Machine de Turing

ALLEGRE–COMMINGES Clément et BROUARD Romain

Table des matières

1 Introduction

1.1 Un peu d'histoire...

En 1928, Le mathématicien allemand David Hilbert énonce le "problème de la décision". Il se demande s'il est possible de trouver une méthode « effectivement calculable » pour décider si une proposition est démontrable. Pour résoudre ce problème, il faut caractériser ce qu'est un procédé effectivement calculable. C'est alors qu'Alan Turing, alors en thèse à Cambridge, conceptualise une machine universelle et prouve grâce à cette dernière que le problème de l'arrêt est indécidable, ce qui permet de donner une réponse négative au problème d'Hilbert pour l'arithmétique. C'est alors qu'Alan Turing introduit les concepts de programme et programmation.
Ce concept de machine universelle, que nous appellerons désormais Machine de Turing Universelle (MTU) n'est pas réalisable puisqu'il s'agit d'un objet mathématique, dont l'on va détailler le fonctionnement plus tard. Néanmoins, une Machine de Turing à état fini peut-être construite, et la première vit donc le jour à Bletchley Park pendand la Seconde Guerre Mondiale, où Turing lui-même et une équipe de scientifique triés sur le volet par le MI6 [1] construisirent une Machine de Turing pour casser les codes allemands générés par la machine Enigma.

1.2 Fonctionnement d'une MTU

Une MTU se compose de deux éléments essentiels :

- un ruban de taille infinie, divisé en cases.

- une tête de lecture/écriture (qu'on appellera simplement tête de lecture, même si elle permet également d'écrire sur le ruban)

Le ruban permet d'accueillir des symboles qui seront lus et écrits par la tête de lecture. Le ruban peut se déplacer vers la gauche ou la droite. L'ensemble des symboles traités est appelé un alphabet.
La tête de lecture permet de lire et écrire un symbole. Elle possède également un état.

Une machine de Turing marche donc de la manière suivante :

1. Services de renseignement britanniques

1. La tête de lecture va lit le symbole qu'elle pointe sur le ruban.

   En fonction de son état et du symbole lu, la tête de lecture écrit un symbole à la place de celui qu'elle a lu précédemment.

2. Un déplacement est choisi en fonction de l'état de la tête de lecture et du symbole lu.

3. la tête de lecture change d'état.
   Le ruban se déplace vers la droite ou vers la gauche selon le déplacement choisi précédemment.

4. Et on recommence depuis le 1.

On peut donc résumer le fonctionnement comme cela : à chaque "cycle", on choisi un symbole à écrire, un nouvel état pour la tête de lecture, et un déplacement, en fonction d'un symbole lu et de l'état actuel. On peut donc définir une fonction de transition, qui va se charger de déterminer l'état futur d'une MTU en fonction de ton état courant. L'ensemble des fonctions de transition permettant de traiter un "mot" peut être représenté sous la forme d'une table de transitions ou d'un graphe de transitions.
Un mot traité est dit accepté si une Machine de Turing s'arrête en état final après l'avoir intégralement traité.
Une MTU possède forcément un nombre fini d'états, et elle a au moins deux états obligatoires : q_O l'état initial, et F un ensemble d'états d'acceptation.

Une machine de Turing est donc définie par :

- Q un ensemble fini d'états.
- q_0 un état initial tel que $q_0 \in Q$.
- F un ensemble d'états d'acceptation tel que $F \subseteq Q$.
- Γ un ensemble fini de symboles.
- Σ un ensemble fini de symboles d'entrée tel que $\Sigma \subseteq \Gamma$.
- B un symbole de ruban vide tel que $B \in \Gamma \setminus \Sigma$.
- δ une fonction de transition.

Une fonction de transition se formalise donc comme ceci :

$$\delta(q, Z) \to (p, Y, D)$$

avec q l'état de la tête de lecture, Z le symbole pointé, p le nouvel état, Y le nouveau symbole et D le déplacement.

