Analyse des spécifications	3
2.3 Reformulation des exigences	4
3 Première modélisation abstraite	4
3.1 Analyse	4
3.2 Contexte du système	5
3.3 Machine :	5
3.4 Preuves	6
4 Premier raffinement	7
4.1 Analyse	7
4.2 Contexte	9
4.3 Glue	9
4.4 Convergence des nouveaux events	10
4.5 Machine	10
4.6 Preuves	12
5 Second raffinement	13
5.1 Analyse	13
5.2 Contexte	13
5.3 Glue	13
5.4 Preuves	17
Conclusion	17

Introduction

L'objet de ce TD est de modéliser un contrôleur automatique de trafic sur aéroport en interaction avec son environnement et de prouver formellement la correction du modèle formel à l'aide de l'outil rodin associé au plugin Atelier B

1 Généralités

Le système est modélisé sous forme d'une machine à états-transitions. Les états sont caractérisés par des constantes respectant des axiomes et des variables respectant des conditions nommées "invariants". Les constantes caractérisent le "contexte" du système i.e. l'aspect statique. Les variables décrivent l'aspect dynamique. Les transitions sont désignées par le terme event. Les events sont des actions conditionnées ou non par une garde. Une garde est une condition qui autorisera ou non l'action du système. La transition du système est supposée instantanée et se produire dès que la garde est vraie. Il peut survenir des cas d'indéterminisme externe lorsque plusieurs gardes sont vraies simultanément. Si elles sont toutes fausses, le système est bloqué. Si, à tout instant, une garde est vraie, le système ne se termine jamais.

La démarche itérative proposée dans l'exercice consiste à écrire les exigences a partir des spécifications ; chaque exigence est identifiée par un label qui permet la traçabilité et le classement en catégories notamment fonctionnelle et environnementale. On part d'un modèle abstrait simple, formalisé dans l'outil rodin. Des preuves notamment d'invariance sont effectuées. Le plugin Atelier B génère des obligations de preuves dans un langage correspondant à la logique du premier ordre associée à la théorie des ensembles. Atelier B utilise notamment une base de règles d'inférence associée à un moteur contenant des heuristiques et un prouveur de type SAT.

Le modèle formel est corrigé le cas échéant par ajout de gardes ou de conditions sur les gardes si on cherche à éviter les blocages. Les exigences sont complétées si nécessaire. Enfin par raffinement successifs, on passe d'une vision synoptique du système à une vue plus détaillée qui peut conduire jusqu'à la phase d'implémentation.

2 Expression textuelle des exigences

2.1 Description du système et de son environnement

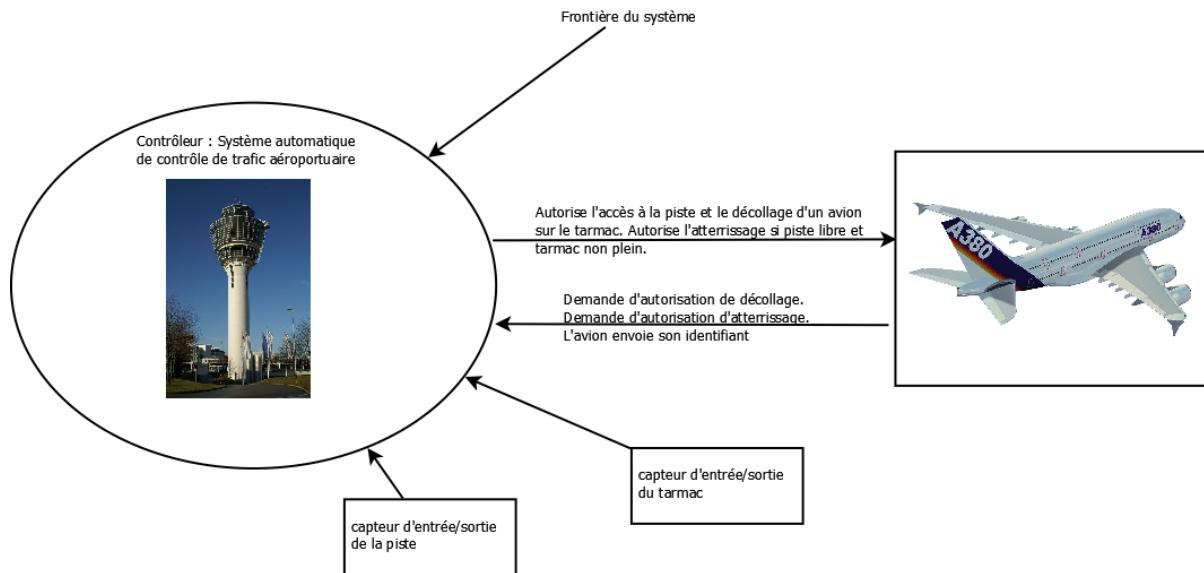


FIGURE 1 – Environnement du système

2.2 Analyse des spécifications

Le système à concevoir est un **contrôleur aérien automatique** chargé de contrôler le trafic sur le tarmac et d'autoriser l'accès à la piste des avions prêts au décollage.

