

Calculabilité et complexité

Cours n° 1

Nicolas (Miki) Hermann

LIX, École Polytechnique

hermann@lix.polytechnique.fr

Machine de Turing **déterministe** $M = (Q, \Sigma, \Gamma, \delta, \text{START}, \text{HALT}, \text{YES}, \text{NO})$

- Q = ensemble fini d'états
- Σ = alphabet (fini) d'entrée
- $\Gamma = \Sigma \cup \{\triangleright, \sqcup\}$ = alphabet de travail
 - \triangleright = limite gauche
 - \sqcup = espace
- $\delta: Q \times (\Sigma \cup \{\sqcup\}) \rightarrow (Q \cup \{\text{HALT}, \text{YES}, \text{NO}\}) \times \Sigma \times \{-1, 0, 1\}$
= transition
- START = départ ($\text{START} \in Q$)
- HALT = arrêt, YES = acceptation, NO = rejet
($\text{HALT}, \text{YES}, \text{NO} \notin Q$)

$M = (Q, \Sigma, \Gamma, \delta, \text{START}, \text{HALT}, \text{YES}, \text{NO})$

Configuration : $(q, u, w) \in (Q \cup \{\text{HALT}, \text{YES}, \text{NO}\}) \times \Gamma^* \times \Gamma^*$

Config. acceptante : (YES, u, w) pour des mots $u, w \in \Gamma^*$

Pas de M : $(q, u, v) \vdash_M (q', u', v')$ pour les configs (q, u, v) et (q', u', v') si $\delta(q, a) = (q', b, D)$ où $a = \text{fst}(v)$ tel que $u' = ub$ et $v = av'$ si $D = 1$, $u' = u$ et $\text{fst}(v') = b$ si $D = 0$, $u = u' \text{lst}(u)$ et $v' = \text{lst}(u) b \text{follow}(v)$ si $D = -1$

Calcul de M : $(q, u, v) \vdash_M^* (q', u', v')$

Langage accepté : $L(M) = \{x \in \Sigma^* \mid (\text{START}, \triangleright, x) \vdash_M^* (\text{YES}, u, v)\}$

Pour une machine M et un mot $x \in \Sigma^*$, $M(x)$ est le **résultat** du calcul de M sur x

- $M(x) = \text{YES}$ si M **accepte** x ($x \in L(M)$)
- $M(x) = \text{NO}$ si M **rejette** x ($x \notin L(M)$)
- $M(x) = \nearrow$ si M **ne s'arrête pas** sur x

Si pour un langage L il existe une machine de Turing M , telle que

- si $x \in L$ alors $M(x) = \text{YES}$
- si $x \notin L$ alors $M(x) = \text{NO}$

nous disons que M **décide** le langage $L \subseteq \Sigma^*$. Un langage $L \subseteq \Sigma^*$ décidé par une machine de Turing M s'appelle **récurisif**.

Machine de Turing déterministe **multi-ruban** (k rubans, $k \in \mathbb{N}$)

$M = (Q, \Sigma, \Gamma, \delta, \text{START}, \text{HALT}, \text{YES}, \text{NO})$

- ...

- $\delta: Q \times (\Sigma \cup \{\sqcup\})^k \rightarrow (Q \cup \{\text{HALT}, \text{YES}, \text{NO}\}) \times (\Sigma \times \{-1, 0, 1\})^k$

Machine de Turing avec k rubans = notre **modèle du calcul**

Resources : le **temps** et l'**espace**

Si pour une machine de Turing M avec k rubans et l'entrée x nous avons

$$(\text{START}, \triangleright, x, (\triangleright, \epsilon)^{k-1}) \vdash_M^t (h, u_1, w_1, \dots, u_k, w_k)$$

pour $h \in \{\text{YES}, \text{NO}\}$ alors le **temps** de travail de M sur x est t . Si $M(x) = \nearrow$ alors le temps de travail de M sur x est infini.

Nous disons que la machine M **travaille en temps** $f(n)$ si pour chaque mot x le temps de travail de M sur x est au plus $f(|x|)$. La fonction $f(n)$ est la **borne temporelle** de M .

Complexité temporelle

Supposons que le langage $L \subseteq \Sigma^*$ est décidé par une machine de Turing M avec k rubans travaillant en temps $f(n)$. Nous allons alors écrire

$$L \in \text{DTIME}(f(n))$$

Alors $\text{DTIME}(f(x))$ est un ensemble de langages. Il contient exactement les langages décidables par une machine de Turing multi-ruban travaillant en temps $f(x)$.

