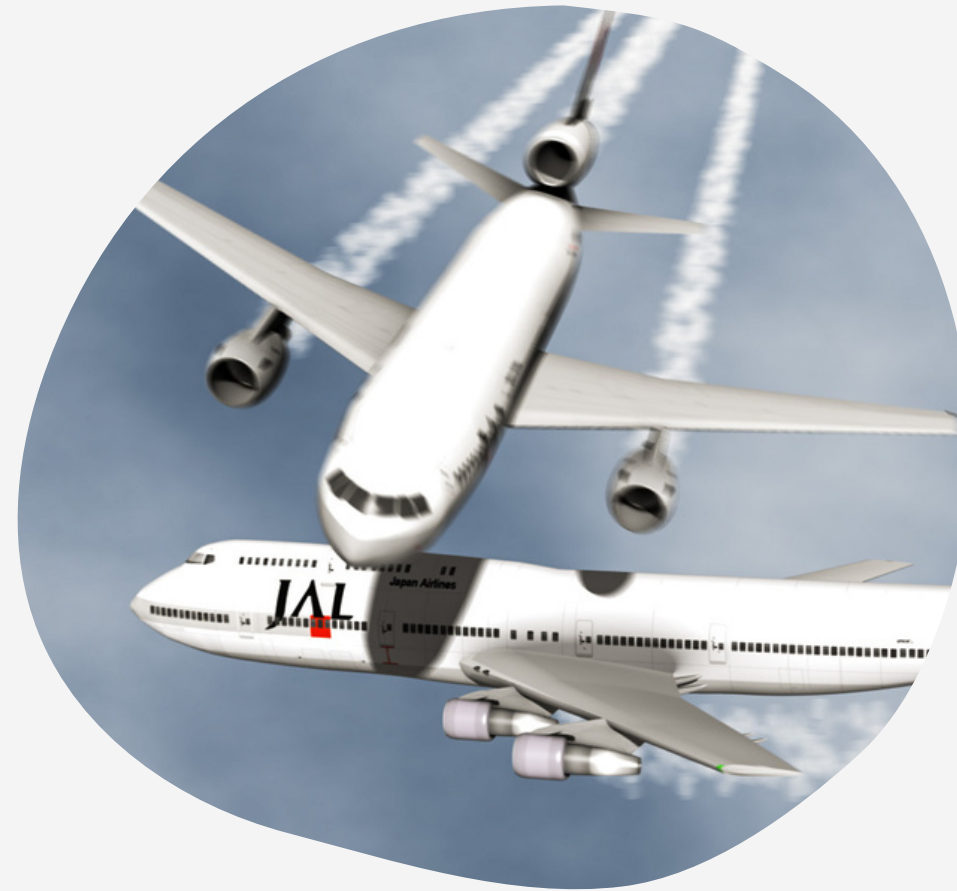




Soutenance EI - Groupe 1

Time Constraints of TCAS II

Maud Valentin, Juliette Kalflèche,
Elisa Faure, Mélanie Augier, Tom Bray,
Florentin Labelle, Marc-Antoine
Godde



Sommaire

- Présentation
- Premier modèle
- Problématiques rencontrées
- Modèle final
- Zenoness
- Méthode de travail
- Conclusion

Présentation



TCAS II



- En 2014, 102 465 vol par jour
- Croissance continue depuis

-
- En 1950, multiples collisions tragiques
 - Développement du TCAS en réponse

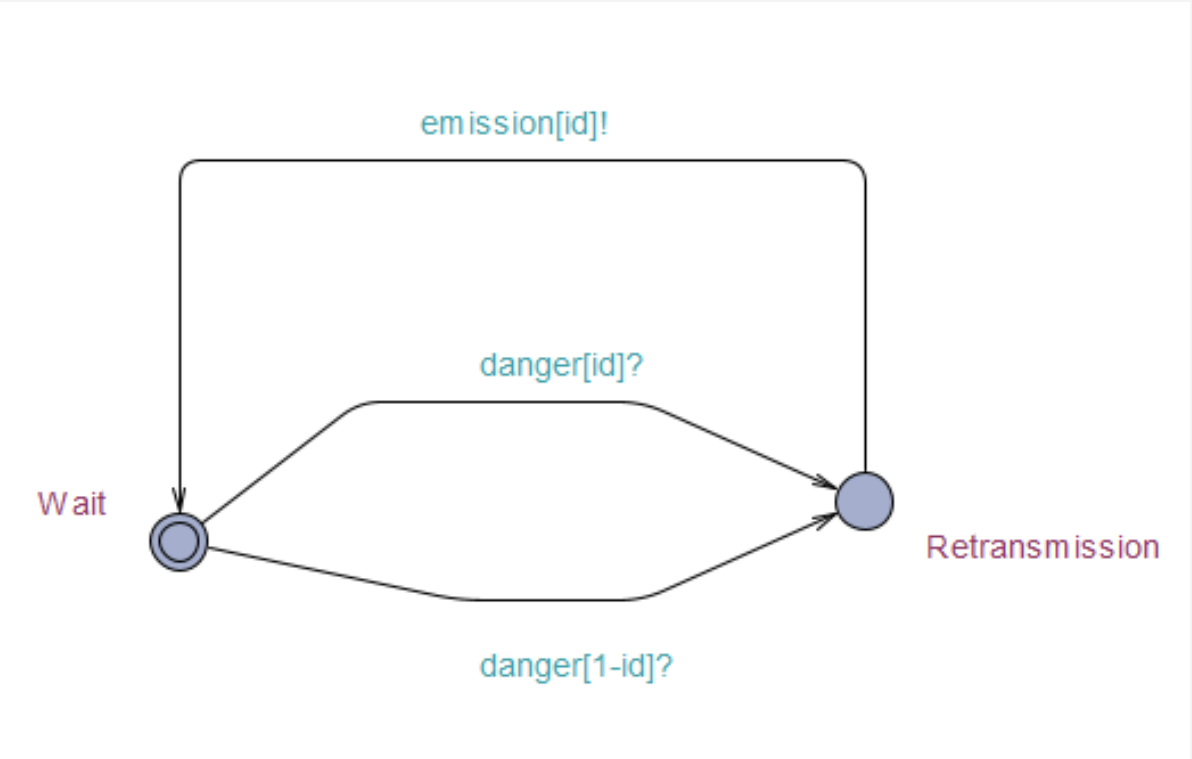
-
- Etude et compréhension du TCAS
 - Développement d'un modèle incluant les différentes propriétés d'un TCAS