Exemple d'une table de transition pour une Machine de Turing acceptant le langage $L = \{a^k b^k \mid k > 0\}$ avec q_0 comme état initial (représenté par une $\to$), et q_4 comme état d'acceptation (représenté par $*$) :

	symboles				
	a	b	X	Y	Blank
→ q0	(q1, X, R)			(q3, Y, R)	
q1	(q1, a, R)	(q2, Y, L)		(q1, Y, R)	
q2	(q2, a, L)		(q0, X, R)	(q2, Y, L)	
q3				(q3, Y, R)	(q4, B, R)
* q4					

Le but de ce projet va donc être de concevoir une Machine de Turing telle que définie ci-dessus.

2 Conception

2.1 Cahier des charges

Le cahier des charges est composé de trois parties. La partie 1 doit être traitée absolument, les parties 2 et 3 seront traitées si le temps nous le permet.

2.1.1 Partie 1

Dans un premier temps, le système conçu doit être capable de :

- exécuter un programme prédéfini (codé en dur dans le programme microcontrôleur).
- avoir un mode pas à pas pour l'exécution du programme.
- avoir un mode continu pour l'exécution du programme.
- gérer l'affichage de l'état du ruban.
- gérer l'affichage de la position de la tête de lecture.
- gérer l'affichage de la table de transition.

2.1.2 Partie 2

Dans un second temps, il faut rajouter :

- la possibilité de sélectionner un programme via un menu.
- le stockage des programmes à sélectionner.
- l'initialisation manuelle du ruban et de la position de la tête de lecture.

2.1.3 Partie 3

Enfin pour obtenir un système complet, il faut implémenter :

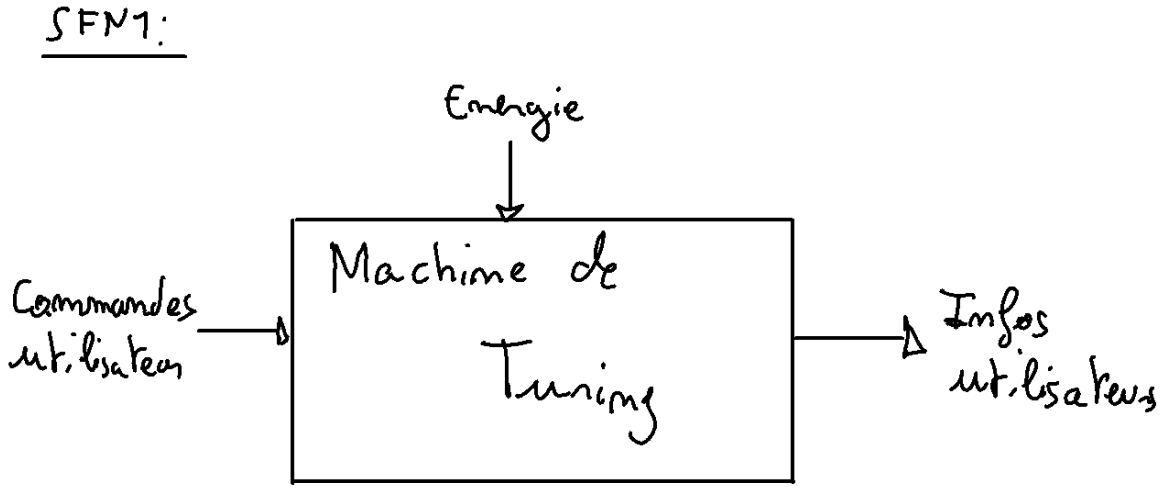
- la programmation directement sur la machine d'une table de transition.
- l'enregistrement de la table de transition programmée dans le support de stockage.
- un reset de la programmation de la ligne en cours.
- l'affichage d'une description du programme.

2.2 Conception Matérielle

Pour concevoir notre système, nous avons décidé de traiter les trois parties en même temps, ce qui nous évite de devoir repasser par une phase de conception et d'adaptation lors de la réalisation des parties 2 et 3.

