12.09.2023

Concours ROBAFIS 2023

CdC : Expression du besoin client

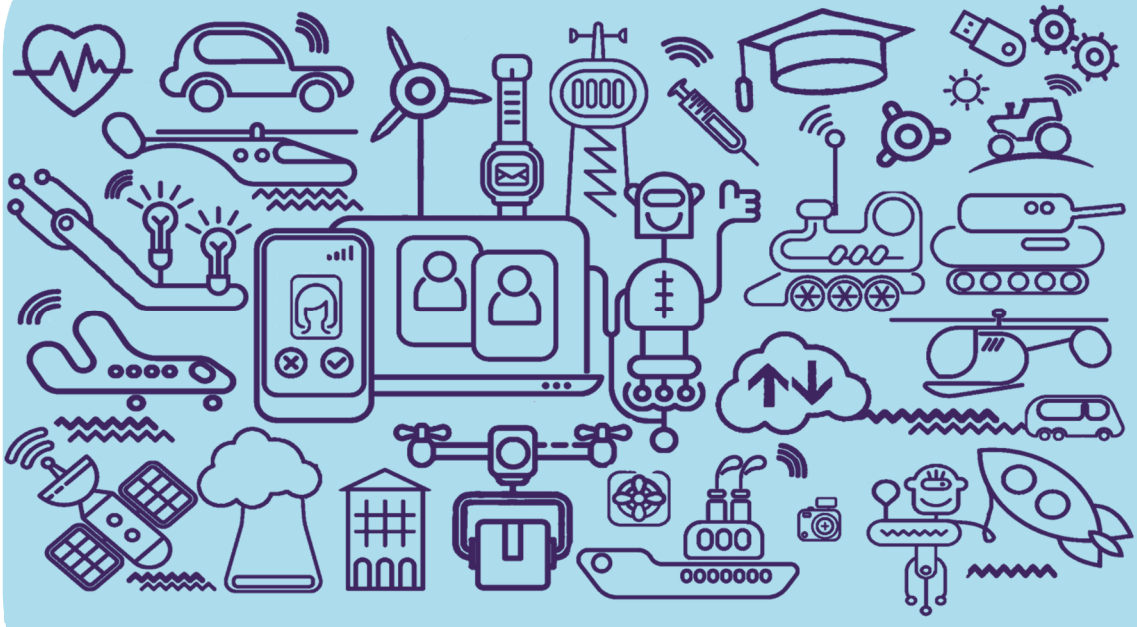


TABLE DES MATIERES

Edition	Nature de l'évolution	Évolution	Date
V0.1	Création		20230302
V0.2	Mise à jour suite à relecture interne	Cf barres de rev	20230627
V0.3	Mise à jour + nouvel exemple	Cf barres de rev (sauf pour l'exemple)	20230709
V0.4	Mise à jour		20230726
V1.0	Version officielle		20230912

Le présent document exprime le besoin client pour le développement, la réalisation et l'évaluation opérationnelle du système ***TuringAFIS***.

Les livrables attendus sont un dossier de développement de niveau système (optionnel), un dossier de développement complet (obligatoire) en version numérique, ainsi qu'un prototype de la solution proposée sur la base de cette étude. Ce prototype est mis en œuvre pour une évaluation opérationnelle lors d'une campagne d'expérimentations comparatives entre plusieurs solutions concurrentes. Ce prototype est capable de remplir la mission spécifiée et de satisfaire les expérimentations définies dans le présent Cahier des Charges (CdC) et dans le règlement.

Cette évaluation et ces expérimentations sont réalisées dans un environnement d'évolution qui est un terrain de jeu.

Afin de permettre une évaluation comparative de toutes les solutions proposées, l'AFIS fournit les composants et sous-systèmes (capteurs, actionneur, cartes de contrôle-commande, éléments de la structure, …) permettant de réaliser ce prototype, ainsi que les éléments du terrain de jeu.

1. Contexte opérationnel

1.1. Mission

Généralités

TuringAFIS est un système qui implémente le fonctionnement d'une machine de Turing universelle. Pour cela, il doit être capable d'effectuer uniquement les opérations suivantes :

- **Lecture** : Lire le contenu de la case mémoire courante (qui peut soit être vide, soit contenir la valeur 0, soit contenir la valeur 1)
- **Ecriture** : En fonction de la valeur lue de la case mémoire courante, modifier le contenu de la case mémoire courante
- **Déplacement** : En fonction de la valeur lue de la case mémoire courante, se déplacer vers la case mémoire située juste à gauche, ou vers la case mémoire située juste à droite. La case mémoire courante devient la nouvelle case mémoire, atteinte suite à ce déplacement.

