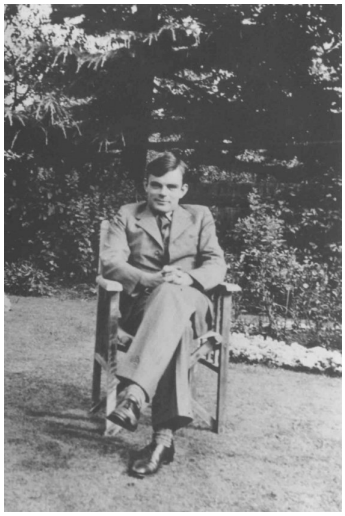
La machine universelle de Turing

Version 3

Lucas Satabin

2007

Table des matières



1 Introduction

Alan Mathison Turing (1912-1954) est un des pères de l'informatique moderne avec, notamment, John Von Neumann. Il a participé aux plus grands projets scientifiques du XX$^{\text{ème}}$ siècle comme le décryptage de la machine ENIGMA durant la Deuxième Guerre Mondiale en contribuant à la mise au point du Colossus I. Mais il a aussi fait des recherches en biologie, et en 1935, il crée la machine dite machine universelle de Turing (Pour une biographie complète, voyez dans la bibliographie [1]). Les possibilités de cette machine sont immenses et je vais essayer par la suite d'en expliquer le fonctionnement de base.

Le plus dur pour moi sera de rester le plus clair possible notamment pour les personnes n'ayant pas de notions mathématiques très poussées, mais le formalisme peut s'avérer nécessaire pour décrire totalement et sans ambiguïté une machine de Turing. Cependant, la première partie est compréhensible par tout le monde car elle n'utilise aucune notion mathématique. Par contre, la deuxième et la troisième sont indispensables pour suivre la suite, et sont peut-être plus théoriques. J'espère que les exemples que j'introduirai permettront d'éclaircir mes définitions. Mon but n'est pas non plus de faire un cours théorique poussé à l'extrême sur la machine de Turing, mais de donner une approche aux personnes intéressées, je ne parlerai donc pas de toutes les possibilités offertes (et elles sont nombreuses) et je ne démontrerai par conséquent pas les justifications de ce que font les machines que je présenterai, je ferai juste appel à votre "bon sens" car les exemples resteront assez simples dans l'ensemble (j'espère que l'on ne m'en voudra pas pour ce manque de rigueur évident mais si on démontre tout, ça devient rapidement compliqué). Pour ceux qui voudront pousser plus loin, des livres sont disponibles sur le sujet (cf la bibliographie à la fin). Je proposerai dans la dernière partie des ouvertures que je n'aborderai pas durant mon exposé, notamment en parlant des machines à plusieurs rubans. Mais nous reviendrons sur ces points plus loin.

2 Le fonctionnement de base

Tout d'abord, il ne faut pas croire que la machine de Turing est un concept impossible à imaginer. Même si elle n'est que théorique à la base, il est très simple de se la représenter. Pour ce faire, imaginez un ruban infini de chaque côté (c'est la partie théorique irréalisable...) sur lequel sont dessinées des cases. Chacune d'elles peut contenir un symbole. On peut aisément se rendre compte que le ruban est analogue à une mémoire d'ordinateur dans laquelle nous pourrons stocker des programmes et des données. La deuxième composante d'une telle machine est une tête de lecture/écriture capable de se déplacer, de lire la valeur contenue dans la case courante, et le cas échéant de modifier ladite valeur. Jusqu'ici rien de très compliqué, nous avons une bande sur laquelle sont inscrites des informations et une tête capable de lire et de modifier ces informations, ce qui est tout à fait similaire à une tête de lecture de magnétoscope sur la bande magnétique d'une cassette (même à l'heure du DVD cette analogie parle encore à tout le monde...). Cependant, cette tête de lecture/écriture possède plusieurs états qui lui permettent d'accomplir différentes tâches selon la valeur lue et l'état courant de la tête.