Premier modèle



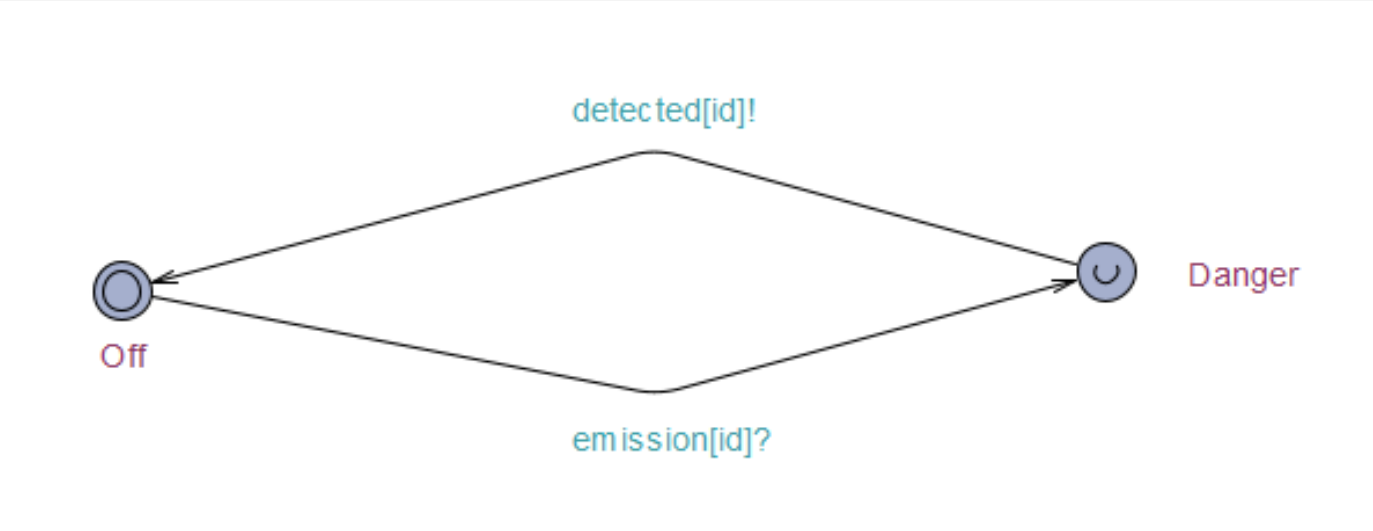
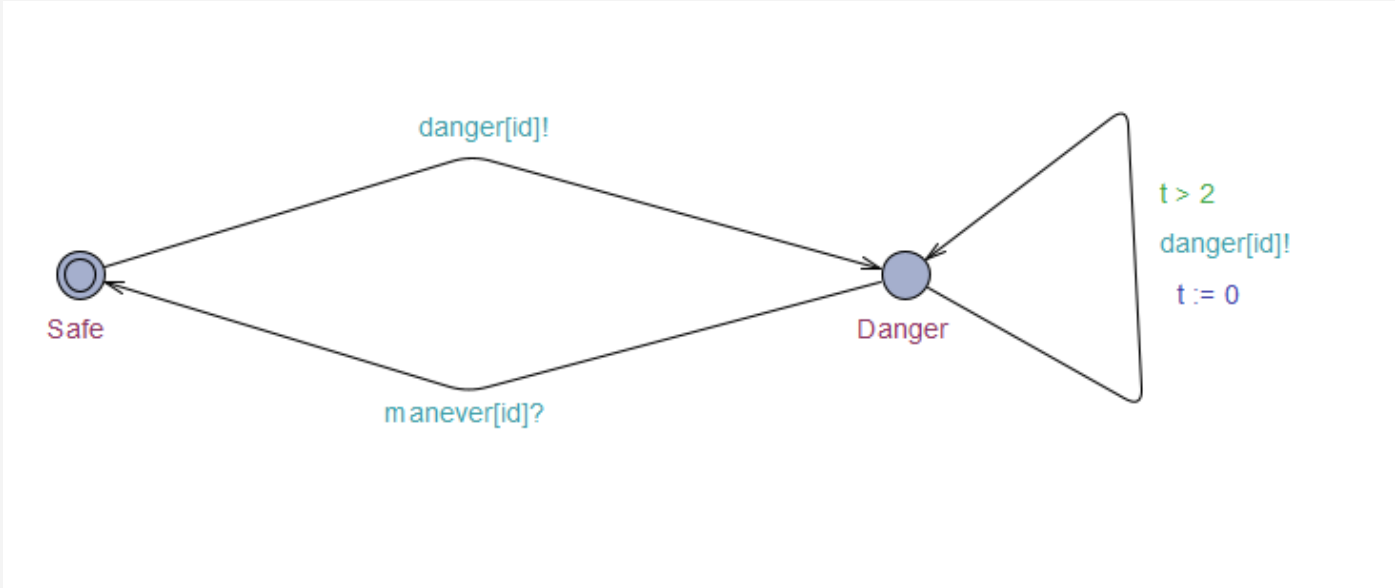
Automates

