Modèles de calcul : principales définitions et théorèmes

Laure Danthony

D'après le cours de Marianne Delorme, janvier - mai 2001

Table des matières

1 Quelques modèles de calcul

On introduit dans ce chapitre les machines de Turing (voir cours de décidabilité), les machines RAM, les algorithmes de Markov (cf décidabilité), les circuits booléens.

1.1 Machines de Turing

La définition est ici d'une machine de Turing déterministe à 1 ruban. On signale l'équivalence avec les machines à plusieurs rubans, et aussi la possibilité de se restreindre à un alphabet $\{0, 1, B\}$

1.2 Machines RAM

DÉFINITION 1
On dispose d'un alphabet $(a_i)_{i \leq k}$ et de registres. Les instructions autorisées sont les suivantes :

1_j X add_j Y, $j \in \{1, \dots k\}$ (concaténer au mot de Y la lettre a_j).

2 X *del* Y (effacer le premier caractère du mot contenu par Y).

3 X *clr* Y (effacer le contenu de Y, c'est à dire le remplacer par ε).

4 X Z$\leftarrow$ Y (substituer au contenu de Z le contenu de Y ; Y n'est pas modifié).

5 X $jmpN$ (aller à la ligne de nom N).

6_j X Yjmp_jN (si le mot contenu par Y commence par a_j, aller à la ligne N).

7 $CONTINUE$ (ne rien faire).

X est un numéro de ligne, Y et Z des registres, N est un nom de ligne : N_a(avant la ligne), N_b(après la ligne).

DÉFINITION 2
Un **programme RAM** est la donnée d'un nombre fini de registres et d'une liste finie d'instructions cohérentes avec une dernière instruction qui est de type 7.

DÉFINITION 3
Une fonction est **RAM-calculable** s'il existe un programme RAM qui calcule $f(u)$ (qui l'inscrit sur R_1 à la fin et termine) ssi $u \in Dom(f)$; sinon elle boucle.

PROPOSITION 1 *Les fonctions RAM calculables sont les fonctions Turing-calculables.*

1.3 Algorithmes de Markov

On redonne la définition :

DÉFINITION 4
Un **algorithme de Markov** est la donnée d'un certain nombre de règles, données dans un ordre défini, et que l'on manipule d'une certaine façon (facteur le plus à gauche, règle la plus petite).
Plus formellement, un algorithme de Markov sur un alphabet fini Σ est une suite finie de règles (ou productions) de la forme $p_i \to q_i$ ou $p_i \to_{(t)} q_i$ (t pour terminal). On transforme un mot en son sucesseur par essai de "matching" des règles en partant par la règle de numéro plus petit. Si elle est étiquetée par (t), l'algorithme termine. Sinon, on réessaie les règles (toujours en partant de la règle 1). L'algo termine aussi si plus aucune règle ne peut plus s'appliquer.

PROPOSITION 2 *Les fonctions MT-calculables sont les fonctions Markov-calculables.*

1.4 Circuits booléens

On choisit les formules contruites sur $\{\neg, \vee, \wedge\}$. On étudie les fonctions booléennes : $\{0,1\}^n \to \{0,1\}$. Le problème c'est que pour étudier une calculabilité, on dispose de fonctions de $\{0,1\}^* \to \{0,1\}$. On va donc être amené à considérer une famille de circuits.

DÉFINITION 5
Un **circuit booléen** est la donnée d'un graphe fini acyclique, avec V l'ensemble des portes : une porte est de type 0, 1, x_i (entrée), $\neg$, $\wedge$ ou $\vee$. La porte $n+q$ est la porte de sortie, de degré sortant 0. On associe à un circuit la fonction qu'il calcule.

DÉFINITION 6
La **taille** d'un circuit est son nombre de portes (cf complexité en espace) ; la **profondeur** d'un circuit est la longueur d'un plus long chemin (cf complexité en profondeur).

PROPOSITION 3 *Pour toute fonction booléenne n-aire, la taille minimum d'un circuit qui calcule f est strictement inférieure à 2^{n+2}.*

PROPOSITION 4 *Pour tout $n \geq 2$, il existe une fonction booléenne f telle que aucun circuit calculant f ne soit de taille inférieure à $\frac{2^n}{2n}$.*

PROPOSITION 5 *Tout langage (ne contenant pas ε) sur $\{0,1\}$ est décidé par une famille de circuits booléens.*

REMARQUE 1 Le problème c'est que l'on ne veut pas récupérer les langages non récursifs.