La case mémoire courante est la case de la mémoire qui peut être lue ou écrite par ***TuringAFIS*** sans aucune opération de déplacement préalable.

Programme

TuringAFIS doit pouvoir être programmé, c'est-à-dire qu'un programmeur humain doit pouvoir lui indiquer à l'avance les opérations de lecture, d'écriture et de déplacement

à effectuer, en fonction du contenu des cases mémoire. On appellera cette description le « programme ».

Concrètement, un programme contient :

- Un ensemble **d'états** identifiés de manière unique, dont un état initial (identifié par « init ») et un état final (identifié par « fin »). Un état représente une étape entre 2 pas d'exécution du programme.
- Pour chaque état, et pour chaque valeur possible de la case mémoire courante, le **pas d'exécution** qui définit les actions à effectuer :
  - Une opération d'écriture à effectuer : définition de la nouvelle valeur avec laquelle la case mémoire courante doit être modifiée (vide, ou la valeur 0, ou la valeur 1),
  - Une opération de déplacement à effectuer : vers la case située juste à gauche ou juste à droite de la case mémoire courante,
  - L'état suivant

Un programme est représenté par un tableau « états – pas d'exécution » qui décrit, pour chaque état et en fonction de la valeur de la case mémoire courante lue, le pas d'exécution à effectuer (c'est à dire l'opération d'écriture à effectuer sur la case mémoire courante, l'opération de déplacement à effectuer et l'état suivant).

Une représentation alternative d'un programme peut être réalisée sous la forme d'un graphe état – transition

Des exemples de représentation de programme par tableau et par graphe sont fournis au chapitre 1.3.

Exécution automatique du programme

Une fois le programme indiqué à ***TuringAFIS*** par le programmeur, ce dernier peut activer ***TuringAFIS. TuringAFIS*** exécute alors automatiquement les opérations qui y sont définies.

Au préalable de l'exécution, ***TuringAFIS*** est configuré de manière à ce que la case courante corresponde à la case « Case 1 » : la case située la plus à gauche de la mémoire (cf chapitre 1.2) doit pouvoir être lue sans opération de déplacement préalable.

A un haut niveau, la procédure d'exécution qui est suivie est la suivante :

Données internes : état courant

Procédure :

L'état courant est positionné à l'état initial,

Tant que l'état courant est différent de « fin » faire :

- Le système lit le contenu de la case mémoire courante

- En fonction de la valeur lue et de l'état courant, le système :
  - Procède à une écriture de la valeur stockée dans case mémoire courante conformément aux indications du programme
  - Procède à un déplacement vers la nouvelle case mémoire conformément aux indications du programme
  - Change son état courant, conformément aux indications du programme

Fin Tant que

1.2. Environnement

L'environnement de ***TuringAFIS*** est constitué :

- **D'un programmeur** en charge de programmer et d'opérer le système.
- **D'une mémoire** représentée physiquement par le dessin de 10 cases disposées horizontalement. Les cases sont identifiées de « Case 1 » à « Case 10 », de gauche à droite. Chaque case est elle-même découpée en 3 sous-cases disposées verticalement, identifiées de bas en haut par « vide », « 0 » ou « 1 ». Une case « Case i » contient un jeton qui selon son positionnement dans les sous-case indique :
  - dans la sous-case du bas (sous-case « vide ») : la mémoire « Case i » est vide
  - dans la sous-case du milieu (sous-case « 0 ») : la mémoire « Case i » est à 0
  - dans la sous-case du haut (sous-case « 1 ») : la mémoire « Case i » est à 1

  Les caractéristiques de la mémoire physique sont les suivantes :

  - Taille d'une sous-case : 10x10 cm
  - Taille et poids d'un jeton : cube en bois avec des arêtes de 3.4 cm (±3 mn) et un poids de 21.5 g ((±3 g)[1]
  - Couleur des sous-cases : blanc
  - Couleur des jetons : rouge
  - Couleur de la séparation entre les cases : noir
  - Couleur de la séparation entre les sous-cases : noir

  La figure 1 ci-dessous représente la mémoire physique vue du dessus. La case 10 contient la valeur 0 (le jeton est dans la sous-case du milieu). La case 2 contient la valeur 1 (le jeton est dans la sous-case du haut) et toutes les autres cases sont vides (les jetons sont en bas).

