Machine de Turing

Yann Strozecki
yann.strozecki@uvsq.fr

1er avril 2021

Objectifs du cours

- Commment est-ce qu'on calcule ?
- Quelles sont les différences entre les dispositifs de calcul ?
- Que peut-on calculer dans l'absolu ?

Rappel des épisodes précédents

Plusieurs modèles de calcul présentés :

- automate fini
- automate fini non-déterministe
- expression régulière
- automate à pile déterministe
- automate à pile non-déterministe
- grammaire non contextuelle

Peut-on imaginer des dispositifs plus généraux ?

Rappel des épisodes précédents

Plusieurs modèles de calcul présentés :

- automate fini
- automate fini non-déterministe
- expression régulière
- automate à pile déterministe
- automate à pile non-déterministe
- grammaire non contextuelle

Peut-on imaginer des dispositifs plus généraux ?

Quelles sont les limites des différents modèles de calcul ?

Rappel des épisodes précédents

Plusieurs modèles de calcul présentés :

- automate fini
- automate fini non-déterministe
- expression régulière
- automate à pile déterministe
- automate à pile non-déterministe
- grammaire non contextuelle

Peut-on imaginer des dispositifs plus généraux ?

Quelles sont les limites des différents modèles de calcul ?

Limite des automates à pile

Le lemme de l'étoile montre que les automates ne peuvent reconnaître le langage $\{a^n b^n \mid n \in \mathbb{N}\}$.
Les automates ne peuvent pas simuler un compteur.

Une généralisation du lemme de l'étoile montre que les automates à pile ne peuvent pas reconnaître le langage $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.
Les automates à pile ne peuvent pas simuler deux compteurs.

Limite des automates à pile

Le lemme de l'étoile montre que les automates ne peuvent reconnaître le langage $\{a^n b^n \mid n \in \mathbb{N}\}$.
Les automates ne peuvent pas simuler un compteur.

Une généralisation du lemme de l'étoile montre que les automates à pile ne peuvent pas reconnaître le langage $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.
Les automates à pile ne peuvent pas simuler deux compteurs.

Quid des automates à k piles ? À voir en TD.

Limite des automates à pile

Le lemme de l'étoile montre que les automates ne peuvent reconnaître le langage $\{a^n b^n \mid n \in \mathbb{N}\}$.
Les automates ne peuvent pas simuler un compteur.

Une généralisation du lemme de l'étoile montre que les automates à pile ne peuvent pas reconnaître le langage $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.
Les automates à pile ne peuvent pas simuler deux compteurs.

Quid des automates à k piles ? À voir en TD.

Machine de Turing

Ajouter de la mémoire à un automate le rend plus puissant.
On remplace la pile par une liste (bande infinie vers la droite), c'est la *machine de Turing*.

Elle est définie par :

1. Un alphabet Σ des caractères d'entrée
2. Un alphabet de travail Γ avec $\Sigma \subseteq \Gamma$ et un caractère spécial blanc noté $\square$ qui n'est pas dans Σ
3. Un ensemble fini d'états Q
4. Un état initial $q_0 \in Q$, deux états spéciaux $ACCEPT \in Q$ et $REJECT \in Q$
5. Une fonction de transition δ de $Q \times \Sigma$ vers $Q \times \Sigma \times \{L, S, R\}$

Machine de Turing

Ajouter de la mémoire à un automate le rend plus puissant.
On remplace la pile par une liste (bande infinie vers la droite), c'est la *machine de Turing*.

Elle est définie par :

1. Un alphabet Σ des caractères d'entrée
2. Un alphabet de travail Γ avec $\Sigma \subseteq \Gamma$ et un caractère spécial blanc noté $\square$ qui n'est pas dans Σ
3. Un ensemble fini d'états Q
4. Un état initial $q_0 \in Q$, deux états spéciaux $ACCEPT \in Q$ et $REJECT \in Q$
5. Une fonction de transition δ de $Q \times \Sigma$ vers $Q \times \Sigma \times \{L, S, R\}$

Configuration d'une MT

La configuration d'une MT est un triplet (B, pos, q) ou B est la bande ou ruban de la machine, pos est une entier désignant la position de la tête de lecture et $q \in Q$.
Une bande B est une suite infinie de symboles de Γ, qu'on peut noter $B_0, B_1, \ldots$. On représente B par un mot fini (les symboles suivant étant $\square$).