DÉFINITION 7
On dit qu'une famille (C_n) de circuits est **polynômiale** si il existe p un polynôme tel que pour tout n, la taille de C_n soit inférieure à $p(n)$.

REMARQUE 2 Là encore il y a un problème car il y a des langages non décidables qui sont décidés par des familles polynômiales de circuits.

DÉFINITION 8
On dit qu'une famille de circuits (C_n) est **uniforme** si il existe une machine de Turing qui sur l'entrée 1^n sort le codage de C_n.

REMARQUE 3 Cette fois on récupère bien les fonctions Turing-calculables, mais on n'a plus une notion intrinsèque.

1.5 Complexité en temps et espace des modèles Turing, Markov et RAM

On donne les définitions pour Turing, elles seront similaires pour les autres modèles.

DÉFINITION 9 (TEMPS)
Soit u une entrée pour une MT $\mathcal{M}$, on désigne par $T_{\mathcal{M}}(u)$ (**complexité en temps sur l'entrée u** le nombre de pas de calcul sur l'entrée u augmenté de $|U|$ lorsque $\mathcal{M}$ s'arrête sur u. Sinon, $T_{\mathcal{M}}(u)$ n'est pas définie.

DÉFINITION 10 (ESPACE)
Sur l'entrée u, si $\mathcal{M}$ s'arrête, on définit $S_{\mathcal{M}}(u)$ (**complexité en espace**) comme le nombre de cases visées par la tête du calcul.

REMARQUE 4 On considère selon les cas que l'entrée elle-même est visée, ou bien on ne la compte pas (ex : LOGSPACE).

FAIT 1 $S_{\mathcal{M}}(u) \leq T_{\mathcal{M}}(u) \leq k_{\mathcal{M}}^{S_{\mathcal{M}}(u)}$.

Comme l'on cherche à exprimer une hiérarchie entre les machines de Turing (c'est à dire classer des "puissances" indépendantes de l'entrée), on définit :

DÉFINITION 11
$Time_{\mathcal{M}} : \mathbb{N} \to \mathbb{N}, n \mapsto Time_{\mathcal{M}}(n) = Max\{T_{\mathcal{M}}(u) / |u| = n\}$

DÉFINITION 12
$Space_{\mathcal{M}} : \mathbb{N} \to \mathbb{N}, n \mapsto Space_{\mathcal{M}}(n) = Max\{S_{\mathcal{M}}(u) / |u| = n\}$

DÉFINITION 13
La **classe de complexité en temps** (resp. en espace) exprimée par f est l'ensemble des langages décidés par une MT (déterministe ou non) opérant en temps (resp. en espace) f : on note $DTIME(f)$ (resp. $DSPACE(f)$) pour les machines déterministes, $NTIME(f)$ (resp. $NSPACE(f)$) pour les machines non déterministes.

2 Isomorphisme de Rogers

DÉFINITION 14
Par **système acceptable de programmation** on entend :

1. énumération de $\mathbb{N}$ dans $\mathbb{N}$ incluant toutes les fonctions ppr.
2. existence d'un algorithme universel, c'est-à-dire u tel que :
$$\varphi_u(< i, j >) = \varphi_i(j)$$
3. existence d'un théorème s-n-m :
$$\varphi_i(< x_1, \ldots, x_m, y_1, \ldots, y_n >) = \varphi_{s_n^m(x_1,\ldots x_m)}(< y_1, \ldots y_n >)$$

PROPOSITION 6 *Pour tout indice k, il existe $j \neq k$, $\varphi_j = \varphi_k$.*

PROPOSITION 7 *Il existe une fonction récursive ρ telle que :*

- $\forall i, x, x \mapsto \rho(< i, x >)$ *est injective.*
- $\forall i, x, \varphi_i = \varphi_{\rho(<i,x>)}$

PROPOSITION 8 *Si (φ_i) et (ψ_i) sont deux sap, alors il existe des fonctions totales récursives g et k, telles que, pour tout i, on ait : $\varphi_i = \psi_{g(i)}$ et $\psi_i = \varphi_{h(i)}$.*

PROPOSITION 9 *Soient (φ_i) et (ψ_i) deux sap. Alors il existe des fonctions g et h récursives totales telles que :*

- $g(0) > 0$, $h(0) > 0$.
- *g et h sont strictement croissantes.*
- $\varphi_i = \psi_{g(i)}$ *et* $\psi_i = \varphi_{h(i)}$.