Avion



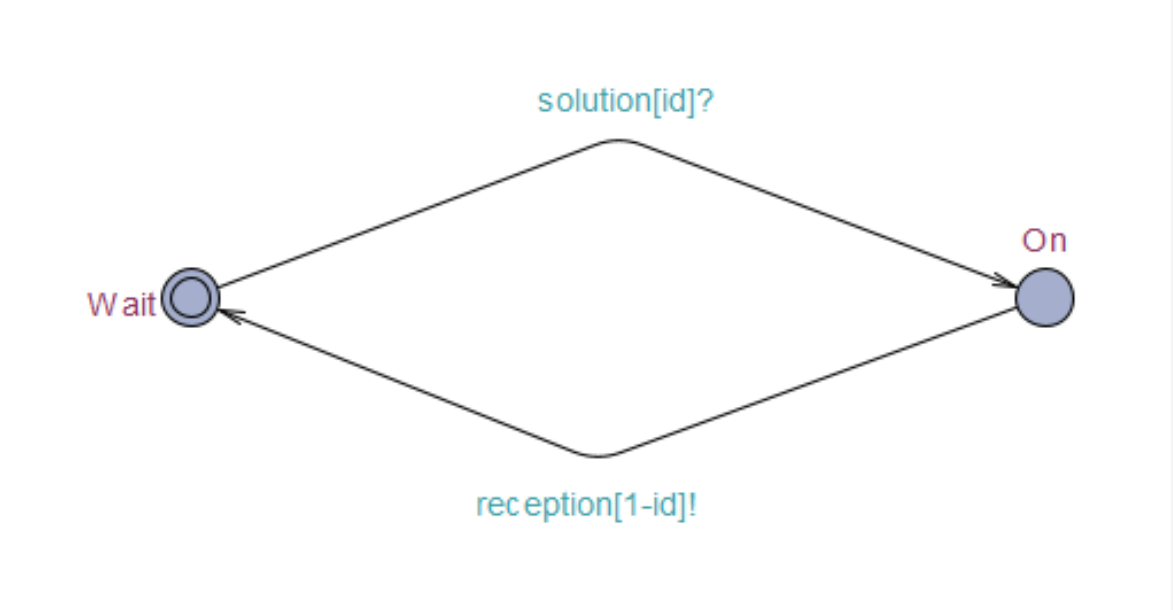
Détecteur

Capteur



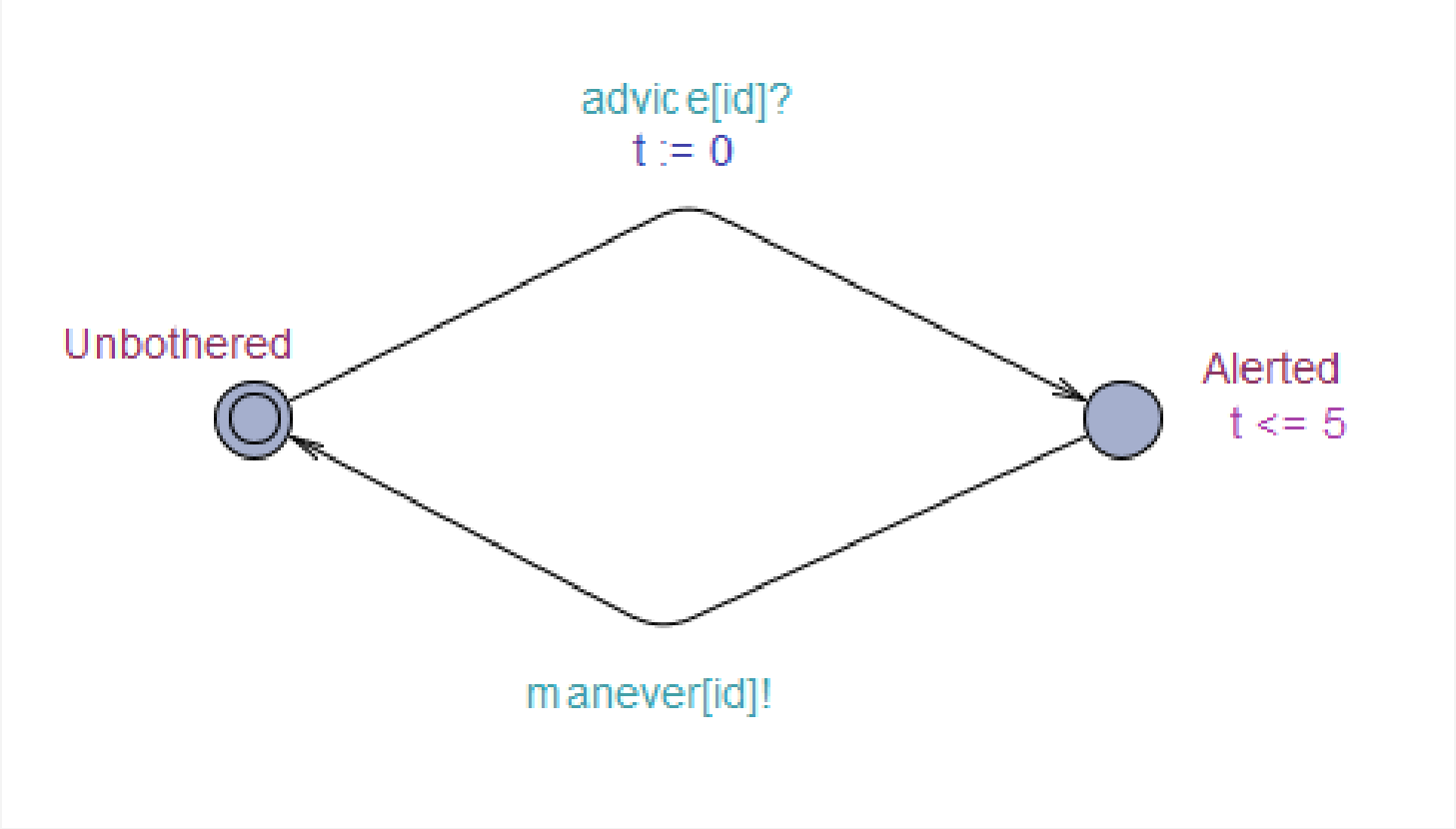
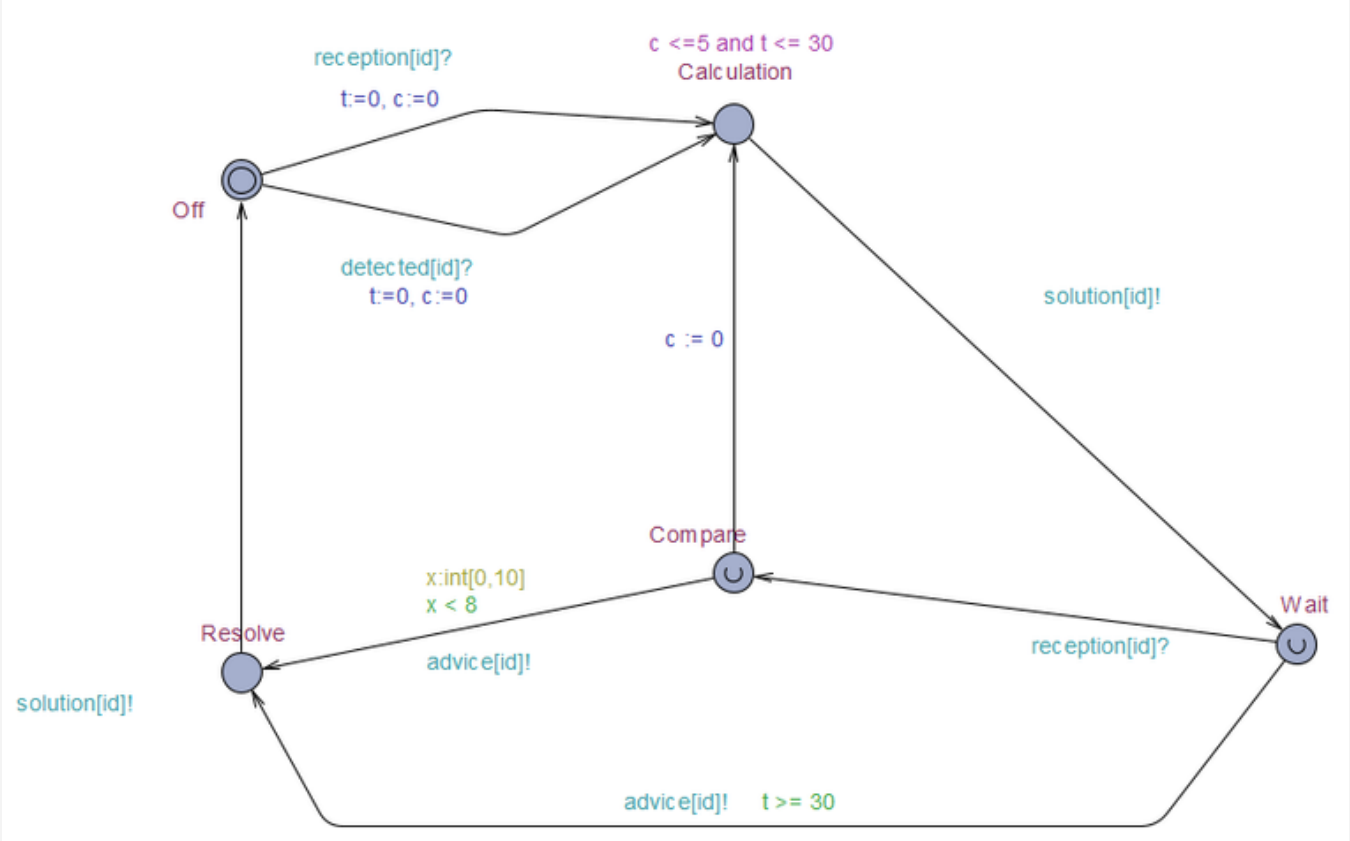
Automates

Résolution



Pilote

Communication



Problématiques rencontrées



Problèmes



Deadlock

Le modèle possédait de nombreuses synchronisations, de nombreux états, ce qui menait à des deadlocks. Débug amplifiait le problème.

Système redondant

Redondance entre aircraft et sensor, que nous ne jugions pas nécessaires.

Non-synchronisation du danger

Difficulté à gérer l'alerte du danger sur les deux avions en même temps, menait également à plusieurs deadlocks.