Ce système est un logiciel qu'on appellera le Contrôleur. Ce contrôleur interagit avec son environnement à savoir les pilotes et des capteurs non précisés dans l'énoncé. Les spécifications sont incomplètes car le contrôleur doit forcément autoriser ou refuser les avions à l'atterrissement également. Il est nécessaire d'avoir un capteur qui permet de compter les avions qui entrent et sortent du tarmac et un autre capteur qui assure le comptage des avions sur la piste. Les avions ne peuvent atterrir que si l'aéroport n'est pas plein, 20 places maximum sont disponibles.

On suppose que les clearances sont données aux avions qui se trouvent sur le tarmac. Dès l'autorisation de décoller donnée, l'avion se rend sur la piste et décolle. Il n'y a pas d'étape intermédiaire. Un système de communication sécurisé de type data-link doit exister pour donner les clearances et recevoir les identifiants et les demandes des pilotes.

2.3 Reformulation des exigences

Label	Exigence
FON-1	Le contrôleur doit autoriser les avions à décoller et atterrir
FON-2	Le nombre d'avions immobilisés sur le tarmac est limité à 20 y compris ceux en attente de décollage
FON-3	Des avions entrent et quittent la piste d'atterrissement décollage
FON-4	Des avions entrent sur le tarmac et le quittent
FON-5	La piste ne peut être occupée par un avion au décollage et un avion à l'atterrissement en même temps.
FON-6	Le nombre de décollages ou atterrissages successifs n'est pas limité
FON-7	Le contrôleur doit fixer et délivrer les clearances à l'avance
FON-8	Le contrôleur ne doit autoriser l'avion qu'après l'envoi de son identifiant
FON-9	Le contrôleur doit soit refuser, soit accepter, soit mettre en attente l'avion demandeur
FON-10	Le contrôleur doit refuser la clearance après 10mn de mise en attente.
ENV-1	Tout avion se dirigeant vers la piste doit avoir une autorisation de décoller
ENV-2	Le système est muni d'un capteur qui permet de compter les avions sur la piste
ENV-3	Le système est muni d'un capteur qui permet de compter les avions sur le tarmac

TABLE 1 – Tableau des exigences V1

3 Première modélisation abstraite

3.1 Analyse

On veut initialement considérer très peu d'exigences. L'énoncé nous indique que seules les entrées et sorties de l'espace de l'aéroport sont étudiées. On effectue donc une étude formelle sur un système très simplifié où la piste et le tarmac ne sont pas différenciés. Nous étudions uniquement les échanges d'avions entre l'espace de l'aéroport et l'espace extérieur, ce qu'on modélise en utilisant deux events¹ : decoller et atterrir.

1. un event est nommé transition dans la sémantique des machines à états finis. C'est un couple (garde/action). L'action s'exécute si la garde est vraie.