Ces propositions sont le cheminement vers le :

THÉORÈME 1 *Soient (φ_i) et (ψ_i) deux sap. Alors il existe une fonction f totale récursive bijective telle que :*
$$\varphi_i = \psi_{f(i)} \text{ et } \psi_i = \varphi_{f^{-1}(i)}.$$

3 Classes de complexité Turing (1) : P, NP, EXPTIME, P-complet, NP-complet

3.1 Théorèmes de compression et d'accélaration linéaire

THÉORÈME 2 (COMPRESSION) *Si $c \in \mathbb{R}^{+*}$, si s est une fonction $\mathbb{N} \to \mathbb{N}$, alors $SPACE(s(m)) = SPACE(c.s(m))$.*

THÉORÈME 3 (ACCÉLÉRATION LINÉAIRE) *Soit $c \in \mathbb{R}^{+*}$. Si L est décidé par une MT (à 2 rubans au moins), en temps $t(n)$ et si Lim $\underset{n\to\infty}{\text{Inf}} \frac{t(n)}{n} = \infty$, alors L est décidé par une machine en temps $c.t(n)$.*

REMARQUE 5 *Lim* $\underset{n\to\infty}{\text{Inf}} \frac{t(n)}{n}$ désigne $\lim_{n\to+\infty} \text{Inf}\{\frac{t(n)}{n}, \frac{t(n+1)}{n+1}, \ldots\}$

3.2 Classe P

$$P = \bigcup_{k \geq 0} TIME(n^k)$$

Notion de réduction polynômiale :

DÉFINITION 15
Si A et B sont deux langages sur Σ, on dit que A **se réduit polynomialement** en B et on écrit $A \leq B$ sit il existe une fonction totale récursive $f : \Sigma^* \to \Sigma^*$ calculable en temps poynomiale et telle que $u \in A \Leftrightarrow f(u) \in B$.

FAIT 2 $\leq$ *est un préordre (relation réflexive et transitive).*

3.3 Classe NP

$$NP = \bigcup_{k \geq 0} NTIME(n^k)$$

C'est la classe des langages décidés en temps polynomial par une machine non déterministe de complexité polynomiale.

DÉFINITION 16
Une **machine de Turing non déterministe** se construit de la même façon qu'une MT classique, sauf que la fonction de transition choisit la lettre et le nouvel état dans un ensemble fini.

PROPOSITION 10 *Toute machine de Turing non déterministe M (temps = $t(n)$) peut être simulée par une machine déterministe M' (temps $2^{O(t(n))}$).*

PROPOSITION 11 *$L \in NP$ ssi il existe un langage $L' \in P$, et un polynôme p, tels que :*

$$L = \{x / \exists y, < x, y > \in L' \text{ et } |y| \leq p(|x|)\}.$$

DÉFINITION 17
L est un **langage NP-complet** ssi :

1. $L \in NP$
2. $\forall L' \in NP, L' \leq L'$ au sens de la réduction polynomiale.

DÉFINITION 18
L est un langage $\mathcal{C}$**-complet** ssi :

1. $L \in \mathcal{C}$
2. $\forall L' \in C, L' \leq L'$ au sens de la réduction polynomiale.

Voici des exemples :

DÉFINITION 19
PfP est défini par : On se donne :

- Un ensemble fini C de couleurs qui contient une couleur particulière, la couleur blanche ;
- une collection de tuiles τ (contenue dans C^4) qui contient une tuile particulière, la tuile blanche ;
- un entier $n \leq |C|$.

Alors existe-t-il un pavage fini, valide (les couleurs identiques se touchent) du plan, non trivial, de taille inférieure à $n * n$?

PROPOSITION 12 *$PfP \in NPc$.*

REMARQUE 6 On obtient ce résultat directement à partir des machines de Turing, en obtenant des tuiles de couleur à partir des états, et un pavage à aprtir des configurations successives de la machine.

DÉFINITION 20
Soit une machine déterministe $\mathcal{M}$, un entier t tel que $t \leq |Q_{\mathcal{M}}|$. Existe-t-il un calcul acceptant de $\mathcal{M}$ sur l'entrée ε, en temps $\leq t$? On note L_{NDTM} le langage qui code ce problème.