$\text{DTIME}(f(n))$ est une **classe de complexité**

Pour une machine M avec k rubans et un mot x , si

$$(\text{START}, \triangleright, x, (\triangleright, \epsilon)^{k-1}) \vdash_M^* (\text{HALT}, u_1, w_1, \dots, u_k, w_k)$$

alors $M(x) = u_k w_k$ est le **résultat** du travail de M sur x

Ruban n° 1 = ruban d'**entrée**

Ruban n° k = ruban de **sortie**

Rubans 2, \dots , $k-1$ = rubans de **travail**

Comparaison de puissance des machines de Turing multi-ruban

Théorème

Pour chaque machine de Turing M avec k rubans travaillant en temps $f(n)$ il existe une machine de Turing M' travaillant en temps $O(f(n)^2)$, telle que pour chaque mot x nous avons $M(x) = M'(x)$.

Idée de la preuve

Les k rubans de la machine M sont considérés comme une table T . Chaque colonne de la table T est un symbole de la machine M' . La table T est le seul ruban de la machine M' . Il faut ajouter les têtes de M dans les colonnes de M' en tant que nouveaux symboles. \square

Conclusion : Il est inutile de distinguer les machines de Turing à un ou plusieurs rubans.

Théorème

Soit $L \in \text{DTIME}(f(n))$. Alors $L \in \text{DTIME}(f'(n))$ où $f'(n) = cf(n) + n + 2$, pour chaque $c > 0$.

Idée de la preuve

Preuve pour $c = 1/2$.

La machine M acceptant L travaille en temps $f(n)$. Soit $x = a_1 \cdots a_n$ le mot d'entrée de M . La machine M' réécrit x en

$$x' = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \cdots \begin{bmatrix} a_{n-1} \\ a_n \end{bmatrix}$$

puis simule deux pas de M par un pas de M' . □

Conclusion : Les constantes multiplicatives sont inutiles.

$L \in \text{DTIME}(f(n))$ et $L \in \text{DTIME}(O(f(n)))$ signifient la même chose.

Si pour une machine de Turing M avec k rubans et l'entrée x nous avons

$$(\text{START}, \triangleright, x, (\triangleright, \epsilon)^{k-1}) \vdash_M^* (h, u_1, w_1, \dots, u_k, w_k)$$

pour $h \in \{\text{HALT}, \text{YES}, \text{NO}\}$ alors l'**espace** de travail de M sur x est

$$\sum_{i=2}^{k-1} u_i w_i$$

(car le ruban est l'entrée et le ruban k est la sortie).

Nous disons que la machine M **travaille en espace** $f(n)$ si pour chaque mot x l'espace de travail de M sur x est au plus $f(|x|)$. La fonction $f(n)$ est la **borne spatiale** de M .

Supposons que le langage $L \subseteq \Sigma^*$ est décidé par une machine de Turing M avec k rubans travaillant en espace $f(n)$. Nous allons alors écrire

$$L \in \text{DSPACE}(f(n))$$

Alors $\text{DSPACE}(f(x))$ est un ensemble de langages. Il contient exactement les langages décidables par une machine de Turing multi-ruban travaillant en espace $f(x)$.

$\text{DSPACE}(f(n))$ est une **classe de complexité**

Théorème

Soit $L \in \text{DSPACE}(f(n))$. Alors $L \in \text{DSPACE}(2 + cf(n))$ pour chaque $c > 0$.

Idée de la preuve

La même idée que pour l'accélération linéaire. □

Conclusion : Les constantes multiplicatives sont inutiles.

$L \in \text{DSPACE}(f(n))$ et $L \in \text{DSPACE}(O(f(n)))$ signifient la même chose.

Machine de Turing **non-déterministe**

$M = (Q, \Sigma, \Gamma, \Delta, \text{START}, \text{HALT}, \text{YES}, \text{NO})$

- ...
- Δ est une **relation** :

$$\Delta \subseteq [Q \times (\Sigma \cup \{\sqcup\})] \times [(Q \cup \{\text{HALT}, \text{YES}, \text{NO}\}) \times \Sigma \times \{-1, 0, 1\}]$$

Pour chaque pair état-symbole il peut y avoir une, plusieurs ou aucune possibilité de transition, correspondant à un **choix non-déterministe**.

Pas de M : ... $(q', b, D) \in \Delta(q, a)$

Toutes les notions connues (acceptation, rejet, décision, accélération linéaire, contraction linéaire, équivalence entre les machines multi-ruban) s'étendent aux machines de Turing non-déterministes **sauf** la définition du temps et de l'espace.

Temps non-déterministe

Supposons que le langage $L \subseteq \Sigma^*$ est décidé par une machine de Turing non-déterministe M travaillant en temps $f(n)$. Nous allons alors écrire

$$L \in \text{NTIME}(f(n))$$

$\text{NTIME}(f(x))$ est un ensemble de langages. Il contient exactement les langages décidables par une machine de Turing non-déterministe travaillant en temps $f(x)$.

