913 : Machines de Turing, Applications

I - Définitions générales

1) Machines de Turing [CAR, p 141-145]

Déf 1 : Une machine de Turing est un septuplet $(Q, \Sigma, \Gamma, \delta, q_0, F, \#)$:
- Q est un ensemble fini d'état
- Σ est un alphabet d'entrée.
- Γ est l'alphabet de travail. $\Sigma \subset \Gamma$: $\# \in \Gamma \setminus \Sigma$ est un symbole spécial dit blanc
- δ est une application partielle de $Q \times \Gamma$ dans $Q \times \Gamma \times \{\leftarrow, \rightarrow\}$
- $q_0 \in Q$ est l'état initial
- $F \subset Q$ est un ensemble d'état finaux

Ex 2 : Voir annexe 1.

Déf 3 : Le ruban d'une machine de Turing est un mot fini de Γ suivi d'une infinité de symbole blanc.

Déf 4 : La configuration d'une MT est la donnée de son état courant, du contenu du ruban et de la position sur le ruban.
On note uqv la configuration d'état q, de ruban uv et positionné sur la première lettre de v.

Ex 5 : Voir annexe 1.

Déf 6 : Une étape de calcul est une paire de configuration (C, C'), notée $C \rightsquigarrow C'$, vérifiant :
- Soit $C = ucpav$ et $C' = uqcbv$ ~~et~~ $\delta(p,a) = (q, b, \leftarrow)$.
- $C = upav$ et $C' = ubqv$ et $\delta(p,a) = (q, b, \rightarrow)$.

Déf 7 : Un calcul est une suite de configuration $C_0 \rightsquigarrow \dots \rightsquigarrow C_k$
Ce calcul est dit acceptant si $C_0 = q_0 w$ avec $w \in \Sigma^*$
et $C_k = uqv$ avec $u,v \in \Sigma^*$ et $q \in Q$

~~Déf~~ 8 : La machine M accepte le mot w si il existe un calcul acceptant, ayant w inscrit initialement sur le ruban.
On note $\mathcal{L}(M)$ le langage des mots acceptés par M.

Ex 9 : La machine en annexe 1 accepte les mots de $\{a,b\}^*$ ayant autant de a que de b.

Déf 10 : Un calcul est dit infini si il y a une suite infinie de configuration $(C_k)_{k \in \mathbb{N}}$ formant des étapes de calcul (C_k, C_{k+1}).

2) Définitions équivalentes

- **Machine à k ruban** : [CAR, p 150]

Une telle machine travaille sur k ruban. Donc sa fonction de transition va de $Q \times \Gamma^k$ dans $Q \times \Gamma^k \times \{\leftarrow, \downarrow, \rightarrow\}^k$. A chaque étape, on lit et réécrit sur chaque ruban et décide de bouger à gauche ou à droite ou rester immobile

Prop 11 : Les MT à un ruban reconnaissent les même langages que celles à k ruban

- **Machine non-déterministe** : [CAR, p 152]

Une machine non-déterministe, à partir de son état et du caractère de la bande, peut choisir parmis un ensemble de réponse.
Ainsi, δ va de $Q \times \Gamma$ dans $\mathcal{P}(Q \times \Gamma \times \{\leftarrow, \rightarrow\})$. La machine accepte un mot si un de ses calculs possibles est acceptant.

Prop 12 : Les MT déterministes et non-déterministes reconnaissent les mêmes langage.

- **Thèse de Church-Turing** :

Les langages reconnus par une procédure effective sont ceux décidés par machine de Turing.

3) Codage

Pas obligatoire

sans Réf

Les machines de Turing travaillent sur des mots. Si l'on souhaite travailler sur un autre objet, il faut le transformer en mot. On parle alors de codage. Ainsi, toutes les propriétés que l'on identifiera à certain problèmes ne valent que pour le codage sur lequel on travaille.

Par exemple, un problème sur les entiers sera différent si il est codé en unaire ou en binaire.
Si il n'y a pas d'ambiguité, on note $\langle x \rangle$ le codage de x.

- Machine de Turing :
Il y a un nombre dénombrable de MT. On peut alors définir un codage des MT.