[1] Les jetons qui seront utilisés sont disponibles ici : https://toutpourlejeu.com/fr/cubes-et-jetons-carre/3512-cube-34-mm-rouge-vintage-bois-pour-jeu-3701525311967.html

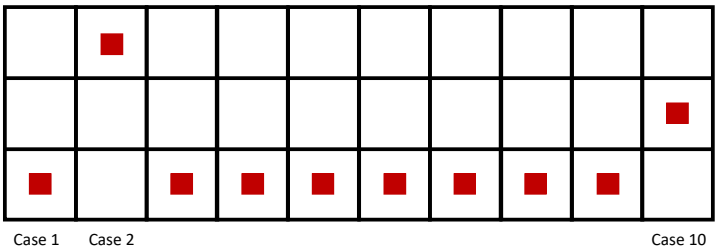


Figure 1 : représentation de la mémoire physique, vue de dessus.

- **D'un plateau de jeu** formé d'une surface plane sur laquelle la mémoire est disposée, de dimension supérieur ou égale à 140cmx140cm, horizontale et rigide, avec une surface dure et de couleur claire uniforme (cf Figure 2).

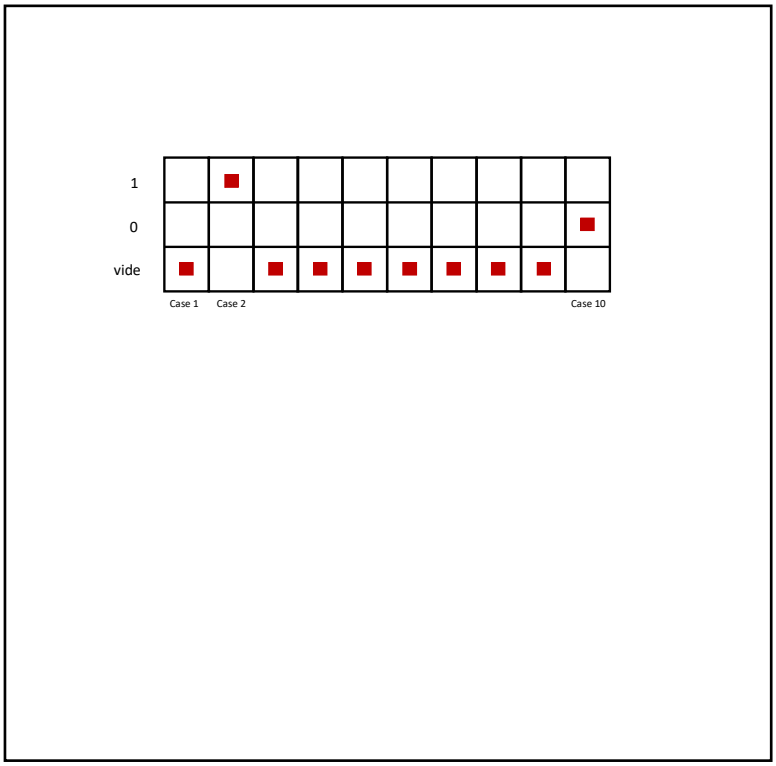


Figure 2 : plateau de jeu vu du dessus

Le plateau de jeu fait l'objet d'un éclairage régulier (lumière ambiante de la salle dans laquelle se trouve le terrain de jeu).

Les conditions climatiques de l'environnement sont les suivantes :

- Température ambiante évoluant dans une plage de 15 à 32 °C.
- Pression atmosphérique comprise entre 990 et 1030 mb.
- Hygrométrie comprise entre 40 et 75 %.

1.3. Exemple

Voici un exemple d'un programme qui réalise une addition de 1 à un nombre stocké en binaire dans la mémoire. Concrètement, le programme consiste à :