$\text{NTIME}(f(n))$ est une **classe de complexité**

Supposons que le langage $L \subseteq \Sigma^*$ est décidé par une machine de Turing non-déterministe M travaillant en espace $f(n)$. Nous allons alors écrire

$$L \in \text{NSPACE}(f(n))$$

$\text{NSPACE}(f(x))$ est un ensemble de langages. Il contient exactement les langages décidables par une machine de Turing non-déterministe travaillant en espace $f(x)$.

$\text{NSPACE}(f(n))$ est une **classe de complexité**

Reconnaissance égale

Les machines non-déterministes reconnaissent-elles plus de langages que les machines déterministes ? **NON !**

Théorème

Pour chaque machine de Turing **non-déterministe** $M = (Q, \Sigma, \Gamma, \Delta, \text{START}, \text{HALT}, \text{YES}, \text{NO})$ il existe une machine de Turing **déterministe** $M' = (Q', \Sigma, \Gamma, \delta', \text{START}, \text{HALT}', \text{YES}', \text{NO}')$ telle que $L(M) = L(M')$

Idée de la preuve

La preuve est constructive. On prend $Q' = 2^Q$ et on simule tous les pas de M par la machine M' qui construit l'arbre de calcul par la recherche en *largeur*. □

Calcul sans borne = indécidabilité

Si on ne borne pas le calcul d'une machine de Turing par une borne temporelle ou spatiale, on encourt des graves difficultés.

Machine de Turing universelle

Il existe une machine de Turing **universelle** U qui prends la pair (M, x) et simule le travail de la machine M sur le mot x = modèle d'ordinateur

Notation : $M(x) = \nearrow$ si la machine M ne s'arrête pas (diverge) sur le mot x

Langage

$$H = \{(M, x) \mid M(x) \neq \nearrow\}$$

Langage H est **récursivement énumérable**, i.e., il existe une machine de Turing qui énumère les éléments (M, x) du langage H

Problème d'arrêt

Langage $H = \{(M, x) \mid M(x) \neq \nearrow\}$ nous aidera à déterminer la décidabilité du problème suivant :

Problème: ARRÊT

Entrée: Une machine de Turing M et un mot x .

Question: La machine de Turing M s'arrête-t-elle sur x ?

Problème d'arrêt

Théorème

Le problème d'arrêt n'est pas décidable.

On dit aussi que le langage H est **indécidable**.

Problème d'arrêt

Théorème

Le problème d'arrêt n'est pas décidable.

Preuve :

Par absurde. Supposons qu'il existe une machine de Turing M_H qui décide H , i.e., qui pour chaque mot x décide si $x \in H$ ou $x \notin H$. Soit D la machine de Turing définie ainsi :

$$D(M) = \text{if } M_H(M, M) == \text{YES then } \nearrow \text{ else YES fi}$$

Quid de $D(D)$?

- Si $D(D) = \nearrow$ alors M_H accepte (D, D) . Donc $(D, D) \in H$ ce qui implique $D(D) \neq \nearrow$. **Contradiction.**
- Si $D(D) \neq \nearrow$ alors M_H rejette (D, D) . Donc $(D, D) \notin H$ ce qui implique $D(D) = \nearrow$. **Contradiction.**



On dit aussi que le langage H est **indécidable**.

Problème d'arrêt (exercice)

Les langages suivants sont indécidables

$$M_1 = \{M \mid M \text{ s'arrête sur chaque entrée}\}$$

$$M_2 = \{(M, x) \mid \exists y M(x) = y\}$$

$$M_3 = \{(M, x) \mid \text{le calcul de } M \text{ sur } x \text{ passe par tous les états } Q\}$$

$$M_4 = \{(M, x, y) \mid M(x) = y\}$$

Problème d'arrêt (exercice)

Les langages suivants sont indécidables

$$M_1 = \{M \mid M \text{ s'arrête sur chaque entrée}\}$$

$$M_2 = \{(M, x) \mid \exists y M(x) = y\}$$

$$M_3 = \{(M, x) \mid \text{le calcul de } M \text{ sur } x \text{ passe par tous les états } Q\}$$

$$M_4 = \{(M, x, y) \mid M(x) = y\}$$

Pour M_1 ; Réduction du problème d'arrêt au problème de décider M_1 .
Etant donné (M, x) , nous construisons la machine de Turing M'
suivante :

$$M'(y) = \text{if } y == x \text{ then } M(x) \text{ else HALT fi}$$

M' s'arrête sur chaque entrée si et seulement si M s'arrête sur x

Construisez les machines de Turing qui décident les langages suivants :

$$M_1 = \{n^2 \mid n \in \mathbb{N}\}$$

$$M_2 = \{w \in \Sigma \mid w = w^R\} \quad \text{palindromes}$$

Prouvez la propriété suivante :

$$\forall k \in \mathbb{N} \exists n_0 \in \mathbb{N} \forall n > n_0 \quad n^k < 2^n$$

Calculs bornés en temps ou en espace

Nous pouvons imposer des **bornes temporelles** ou **spatiales** sur les calculs des machines de Turing déterministes ou non-déterministes.