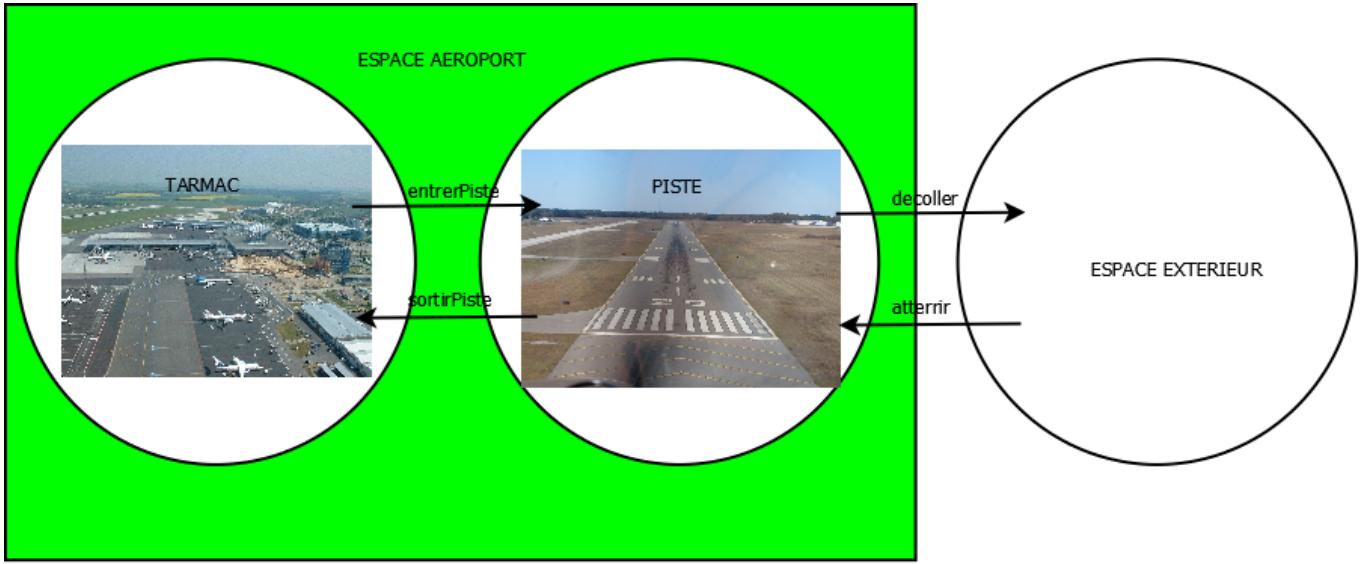


FIGURE 2 – Modèle abstrait : seuls les échanges avec l'extérieur sont pris en compte.

3.2 Contexte du système

L'état statique du système, nommé contexte dans la sémantique event-b, est caractérisé par $ntmax$, le nombre total maximum d'avions dans l'espace de l'aéroport i.e. la capacité de l'aéroport, tarmac et piste confondues.

```

CONTEXT contexte0
CONSTANTS
    ntmax
AXIOMS
    axm1: ntmax ∈ N
    axm2: 0 < ntmax
    axm3: ntmax = 20
END

```

FIGURE 3 – Etat statique du système avec deux axiomes redondants

3.3 Machine :

On fait abstraction de la piste. Ainsi, la seule variable à considérer est le nombre total d'avions présents dans l'espace de l'aéroport à un instant donné. On note nt cette variable. L'état dynamique du système, nommé machine dans la sémantique rodin/event-b, n'est constitué que de cette seule variable. Elle respecte deux conditions ou *invariants* :

```

MACHINE machine0
SEES contexte0
VARIABLES
    nt
INVARIANTS
    inv1: nt ∈  $\mathbb{N}$ 
    inv2: nt ≤ ntmax
    DLF0: ⟨theorem⟩ nt < ntmax ∨ nt > 0
EVENTS
Initialisation
begin
    act1: nt := 0
end
Event decoller ⟨ordinary⟩ ≡
when
    grd1: nt > 0
then
    act1: nt := nt - 1
end
Event atterrir ⟨ordinary⟩ ≡
when
    grd1: nt < ntmax
then
    act1: nt := nt + 1
end
END

```

FIGURE 4 – Etat dynamique du système

3.4 Preuves

Pour déboguer le modèle formel initial, il a fallu ajouter deux gardes pour garantir un nombre d'avions compris entre 0 et 20 et éviter les deadlock. Une preuve interactive, dans la perspective "proving", icône PP du "Proof control Panel", est nécessaire pour démontrer le théorème¹ d'invariant DLF0 pour prendre en compte toutes les hypothèses. C'est à ce stade que les règles d'inférences, en l'occurrence MON et OR_L, mettent en évidence qu'un axiome supplémentaire est nécessaire qui impose ntmax > 0. Ici, on fixe ntmax = 20. En effet, si ntmax = 0, le système est bloqué dès l'origine.

1. un théorème d'invariant est dépendant d'autres invariants placés avant

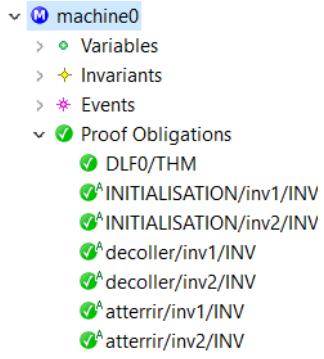


FIGURE 5 – Ajout de 2 gardes et d'un théorème pour éviter les blocages (invariant Deadlock free : DLF0)

Les exigences sont mises à jour pour tenir compte de ces oublis.