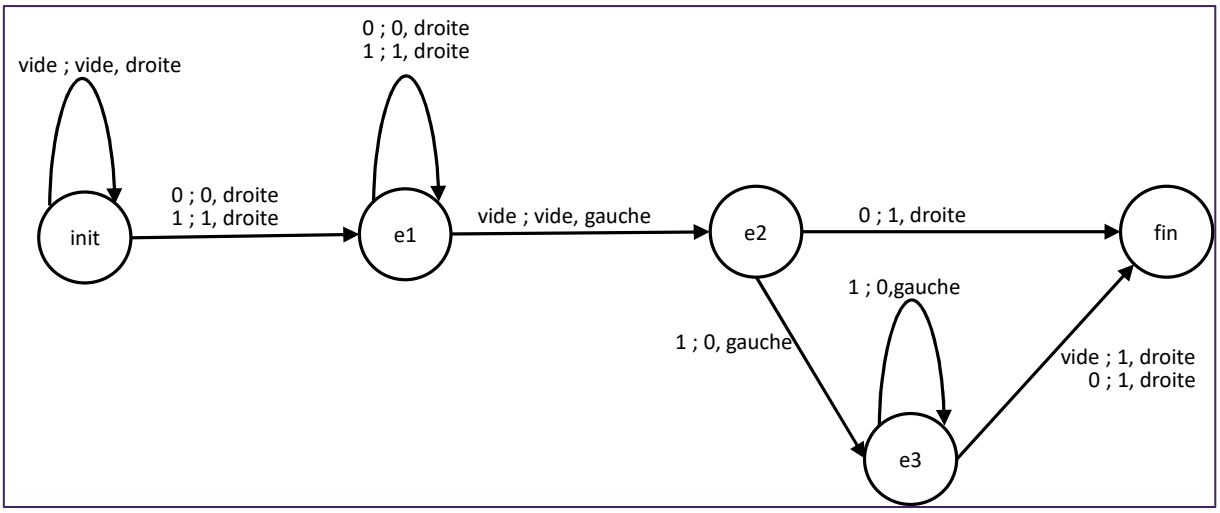
- Rechercher le début du nombre stocké en mémoire : positionner la case courante sur la case mémoire la plus à gauche du nombre (instructions 1 à 3), en recherchant la première case non vide ;
- Rechercher la fin du nombre stocké : positionner la case courante sur la case mémoire la plus à droite du nombre (instructions 4 à 6) en recherchant la prochaine case vide :
- Si la valeur de cette case est 0 alors on écrit 1 et c'est fini (instruction 7) ;
- Sinon, on écrit 0 et le programme propage la retenue vers la gauche (instructions 8 à 12).

Le programme est représenté par le tableau « états-pas d'exécution » suivant :

Instruction	Etat	Valeur lue	A écrire	Déplacement	Etat suivant
1	init	vide	vide	droite	init
2	init	0	0	droite	e1
3	init	1	1	droite	e1
4	e1	vide	vide	gauche	e2
5	e1	0	0	droite	e1
6	e1	1	1	droite	e1
7	e2	0	1	droite	fin
8	e2	1	0	gauche	e3
9	e3	vide	1	droite	fin
10	e3	0	1	droite	fin
11	e3	1	0	gauche	e3
12	fin				

Le graphe état-transition correspondant est le suivant :

- Les états sont représentés par les cercles
- Les pas d'exécution sont représentés par les arcs orientés. Les arcs sont libellés de la manière suivante : « valeur lue ; valeur à écrire, déplacement à effectuer ». L'arc indique l'état suivant.



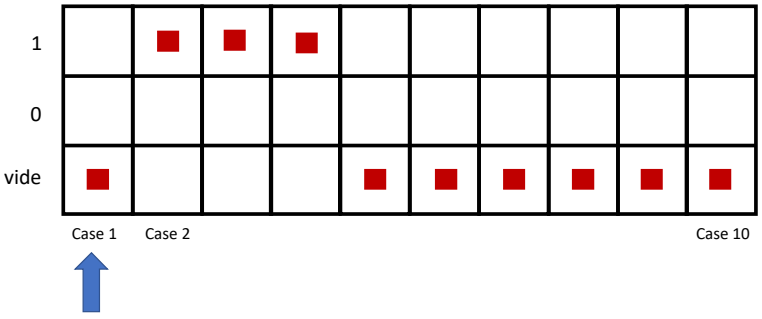
Exécution

Nous montrons ci-dessous l'exécution pas à pas du programme sur le nombre 111.

La flèche verticale bleue placée sous les cases mémoire représente la case mémoire courante.