Ex 13 : On peut définir une machine, dite universelle, qui, étant donné le codage d'une machine M et un mot w, simule l'exécution de M sur w.
Son langage accepté, noté LU est égal à $\{\langle M, w\rangle \mid w \in \mathcal{L}(M)\}$

II - Calculabilité et décidabilité

1) Fonction calculable. [CAR, p 160]

Déf 14 : Une fonction $f : \Sigma^* \to \Gamma^*$ est dite calculable si il existe une MT qui, à partir de w en entrée, écrit $f(w)$ sur le ruban pour tout $w \in \Sigma^*$.
De même, en fixant un codage des entiers, on définit les fonctions de $\mathbb{N}$ dans $\mathbb{N}$ calculable.

Prop 15 : Les fonctions de $\mathbb{N}$ dans $\mathbb{N}$ calculables sont exactement les fonctions ~~μ~~ récursives.

Ex 16 : $n \mapsto n^2$ est calculable.

Déf 17 : Le castor affairé est une fonction $\Sigma : \mathbb{N} \to \mathbb{N}$; $\Sigma(n)$ défini par le nombre de 1 maximal écrit sur le ruban par une machine à n état et un autre final, partant d'un ruban vide et qui s'arrête. [D]

écrire plus formellement

Prop 18 : Soit $f : \mathbb{N} \to \mathbb{N}$ calculable
~~A~~ A partir d'un certain rang, $\Sigma(n) > f(n)$ DEV

Corollaire 19 : Σ n'est pas calculable. [D]

2) Langages [CAR, p 155-160]

Déf 20 : Soit RE l'ensemble des langages acceptés par une MT.

Ex 21 : $LU \in RE$ et $\overline{LU} \notin RE$

Déf 22 : Un énumérateur est une machine dont le calcul est possiblement infinie, et qui écrit des mots de Σ^* séparé par un séparateur spécial $\$$.

Prop 23 : $L \in RE$ si il existe un énumérateur listant tous les mots de L.

Déf 24 : Soit R l'ensemble des langage acceptés par des machines n'admettant aucun calcul infini.
On dit alors que la machine décide le langage.

Prop 25 : $R \subset RE$

Prop 26 : $L \in RE$ et $\overline{L} \in RE$ impliquent $L \in R$.

Prop 27 : Les langages algébriques sont dans R.

Prop 28 : R et RE sont stable par union, intersection, concaténation et étoile de Kleene.
sans ref.

3) Problèmes. [CAR, p161-168]

Déf 29 : Un problème est indécidable si le langage des ses instance positive après codage n'appartient pas à R.

Déf 30 : Une instance de PCP est un ensemble $\{(U_i,V_i) | 1\leq i\leq k\}$
Cette instance est positive ssi il existe $i_1, \dots, i_m$ tq

$$U_{i_1} \dots U_{i_m} = V_{i_1} \dots V_{i_m}.$$

Prop 31 : PCP est indécidable.

Déf 32 : Soient A et B deux problèmes d'alphabets Σ_A et Σ_B.
Une réduction de A à B, notée $A < B$, est une fonction $f : \Sigma_A^* \rightarrow \Sigma_B^*$ calculable telle que $w \in L_A \Leftrightarrow f(w) \in L_B$.

Prop 33 : Si $A < B$ et A indécidable. Alors B indécidable.

Corollaire 34 : Soient deux grammaires algébriques
La vacuité de leur intersection est indécidable.

Prop 35 : Théorème de Rice
Soit P une propriété sur RE non triviale.
Etant donné $\langle M \rangle$ le codage d'une MT,
le problème de savoir si $\mathcal{L}(M)$ vérifie P est indécidable.

DEV

creux

casques très symboliques → chercher

Corollaire 36 : Si l'on choisit P=R dans le théorème de Rice on a l'indécidabilité du problème de l'arrêt

questions : - quelles sont les casques ?
- quelles sont les solutions ?

Le thm de l'arrêt n'est pas introduit ?!

III - Complexité [CAR] p 196-197

Déf 37 : Si une machine de Turing décide une instance de taille n en un nombre d'étape de calcul inférieur à P(n) avec P un polynôme, alors le problème appartient à la classe P

Ex 38 : Trouver une arbre couvrant minimal est dans P.

Rq 39 : On étudie P car c'est une classe de problème "rapide" et assez stable par changement d'architecture

Déf 40 : Une définition analogue à P avec des machines non-déterministe définit la classe NP.

Ex 41 : SAT ∈ NP

[CAR] p118-222

Déf 42 : De manière analogue à la complexité en temps, on définit P-SPACE et NP-SPACE pour la complexité en espace

Ex 43 : Savoir si une formule propositionnelle est auto-duale (c'est-à-dire si $\varphi(\neg x_1, \dots, \neg x_n) \equiv \neg\varphi(x_1, \dots, x_n)$) n'est pas NP, mais est NP-SPACE.