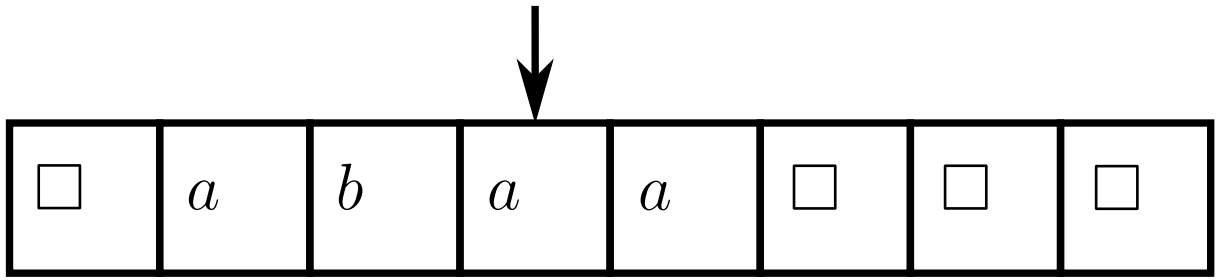
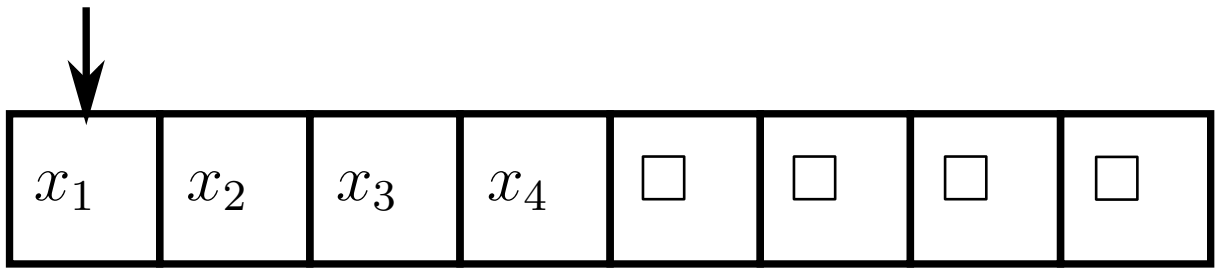


Figure – Bande d'une machine de Turing, la position de la tête de lecture (3) représentée par une flèche.

Configuration initiale

Comme expliqué précédemment, on note x la bande qui contient le mot x sur ses premières cases et $\square$ sur ses cases suivantes. La configuration initiale d'une MT est $(x, 0, q_0)$.



Transition entre configurations

On définit la transition de la configuration (B, pos, q) à la configuration (B', pos', q') par la fonction de transition δ :

- On note $(q', c, D) = \delta(q, B_{pos})$ (calcul du nouvel état)
- $B_i = B'_i$ pour $i \neq pos$ et $B'_{pos} = c$ (écriture d'un symbole sur la bande)
- Si $D = L$, $pos' = pos - 1$, si $D = S$, $pos' = pos$ et si $D = R$, $pos' = pos + 1$.
  Attention, si $pos = 0$ et $D = L$, on ne peut pas aller vers la gauche et on a $pos' = 0$. Pour éviter ce problème, on peut faire commencer le ruban par un symbole spécial.
  (calcul du mouvement de la tête)
- On ne fait pas de transition à partir des états ACCEPT et REJECT.