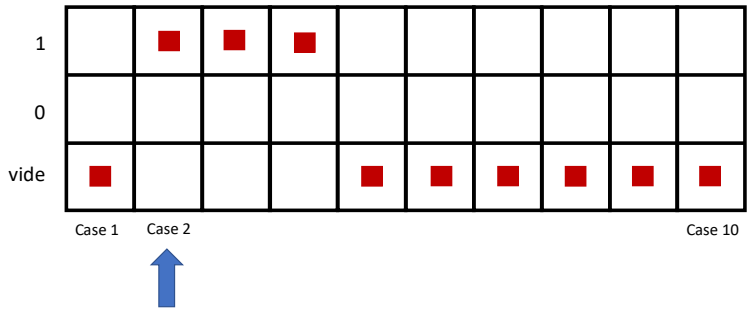
Etat initial :

Le nombre 111 est stocké en mémoire à partir de la case 2. La case mémoire courante est positionnée sur la case 1. L'état courant est positionné à init



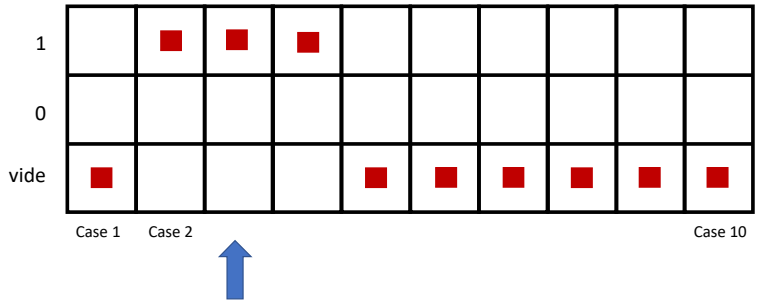
Etape 1 :

Dans l'état init, la valeur de la case 1 est « vide », ce qui conduit le système à écrire « vide » dans la case 1, à positionner la case mémoire courante sur la case 2, et à aller dans l'état init.



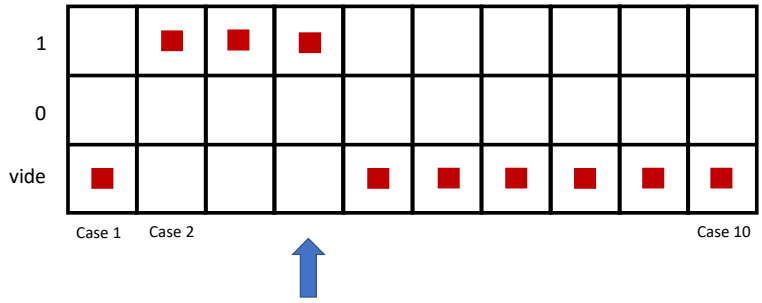
Etape 2 :

Dans l'état init, la valeur de la case 2 est « 1 », ce qui conduit le système à écrire « 1 » dans la case 2, à positionner la case mémoire courante sur la case 3, et à aller dans l'état e1.



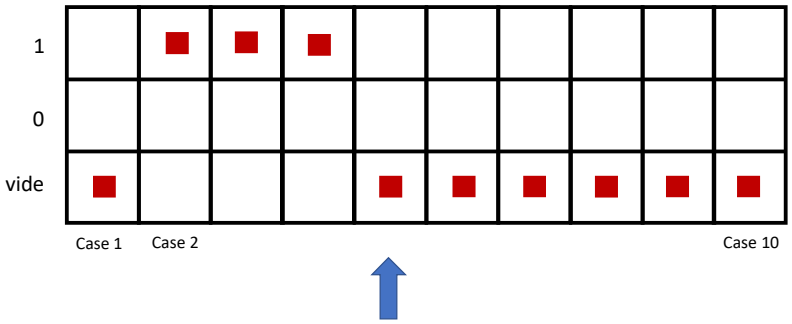
Etape 3 :

Dans l'état e1, la valeur de la case 3 est « 1 », ce qui conduit le système à écrire « 1 » dans la case 3, à positionner la case mémoire courante sur la case 4, et à rester dans l'état e1.



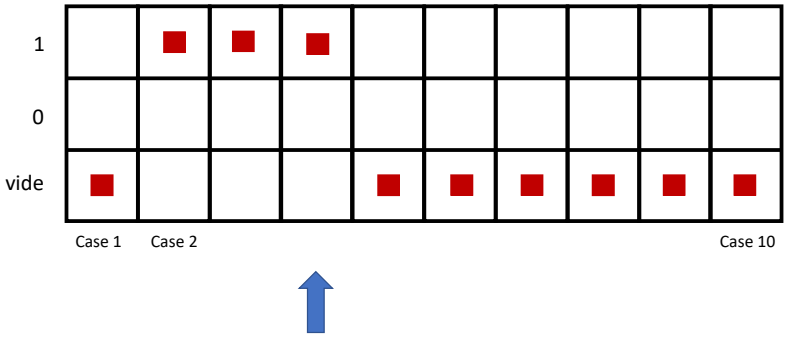
Etape 4 :

Dans l'état e1, la valeur de la case 4 est « 1 », ce qui conduit le système à écrire « 1 » dans la case 4, à positionner la case mémoire courante sur la case 5, et à rester dans l'état e1.