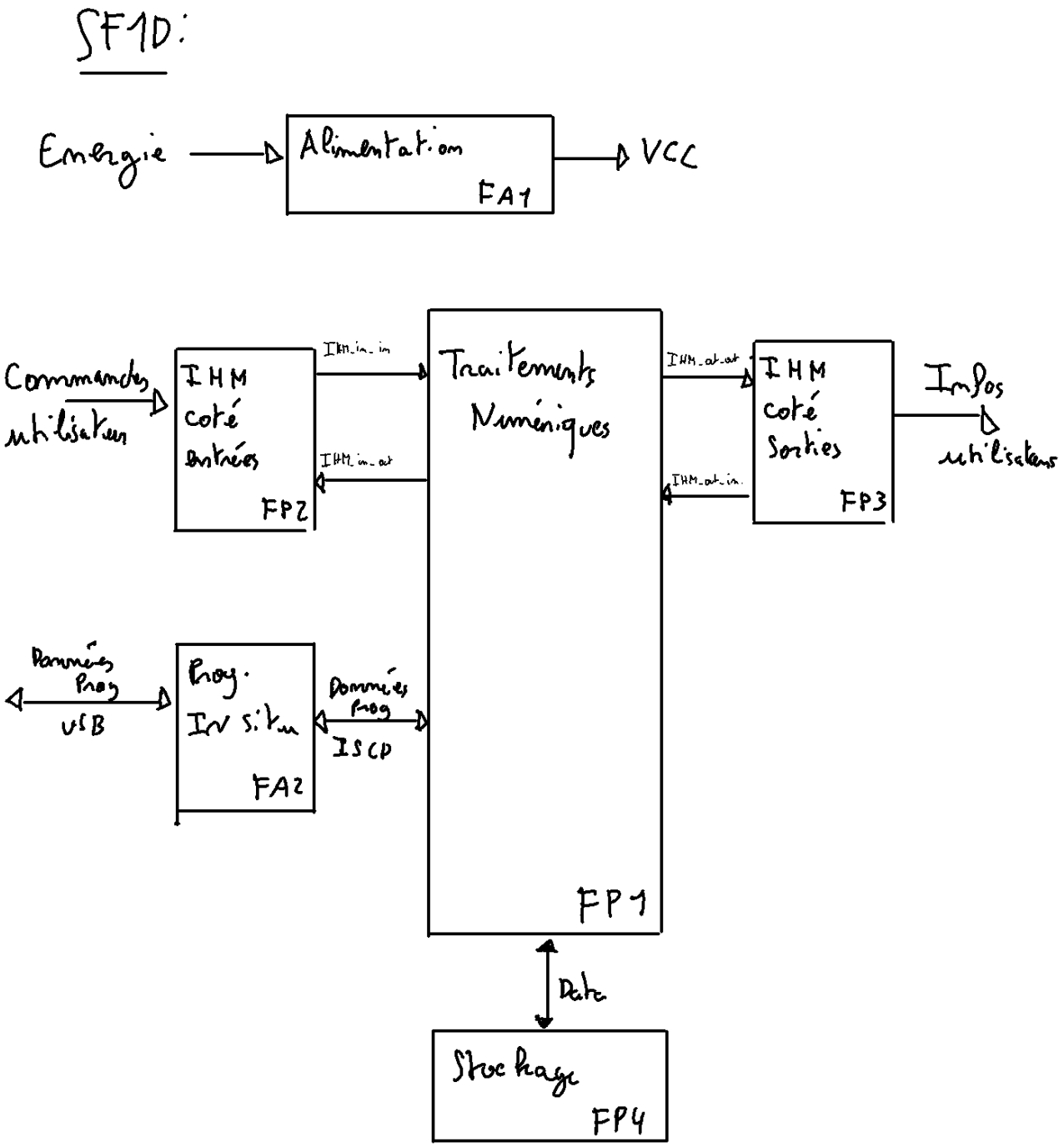
2.2.1 SFN1

Nous avons donc commencé par dessiner un Schéma fonctionnel de premier niveau, pour définir les différentes entrées et sorties de notre système :