Prop 44 : • P ⊂ NP et P-SPACE ⊂ NP-SPACE
• P ⊂ P-SPACE et NP ⊂ NP-SPACE

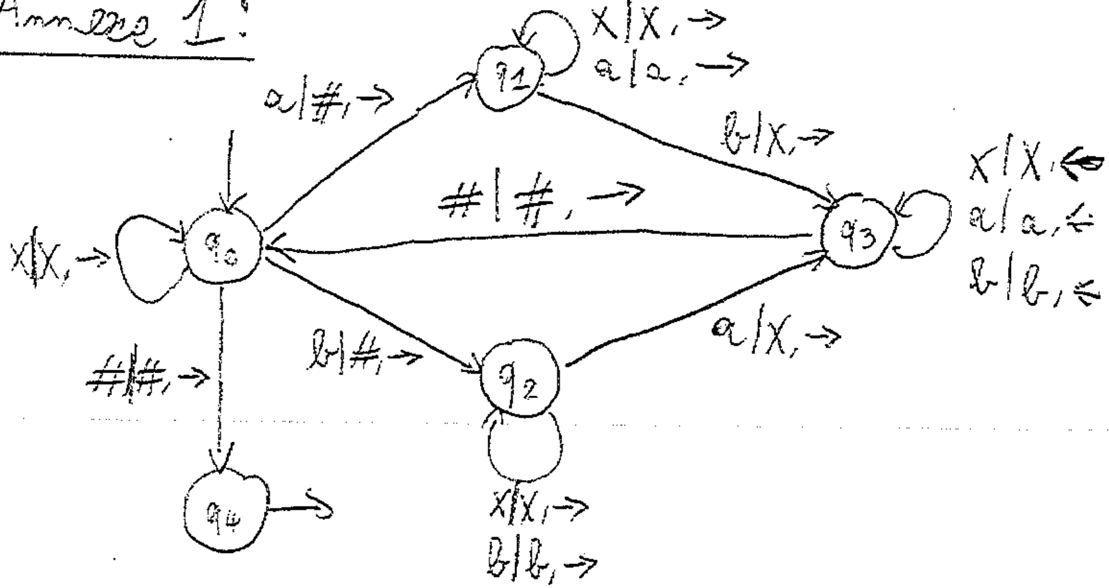
Prop 45 : Théorème de Savitch
P-SPACE = NP-SPACE.

Savitch c'est pas ça c'est un cor.

accepté / décidé ?

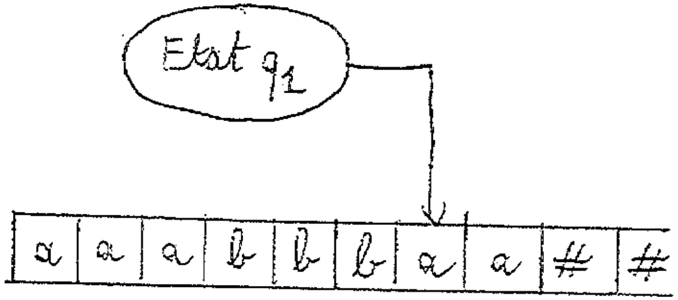
accepté : qd c'est bon, on s'arrête et rpd oui
décidé : ∃ la machine qui s'arrête tt le temps

Annexe 1 :



Exemple de machine de Turing

Annexe 2 :



Exemple de configuration : aaabbb q_1 aa.

Questions :

- (MT, λ-calcul, ...?)
- Machine universelle ⇝ quel intérêt ?

Dvpts possibles :

↳ Cook
↳ PCP (dur)
↳ Applications de PCP (langages algébriques)
↳ Savitch
↳ Calculable par MT ⇒ récursif.
↳ Indécidabilité de la terminaison d'un syst. de réécriture
↳ RE eng. par gramm. générale.

Référence :

- Olivier Carton. Langages formels, Calculabilité et complexité
- Pierre Wolper. Introduction à la calculabilité (un peu)
- Dehornoy, Mathématiques de l'informatique

Le castor affairé

Définition — On appelle "castor" une machine de Turing M telle que :
- Le ruban, infini à gauche et à droite, utilise l'alphabet $\{1, \sqcup\}$.
- M possède un état d'arrêt, qu'elle atteint si on la lance avec l'entrée vide.