Label	Exigence
FON-1	Le contrôleur doit autoriser les avions à décoller et atterrir
FON-1.1	Le contrôleur doit fonctionner indéfiniment une fois lancé.
FON-2	Le nombre d'avions immobilisés sur le tarmac est limité à 20 y compris ceux en attente de décollage mais doit rester positif
FON-3	Des avions entrent sur et quittent la piste d'atterrissement décollage
FON-4	Des avions entrent sur le tarmac et le quittent
FON-5	La piste ne peut être occupée par un avion au décollage et un avion à l'atterrissement en même temps.
FON-6	Le nombre de décollages ou atterrissages successifs n'est pas limité
FON-7	Le contrôleur doit fixer et délivrer les clearances à l'avance
FON-8	Le contrôleur ne doit autoriser l'avion qu'après l'envoi de son identifiant
FON-9	Le contrôleur doit soit refuser, soit accepter, soit mettre en attente l'avion demandeur
FON-10	Le contrôleur doit refuser la clearance après 10mn de mise en attente.
ENV-1	Tout avion se dirigeant vers la piste doit avoir une autorisation de décoller
ENV-2	Le système est muni d'un capteur qui permet de compter les avions sur la piste
ENV-3	Le système est muni d'un capteur qui permet de compter les avions sur le tarmac

TABLE 2 – Tableau des exigences V2

4 Premier raffinement

4.1 Analyse

Dans cette partie, on introduit la piste pour évoluer vers un modèle plus concret. On peut ainsi spécifier le nombre d'avions se trouvant sur la piste en vue d'un décollage et après un atterrissage ainsi que le nombre d'avions sur le tarmac. Ce fait implique un "raffinement" des exigences environnementales ; on doit avoir un capteur capable de compter les avions au décollage et un capteur pour les avions à l'atterrissement. Ce qui donne les exigences modifiées suivantes :

Label	Exigence
FON-1	Le contrôleur doit autoriser les avions à décoller et atterrir
FON-1.1	Le contrôleur doit fonctionner indéfiniment une fois lancé.
FON-2	Le nombre d'avions immobilisés sur le tarmac est limité à 20 y compris ceux en attente de décollage mais doit rester positif
FON-3	Des avions entrent sur et quittent la piste d'atterrissement décollage
FON-4	Des avions entrent sur le tarmac et le quittent
FON-5	La piste ne peut être occupée par un avion au décollage et un avion à l'atterrissement en même temps.
FON-6	Le nombre de décollages ou atterrissages successifs n'est pas limité
FON-7	Le contrôleur doit fixer et délivrer les clearances à l'avance
FON-8	Le contrôleur ne doit autoriser l'avion qu'après l'envoi de son identifiant
FON-9	Le contrôleur doit soit refuser, soit accepter, soit mettre en attente l'avion demandeur
FON-10	Le contrôleur doit refuser la clearance après 10mn de mise en attente.
ENV-1	Tout avion se dirigeant vers la piste doit avoir une autorisation de décoller
ENV-2	Le système est muni d'un capteur qui permet de compter les avions sur la piste à l'atterrissement et un capteur pour les avions sur la piste au décollage.
ENV-3	Le système est muni d'un capteur qui permet de compter les avions sur la tarmac