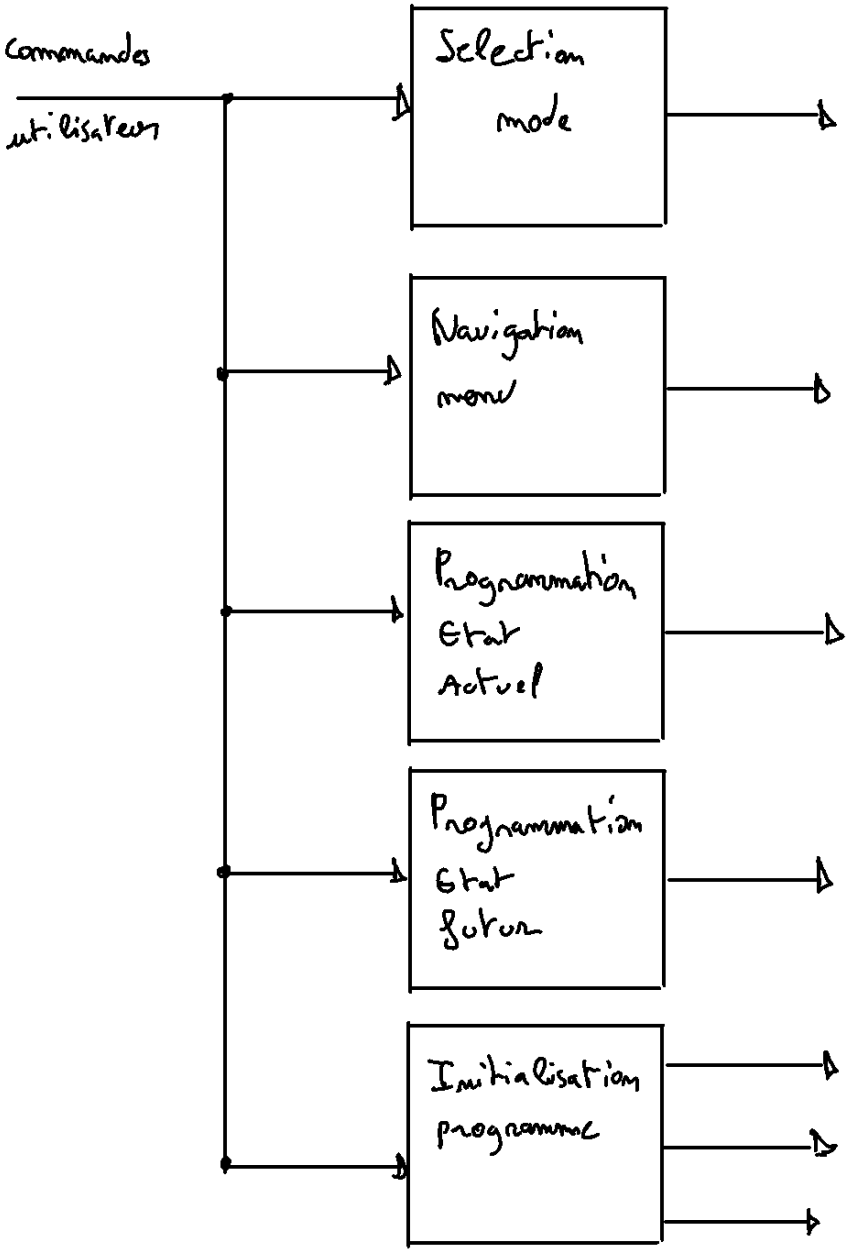
2.2.2 SFnD

On peut désormais rentrer un peu plus dans le détail avec un SF1D :