PROPOSITION 13 *L_{NDTM} est dans NPc.*

3.4 $NP \cap CoNP$

DÉFINITION 21
L_{prem} est le langage qui code le problème :

- n entier.
- n est-t-il premier?

PROPOSITION 14 *$L_{prem} \in NP \cap CoNP$.*

REMARQUE 7 On ne sait toujours pas si $L_{prem} \in P$ ou si $L_{prem} \in NPc$.

3.5 Langages P-complets

Les langages de P sont équivalents modulo la relation d'ordre induite par $\leq$. Pour établir une hiérarchie et parler de langage P-complet, on a besoin de la notion de réduction en espace logarithmique :

DÉFINITION 22
$A \leq_L B$ (A se **réduit logarithmiquement en** B) ssi il existe une fonction f totale calculable en espace logarithmique de Σ^* dans Σ^* telle que $u \in A \Leftrightarrow f(u) \in B$.

PROPOSITION 15 *$\leq_L$ est un préordre.*

DÉFINITION 23
L est **P-complet** si $L \in P$ et tout langage L' s'y réduit en espace logarithmique : $L' \leq_L L$.

Il existe des langages P-complets :

EXEMPLE 1 *On se donne X un ensemble fini, R une relation ($R \subseteq X \times X \times X$), et X_s, X_t deux sous-ensembles de X. La question posée est : X_t contient-t-il au moins un élément du plus petit ensemble $A \subseteq X$ tel que :*

- *$X_s \subseteq A$;*
- *si $y, z \in A$ et $< x, y, z > \in R$, alors $x \in A$.*

Le langage codant ce problème est SPS.

PROPOSITION 16 *SPS est P-complet.*

4 Classes de complexité Turing (2) : LOGSPACE, NLOGSPACE, PSPACE, PSPACE-complet

4.1 Définitions

On considère les machines à 1 ruban d'entrée de pure lecture et l'espace est évalué sur les rubans de travail.

DÉFINITION 24
Etant donnée une MT $\mathcal{M}$ de caractéristiques ci-dessus, sa **complexité en espace** est le nombre de cases visitées au moins un fois par la tête.

DÉFINITION 25
Une machine $\mathcal{M}$ est de **complexité en espace bornée par** $f : \mathbb{N} \to \mathbb{N}$ si pour (presque-) tout n, $SPACE_{\mathcal{M}}(n) \leq f(n)$ (ou $O(f(n))$.

DÉFINITION 26
On dénote :

$SPACE(f(n)) = \{L / \exists \mathcal{M}$ déterministe qui décide L et de complexité $f(n)\}$

- $NSPACE(f(n)) = \{L / \exists \mathcal{M} MTND$ non déterministe qui décide L et de complexité $f(n)\}$
- d'où $LOGSPACE, NLOGSPACE \ldots$
- $PSPACE = \bigcup_{k \geq 0} PSPACE(n^k)$.
- $NSPACE = \bigcup_{k \geq 0} NSPACE(n^k)$.

Quelques exemples :

EXEMPLE 2 *SAT peut être décidé en espace $O(n)$, appartient donc à $PSPACE$.*

DÉFINITION 27
Soit G un graphe à n sommets. x, y sont deux sommets, existe-t-il un chemin entre x et y? L_{att} est le langage codant ce problème. Les données sont les sommets de 1 à n, la matrice du graphe représentée par ses lignes successives (mots sur $\{0,1\}$ de longueur n), et les deux sommets x et y.

PROPOSITION 17 $L_{att} \in SPACE\big(log(n)^2\big)$.

4.2 Théorème de Savitch

THÉORÈME 4 *Soit $f : \mathbb{N} \to \mathbb{N}$ complétement constructible en espace et telle que $f(n) \geq log(n)$. Alors :*

$$NSPACE\big(f(n)\big) \subseteq SPACE\big(f(n)^2\big).$$

REMARQUE 8 "Complétement contructible en espace" signifie qu'il existe une machine de Turing qui sur toute entrée n occupe exactement $f(n)$ cases.

COROLLAIRE 1 $NSPACE = PSPACE$.

A ce stade, on a :

$$LOGSPACE \subseteq NLOGSPACE \subseteq P \subseteq NP \subseteq PSPACE = NSPACE$$

4.3 Il existe des langages PSPACE-complets

DÉFINITION 28
A est $PSPACE$-complet si $A \in PSPACE$ et tout autre langage de $PSPACE$ s'y réduit polynomialement en temps.