TABLE 3 – Tableau des exigences V3

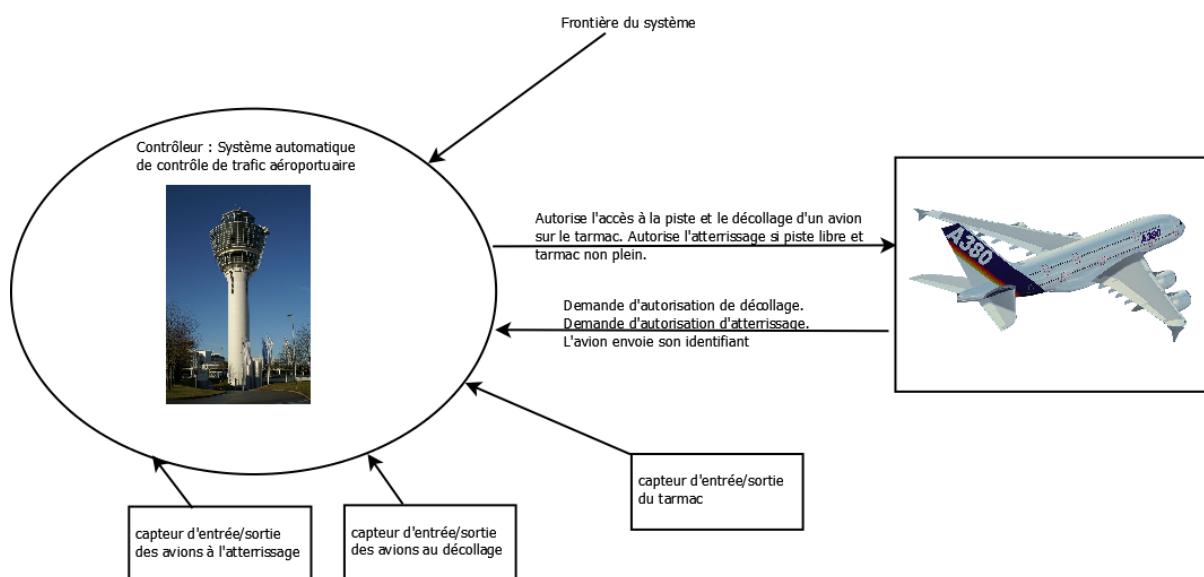


FIGURE 6 – Environnement du système : version 2



FIGURE 7 – Premier raffinement du système : piste et tarmac sont différenciés

Contrairement à ce qui se passe dans la réalité, les spécifications laissent la possibilité d'avoir au même moment plusieurs avions sur la piste au décollage ou à l'atterrissement dès lors qu'on ne mélange pas les décollages et atterrissages. La piste est, à un instant donné, en sens unique.

4.2 Contexte

Le CONTEXTE n'est pas modifié. La constante ntmax du système caractérise l'état statique du système concret associé à ce premier raffinement.

```

CONTEXT contexte0
CONSTANTS
    ntmax
AXIOMS
    axm1: ntmax ∈ N
    axm2: 0 < ntmax
    axm3: ntmax = 20
END

```

FIGURE 8 – Contexte premier raffinement

4.3 Glue

Ces variables vérifient $\text{nbAvionsDecollage} + \text{nbAvionsAtterrissage} + \text{nbAvionsTarmac} = \text{nt}$. C'est l'invariant de "glue" qui fait le lien entre les 3 variables concrètes et la variable abstraite nt.

4.4 Convergence des nouveaux events

On définit un variant entier naturel, en l'occurrence, $nbAvionsTarmac + 2 * nbAvionsAtterrissage$ dont on a vérifié qu'il est bien décrémenté par les deux nouveaux events introduits : *entrerPiste* et *sortirPiste* pour éviter qu'ils ne soient indéfiniment activés avec pour conséquence une interdiction des décollages et atterrissages.

4.5 Machine

On définit une nouvelle machine nommée "raffinement1" pour prendre en compte les nouvelles variables. La Machine raffinement1 est obtenue par raffinage de la machine0.

```

MACHINE raffinement1
REFINES machine0
SEES contexte0
VARIABLES
    nbAvionsDecollage
    nbAvionsAtterrissage
    nbAvionsTarmac
INVARIANTS
    inv1: nbAvionsDecollage ∈ ℕ
    inv2: nbAvionsAtterrissage ∈ ℕ
    inv3: nbAvionsTarmac ∈ ℕ
    inv4: nbAvionsDecollage + nbAvionsAtterrissage + nbAvionsTarmac = nt
    inv5: nbAvionsAtterrissage = 0 ∨ nbAvionsDecollage = 0
        piste à sens unique à t donné
    thm1: (theorem) nbAvionsDecollage + nbAvionsAtterrissage + nbAvionsTarmac ∈ ℕ
    DLFO: (theorem)
        (nbAvionsDecollage > 0) ∨
        (nbAvionsAtterrissage > 0) ∨
        (nbAvionsAtterrissage + nbAvionsTarmac < ntmax ∧ nbAvionsDecollage = 0) ∨
        (0 < nbAvionsTarmac ∧ nbAvionsAtterrissage = 0)
VARIANT
    nbAvionsTarmac + 2 * nbAvionsAtterrissage variant décrémenté par entrerPiste et sortir piste
EVENTS
Initialisation
    begin
        act1: nbAvionsDecollage := 0
        act2: nbAvionsAtterrissage := 0
        act3: nbAvionsTarmac := 0
    end
Event decoller ⟨ordinary⟩ ≡
refines decoller
when
    grd1: nbAvionsDecollage > 0
then
    act1: nbAvionsDecollage := nbAvionsDecollage - 1
end
Event atterrir ⟨ordinary⟩ ≡
refines atterrir
when
    grd1: nbAvionsAtterrissage + nbAvionsTarmac < ntmax
        la capa aéroport ne doit pas être dépassée
    grd2: nbAvionsDecollage = 0
        décollage interdit si atterrissage : piste à sens unique à t donné
then
    act1: nbAvionsAtterrissage := nbAvionsAtterrissage + 1
end
Event entrerPiste ⟨convergent⟩ ≡
when
    grd1: nbAvionsTarmac > 0
        tarmac non vide
    grd2: nbAvionsAtterrissage = 0
        pas d'atterrissage si entrée piste
then
    act1: nbAvionsDecollage := nbAvionsDecollage + 1
    act2: nbAvionsTarmac := nbAvionsTarmac - 1
end

```

FIGURE 9 –

4.6 Preuves

Comme pour la machine abstraite, le théorème DLF0 doit être prouvé interactivement via la perspective "Proving" pour prendre en compte toutes les hypothèses.