On note $\Phi(M)$ le nombre de 1 sur le ruban après le calcul sur entrée vide.

On définit, pour $N \in \mathbb{N}^*$, $U(N)$ comme la borne supérieure des $\Phi(M)$, où M est un castor à N états.

Ce nombre existe bien puisqu'il n'y a qu'un nombre fini de machines de Turing à N états d'alphabet $\{1, \sqcup\}$.

Proposition La fonction U est strictement croissante.

Preuve Soit M un castor à N états tel que $\Phi(M) = U(N)$.

On construit à partir de M un castor M' à $N+1$ états en ajoutant les instructions :

- $(1, N) \mapsto (1, \rightarrow, N)$ — si on lit un 1 à l'état N, on déplace la tête de ruban vers la droite sans rien changer.
- $(\sqcup, N) \mapsto (1, \rightarrow, N+1)$ — si on lit un $\sqcup$, on le remplace par un 1 et on s'arrête.

N étant ici l'état d'arrêt de M.

On voit alors que $\Phi(M') = U(N) + 1$.

D'où $U(N+1) > U(N)$. $\square$

Soit $F : \mathbb{N}^* \to \mathbb{N}^*$, on note $[F]_1 : \begin{cases} 1^* \longrightarrow 1^* \\ 1^n \longmapsto 1^{F(n)} \end{cases}$ la fonction unaire associée à F.

Lemme F est calculable ssi $[F]_1$ est calculable.) ADMIS c'est classique.

Preuve ADMIS.

Mais pour en donner une idée

- On peut simuler une machine sur l'alphabet $\{0, 1, \sqcup\}$ (et même sur n'importe quel alphabet Σ) par une autre sur $\{1, \sqcup\}$ en codant chaque lettre par un bloc de 2 : par exemple $\begin{pmatrix} 1 \text{ par } 11 \\ 0 \text{ par } \sqcup 1 \\ \sqcup \text{ par } \sqcup\sqcup \end{pmatrix}$.

• Si on peut calculer F avec une machine à k rubans, il existe une machine à un seul ruban sur le même alphabet qui calcule F.

~~• On peut supposer~~

• On peut passer d'un nombre en base 2 à une base unaire avec une machine à 3 rubans, et de la base unaire à la base 2 par divisions euclidiennes successives sur une machine à 3 rubans aussi.

⇝ Donc il revient au même de savoir calculer F en base deux et en base unaire.

Théorème | Si $F : \mathbb{N}^* \to \mathbb{N}^*$ est calculable, il existe un rang N tel que :

$$F(n) < U(n) \quad \text{pour tout } n \geq N.$$

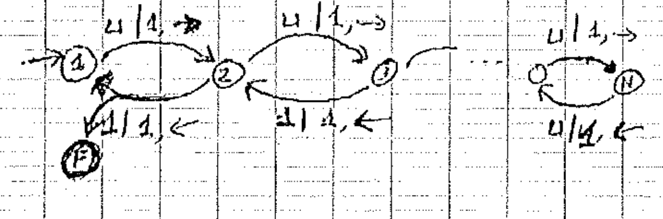
Et donc U n'est pas calculable.

Preuve Quitte à remplacer F par $n \mapsto \sum_{i=1}^{n} F(i)$, on peut supposer F croissante.

De plus, $n \mapsto n^2$ est μ-récursive, donc calculable, de même pour $G : n \mapsto F(n^2)$.

⇝ Soit M une machine de Turing sur l'alphabet $\{1, \sqcup\}$ calculant $[G]_1$. On note K le nombre d'états de M.

⇝ Soit M_N, pour un entier N fixé, une machine de Turing à $N+1$ états passant de la configuration vide au mot $[N]_1$, où le premier caractère est accessible.



⇝ En enchaînant M à M_N, on obtient une machine à $N+K+1$ états qui passe de la configuration vide à $[F(N^2)]_1$.

D'où, par définition de U,

$$F(N^2) \leq U(N+K+1).$$

Pour $N > K$ et $K > 3$, on a ~~donc~~ $(N-1)^2 > K(K-3) + N + 1 > N+K+1$

Et donc

$$F(N^2) < U((N-1)^2)$$

Par croissance de U et F, si $M > K^2$, et N est l'unique entier tel que $(N-1)^2 < M \leq N^2$,

On a $F(M) \leq F(N^2) < U((N-1)^2) < U(M)$ □