



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 raffinement de l'état du système clos	9
4.3 Contexte	9
4.4 Glue	10
4.5 Convergence des nouveaux events	10
4.6 Machine	10
4.7 Preuve	10
5 Second raffinement	13
Conclusion	13

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

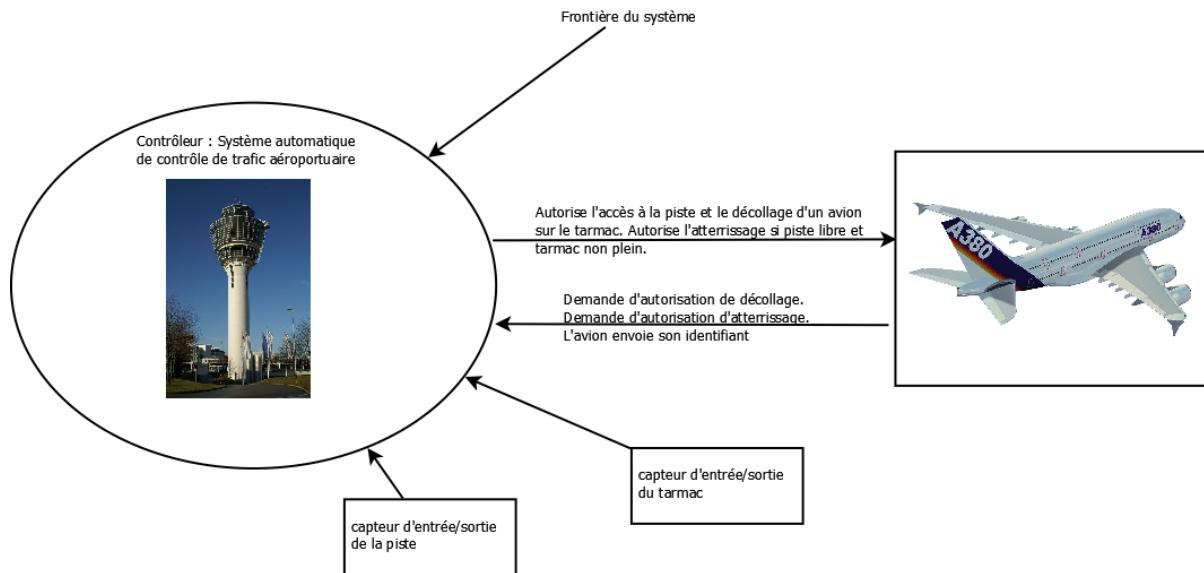


FIGURE 1 – Environnement du système

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.

On suppose qu'il ne peut y avoir qu'un seul avion sur la piste ; pas de possibilité de mettre deux avions au décollage en même temps.

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 :

- extout lorsqu'un avion sort de l'espace extérieur pour rentrer dans l'espace aéroportuaire,
- extin lorsqu'un avion entre dans l'espace extérieur.

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.