DÉFINITION 29
Une formule QBF (totally quantified boolean formula) est une formule du type $Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \Phi(x_1, \ldots, x_n)$. Les $Q_i \in \{\exists, \forall\}$ et les variables de Φ sont toutes dans les champs des quantificateurs. $TQBF$ est le langage des fQBF "vraies".

PROPOSITION 18 *$TQBF$ est un langage $PSPACE$-complet.*

5 Relations entre classes de Complexité Turing

5.1 Quelques relations

THÉORÈME 5 *On a les relations générales suivantes :*

1. *Si $L \in TIME\big(f(n)\big)$ alors $L \in SPACE\big(f(n)\big)$.*

2. *Si* $L \in SPACE\big(f(n)\big)$, *si* $f(n) \geq log(n)$ *et si* f *est totalement constructible en espace, alors il existe une constante ne dépendant que de* L *telle que* $L \in TIME\big(c^{f(n)}\big)$.

3. *Si* $L \in NTIME\big(f(n)\big)$ *alors il existe une constante* $c_L > 0$ *telle que* $L \in TIME\big(c^{f(n)}\big)$.

5.2 Il n'existe pas de classe "maximum"

THÉORÈME 6 *Soit* f *une fonction totale calculable. Il existe un langage récursif* L *qui n'appartient pas à* $TIME(f)$ *et il existe un langage récursif* L' *qui n'appartient pas à* $SPACE(f)$.

5.3 Hiérarchie déterministe en espace

THÉORÈME 7 *Soit* $s_2(n)$ *une fonction totalement constructible en espace, et* $s_1(n)$ *une fonction telle que* $s_2(n) \geq s_1(n) \geq log(n)$ *et* $Lim \inf_{n\to\infty} \frac{s_1(n)}{s_2(n)} = 0$, *alors :*

$$SPACE\big(s_1(n)\big) \subsetneq SPACE\big(s_2(n)\big)$$

5.4 Hiérarchie déterministe en temps

THÉORÈME 8 *Soient* $t_1(n)$ *et* $t_2(n)$ *des fonctions* $\mathbb{N} \to N$ *telles que* $t_2(n) \geq t_1(n) \geq n$ *et aussi telles que* $Lim \inf_{n\to\infty} \frac{t_1(n)\log(t_1(n))}{t_2(n)} = 0$. *Alors :*

$$TIME\big(t_1(n)\big) \subsetneq TIME\big(t_2(n)\big)$$

DÉFINITION 30
Une fonction "**super polynomiale**" est une fonction non décroissante de $\mathbb{N}$ dans $\mathbb{N}$ telle que :

$$Lim \inf_{n\to\infty} \frac{n^k}{f(n)} = 0$$

En corollaire au théorème précédent, on obtient :

PROPOSITION 19 *Pour toute fonction* f *super polynômiale, on a :*

- $P \subsetneq TIME\big(f(n)\big)$
- $PSPACE \subsetneq SPACE\big(f(n)\big)$

On prouve par ailleurs que $EXPTIME \neq PSPACE$, et finalement :

PROPOSITION 20

- $P \subsetneq EXPTIME = \bigcup_{k>0} TIME\big(2^{kn}\big)$
- $LOGSPACE \subsetneq PSPACE \subsetneq EXPTIME$

6 Machines de Turing avec oracle, hiérarchie polynomiale

6.1 Machine de Turing avec oracle

DÉFINITION 31 (MT AVEC ORACLE)
C'est une machine de Turing (déterministe ou non), à laquelle on ajoute 3 états : $q_?$ qui sert à interroger l'oracle, q_{oui} et q_{non} qui représenteront la réponse de l'oracle. On se donne également un langage A auquel sera réservé un ruban de la machine. L'oracle dira "oui" si le mot de ce ruban à l'instant de la question est dans le langage A, il dit "non" sinon (et ceci en 1 seul pas de calcul!)

NOTATION : on note $L(\mathcal{M}, A)$ ou $L(\mathcal{M}^A)$ le langage reconnu par la machine $\mathcal{M}$ avec A comme oracle.

Comme langage d'oracle, on prend uniquement les *langages récursifs*, sinon on pourrait récupérer des langages non récursifs, ce qui n'est pas raisonnable !

6.2 $P = NP$ ou $P \neq NP$?