Théorème

Supposons que le langage L est décidable par une machine de Turing **non-déterministe** N en temps $f(n)$. Alors, L est décidable par une machine de Turing **déterministe** M à 3 rubans en temps $O(c^{f(n)})$, où $c > 1$ est une constante qui dépend de N .

Autrement dit, nous avons la relation

$$\text{NTIME}(f(n)) \subseteq \bigcup_{c>1} \text{DTIME}(c^{f(n)})$$

Problème : Le **temps nécessaire** pour simuler N par M . Explosion exponentielle !

Remarque sur la fonction $f(n)$

Il faut que

- 1 f soit une fonction sur les entiers naturels, i.e., $f: \mathbb{N} \rightarrow \mathbb{N}$
- 2 f soit **non-décroissante**, i.e., $f(n) \leq f(n+1)$
- 3 f soit **effectivement calculable** (par une machine de Turing) pour chaque n . I.e., il existe une machine de Turing déterministe M qui sur le mot x s'arrête après $O(|x| + f(|x|))$ pas, utilise l'espace $O(f(|x|))$ et calcule $a^{f(|x|)}$ pour un symbole $a \in \Sigma$.

alors f est une **fonction de complexité**.

Classes de complexité

$$\mathbf{P} = \bigcup_{k=1}^{\infty} \text{DTIME}(n^k)$$

$$\mathbf{NP} = \bigcup_{k=1}^{\infty} \text{NTIME}(n^k)$$

$$\mathbf{PSPACE} = \bigcup_{k=1}^{\infty} \text{DSPACE}(n^k)$$

$$\mathbf{NPSPACE} = \bigcup_{k=1}^{\infty} \text{NSPACE}(n^k)$$

Classes de complexité

Montons d'une étage plus haut :

$$\text{EXP} = \bigcup_{k=1}^{\infty} \text{DTIME}(2^{n^k})$$

$$\text{NEXP} = \bigcup_{k=1}^{\infty} \text{NTIME}(2^{n^k})$$

$$\text{EXPSPACE} = \bigcup_{k=1}^{\infty} \text{DSPACE}(2^{n^k})$$

On peut continuer ainsi à l'infini avec 2EXP , 2NEXP , 2EXPSPACE , \dots , $k\text{EXP}$, $k\text{NEXP}$, $k\text{EXPSPACE}$ pour $k \in \mathbb{N}$, mais :

$$\text{ELEM} = \bigcup_{k=1}^{\infty} k\text{EXP}$$

$$\text{ELEM} \subsetneq \text{REC}$$

Plongeons-nous dans P :

$$L = \text{DSPACE}(\log n)$$

$$\text{NL} = \text{NSPACE}(\log n)$$

Plongeons-nous dans P :

$$L = \text{DSPACE}(\log n)$$

$$\text{NL} = \text{NSPACE}(\log n)$$

Question : Peut-on descendre en dessous de L ?

Plongeons-nous dans P :

$$\begin{aligned}L &= \text{DSPACE}(\log n) \\ \text{NL} &= \text{NSPACE}(\log n)\end{aligned}$$

Question : Peut-on descendre en dessous de L ?

Réponse 1 : **NON** avec se modèle de calcul.

Réponse 2 : **OUI** en utilisant le modèle de circuits.

Classes complémentives

Soit $L \subseteq \Sigma^*$ un langage. Le **langage complémentaire** de L est

$$\bar{L} = \Sigma^* \setminus L$$

Pour chaque classe de complexité C , nous avons

$$\text{co}C = \{\bar{L} \mid L \in C\}$$

Pour chaque classe C **déterministe** (temps ou espace)

$$\text{co}C = C$$

Si $L = L(M)$ pour une machine de Turing déterministe $M = (Q, \Sigma, \Gamma, \delta, \text{START}, \text{HALT}, \text{YES}, \text{NO})$, alors il existe une machine de Turing déterministe $M' = (Q, \Sigma, \Gamma, \delta, \text{START}, \text{HALT}', \text{YES}', \text{NO}')$ telle que $\bar{L} = L(M')$ avec **YES' = NO** et **NO' = YES**.