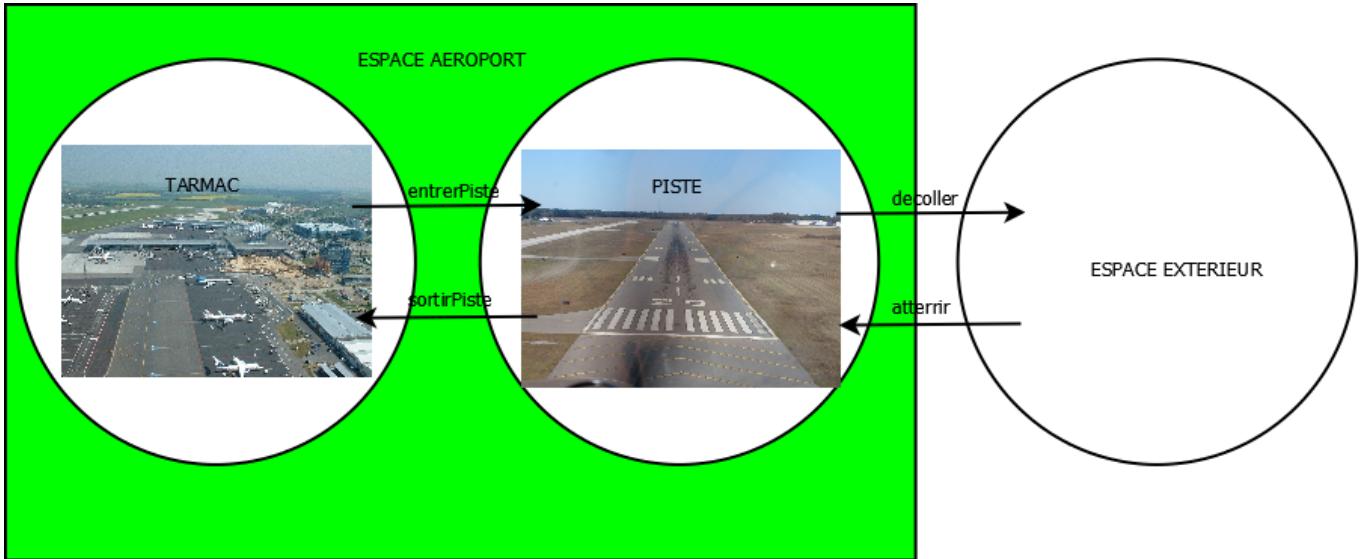


FIGURE 2 – Modèle abstrait : seuls les échanges avec l'extérieur sont pris en compte.

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

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 est 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 débuguer 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 théorème 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.

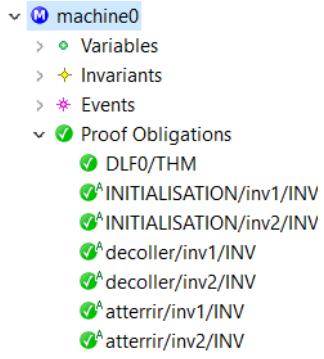


FIGURE 5 – Ajout de 2 gardes et d'un théorème pour éviter les blocages (invariant Deadlock free : DLF0)

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

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 :