THÉORÈME 9 *Il existe un oracle A tel que $P^A = NP^A$*

REMARQUE 9 Pour la preuve, on utilise un langage $PSPACE$ complet, et on utilise le fait que $NSPACE = PSPACE$.

THÉORÈME 10 *Il existe un oracle B tel que $P^B \neq NP^B$.*

REMARQUE 10 Ici la démo est sensiblement plus difficile !

REMARQUE 11 Dommage !

6.3 Notion de réduction Turing

DÉFINITION 32
Etant donnés 2 langages A et B (sur un alphabet Σ), on dit que A est Turing-réductible en B et on note $A \leq_T B$ si il existe une machine de Turing déterministe de complexité en temps polynomiale qui décide A avec l'oracle B.

FAIT 3 *$\leq_T$ est un préordre.*

PROPOSITION 21 $A \leq B \Rightarrow A \leq_T B$

REMARQUE 12 Si $A \leq_T$ alors :

- B récursif $\Rightarrow$ A récursif.
- B réc. énum. $\Rightarrow$ A réc.énum.

FAIT 4 *On a $\overline{A} \leq_T A$. On ne sait pas si $\overline{A} \leq A$. Par contre, si $NP \neq Co-NP$, alors $\overline{SAT} \not\leq SAT$.*

DÉFINITION 33 (CLÔTURE)
On dit que $\mathcal{C}$ est close par $\leq_T$ si la classe vérifie : si $L \in \mathcal{C}$ et $L' \leq_T L$, alors $L' \in \mathcal{C}$

PROPOSITION 22

- *P et $PSPACE$ sont closes pour $\leq_T$.*
- *$NP = CoNP$ ssi NP est close pour $\leq_T$.*

6.4 Classes relativisées de P et de NP

DÉFINITION 34
- A étant un langage, $P^A = \{L \in P,$ décidés avec l'oracle $A\}$
- de même, $NP^A = \{L \in NP,$ décidés avec l'oracle $A\}$
- de même avec une classe $\mathcal{C}$ quelconque.

PROPOSITION 23 (FACILES)

- *$A \in NP^A$;*
- *$A \in P^B \Rightarrow A \in NP^B$;*
- *$A \in NP^B \Rightarrow A \in NP^{\overline{B}}$;*
- *$A \in NP^B$ et $B \in P^C \Rightarrow A \in NP^C$.*

Par abus de langage, on a noté A pour $\{A\}$ dans la proposition précédente.

PROPOSITION 24

- *$NP^P = NP$;*
- *$NP^{PSPACE} = PSPACE$.*

6.5 Hiérarchie polynômiale, V1

Il s'agit de la famille $(\Delta_n^p, \Sigma_n^p, \Pi_n^p)_{n \geq 0}$ avec :

- $\Delta_0^p = \Sigma_0^p = \Pi_0^p = P$
- $\Sigma_{n+1}^p = NP^{\Sigma_n^p}$
- $\Pi_{n+1}^p =$ Co-Σ_{n+1}^p
- $\Delta_{n+1}^p = P^{\Sigma_{n+1}^p}$

NOTATION : On note $PH = \bigcup_{n \geq 0} \Sigma_n^p$.

REMARQUE 13 $\Sigma_1^p = NP^P = NP, \Pi_1^p =$ Co-$NP, \Delta_1^p = P$

PROPOSITION 25 $\forall n \geq 0, \Sigma_n^p \cup \Pi_n^p \subseteq \Delta_{n+1}^p \subseteq \Sigma_{n+1}^p \cap \Pi_{n+1}^p$.

REMARQUE 14 On obtient donc une hiérarchie que l'on visualise habituellement par des patatoïdes enlacés.

THÉORÈME 11 $PH = PSPACE$

REMARQUE 15 La portée de ce théorème est très importante !

6.6 Hiérarchie polynômiale, V2

Soit x un mot, $R(x)$ une propriété sur les mots, $p(n)$ un polynôme :

DÉFINITION 35
- $\exists^{p(n)} x R(x)$ signifie qu'il existe un mot x, $|x| \leq p(n)$, qui satisfait $R(x)$.
- $\forall^{p(n)} x R(x)$ signifie pour que tout mot x, tel que $|x| \leq p(n)$, on a $R(x)$.