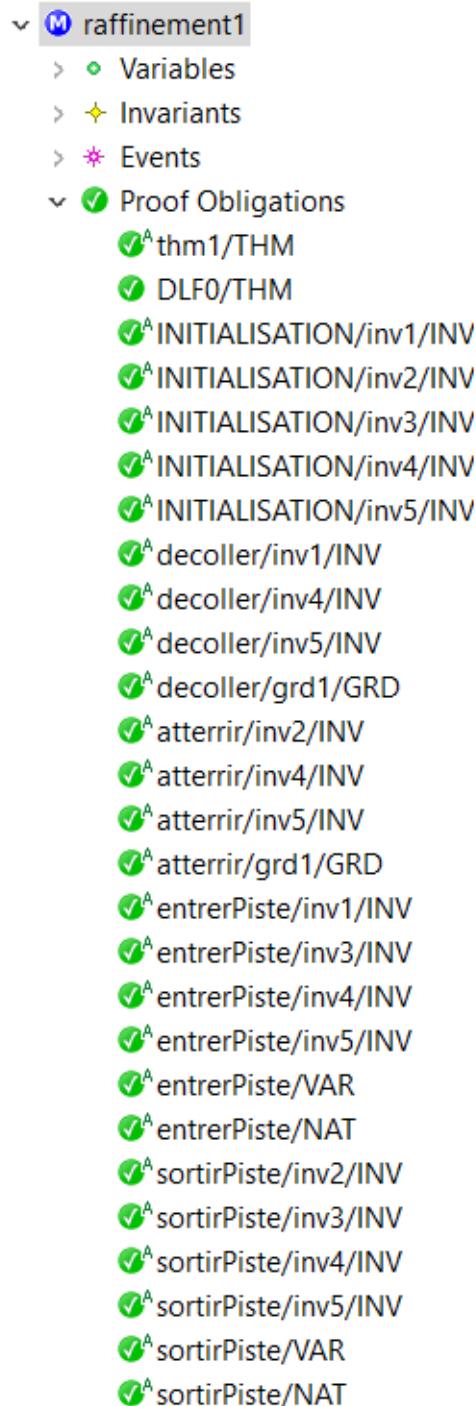


FIGURE 10 – Preuves du premier raffinement

5 Second raffinement

5.1 Analyse

On introduit dans ce dernier raffinement les autorisations d'accès à la piste et les identifiants uniques à chaque avions. Un pilote effectue une demande d'accès à la piste. Le système répond en accordant ou refusant le droit d'accès.

- Si l'accès est accordé, le pilote a 15 mn pour libérer la piste.
- S'il est refusé, le pilote doit attendre 5 mn avant de pouvoir renouveler sa demande.

5.2 Contexte

On étend le contexte0 précédent avec le contexte2

```
CONTEXT contexte2
EXTENDS contexte0
SETS
    ensAvions set of planes
    ensPistes ensemble de pistes
CONSTANTS
    clear cte clearance
    tarmac cte tarmac
AXIOMS
    axm1: clear ∈ ensAvions ↔ ensPistes
        relation binaire entre ensembles avions et pistes
    axm2: tarmac ∈ ensPistes
        le tarmac est élément de ensPistes
    axm3: (theorem) ensAvions × {tarmac} ⊆ clear
        tous les avions sont autorisés à se trouver sur le tarmac
END
```

FIGURE 11 – Contexte second raffinement

5.3 Glue

Le lien avec la machine raffinement1 est réalisé par le biais du nombre d'avions respectivement sur le tarmac, la piste décollage et la piste d'atterrissage. Une approche ensembliste, analogue à celle utilisée dans un système de contrôle d'accès à des bâtiments d'un site, conduit à introduire les ensembles d'avions correspondant. Les invariants de cardinalité, inv 16, 17 et 18 de la figure 12 assure la liaison avec la machine parente.

On crée une nouvelle machine nommée raffinement2 comme ci-après. Les commentaires ont été directement saisis dans l'interface rodin. Les livrables sont accessibles en ligne via l'URI https://github.com/kad15/eventb/tree/master/LIVRABLE_HELLER_BELDJILALI/workspace_rodin.