Changements

Le channel danger en broadcast

Pour synchroniser le danger chez les deux avions

Simplification du système

Suppression des sensor et un seul ciel plutôt que deux avions

Ajout d'états Safe

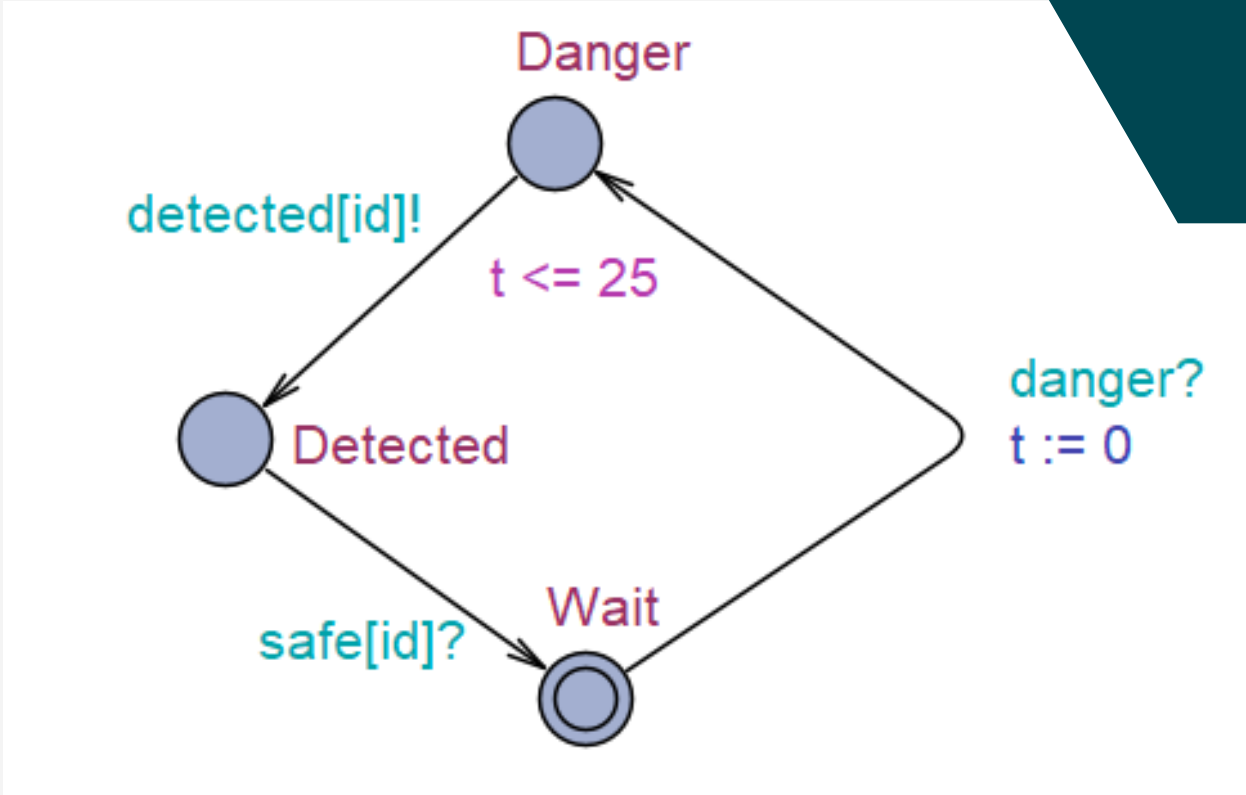
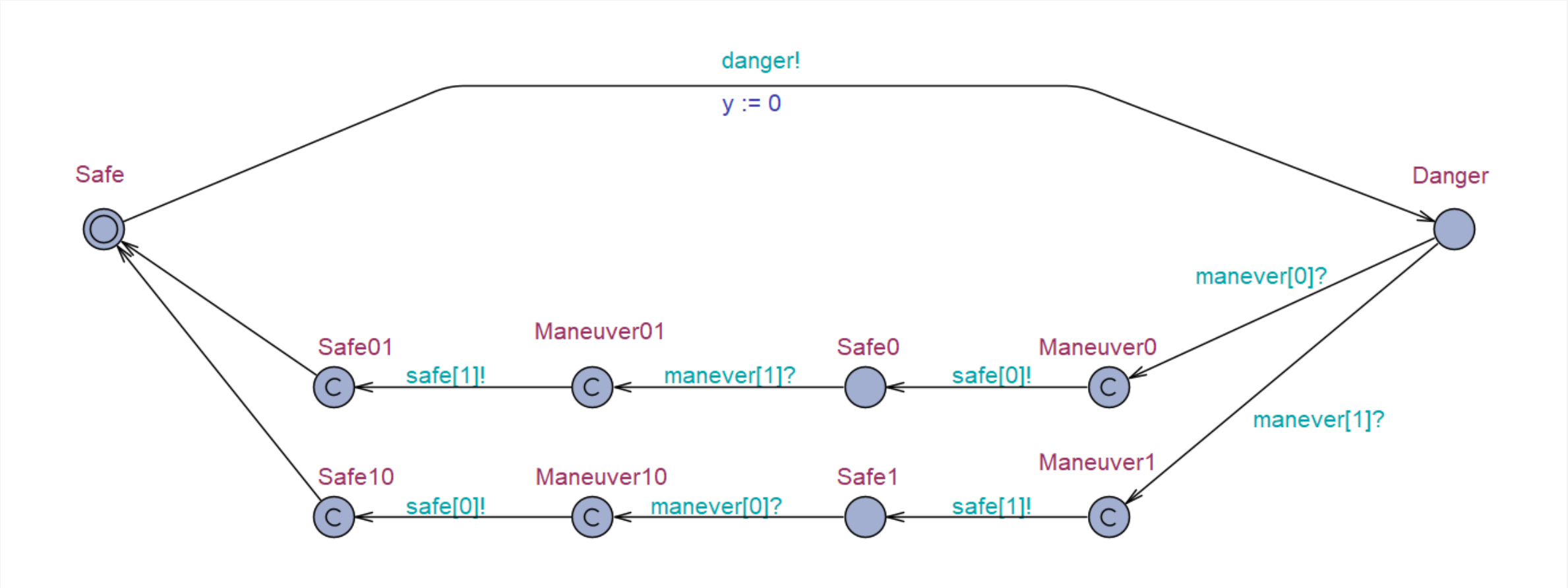
Pour pouvoir établir la fin du danger pour chaque avion