Une première représentation d'une machine de Turing peut se faire si l'on a à portée de main du papier, un crayon et une gomme. Bien sûr nous aurons beaucoup de difficultés à reproduire le caractère infini du ruban, mais nous concevons que la mémoire n'est pas illimitée, même dans nos machines les plus modernes, ce caractère infini n'est qu'une vue de l'esprit de mathématicien. Dans cette machine la tête de lecture/écriture est représentée par le crayon, la gomme et... par vous. En effet c'est vous qui représenterez la liste des états, réagissant différemment suivant les cas.

D'autres représentations concrètes peuvent être imaginées, qu'elles soient mécaniques ou que ce soit des interpréteurs programmés sur un ordinateur. Cependant toutes ces représentations resteront toujours limitées par la taille du ruban qui ne peut physiquement pas être infini.

Nous avons donc vu les différents composants d'une telle machine. Nous allons maintenant voir le fonctionnement de base de celle-ci. Ici commencent donc les choses sérieuses.

Les actions s'effectuent par étapes successives. A chacune d'elles, la tête lit la valeur inscrite dans la case devant laquelle elle se trouve, modifie son contenu suivant l'état courant et bouge d'une case sur la gauche ou la droite, ou bien reste sur place. Il ressort donc que nous devons enregistrer les différents états dans une table avec les actions à effectuer dans chacun d'eux. Ces états sont donc la représentation d'un programme dans un langage vraiment basique. Ce qui apparaît aussi comme une obligation à ce stade est le caractère fini du nombre d'états ainsi que du nombre de caractères possibles (c'est assez simple à s'imaginer car notre alphabet et somme toute limité... de plus la plupart du temps nous n'utiliserons que deux caractères : 1 et 0, voire éventuellement la case vide, l'informatique aime le binaire...).

Cette partie termine donc la description de base d'une machine de Turing, nous allons passer à une fomalisation de la chose...

3 Formalisons un peu

Dans la partie précédente, nous avons vu le principe de base d'une machine de Turing. Ici nous allons poser les notations qui nous permettrons de décrire une machine par la suite.

3.1 Un peu de vocabulaire

Définition 3.1 *Un alphabet est un ensemble Σ fini non vide dont les éléments sont appelés lettres ou caractères.*

Exemple 3.1 *Voici quelques exemples d'alphabets :*
- $\{a\}$ *est un alphabet ne comprenant que la lettre "a"*
- $\{allumer\,;\,éteindre\}$ *est aussi un alphabet où les lettres (ou caractères) sont en fait des mots dans notre langage courant. Ce sont pourtant bien des lettres si l'on définit un alphabet comme tel.*

3.2 Et la machine dans tout ça ?

Fort de ces quelques définitions, nous pouvons désormais décrire précisément une machine de Turing.

Définition 3.2 *Une machine de Turing est un quintuplet (Σ, Q, T, I, F), avec :*
- Σ *l'alphabet sur lequel s'effectuent les actions*
- *Q l'ensemble des états de la tête de lecture/écriture*
- *T l'ensemble des transitions*
- *I l'ensemble des états initiaux*
- *F l'ensemble des états finaux*

Par la suite nous n'utiliserons que le quadruplet (Q, T, I, F) et on précisera l'alphabet au début de l'utilisation dans les exemples par exemple, afin d'alléger les notations. Par ailleurs l'alphabet utilisé par défaut sera l'alphabet binaire à savoir $\Sigma = \{0, 1, \epsilon\}$ où ϵ est le caractère vide ou absence de caractère.

L'ensemble T des transitions définit les actions à effectuer dans les différents états de la machine, suivant les situations rencontrées.

Notation :

Nous noterons $G = \{-1; 0; 1\}$ l'ensemble décrivant les mouvements possibles de la tête. Par convention, "-1" décrit un déplacement sur la gauche, "0" aucun déplacement, et "1" un déplacement sur la droite. Tous les déplacements se font d'une seule case sur le ruban.