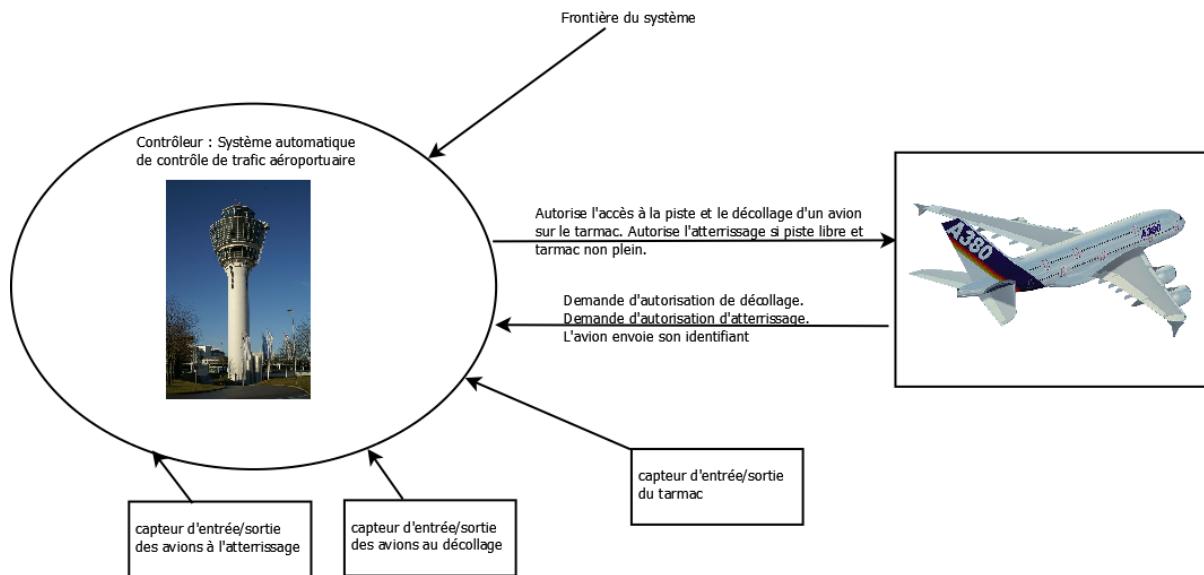


FIGURE 6 – Environnement du système : version 2

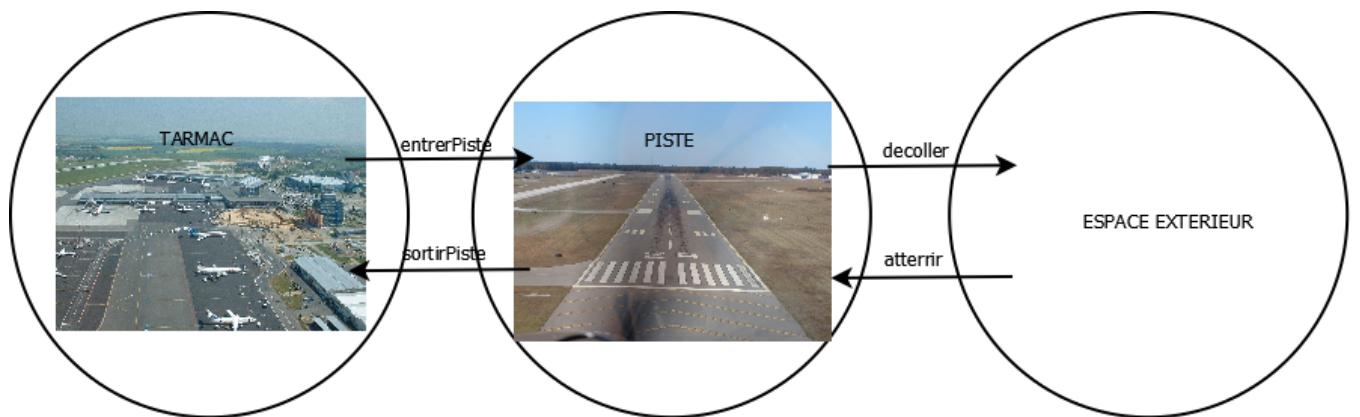


FIGURE 7 – Premier raffinement du système : piste et tarmac sont différenciés

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

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 raffinement de l'état du système clos

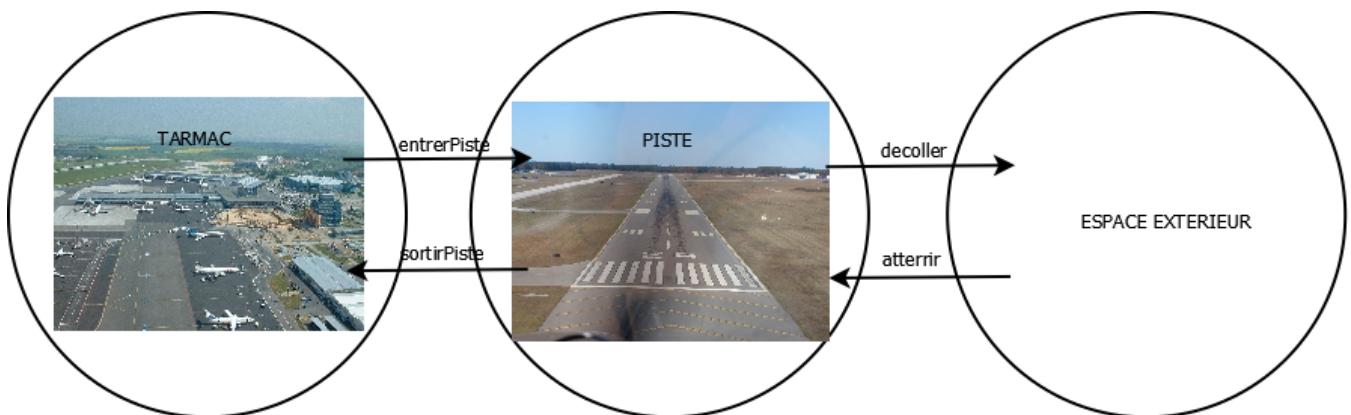


FIGURE 8 – Premier raffinement du système : la piste et le tarmac sont différenciés

4.3 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 9 – Contexte premier raffinement

4.4 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 à la variable abstraite nt.

4.5 Convergence des nouveaux events

On définit un variant entier naturel, en l'occurrence, $\text{nbAvionsTarmac} + 2 * \text{nbAvionsAtterrissage}$ 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.

4.6 Machine

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

4.7 Preuve

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

- ✓ M raffinement1
 - > ● Variables
 - > ✦ Invariants
 - > ✯ Events
 - > ✓ Proof Obligations
 - ✓^Athm1/THM
 - ✓ DLF0/THM
 - ✓^AINITIALISATION/inv1/INV
 - ✓^AINITIALISATION/inv2/INV
 - ✓^AINITIALISATION/inv3/INV
 - ✓^AINITIALISATION/inv4/INV
 - ✓^AINITIALISATION/inv5/INV
 - ✓^Adecolle/inv1/INV
 - ✓^Adecolle/inv4/INV
 - ✓^Adecolle/inv5/INV
 - ✓^Adecolle/grd1/GRD
 - ✓^Aatterrir/inv2/INV
 - ✓^Aatterrir/inv4/INV
 - ✓^Aatterrir/inv5/INV
 - ✓^Aatterrir/grd1/GRD
 - ✓^AentrerPiste/inv1/INV
 - ✓^AentrerPiste/inv3/INV
 - ✓^AentrerPiste/inv4/INV
 - ✓^AentrerPiste/inv5/INV
 - ✓^AentrerPiste/VAR
 - ✓^AentrerPiste/NAT
 - ✓^AsortirPiste/inv2/INV
 - ✓^AsortirPiste/inv3/INV
 - ✓^AsortirPiste/inv4/INV
 - ✓^AsortirPiste/inv5/INV
 - ✓^AsortirPiste/VAR
 - ✓^AsortirPiste/NAT

FIGURE 10 – Preuves du premier raffinement

```

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 11 –

5 Second raffinement

Conclusion