Calcul d'une MT

On définit une suite de configuration $C_i(x)$ par :

- $C_0(x)$ est la configuration initiale avec x sur la bande
- $C_i(x)$ est obtenue de $C_{i-1}(x)$ par transition

Une suite de configurations est finie et termine par une configuration contenant ACCEPT ou REJECT, ou infinie et ne contient pas ACCEPT ou REJECT.

La suite de configurations représente le calcul de la machine de Turing et donc sa taille correspond au *temps de calcul*.

Configuration finie

Lemma

Soit M une machine de Turing, alors si $C_i(x) = (B, pos, q)$, on a pour tous les $j > \max(i, |x|), B_j = \square$.

Ce lemme montre qu'en partant d'une configuration initiale finie, on obtient que des configurations finies. Cela justifie la représentation de la bande par *un mot fini*.

Langage reconnu

Attention, on se restreint aux machines de Turing qui terminent quelle que soit leur entrée : les suites $C_i(x)$ doivent être finies, c'est à dire terminer par ACCEPT our REJECT.
On note $L(M)$ le langage reconnu par une MT M.

$$L(M) = \{x \mid \text{La suite des configuration est finie et de dernier élément } (B, pos, ACCEPT)\}$$

Exemple d'une MT

Donnons ici une machine qui accepte les mots finissant par a :

- $\Sigma = \{a, b\}$
- $\Gamma = \{a, b, \square\}$
- $Q = \{q_0, q_a, q_b, ACCEPT, REJECT\}$
- $\delta(q_0, a) = \delta(q_a, a) = \delta(q_b, a) = (q_a, a, R)$
  $\delta(q_0, b) = \delta(q_a, b) = \delta(q_b, b) = (q_b, b, R)$
  $\delta(q_a, \square) = (ACCEPT, \square, S)$

La description d'une MT peut être *incomplète*, les transitions manquantes mènent à l'état REJECT.

Représentation graphique d'une MT

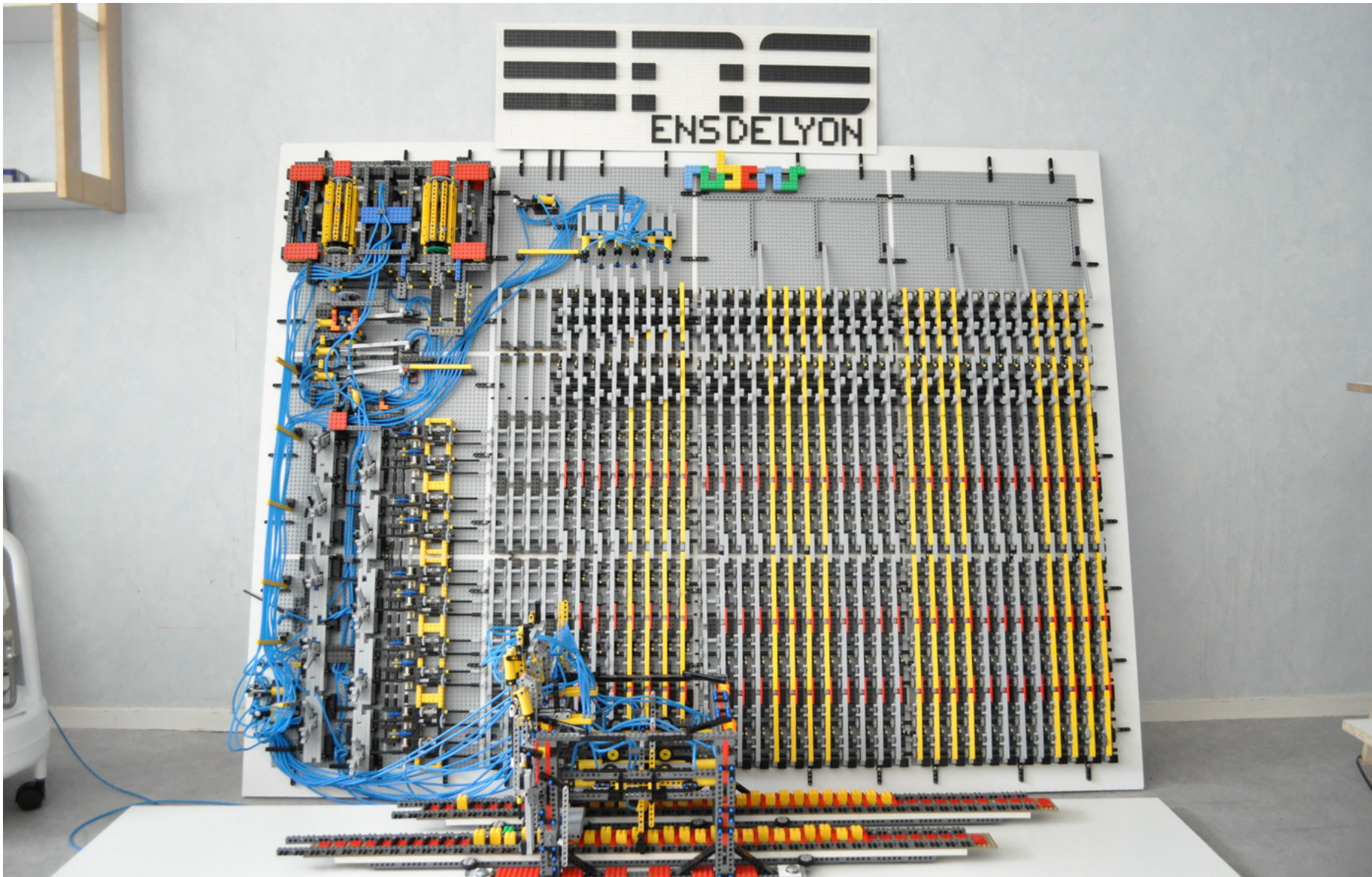
La représentation est similaire aux automates finis et automates à pile :