Définition 3.3 *Avec les notations précédentes, nous obtenons $T \subset Q \times \Sigma \times \Sigma \times G \times Q$. Une transition est entièrement définie par $t = (q, a, a', m, q') \in T$, où :*
- *q est l'état de départ*
- *a le caractère lu*
- *a' le caractère à écrire*

- *m est le mouvement à effectuer*
- *q' est l'état d'arrivée*

Par convention, nous dirons que si la machine lit un caractère inconnu dans l'état courant, elle s'arrête à cette étape.

Exemple 3.2 *Pour commencer nous allons réaliser une machine toute simple qui se comporte comme un porte NON. Si elle lit 1 en entrée elle rend 0 et réciproquement. Avec cet exemple très simple, nous comprendrons le fonctionnement et les notations.*
Soit donc M une telle machine sur l'alphabet $\Sigma = \{1; 0\}$ *définie par :*

- $Q = \{q_0\}$
- $T = \{(q_0,\ 1,\ 0,\ 1,\ q_0)\ ;\ (q_0,\ 0,\ 1,\ 1,\ q_0)\}$
- $I = \{q_0\}$
- $F = \{q_0\}$

Cette machine n'a qu'un seul état nommé q_0*. Si elle lit un 1, elle écrit 0 avance d'une case et reste dans le même état. Inversement, si elle lit un 0, elle écrit 1, avance et reste dans cet état aussi. A l'itération suivante, elle va lire un caractère vide, qui n'appartient pas à l'alphabet, donc inconnu et la machine s'arrête. Comme l'état courant est un état final, l'entrée est acceptée.*

Ici apparaît un problème. Si en entrée, nous avions rencontré par exemple le caractère "a". Ce caractère étant inconnu, la machine s'arrête, dans son unique état qui est final. L'entrée est donc acceptée par la machine. Cependant, le caractère n'appartenant pas à l'alphabet, on ne peut pas dire que l'entrée est acceptable. Nous pouvons résoudre ce problème en ajoutant un état, qui lui sera final, alors que le premier ne le sera plus. Ainsi, lors des deux transitions, on passe dans l'état final si on rencontre un caractère autorisé, sinon on reste dans le même état, et comme celui ci n'est pas final, l'entrée n'est pas acceptée.

3.3 Le déterminisme

Dans l'exemple précédent nous avons vu qu'il n'y avait qu'un état initial. Que se passerait-il s'il y en avait deux ou plus ? De même, dans un état donné pour un caractère lu donné, il n'y a qu'une transition. Si plusieurs choix s'offraient à la machine, comment réagirait-elle ?
Quand dans la description d'une machine de Turing, de telles ambiguïtés apparaissent, on dit que la machine est *non déterministe*

Définition 3.4 *Une machine de Turing (Q, T, I, F) sur l'alphabet* Σ *est dite déterministe ssi :*

i *L'ensemble I des états initiaux est un singleton.*

ii $\forall q \in Q, \forall a \in \Sigma,$
si $\exists (p, p') \in Q^2, \exists (b, b') \in \Sigma^2, \exists (m, m') \in G^2$ *tq* $(q, a, b, m, p) \in T$ *et* $(q, a, b', m', p') \in T$
Alors
- $p = p'$

- $b = b'$
- $m = m'$

Autrement dit, Dans un état donné, pour un caractère lu donné, il n'existe au plus qu'une seule transition.

A partir de ce moment, on peut se demander si une machine de Turing non déterministe peut être remplacée par une autre déterministe qui effectue le même traitement. On obtient :

Propriété 3.1 *Toute machine de Turing non déterministe peut être émulée par une machine de Turing déterministe*

Nous aborderons dans la section 6, le problème de déterminisation d'une machine de Turing.

4 Représentation d'une machine de Turing

Maintenant que nous avons décrit formellement une machine universelle, nous povons constater que ce formalisme n'est pas facilement compréhenseible pour se représenter la machine décrite. Il est donc plus simple de représenter une machine de Turing graphiquement à l'aide (justement) de graphes. Notre but n'est pas de faire un cours sur la théorie des graphes, mais quelques éléments sont essentiels, c'est pourquoi nous allons définir les termes utiles.