MACHINE raffinement2

machine pour prendre en compte les autorisations accès piste

REFINES raffinement1

SEES contexte2

VARIABLES

ensAvionsTarmac ensemble avions sur tarmac inclus dans ensAvions
 ensAvionsDecollage ensemble avion sur piste decollage inclus dans ensAvions
 ensAvionsAtterrissage ensemble avion sur piste atterrissage inclus dans ensAvions
 demandeAcces demande d'accès
 pisteOK ensemble de pistes dont l'accès a été autorisée à un avion
 pisteKO ensemble de pistes dont l'accès a été refusé à un avion

INVARIANTS

inv1: $\text{ensAvionsTarmac} \subseteq \text{ensAvions}$
 inclus dans ensAvions
inv2: $\text{ensAvionsDecollage} \subseteq \text{ensAvions}$
 inclus dans ensAvions
inv3: $\text{ensAvionsAtterrissage} \subseteq \text{ensAvions}$
 inclus dans ensAvions
inv4: $\text{ensAvionsTarmac} \cup \text{ensAvionsDecollage} \cup \text{ensAvionsAtterrissage} \subseteq \text{dom}(\text{clear})$
inv5: $\text{demandeAcces} \subseteq \text{clear}$
 une demande d'accès est une clearance
inv6: $\text{demandeAcces} \in \text{ensAvions} \leftrightarrow \text{ensPistes} \setminus \{\text{tarmac}\}$
 injection partielle entre les avions et les pistes hors tarmac
inv7: $\text{ensAvionsTarmac} \cap \text{ensAvionsDecollage} = \emptyset$
inv8: $\text{ensAvionsDecollage} \cap \text{ensAvionsAtterrissage} = \emptyset$
inv9: $\text{finite}(\text{ensAvionsTarmac})$
 ensembles finis pour faciliter la tâche du prouver
inv10: $\text{finite}(\text{ensAvionsDecollage})$
 ensembles finis
inv11: $\text{finite}(\text{ensAvionsAtterrissage})$
 ensembles finis
inv12: $\text{pisteOK} \subseteq \text{ensPistes}$
inv13: $\text{pisteKO} \subseteq \text{ensPistes}$
inv14: $\text{finite}(\text{pisteKO})$
 ensembles finis
inv15: $\text{finite}(\text{pisteOK})$
 ensembles finis
inv16: $\text{nbAvionsDecollage} = \text{card}(\text{ensAvionsDecollage})$
 invariant de glue avec raffinement1
inv17: $\text{nbAvionsTarmac} = \text{card}(\text{ensAvionsTarmac})$
 invariant de glue avec raffinement1
inv18: $\text{nbAvionsAtterrissage} = \text{card}(\text{ensAvionsAtterrissage})$
 invariant de glue avec raffinement1

EVENTS

Initialisation

```
begin
  act1:  $\text{ensAvionsTarmac} := \emptyset$ 
  act2:  $\text{ensAvionsDecollage} := \emptyset$ 
  act3:  $\text{ensAvionsAtterrissage} := \emptyset$ 
  act5:  $\text{demandeAcces} := \emptyset$ 
  act6:  $\text{pisteOK} := \emptyset$ 
  act7:  $\text{pisteKO} := \emptyset$ 
end
```

Event decoller $\langle \text{ordinary} \rangle \hat{=}$

FIGURE 12 – Machine : second raffinement

```

refines decoller
  any
    a un avion
  where
    grd1: a ∈ ensAvionsDecollage
  then
    act1: ensAvionsDecollage := ensAvionsDecollage \ {a}
  end
Event atterrir ⟨ordinary⟩ ≡
refines atterrir
  any
    a
  where
    grd1: a ∈ ensAvions
    grd2: card(ensAvionsAtterrissage) + card(ensAvionsTarmac) < ntmax
    grd3: ensAvionsDecollage = ∅
  then
    act1: ensAvionsAtterrissage := ensAvionsAtterrissage ∪ {a}
  end
Event entrerPiste ⟨ordinary⟩ ≡
refines entrerPiste
  any
    a
    p
  where
    grd1: a ∈ ensAvionsTarmac
    grd2: p ∈ pisteOK
    grd3: ensAvionsAtterrissage = ∅
    grd4: a ↦ p ∈ demandeAcces
  then
    act1: ensAvionsTarmac := ensAvionsTarmac \ {a}
    act2: ensAvionsDecollage := ensAvionsDecollage ∪ {a}
  end
Event sortirPiste ⟨ordinary⟩ ≡
refines sortirPiste
  any
    a
    p
  where
    grd1: a ∈ ensAvionsAtterrissage
    grd2: p ∈ pisteOK
  then
    act1: ensAvionsTarmac := ensAvionsTarmac ∪ {a}
    act2: ensAvionsAtterrissage := ensAvionsAtterrissage \ {a}
  end
Event accepterDemande ⟨ordinary⟩ ≡
any
  a un avion
  p une piste
where
  grd1: a ∈ ensAvions
    a est un avion
  grd2: p ∈ ensPistes \ {tarmac}
    la piste est une piste mais pas un tarmac
  grd3: a ∉ dom(demandeAcces)
    a ne fait pas partie du domaine
  grd4: p ∉ ran(demandeAcces)
    p ne fait pas partie de l'image de la relation a vers p