- Un sommet est représenté par un cercle contenant son nom.
- Une flèche entrante dénote le sommet initial
- Une transition $\delta(q_1, a) = (q_2, b, R)$ se note par un arc entre q_1 et q_2 avec comme label (a, b, R)

Dessiner la machine qui reconnaît les mots finissant par a.
Dessiner la suite de ses configurations en partant de l'entrée bba.

Une machine concrète

La machine de Turing existe, je l'ai rencontrée.



Des machines qui calculent

On peut définir f_M, la fonction partielle calculée par une machine de Turing M de la manière suivante :

- La suite de configurations $C_i(x)$ termine sur la configuration $(y, ACCEPT, pos)$, alors $f_M(x) = y$
- La suite de configurations $C_i(x)$ termine sur la configuration $(y, REJECT, pos)$, ou la suite des configurations est infinie, alors $f_M(x) = \perp$ (f n'est pas définie sur x).

Au lieu de reconnaître des langages, on calcule des fonctions comme en programmation et en algorithmique !

Exemples

Construire une machine de Turing qui :

- reconnaît les nombres pairs écrits en binaire
- ajoute 1 à un nombre écrit en unaire
- ajoute 1 à un nombre écrit en binaire

Réaliser ces exemples sur
`https://turingmachinesimulator.com/`.

Question Philosphique

Peut-on encore généraliser nos modèles de calcul pour calculer plus de fonctions ?

Il y a plein d'autres modèles de calcul :

- Machine de Turing à plusieurs rubans
- Machine de Turing non déterministe
- Machine RAM ou de von Neumann (assembleur/processeur)
- Pseudo-code (Programmation Impérative)
- Fonctions récursives (Programmation fonctionnelle)
- Le cerveau

Question Philosphique

Peut-on encore généraliser nos modèles de calcul pour calculer plus de fonctions ?

Il y a plein d'autres modèles de calcul :

- Machine de Turing à plusieurs rubans
- Machine de Turing non déterministe
- Machine RAM ou de von Neumann (assembleur/processeur)
- Pseudo-code (Programmation Impérative)
- Fonctions récursives (Programmation fonctionnelle)
- Le cerveau

Calculabilité

La calculabilité est le domaine de l'informatique qui s'intéresse au pouvoir de calcul des différents modèles de calcul.