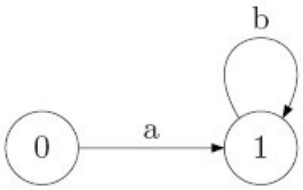
4.1 Définitions

Définition 4.1 *Un graphe orienté G est formé de deux ensembles finis : un ensemble X dont les éléments sont appelés sommets et un ensemble $A \subset X \times X$ dont les éléments sont appelés arcs. On note $G = \{X,A\}$.*
Si $a = (x, y) \in A$, on dit que x est l'extrémité initiale de A et y l'extrémité finale de A.

Plus simplement, s'il existe un arc orienté de x vers y, on trouve dans A le couple (x,y) ($\neq (y, x)$).

Définition 4.2 *Un graphe est dit étiqueté si à chaque arc on affecte une valeur. Dans ce cas on prendra $A \subset X \times X \times \Sigma$ où Σ est l'alphabet sur lequel les valeurs des étiquettes sont prises.*

Exemple 4.1 *Voici une représentation du graphe orienté étiqueté $G = (X, A)$ où :*
- $X = \{1; 2\}$
- $A = \{(1, 2, a); (2, 2, b)\}$

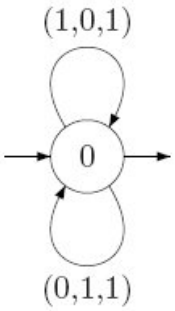


4.2 Graphe d'une machine

A partir de maintenant, nous allons représenter une machine de Turing par un graphe orienté étiqueté. Les états initiaux seront représentés par une flèche entrante sur lesdits états, et les états finaux porteront une flèche sortante. Les sommets représenteront les états de notre machine, et porteront leur nom, et les arcs représenteront les transitions. Les étiquettes des arcs porteront les informations de la transition. Ainsi nous pouvons définir E l'ensemble des étiquettes du graphe, et nous avons : $E \subset \Sigma \times \Sigma \times G$ en reprenant les notations de la section 3.
Ainsi soit $e = (a, b, d) \in E$:

- a est le caractère lu
- b est le caractère à écrire
- d est le déplacement à effectuer

Exemple 4.2 *Si on reprend l'exemple 3.2 de la section 3 qui décrit la machine NON, nous obtenons la représentation sous forme de graphe suivante*



Ce genre de représentation graphique permet de se représenter plus facilement la machine décrite et donc de mieux comprendre son but. Bien sur il n'est pas obligatoire de représenter ainsi une machine de Turing, par exemple nous pourrions la rpésenter sous forme d'un tableau qui indique, suivant l'état et le caractère lu, le caractère à écrire, le déplacement et l'état suivant. Malgré tout, il me semble que la représentation sous forme de graphe est plus lisible et permet de suivra plus facilement le déroulement de l'exécution du programme, et même de l'exécuter manuellement.

5 Mise en pratique sur des exemples

Nous avons, à mon sens, posé suffisamment de bases pour explorer maintenant des exemples intéressants de machines de Turing. De même vous pouvez tout aussi bien vous

plonger dans la conceptions de machines de Turing sans aucun problème. Sur ce point, je ne peux vous donner aucune méthode miracle, mais je ne peux que vous conseiller de commencer par écrire sur papier ce que vous voulez exactement réaliser et l'alphabet sur lequel vous allez travailler. Ensuite c'est une machine à état, donc décomposez bien en états (justement) votra algorihme...

Dans les exemples qui vont suivre, je ne prétend en aucun cas donner LA solution optimale (s'il en est) au problème, mais une solution éventuellement parmi d'autres, compréhensible par le plus de monde possible. Il est toujours possible de chercher à améliorer une machine en terme de nombre d'états mais la lisibilité peut en être affectée.