```

16.12.2017 20:17

Page 2 of 3

FIGURE 13 – Machine : second raffinement (suite)

```

grd5: a ↦ p ∈ clear
grd6: p ∉ pisteOK ∪ pisteKO
    l'avion n'est ni en attente ni déjà autorisé à accéder piste p
then
    act1: demandeAcces(a) := p
end
Event refuserDemande ⟨ordinary⟩ ≡
any
    a un avion
    p une piste
where
    grd1: a ∈ ensAvions
    grd2: p ∈ ensPistes \ {tarmac}
    grd3: p ∉ pisteOK ∪ pisteKO
then
    act1: pisteKO := pisteKO \ {p}
end
Event libererPisteOK ⟨ordinary⟩ ≡
any
    p
where
    grd1: p ∈ pisteOK
then
    act1: pisteOK := pisteOK \ {p}
    act2: demandeAcces := demandeAcces ▷ {p}
end
Event libererPisteKO ⟨ordinary⟩ ≡
any
    p
where
    grd1: p ∈ pisteKO
then
    act1: pisteKO := pisteKO \ {p}
end
END

```

FIGURE 14 – Machine : second raffinement (suite)

5.4 Preuves

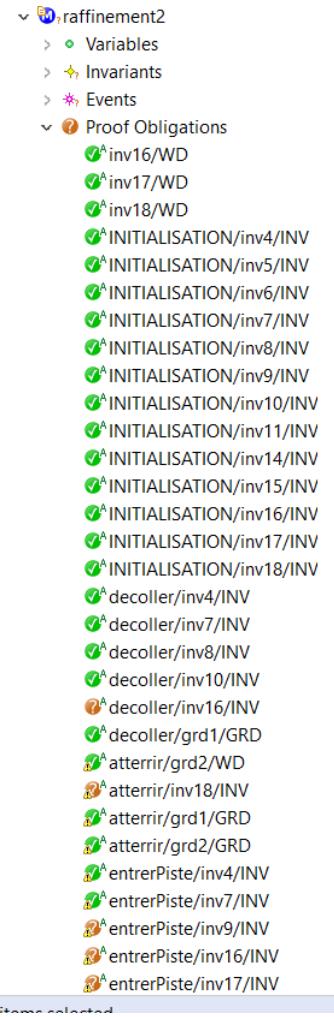


FIGURE 15 – Résultat des preuves des propriétés générées par l'outil rodin - atelier B

On constate que le système n'est pas entièrement validé. En outre, l'outil rodin indique une syntaxe incorrecte pour l'instruction $ensAvionsTarmac := ensAvionsTarmac \cup \{a\}$ alors qu'elle semble correcte.

Conclusion

Ce BE aura été l'occasion de se familiariser avec la méthode formelle event-b qui par modélisation d'un système sous la forme d'une machine à états-transitions propose une approche amont de la conception et vérification des systèmes, reposant sur un formalisme mathématique rigoureux, par opposition à une conception avalé contrainte de s'appuyer sur des batteries de tests a posteriori pour "démontrer" empiriquement la correction du système.

La gestion de la complexité dans cette méthode repose sur une raffinage successif du modèle par héritage et extension des attributs et des propriétés entre machines successives validées progressivement sur la base de règles d'inférences. Elle permet également d'approcher l'exhaustivité et l'exactitude des exigences en les complétant.

Table des figures

1	Environnement du système	3
2	Modèle abstrait : seuls les échanges avec l'extérieur sont pris en compte.	5
3	Etat statique du système avec deux axiomes redondants	5
4	Etat dynamique du système	6
5	Ajout de 2 gardes et d'un théorème pour éviter les blocages (invariant Deadlock free : DLF0)	7
6	Environnement du système : version 2	8
7	Premier raffinement du système : piste et tarmac sont différenciés	9
8	Contexte premier raffinement	9
9	11
10	Preuves du premier raffinement	12
11	Contexte second raffinement	13
12	Machine : second raffinement	14
13	Machine : second raffinement (suite)	15
14	Machine : second raffinement (suite)	16
15	Résultat des preuves des propriétés générées par l'outil rodin - atelier B	17

Liste des tableaux

1	Tableau des exigences V1	4
2	Tableau des exigences V2	7
3	Tableau des exigences V3	8