Conjecture (Thèse de Church)

Toute fonction calculable par un dispositif physique est calculable par une machine de Turing.

Cette conjecture est renforcée par l'expérience, pour l'instant tous les modèles proposés ont le même pouvoir de calcul, c'est à dire qu'ils permettent de reconnaître les mêmes langages.

Calculabilité

La calculabilité est le domaine de l'informatique qui s'intéresse au pouvoir de calcul des différents modèles de calcul.

Conjecture (Thèse de Church)

Toute fonction calculable par un dispositif physique est calculable par une machine de Turing.

Cette conjecture est renforcée par l'expérience, pour l'instant tous les modèles proposés ont le même pouvoir de calcul, c'est à dire qu'ils permettent de reconnaître les mêmes langages.

Les machines à k rubans

On généralise les machine de Turing en leur permettant d'utiliser k rubans avec une tête de lecture sur chaque ruban.

- Une configuration est donnée par $(B_1, pos_1, B_2, pos_2, \ldots, B_k, pos_k, q)$ avec B_i le contenu de la $i$ème bande et pos_i la position de la tête de lecture sur la $i$ème bande.
- La fonction de transition δ est définie de $\Gamma^k \times Q$ vers $\Gamma^k \times Q \times \{L, H, R\}^k$, c'est à dire qu'à chaque étape tous les symboles sous les têtes de lecture sont lus, modifiés et les têtes déplacées.

On peut également ajouter un ruban spécial destiné à contenir la sortie de la fonction calculée par la MT.

Example de machine à k rubans

Avec deux rubans, on va faire les machines suivantes :

- Écrit l'entrée à l'envers sur le second ruban.
- Avec une entrée $x\sharp y$ finit avec x sur le premier ruban et y sur le second.
- Teste si une entrée est un palindrome.

Réaliser ces exemples sur `https://turingmachinesimulator.com/`.

Équivalence de modèles de calcul

Deux modèles de calcul sont équivalents si ils reconnaissent les mêmes langages.

On dit que le modèle de calcul $\mathcal{M}_1$ simule le modèle de calcul $\mathcal{M}_2$ si pour tout $M \in \mathcal{M}_2$, il existe $M' \in \mathcal{M}_1$ tel que $L_M = L_{M'}$.

Équivalence de modèles de calcul

Deux modèles de calcul sont équivalents si ils reconnaissent les mêmes langages.

On dit que le modèle de calcul $\mathcal{M}_1$ simule le modèle de calcul $\mathcal{M}_2$ si pour tout $M \in \mathcal{M}_2$, il existe $M' \in \mathcal{M}_1$ tel que $L_M = L_{M'}$.

Pour montrer qu'un modèle simule un autre, généralement on représente un calcul de $M \in \mathcal{M}_2$, c'est à dire une suite finie de configurations, par une machine $M' \in \mathcal{M}_1$ qui reproduit cette suite de configurations.

Équivalence de modèles de calcul

Deux modèles de calcul sont équivalents si ils reconnaissent les mêmes langages.

On dit que le modèle de calcul $\mathcal{M}_1$ simule le modèle de calcul $\mathcal{M}_2$ si pour tout $M \in \mathcal{M}_2$, il existe $M' \in \mathcal{M}_1$ tel que $L_M = L_{M'}$.

Pour montrer qu'un modèle simule un autre, généralement on représente un calcul de $M \in \mathcal{M}_2$, c'est à dire une suite finie de configurations, par une machine $M' \in \mathcal{M}_1$ qui reproduit cette suite de configurations.

Variations sur un ruban

Théorème

Les machines de Turing à k rubans sont équivalentes aux machines à 1 ruban.