5.1 Exemples expliqués

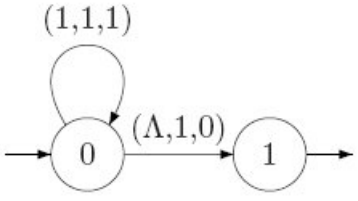
Je vais ici vous présenter quelques exemples de base, expliqués entièrement (de manière intuitive ceci dit...). Ensuite je poserai quelques problèmes et idées sans les résoudre, vous verrez vous y arriverez c'est certain. Cependant si tel n'est pas le cas, vous pouvez me contacter par le biais du site donné dans les remerciements.

5.1.1 Les tas de bâtons

On aime beaucoup les bâtons dans les machines de Turing, c'est très simple à manipuler et à comprendre. Le premier exemple que je vais donner est le suivant : *ajouter un bâton à la fin d'un tas de bâtons donné en entrée*. Simple, n'est-il pas ?
Il est aussi très simple de le résoudre, dans un premier temps, nous voulons que la machine parcourt le tas de bâtons en entier, et une fois qu'elle est arrivée au bout qu'elle ajoute ce bâton. Nous représenterons ici un bâton par un "1", et comme c'est le seul caractère que nous rencontrerons (du moins que nous saurons interpréter) ou presque, notre alphabet Σ sera restreint à celui-ci.
Dans l'explication du problème donnée ci dessus nous voyons apparaître deux états :

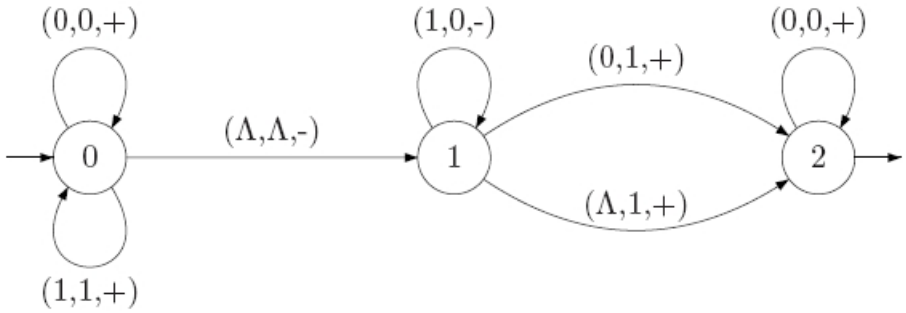
- On parcourt le tas
- On a posé de le nouveau bâton

Que fait-on lorsqu'on parcourt le tas ? On laisse le bâton lu en place et on avance d'une case. On détecte la fin du tas par la lecture du *mot vide*, noté Λ. Il représente l'absence de caractère dans la case lue. Ainsi lorsque qu'on trouvera cette absence de caractère, il nous suffira d'écrire un bâton (un "1") et de passer dans l'état *j'ai ajouté le bâton*, et la machine peut s'arrêter. Il en résulte donc le graphe suivant :



5.1.2 Incrémenter un nombre en binaire

Après les bâtons et un peu plus évolué, on manipule très souvent des nombres binaires. Pour illustrer ceci nous allons réaliser ici une machine permettant d'incrémenter un nombre écrit en binaire donné en entrée.
Analysons le problème comme dans l'exemple précédent. L'alphabet des nombres binaires n'est pas beaucoup plus important étant donné qu'il ne comporte que deux symboles : 1 et 0.
Pour résoudre ce problème nous partons d'un état dans lequel nous parcourerons le nombre pour en déduire le résultat. Une fois ce nombre parcouru, nous ajouterons un, pour cela un nombre impair devient pair et réciproquement. Il faut ensuite reporter la retenue et la propager jusqu'à ce que ce soit inutile. Typiquement elle remontera tant que nous rencontrerons des *1* dans le nombre. Si nous renconcontrons un *0* ou le mot vide Λ en ayant remonté tout le nombre binaire (ce qui revient au même car on peut considérer que le nombre commence par des *0*), nous écrivons un *1*, puis nous reparcourons le nombre pour que la tête de lecture/écriture finisse après ce nouveau nombre. Ce qui donne le graphe suivant :



NB : On peut remarquer que pour finir de parcourir le nombre nous ne regardons que la présence de *0*, ceci est dû au fait que la propagation de la retenue n'a produit que des *0*.