Modèle final



Automates

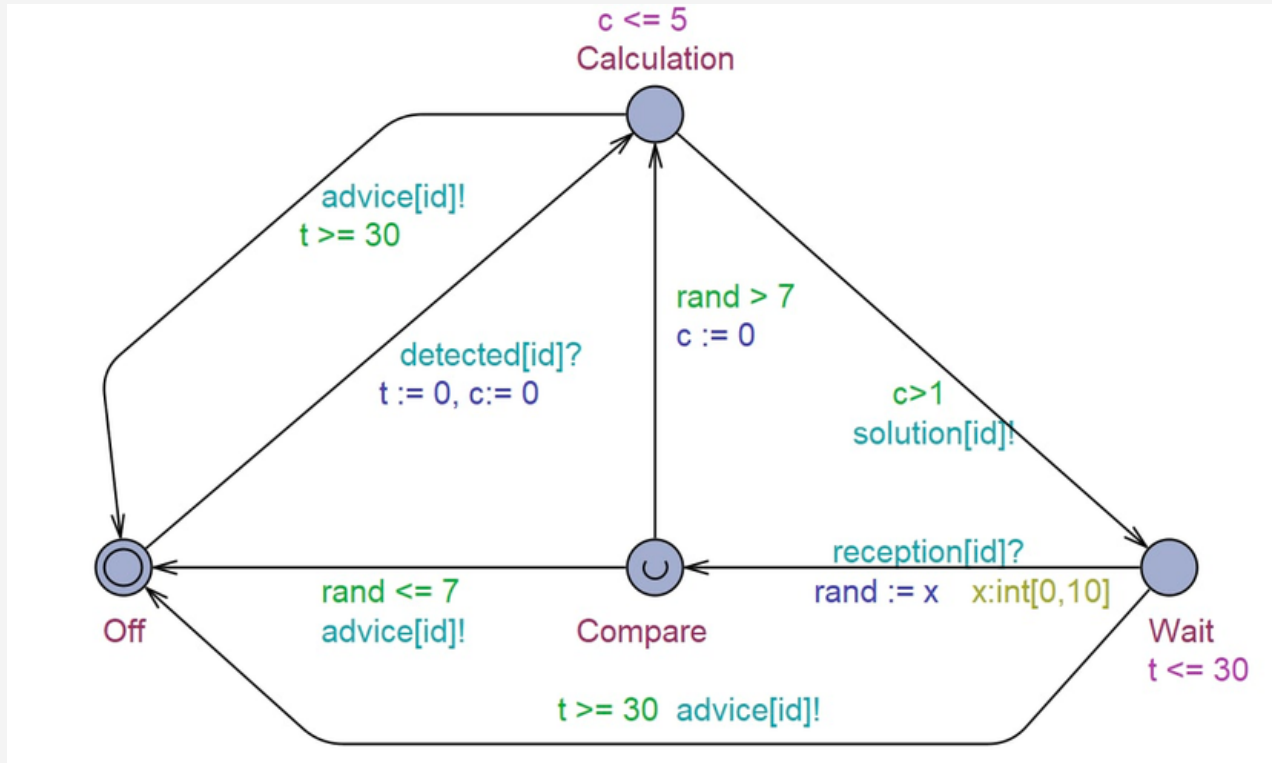


Avions

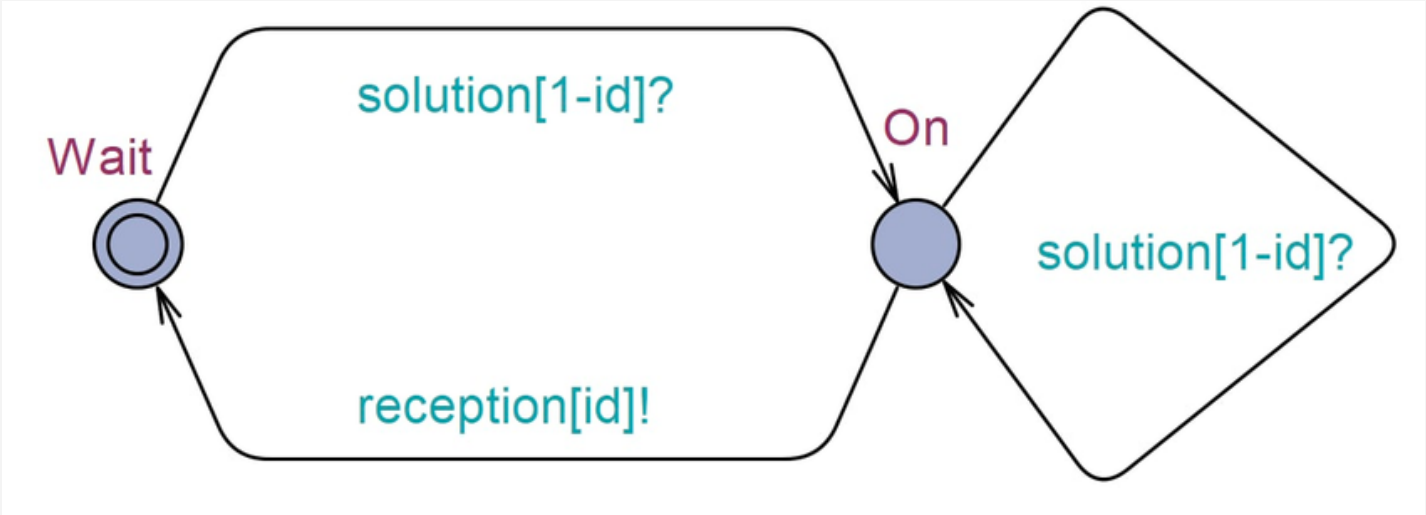
Détecteur

Automates

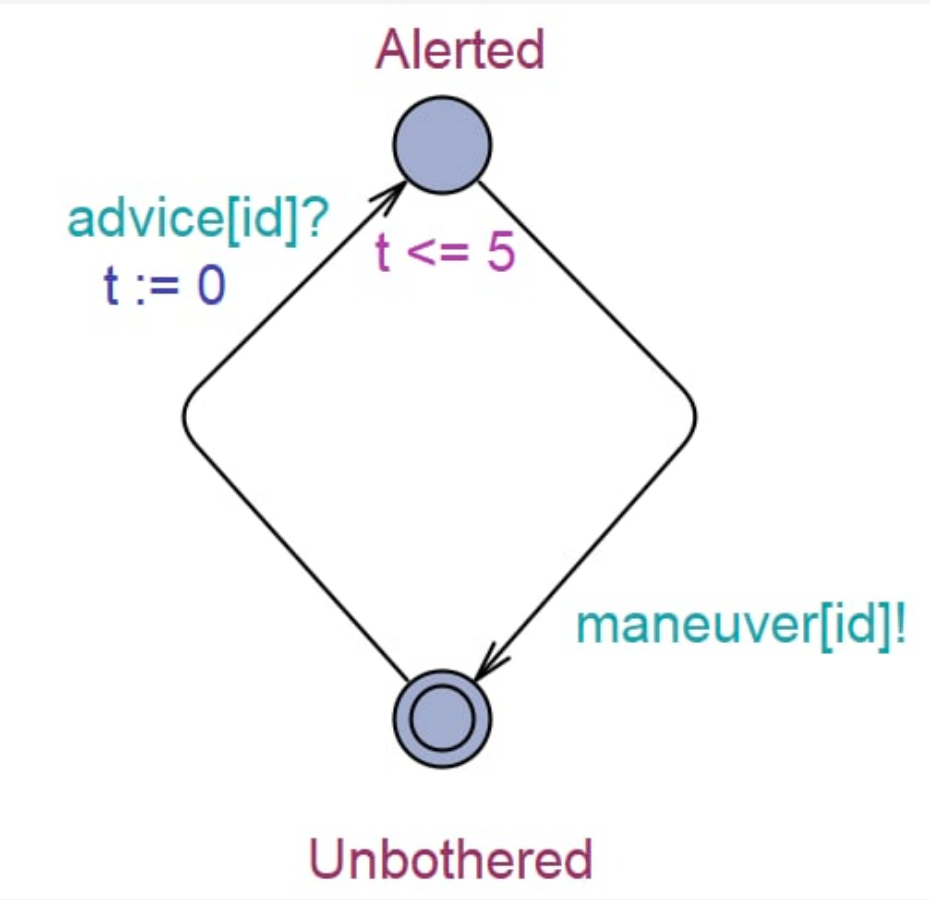
Résolution



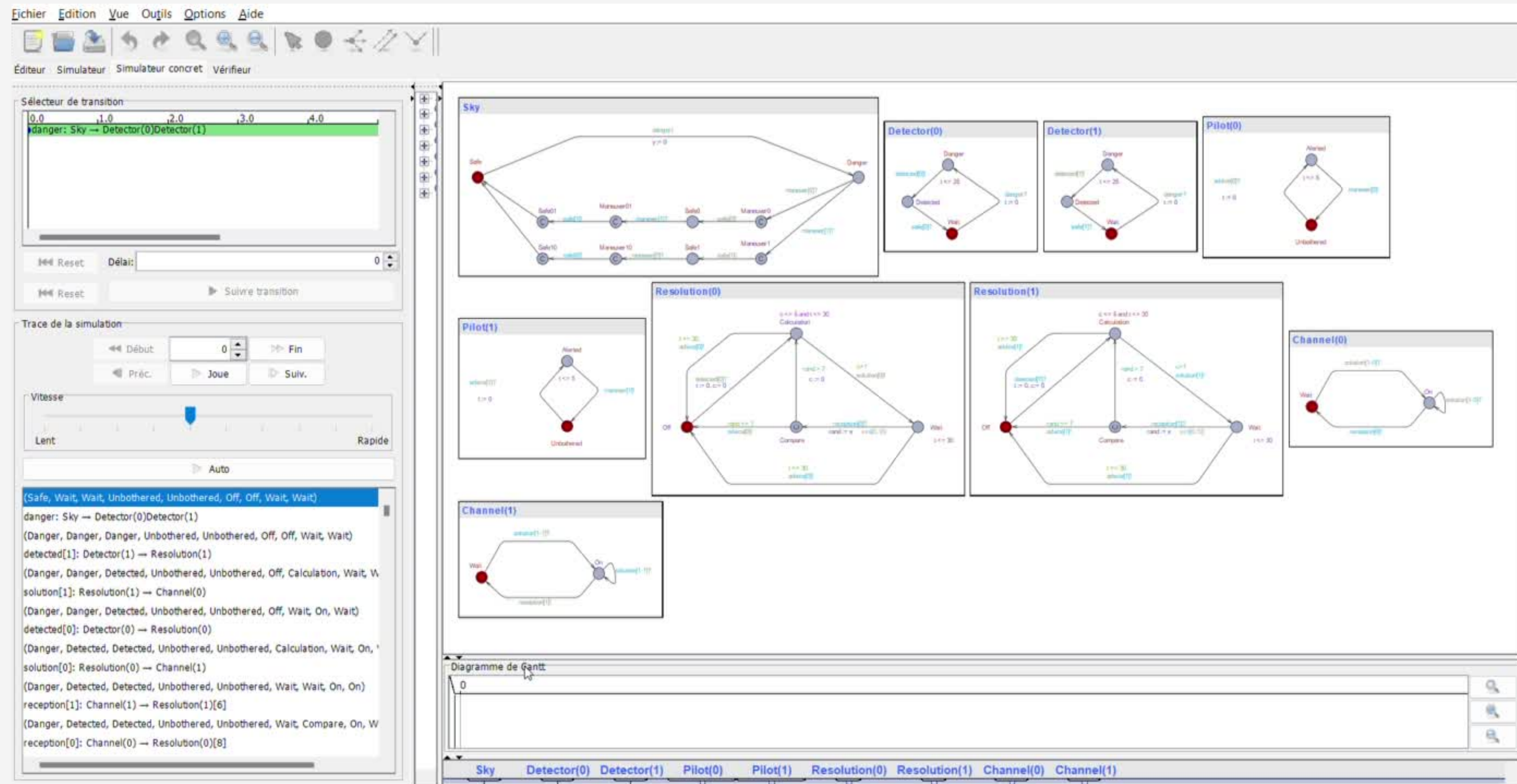
Communication



Pilote



Démonstration



Zenoness checking



Découpage du travail



Codage en python de 4 fonctions :

- Parser le fichier .xml en listes et dictionnaires
- Trouver pour chaque composants toutes les boucles locales
- Regrouper les boucles locales globalement en déterminant si elles sont les mêmes synchronised actions
- Vérifier la propriété de strongly non-zenoness

Parsing

Pour chaque template, on isole:

- La Liste des états
 - nom
 - Invariant
- Une "matrice" de transition
 - Synchronisation
 - Affectation
 - Garde
- Une liste d'horloges

 **parsed.py**  365 bytes

```
1 {
2   "templates": [
3     {
4       "name": "Train",
5       "states": {
6         "id0": { "name": "Safe", "id": "id0", "invariant": None },
7       },
8       "matrix": {
9         "id1": {
10          "id0": {
11            "synchronization": "leave[id]!",
12            "assignment": None,
13            "guard": "x>=3",
14          }
15        },
16      },
17    },
18  ],
19  "clocks": ["x"]
20 }
```

Détection de boucles

- Pour chaque automate :
 - Etat de départ et horloges réinitialisées
 - Parcours en profondeur des chemins possibles
 - Test des gardes des transitions
 - Test des invariants de l'état suivant
 - Mise à jour des horloges
 - Vérification si boucle terminée
- Puis concaténation des listes de boucles de chaque automate



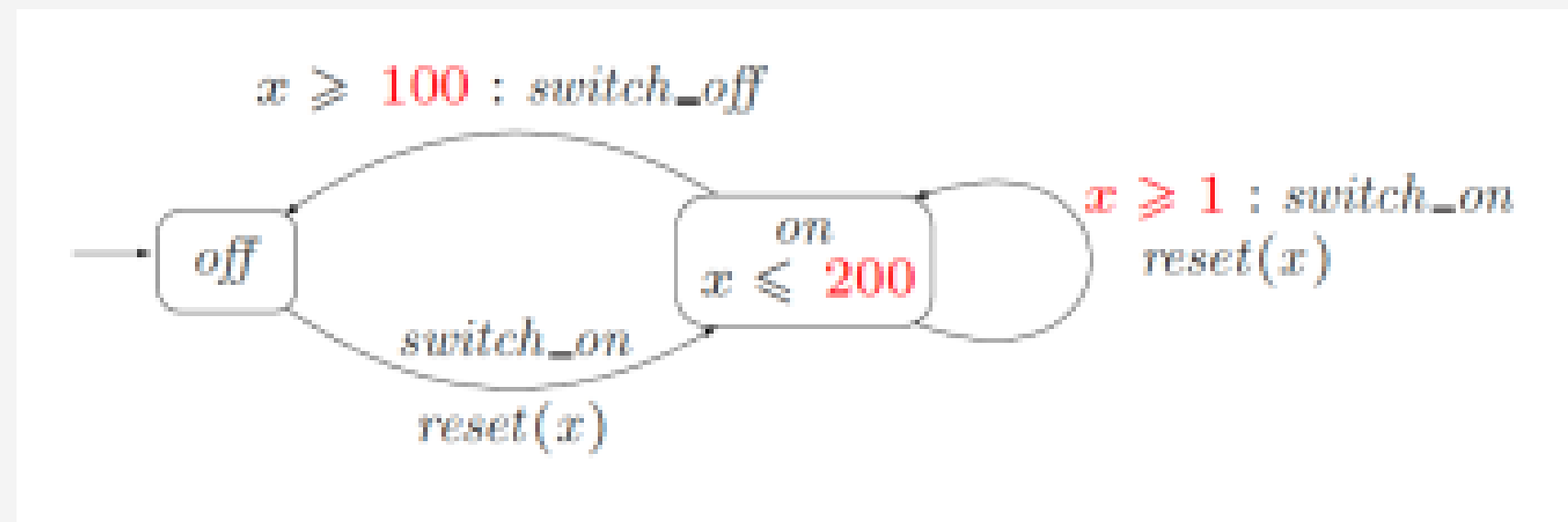
Regroupement de boucle

4 étapes :

1. Création d'une nouvelle liste
2. Itération sur tout les boucles
3. Deuxième itération pour comparer 2 à 2
4. Comparaison des "synchronized actions"



Vérification de la strong non-zenoness



Sur une boucle, pour chaque horloge :

- celle-ci doit être remise à zéro durant l'exécution
- Il existe une garde concernant cette horloge qui garantie un temps minimum d'exécution

Méthode de travail



Découpage du travail

Task I :

Schématisation ensemble afin de nous approprier le problème.

Task II & III :

Rédaction du rapport

Modélisation sur UPPAAL

Extension (Zeno) :

Chacun essaie de coder une fonction différente (sur les 4 nécessaires)

Répartition du travail



Elisa FAURE
Modélisation sur UPPAAL &
programme de non zenosness



Florentin LABELLE
Modélisation sur UPPAAL & programme de
non zenosness



Tom BRAY
Modélisation, programme de non zenoness



Mélanie AUGIER
Modélisation sur UPPAAL



Maud VALENTIN
Modélisaton, écriture du rapport et présentation



Juliette KALFLECHE
Travail sur le programme de non zenoness,
écriture du rapport



Marc-Antoine GODDE
Ecriture du rapport et présentation

UPPAAL



Prise en main facile du logiciel.

Pratique pour modéliser de systèmes simples, mais difficulté à déboguer pour des systèmes complexes.

Une option collaborative serait la bienvenue.

Difficulté à installer le logiciel sur certains ordinateurs.

Principales difficultés

- Mauvaise compréhension des objectifs de l'EI (volonté de modéliser exactement la réalité)

-> *Résolution : les entretiens*

- Un premier modèle **trop compliqué à déboguer** (énormément de deadlock, pas évident à corriger), on se perdait dans les états urgents, committed et les channels urgentes

-> *Résolution : recommencer avec un modèle plus simple, en prenant des libertés par rapports aux consignes initiales*

- Zenoness : difficulté de la tâche, manque de temps et d'expérience



Conclusion



Conclusion

- Découverte de UPPAAL et de la vérification formelle
- Développement concret d'un système critique à logiciel prépondérant
- Travail avec les problématiques concrètes d'un ingénieur
- Prise en compte de la responsabilité de l'ingénieur dans la sécurité des technologies développées