Idée de preuve :

On code les k rubans sur un seul ruban, k cases consécutives (de ik à $(i+1)k-1$) représentent les k cases en position i des k rubans. On enrichit également l'alphabet pour noter sur quelles cases sont les têtes de lecture. Chaque étape de simulation nécessite de balayer le ruban pour trouver les k têtes de lecture et la lettre qu'elles lisent puis de nouveau pour modifier cette lettre et la position de la tête.

Exemple

Alphabet d'une MT

Un codage est une fonction de Σ vers Σ' qui s'étend aux mots.
Par exemple on peut coder $\{a, b, c, d\}$ par $\{00, 01, 10, 11\}$.

Théorème

Les machines de Turing sur l'alphabet Γ sont équivalentes aux MT sur l'alphabet $\{0, 1\}$ (à codage près).

Idée de preuve :
Pour réduire un alphabet avec $\Sigma = 2^k$ symboles à l'alphabet $\{0, 1\}$, on numérote chaque symbole et on le remplace par son numéro en binaire.
On simule une étape de calcul d'une machine avec alphabet Σ par k étapes de calcul qui lisent k cases consécutives avant de changer d'état selon le résultat de la lecture.
Exemple

Machine de Turing non déterministe

On définit les MTs non déterministes exactement comme les automates non déterministes :

- La fonction de transition est remplacée par une relation de transition $\delta \subseteq \Gamma \times Q \times \Gamma \times Q \times \{L, H, R\}$.
- Le fonctionnement d'une MT non déterministe est représenté par une suite *d'ensemble de configurations*.
- On passe d'un ensemble de configurations à un autre en appliquant toutes les transitions possibles à toutes les configurations de l'ensemble.
- Une entrée est acceptée, s'il existe un état acceptant dans un ensemble de configuration de la suite que la machine définit.

Déterminisation des Machine de Turing non détrministes

Theorem

Les machines de Turing déterministes et non déterministes sont équivalentes.

Preuve en TD.

Encore améliorer la mémoire

Jusqu'à présent automate avec la mémoire suivante :

- pas de mémoire
- une pile
- une liste doublement chaînée (ou deux piles)

Peut-on utiliser un structure de données plus puissante ?

Mémoire organisée en tableau : chaque case est accessible directement, si on a son numéro.

C'est ce que modélise les machine RAM, le modèle le plus proche des ordinateurs modernes avec architecture de Von Neumann. Similaire à un langage assembleur.

Encore améliorer la mémoire

Jusqu'à présent automate avec la mémoire suivante :

- pas de mémoire
- une pile
- une liste doublement chaînée (ou deux piles)

Peut-on utiliser un structure de données plus puissante ?

Mémoire organisée en tableau : chaque case est accessible directement, si on a son numéro.

C'est ce que modélise les machine RAM, le modèle le plus proche des ordinateurs modernes avec architecture de Von Neumann. Similaire à un langage assembleur.

Définition des machines RAM

Une machine RAM possède une suite infinie de registres $r1, r2, \ldots$ qui contiennent des entiers. Elle est définie par une suite d'instructions :

- arithmétiques : ADD(r1,r2,r3), SUB(r1,r2,r3), MULT(r1,r2,r3), DIV(r1,r2,r3) (les deux opérandes sont dans r1, r2 et le résultat dans r3)
- contrôle : JUMP(z), JE(r1,r2,r3) : si la valeur de r1 est égale à r2 sauter de r3 instructions.

Les arguments peuvent être :

- des constantes, par exemple ADD(2,3,r3).
- des registres, par exemple JE(r1,r2,r3).
- des références, on utilise de l'indirection, ADD([r1],r2,r3) avec [r1] le registre dont le numéro est stocké dans r1.

Un exemple de programme

Compter la taille d'un nombre en base 2 donné dans le registre r_1 :

- ADD(0,0,r2) // compteur de taille initialisé à 0
- ADD(r2,1,r2)// compteur de taille incrémenté de 1
- DIV(r1,2,r1)// divise l'entrée par 2
- JE(r1,0,1) // détecte la fin du programme
- JUMP(-3) // recommence la boucle 3 lignes avant
- //fin du programme le résultat se trouve dans r2

Allumons Human Ressource Machine . . .