5.2 Problèmes ouverts

A présent je peux vous proposer quelques machines marrantes à réaliser et pas trop compliquées.

- Les portes logiques : étant donné en entrée deux bits (0 ou 1), faire calculer le *et*, le *ou*, le *ou exclusif*.
- Etant donné un tas de bâtons en entrée, écrire une machine qui triple le nombre de bâtons dans le tas
- Une machine qui décrémente un nombre en binaire.
- En combinant une machine qui incrémente à une autre qui décrémente, on peut écrire un additionneur de deux nombres binaires très facilement.

NB : Pour séparer deux données différentes en entrée, il est coutume de laisser une case vide entre les deux.

6 Pour aller plus loin

Le domaine des machines de Turing est très vaste et très complexe dès lors que l'on souhaite pousser un peu plus nos études. Dans cette section, nous allons ouvrir quelques portes sur des idées intéressantes et à creuser (si vous en avez le courage et l'envie...).

6.1 La thèse de Church-Turing

> LCMs [logical computing machines : Turing's expression for Turing machines] can do anything that could be described as "rule of thumb" or "purely mechanical".[1]

Alan Mathison Turing

Cette citation est l'énoncé d'Alan Turing sur sa thèse qui, pour résumer, prétend que tout algorithme est programmable sur une machine de Turing, et que, réciproquement, une fonction calculable par une machine de Turing est récursive, ou encore implémentable sur une machine comme on a l'habitude d'en manipuler. Je ne vais pas m'attarder ici sur cette thèse mais un article très intéressant peut être trouvé à ce sujet sur le net (cf la référence [5]).

6.2 Le déterminisme

Nous avons vu dans la section 3 qu'à toute machine de Turing non déterministe, nous pouvions associer une machine déterministe équivalente, ie qui calcule la même chose avec les mêmes données en entrée. Il nous vient donc à l'idée d'automatiser ce passage d'une machine à l'autre comme on peut le faire pour un automate fini.
Cette section ne sera pas très remplie dans cette version, étant donné que je n'ai pas beaucoup d'informations sur la manière de procéder. Mes recherches sont restées pour le moment infructueuses.

6.3 Arrêt d'une machine de Turing

Quand on écrit une machine (ou tout algorithme dans tout langage de programmation), on aimerait savoir si cette machine va s'arrêter ou continuer à boucler indéfiniment. Il s'avère en fait qu'il est impossible de prédire dans le cas général si une machine va s'arrêter. Le problème de l'arrêt d'une machine de Turing est donc *indécidable*. Nous allons le montrer par l'absurde en utilisant les machines de Turing.

Supposons qu'il existe une machine de Turing MA, qui, si on lui donne en entrée les instructions d'une autre machines de Turing et un entier n, répond *oui* sur le ruban si la machine donnée en entrée s'arrête avec la valeur n donnée, et *non* si elle ne s'arrête pas. Une telle machine existe vu que l'on suppose que le problème de l'arrêt d'une machine de Turing est décidable. Supposons maintenant que l'on a numéroté les machines de Turing (pourquoi pas ?) et créons la machine MB qui prend en entrée un entier n, et qui le transforme en la suite d'instructions de la machine de Turing numérotée n, suivie de

[1] Les machines à calculer logiques peuvent faire tout ce qui peut être décrit "en règle générale" [non basé sur une définition mathématique précise] ou de manière purement mécanique.

l'entier n, et qui ensuite se comporte comme la machine MA mais qui s'arrête uniquement si la machine MA aurait renvoyé *non*, boucle indéfiniment sinon. Comme MB est une machine de Turing, elle est numérotée. Soit k le numéro de la machine NB. Donnons en entrée à MB l'entier k (qui est son numéro). Si elle s'arrête, cela signifie que la machine numéro k (MB elle même) ayant pour entrée la donnée k boucle indéfiniment,et pourtant elle vient de s'arrêter avec cette donnée. Réciproquement, si elle boucle indéfiniment, cela signifie qu'elle s'arrête avec la valeur k en entrée. Nous aboutissons ainsi à une contradiction. L'hypothèse de base n'est donc pas fondée et le problème de l'arrêt des machines de Turing est indécidable (une aspirine ?).