DÉFINITION 36
Soit $\mathcal{C}$ une classe de langages, $\exists\mathcal{C}$ désigne la classe des langages A pour lesquels il existe $B \in \mathcal{C}$ tel que :

$$x \in A \Leftrightarrow \exists^{p(|x|)} y(< x, y > \in B$$

DÉFINITION 37
Soit $\mathcal{C}$ une classe de langages, $\forall\mathcal{C}$ désigne la classe des langages A pour lesquels il existe $B \in \mathcal{C}$ tel que :

$$x \in A \Leftrightarrow \forall^{p(|x|)} y(< x, y > \in B$$

THÉORÈME 12
- $\exists P = NP = \Sigma_1^p$
- $\forall P = Co\text{-}NP = \Pi_1^p$
- $(k > 0), \exists\Sigma_k^p = \Sigma_k^p$
- $(k > 0), \forall\Pi_k^p = \Pi_k^p$
- $(k \geq 0), \exists\Pi_k^p = \Sigma_{k+1}^p$
- $(k \geq 0), \forall\Sigma_k^p = \Pi_{k=1}^p$

REMARQUE 16 Donc les 2 hiérarchies sont les mêmes !

THÉORÈME 13
- *$A \in \Sigma_k^p$ ssi il existe $B \in P$ et $p(n)$ un polynôme tels que :*

$$x \in A \Leftrightarrow \exists^{p|x|} y_1 \forall^{p|x|} y_2 \ldots Q^{p|x|} y_k (< x, y_1, \ldots, y_k > \in B),$$

  où $Q = \forall$ si k pair, $\exists$ sinon.
- *résultat analogue pour Π_k^p.*

6.7 Résultats généraux sur les 2 hiérarchies

THÉORÈME 14 *Si il existe k tel que $\Sigma_k^p = \Pi_k^p$, alors $\forall j \geq 0, \Sigma_{k+j}^p = \Pi_{k+j}^p$*

REMARQUE 17 Ce théorème se montre par récurrence sur k.

COROLLAIRE 2 $\exists k > 0, P = \Sigma_0^p \neq \Sigma_k^p \Rightarrow P \neq NP$

COROLLAIRE 3 $P \neq NP \Leftrightarrow P \neq PH$.

THÉORÈME 15 $PH = PSPACE \Rightarrow \exists k, \Sigma_{k+1}^p = \Sigma_k^p$.

6.8 Intérêts (!?) de la hiérarchie polynômiale

6.8.1 Sa grande ressemblance avec la hiérarchie mathématique

- Le problème de l'arrêt donne un exemple de langage récursivement énumérable non récursif: $\varphi^{(1)}$.
- le problème de l'arrêt des MT avec l'oracle $\varphi^{(1)}$, donne l'existence d'un langage énumérable avec $\varphi^{(1)}$ mais non récursif avec $\varphi^{(1)}$: $\varphi^{(2)}$.

D'où une hiérarchie que l'on peut présenter autrement :

DÉFINITION 38
- Une relation R ($\subseteq (\Sigma^*)^k$) est dite récursive si $L_R = \{< x_1, \ldots, x_k > / R(x_1, \ldots, x_k)\}$ est récursif.
- $\forall k \geq 0$, on définit Σ_k comme la classe des langages L pour lesquels $\exists$ une relation $(k+1)$-aire R récursive telle que : $L = \{x / \exists x_1, \forall x_2, \ldots Q_k x_k R(x, x_1, \ldots, x_k)\}$.

REMARQUE 18 Σ_0 est l'ensemble des fonctions récursives ; Σ_1 l'ensemble des fonctions récursivement énumérables.

PROPOSITION 26 *Si on note Π_k = Co-Σ_k, on a :*

- $\forall k \geq 0, \exists$ *un langage dans* $\Pi_k \setminus \Sigma_k$;
- $\forall k \geq 0, \exists$ *un langage dans* $\Sigma_k \setminus \Pi_k$;
- $\forall k \geq 0, \Sigma_k \cup \Pi_k \subseteq \Sigma_{k+1} \cup \Pi_{k+1}$;
- $\forall k \geq 0, \Sigma_k \neq \Sigma_{k+1}$.