Un exemple de programme

Compter la taille d'un nombre en base 2 donné dans le registre r_1 :

- ADD(0,0,r2) // compteur de taille initialisé à 0
- ADD(r2,1,r2)// compteur de taille incrémenté de 1
- DIV(r1,2,r1)// divise l'entrée par 2
- JE(r1,0,1) // détecte la fin du programme
- JUMP(-3) // recommence la boucle 3 lignes avant
- //fin du programme le résultat se trouve dans r2

Allumons Human Ressource Machine ...

Équivalence RAM et machine de Turing

Theorem

Les machines RAM sont équivalentes aux machines de Turing.

Idée de preuve :
On simule une machine de Turing par une machine RAM. On simule le ruban par une suite de registres. La position de la tête de lecture est conservé dans un registre spécifique, ainsi que le numéro de l'état courant. Le programme de la RAM est une grande énumération de cas, décrivant la fonction de transition de la MT.

Divertissement mathématique

Une autre forme d'automate : le jeu de la vie est un automate cellulaire. Il a été inventé par John Conway en 1970.

Traditionnellement le jeu se joue sur une grille infinie.
Éventuellement la grille peut être finie et toroïdale.

Divertissement mathématique

Une autre forme d'automate : le jeu de la vie est un automate cellulaire. Il a été inventé par John Conway en 1970.

Traditionnellement le jeu se joue sur une grille infinie.
Éventuellement la grille peut être finie et toroïdale.

On appelle chaque élément de la grille une cellule. Une cellule a deux états possibles : vivante ou morte.

Divertissement mathématique

Une autre forme d'automate : le jeu de la vie est un automate cellulaire. Il a été inventé par John Conway en 1970.

Traditionnellement le jeu se joue sur une grille infinie. Éventuellement la grille peut être finie et toroïdale.

On appelle chaque élément de la grille une cellule. Une cellule a deux états possibles : vivante ou morte.

Le "joueur" créé un état initial en fixant un nombre fini de cellules à l'état vivant et les règles du jeu permettent de faire évoluer cet état dans le temps.

Divertissement mathématique

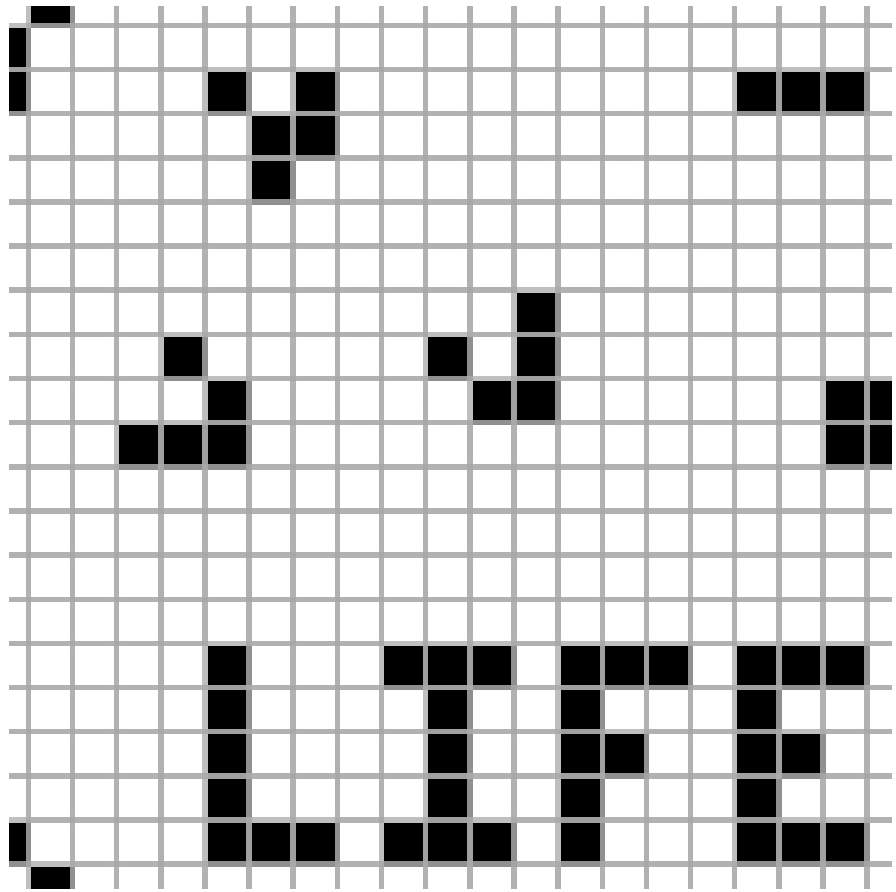
Une autre forme d'automate : le jeu de la vie est un automate cellulaire. Il a été inventé par John Conway en 1970.