Bien sûr, dans certains cas, il est possible de prouver l'arrêt de machines de Turing, mais dans le cas général c'est impossible.

6.4 Variantes des machines de Turing

Nous avons vu dans les sections précédentes un type de base de machines de Turing, simples avec un ruban infini de chaque côté. Nous pouvons imaginer des variantes à ces machines qui donnent de nouvelles propriétés. Par exemple on peut imaginer un ruban fini d'un côté et infini de l'autre, ou encore une machine qui possède plusieurs rubans.

6.5 Turing et les autres

Alan Turing fait partie d'une bande de joyeux drilles du début du XX$^{\text{ème}}$ siècle, parmi lesquels on peut retrouver Gödel, Hilbert, Church ou encore Von Neumann, et qui ont posé les bases de l'informatique moderne (on parle toujours des architectures Von Neumann pour les ordinateurs). Ils ont conduit d'importantes recherches en mathématiques modernes, ouvert des domaines (le $\lambda - calcul$ par exemple), révolutionné notre idée des mathématiques (théorèmes d'incomplétude de Gödel). Si cela vous intéresse, je vous invite à vous plonger dans leur biographie et leurs travaux.

7 Conclusion

Alan Turing passé sa vie à vouloir construire une machine pensante, et a posé (avec d'autres) les bases de l'informatique moderne. Cette machine universelle nous paraît bien dérisoire mais reste une des premières inventées pour résoudre tout algorithme de manière systèmatique. Bien que son efficacité ne soit pas des plus impressionnantes en terme de rapidité, elle en reste néanmoins, à mes yeux, fascinante et complète. Beaucoup de choses peuvent encore être découvertes dans ce domaine, tant il est vaste, mais il n'en est pas moins complexe. Vous trouverez ici et là des documents de recherche sur des domaines bien particuliers de ces machines, je vous laisse parcourir toutes vos ressources disponibles pour approfondir ce sujet ô combien intéressant.

8 Annexes

Remerciements

Je tiens à remercier Hervé Devie dont les cours sur les automates finis m'ont été très utiles, et dont j'ai repris certaines définitions dans ce texte. Je remercie aussi mon premier lecteur, et donc correcteur, Gwen pour avoir accepté, sans râler, de lire mes délires alors que l'idée ne lui serait pas venue seule. Merci aussi à Anthony pour avoir corrigé une énormité mathématique... Merci aussi bien évidemment à vous de lire ce que j'écris.
Si vous avez des questions, commentaires, suggestions pour une version future, si vous avez remarqué des fautes, ou si vous voulez me parler de vos travaux personnels sur les machines de Turing, je suis tout ouïe, contactez moi par le biais de notre site : http ://gnieh.homelinux.org (site de projets libres, n'hésitez pas à y faire un tour...)

Références

[1] *Alan Turing, l'homme qui a croqué la pomme*, de Laurent Lemire chez Hachette

[2] *La machine de Turing*, par Alan Turing et Jean-Yves Girard, chez Seuil

[3] *Alan Turing : The enigma*, (titre français : *Alan Turing ou l'énigme de l'intelligence*) de Andrew Hodges, chez Payot & Rivages

[4] http ://www.turing.org.uk/ site de Andrew Hodges avec notamment un émulateur de machine de Turing.

[5] http ://plato.stanford.edu/entries/church-turing/ article sur la thèse de Church-Turing.

Licence