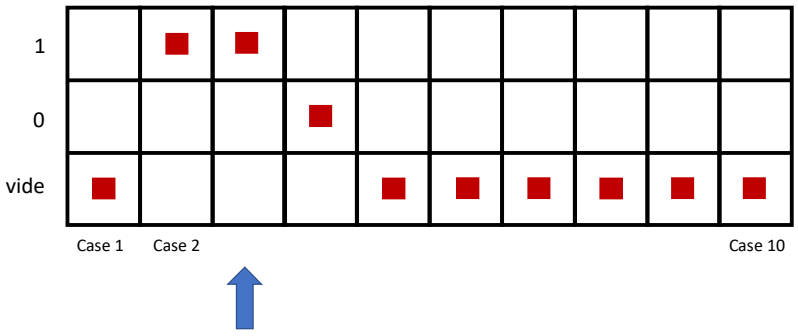
Etape 5 :

Dans l'état e1, la valeur de la case 5 est « vide », ce qui conduit le système à écrire « vide » dans la case 5, à positionner la case mémoire courante sur la case 4, et à aller dans l'état e2.



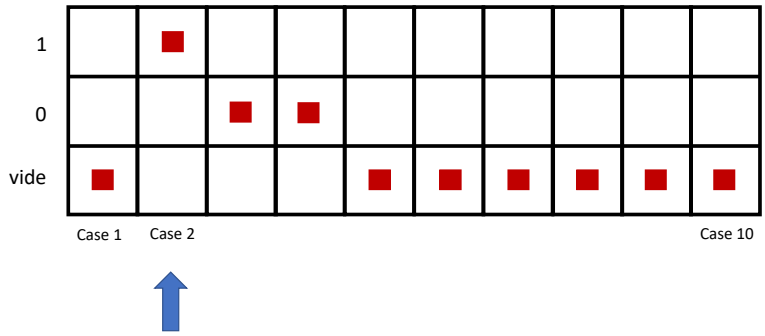
Etape 6 :

Dans l'état e2, la valeur de la case 4 est « 1 », ce qui conduit le système à écrire « 0 » dans la case 4, à positionner la case mémoire courante sur la case 3, et à aller dans l'état e3.



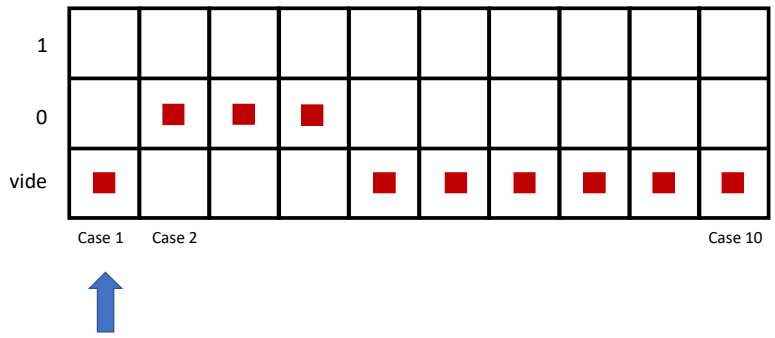
Etape 7 :

Dans l'état e3, la valeur de la case 3 est « 1 », ce qui conduit le système à écrire « 0 » dans la case 3, à positionner la case mémoire courante sur la case 2, et à rester dans l'état e3.



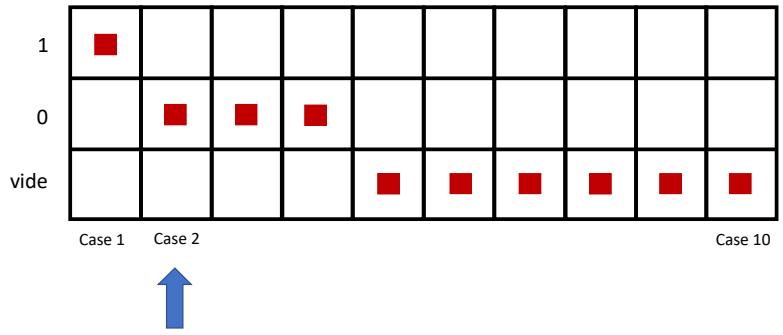
Etape 8 :

Dans l'état e3, la valeur de la case 2 est « 1 », ce qui conduit le système à écrire « 0 » dans la case 2, à positionner la case mémoire courante sur la case 1, et à rester dans l'état e3.