Traditionnellement le jeu se joue sur une grille infinie. Éventuellement la grille peut être finie et toroïdale.

On appelle chaque élément de la grille une cellule. Une cellule a deux états possibles : vivante ou morte.

Le "joueur" créé un état initial en fixant un nombre fini de cellules à l'état vivant et les règles du jeu permettent de faire évoluer cet état dans le temps.

Exemple de grille



Les règles du jeu

À chaque pas de temps on applique ces règles sur toutes les cellules simultanément.

La naissance et la mort

1. Une cellule vivante entourée de 2 ou 3 cellules vivantes reste vivante.
2. Une cellule morte entourée de trois cellules vivantes devient vivante.
3. Dans les autres cas la cellule reste ou devient morte.

Exemple sur Golly.

Les règles du jeu

À chaque pas de temps on applique ces règles sur toutes les cellules simultanément.

La naissance et la mort

1. Une cellule vivante entourée de 2 ou 3 cellules vivantes reste vivante.
2. Une cellule morte entourée de trois cellules vivantes devient vivante.
3. Dans les autres cas la cellule reste ou devient morte.

Exemple sur Golly.

Langage reconnu par un automate cellulaire

Comment définir le langage reconnu par un automate cellulaire ?

- Les entrées qui donnent une grille vide.
- Les entrées qui donnent une configuration stable ou périodique.
- Une fonction des entrées vers les configurations stables.
- L'existence d'une configuration avec une cellule dans un état spécial acceptant.

Langage reconnu par un automate cellulaire

Comment définir le langage reconnu par un automate cellulaire ?

- Les entrées qui donnent une grille vide.
- Les entrées qui donnent une configuration stable ou périodique.
- Une fonction des entrées vers les configurations stables.
- L'existence d'une configuration avec une cellule dans un état spécial acceptant.

Theorem

Les automates cellulaires et les machines de Turing sont équivalent.

Langage reconnu par un automate cellulaire

Comment définir le langage reconnu par un automate cellulaire ?

- Les entrées qui donnent une grille vide.
- Les entrées qui donnent une configuration stable ou périodique.
- Une fonction des entrées vers les configurations stables.
- L'existence d'une configuration avec une cellule dans un état spécial acceptant.

Theorem

Les automates cellulaires et les machines de Turing sont équivalent.

À voir attentivement pour la prochaine fois !
https://www.youtube.com/watch?v=1YrbUBSo4Os&list=PLxzM9a5lhAuk-LctI0hTARO2yIMJtvPgJ&index=8&t=604s
Autres ressources, si vous êtes curieux :

- MT rapide
  https://www.youtube.com/watch?v=P66h8D5Lkwk
- MT plus avancée
  https://www.youtube.com/watch?v=zlPWnNxBUCY
- MT physique
  https://www.dailymotion.com/video/xrn0yi
- Jeu de la vie https://www.youtube.com/watch?v=S-W0NX97DB0&list=PLxzM9a5lhAuk-LctI0hTARO2yIMJtvPgJ&index=17