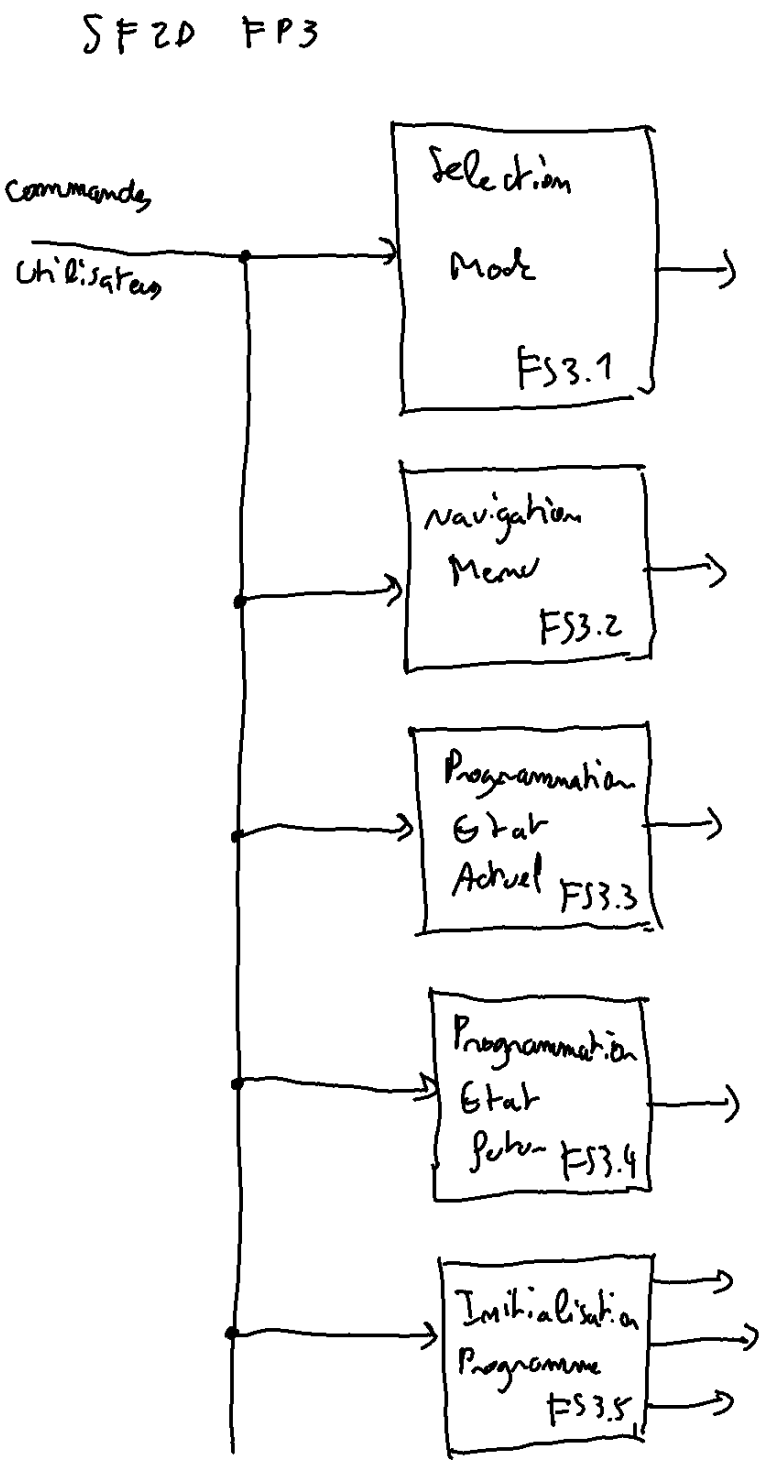


On voit qu'on a quatre fonctions principales : une qui s'occupe de la gestion des entrées (FP2), une qui gère les sorties (FP3), une fonction de stockage pour stocker les tables de transitions (FP4), et du traitement numérique qui va piloter tout ça (FP1).
On a aussi deux fonctions annexes : la fonction alimentation qui va se charger de fournir le courant nécessaire pour que la machine puisse fonctionner, et la fonction de programmation in-situ qui va faciliter le chargement du code dans la machine et nous éviter d'avoir à sortir le micro-contrôleur (MCU) à chaque fois.

On voit que les fonctions FP2 et FP3 sont encore floues, on va donc les affiner en faisant des SF2D pour qu'on se rende compte de leur fonctionnement :



FP2 :
On voit clairement que FP2 est bien plus complexe qu'elle ne le parait, et forme un ensemble de fonctions plus spécifiques qui vont permettre à l'utilisateur d'interagir physiquement avec la machine, dans le but de la configurer, programmer, etc.



FP3 :
De la même manière que pour FP2, FP3 est en réalité un ensemble de fonctions plus spécifiques qui vont chacune avoir un rôle bien précis dans le retour des informations utilisateur. On a donc une grande variété de fonctions d'affichage qui vont, de la même manière que pour les sous-fonctions de FP2, être spécifiées ci-dessous.

2.2.3 Spécifications des fonctions et de leurs signaux de communication

2.3 Conception Logiciel

3 Annexes

3.1 Sources