Etape 9 :

Dans l'état e3, la valeur de la case 1 est « vide », ce qui conduit le système à écrire « 1 » dans la case 1, à positionner la case mémoire courante sur la case 2, et à aller dans l'état fin.



Etape 10 :

Dans l'état fin, l'exécution se termine.

On obtient bien le résultat suivant : 111+1=1000.

1.4. Scenarii opérationnels

Concrètement, l'utilisation de ***TuringAFIS*** suit un scénario opérationnel qui contient les étapes suivantes :

1. Le Jury fournit au programmeur le programme à exécuter sous la forme d'un document papier décrivant le tableau « états-transitions »
2. Le programmeur saisit dans ***TuringAFIS*** le programme à exécuter communiqué par le Jury.
3. Le Jury positionne la mémoire dans une configuration initiale de son choix
4. Le programmeur configure ***TuringAFIS*** de façon à ce que la case courante soit la « Case 1 ».
5. Le programmeur déclenche l'exécution automatique du programme
6. ***TuringAFIS*** exécute automatiquement les instructions du programme. Lorsque l'état « fin » est atteint, il s'immobilise et signale la fin de l'exécution.

- **Scénario « auto test »**

Ce scénario consiste à exécuter le programme suivant qui doit être stocké de manière permanente dans la mémoire du système :

Programme « auto test » :

Etat	Valeur lue	A écrire	Déplacement	Etat suivant
init	vide	vide	droite	e1
e1	1	0	droite	e1
e1	0	0	droite	e1
e1	vide	1	gauche	e2
e2	vide	1	droite	fin
e2	0	0	gauche	e2
fin				

La configuration initiale de la mémoire sera communiquée par le jury au début de ce scénario.

- **Scénario « programme simple »**

Ce scénario consiste à saisir un programme dans le système, puis à l'exécuter sur une configuration mémoire donnée.
Le programme ainsi que la configuration initiale mémoire seront communiqués par le jury au début de ce scénario.

Caractéristiques :

› Un programme simple ne contient pas de bugs (cf. paragraphe 2.2 pour la définition des bugs possibles)

- **Scénario « programme complexe »**

Ce scénario consiste à saisir un programme complexe dans le système, puis à l'exécuter sur une configuration mémoire donnée.

Le programme ainsi que la configuration initiale mémoire seront communiqués par le jury au début de ce scénario.

Caractéristiques :

› Un programme complexe pourra contenir des bugs d'accès mémoire, des bugs dus à des instructions manquantes ou à une configuration mémoire erronée (cf. paragraphe 2.2 pour la définition des bugs possibles)

Durant ces scénarios, les comportements suivants doivent être suivis :

- ***TuringAFIS*** ne peut effectuer que des opérations de lecture, écriture et déplacement telles que stipulées dans le programme en cours d'exécution
- ***TuringAFIS*** ne doit pas empiéter sur le bord du plateau de jeu
- ***TuringAFIS*** ne doit pas modifier le positionnement d'un jeton hormis celui de la case mémoire courante
- ***TuringAFIS*** ne doit pas positionner les jetons à cheval sur plusieurs cases ou sous-cases mémoire.
- À tous moments le programmeur peut demander un arrêt d'urgence de l'exécution du programme. Dans ce cas, ***TuringAFIS*** doit immédiatement s'arrêter et attendre une opération de maintenance.
  Une opération de maintenance est alors réalisée par un opérateur de maintenance et peut consister en :
  - Un déplacement manuel de ***TuringAFIS***
  - La modification manuelle d'une ou plusieurs cases mémoire
  - Une opération de maintenance (changement de piles, réparation…)

  A l'issue de cette opération de maintenance, l'opérateur de maintenance peut demander au programmeur la reprise de l'exécution du programme.
- A la fin de la réalisation du programme, ***TuringAFIS*** doit notifier le programmeur que la mission est terminée et qu'il est disponible pour l'exécution d'un autre programme.

1.5. Interactions avec le programmeur

Programmeur -> *TuringAFIS*

Le programmeur peut :

- Saisir un nouveau programme.
- Enregistrer un nouveau programme
- Lancer l'exécution d'un programme préalablement enregistré ou du programme « auto-test ».
- Pendant l'exécution d'un programme :
  - Effectuer une commande d'arrêt d'urgence
  - Effectuer une commande de reprise, suite à un arrêt d'urgence

***TuringAFIS* -> Programmeur**

A tout instant ***TuringAFIS*** doit communiquer des informations permettant sa supervision :

- Si une exécution est en cours, ou non
- Si un arrêt d'urgence est en cours ou non. Dans le cas d'un arrêt d'urgence : la raison (détection de bug à l'exécution, arrêt demandé par le programmeur).
- Dans le cas où l'exécution d'un programme est en cours :
  - L'identification du programme qui est exécuté
  - L'état courant, la case mémoire courante ainsi que l'action en cours d'exécution
  - La fin d'une exécution

1.6. Interactions avec la mémoire (plateau de jeu)

***TuringAFIS* -> Mémoire**

TuringAFIS doit être capable d'accéder à la case mémoire située à gauche (resp. à droite) de la case mémoire courante

TuringAFIS doit être capable de lire le contenu de la case mémoire courante (si sa valeur est 0, 1 ou si elle est « vide »)

TuringAFIS doit être capable de modifier le contenu de la case mémoire courante en déplaçant physiquement le jeton d'une sous-case à l'autre de la case mémoire considérée.

2. Besoins

2.1. Contraintes opérationnelles

TuringAFIS respecte les contraintes opérationnelles suivantes :

- ***TuringAFIS*** est capable de réaliser les 3 scénarios opérationnels « auto test », « programme simple » et « programme complexe ».
- ***TuringAFIS*** fournit les informations pour être supervisé et contrôlé.
- ***TuringAFIS*** est capable de traiter des programmes contenant au maximum 6 états (en comptant les états « init » et « fin »).

2.2. Besoins de sûreté

TuringAFIS répond aux besoins de sûreté suivants :

- ***TuringAFIS*** assure l'intégrité de la mémoire :

- il ne doit pas modifier une case mémoire hormis celle de la case courante.
- Il ne doit pas positionner de jeton à cheval entre plusieurs sous-cases mémoire

- ***TuringAFIS*** assure sa propre intégrité (pas de chevauchement de la limite du plateau de jeu)
- ***TuringAFIS*** ne doit pas effectuer d'opérations contenant des bugs. Si une telle opération se présente, il doit interrompre son exécution via un arrêt d'urgence, et signaler le bug au programmeur.

  Un bug dans un programme peut être par exemple :

  - tentative d'accès à la case mémoire située à gauche de la case 1,
  - tentative d'accès à la case mémoire située à droite de la case 10,
  - une instruction manquante, un état suivant indeterminé
  - case mémoire occupée par plusieurs jetons

- ***TuringAFIS*** gère ses défaillances de la manière suivante :
  - Suite à une défaillance de l'alimentation en énergie, ***TuringAFIS*** est maintenable avec les pièces et outils du kits en moins de 2 min.
  - Suite à une défaillance mécanique, ***TuringAFIS*** est maintenable avec les pièces et outils du kits en moins de 5 min.
  - Suite à une demande explicite du programmeur, ***TuringAFIS*** doit interrompre l'exécution en cours, et attendre un ordre de reprise du programmeur après qu'une opération de maintenance ait été réalisée.

2.3. Besoins de performances

Les performances suivantes sont données :

- Il doit être possible pour le programmeur de saisir un programme en moins de 3 minutes
- Le scénario « auto-test » doit être implanté et exécuté en moins de 3 minutes.

2.4. Contraintes de réalisation

Le client impose l'utilisation de certains éléments pour réaliser ***TuringAFIS***:

- En plus d'un robot mobile, ***TuringAFIS*** comporte une interface permettant au programmeur de saisir un programme, et de superviser l'exécution et de contrôler l'exécution (démarrage, arrêt, interruption, reprise)

- ***TuringAFIS*** ne peut conserver dans sa mémoire que :
  - Les différents programmes saisis via l'interface programmeur ainsi que le programme « auto-test » décrit plus haut.
  - L'état courant au cours d'une exécution
- En particulier, il est interdit de stocker l'état de la mémoire physique.
- *Le robot mobile de* ***TuringAFIS*** est constitué des composants du kit Ultimate 2.0 (ref 90040) et des extensions suivantes : 1 capteur de couleur (ref MB 11050) sans ajout (e.g. de pièces, matériaux, capteurs, ou colle), ni modification d'un quelconque constituant du kit. Il utilise des piles de type AA/LR6 - 1,5 V.