6.8.2 Il existe des problèmes "naturels" dont les langages associés sont dans les différents niveaux de la hiérarchie

EXEMPLE 3 *CIRCUIT MINIMUM :*

- *Soit C un circuit booléen.*
- *Est-il minimum, au sens qu'un circuit ayant moins de portes ne peut calculer la même fonction.*

PROPOSITION 27 *$CM \in \Pi_2^p$. On ne sait pas s'il appartient à un niveau inférieur, ni s'il est Π_2^p-complet.*

EXEMPLE 4 *GRN :*
On dit qu'un graphe $G = (V, E)$ est k-coloré si il est associé à une fonction $E \to \{1, \ldots, k\}$. Pour $k > 0$, on définit R_k (nombre de Ramsey) comme l'entier n minimum tel que tout graphe complet 2-coloré de taille n contienne une clique monocolore de taille k. On sait que R_k existe.
GNR : problème des nombres de Ramsey généralisés. Soit un graphe complet $G = (V, E)$ partiellement 2 coloré : $C : E \to \{0, 1, *\}$, et soi $k > 0$. Est-il vrai que $\forall C' : E \to \{0, 1\}$ tel que $C'(e) = C(e)$ si $c(e) \neq *$, il existe une clique monocolore de taille k?

PROPOSITION 28 *GRN est Π_2^p-complet.*

REMARQUE 19 On le démontre en le réduisant à partir de $3 - SAT_2$.

EXEMPLE 5 *SAT_k :*
Pour $k \geq 1$, généralisons SAT. Si X est un ensemble de variables, $\tau : X \to \{0, 1\}$.
Soit $X_1, \ldots, X_k$ des ensembles de variables, et φ une formule booléenne. Est-il vrai que :

$$\exists \tau_1 : X_1 \to \{0,1\}, \forall \tau_2 : X_2 \to \{0,1\}, \ldots Q_k \tau_k : X_k \to \{0,1\}, \varphi_{|\tau_i} = 1?$$

THÉORÈME 16 *$\forall k \geq 1$, SAT_k est Σ_k^p-complet.*

COROLLAIRE 4

- *$\forall k \geq 1$, $2 - CNF - SAT_k$ est Σ_k^p-complet pour k impair ;*
- *$\forall k \geq 1$, $3 - CNF - SAT_k$ est Π_k^p-complet pour k impair ;*

DÉFINITION 39
Quelques nouvelles classes de complexité :

- $2 - EXP = \bigcup_{k \geq 1} TIME(2^{(2^{n^k})})$
- $ELEM$ est la réunion des langages sont la complexité s'écrit sous forme d'une tour finie d'exponentielles.

PROPOSITION 29 *Les problèmes d'équivalence des expressions rationnelles sont partout dans la hiérarchie :*

- *l'équivalence sur $(+, \cdot, *)$ est $PSPACE$-complet ;*
- *l'équivalence sur $(+, \cdot)$ est Co-NP-complet ;*
- *l'équivalence sur $(+, \cdot, *, ^2)$ est $EXPSPACE$-complet ;*
- *l'équivalence sur $(+, \cdot, ^2)$ est Co-$NEXPSPACE$-complet ;*
- *l'équivalence sur $(+, \cdot, *, ^2, \neg)$ où $\neg$ est le complémentaire appartient à $ELEM$;*

7 Quelques résultats généraux. Une idée d'une théorie axiomatique de la complexité

Dans ce chapitre on donne des idées de preuves pour le cas de l'espace déterministe , mais des résultats similaires existent pour le cas du temps, et aussi pour le cas non déterministe.

7.1 Théorème de la lacune

THÉORÈME 17 (LACUNE) *Soit $g : \mathbb{N} \to \mathbb{N}$ une fonction totale récursive, vérifiant $\forall n, g(n) \geq n$. Alors il existe une fonction s totale récursive telle que*

$$SPACE\big(s(n)\big) = SPACE\Big(g\big(s(n)\big)\Big).$$

REMARQUE 20 Cela veut dire qu'il existe un fossé entre les limites de la hiérarchie.

7.2 Théorème d'accélération de Blum

THÉORÈME 18 *Soit r une fonction totale récursive. Il existe un langage L récursif tel que pour toute machine de Turing $\mathcal{M}_i$ qui le décide, il existe une machine $\mathcal{M}_j$ qui le décide et telle que : $r\big(s_j(n)\big) \leq s_i(n)$ presque partout.*

7.3 Théorème de l'union

THÉORÈME 19 *Soit $\big(f_i(n)\big)_{i\geq 1}$ une famille récursivement énumérable de fonctions récursives (c'est-à-dire qu'il existe une machine de Turing qui énumère les machines $\mathcal{M}_i$ qui calculent les f_i). On suppose aussi que $\forall i, \forall n, f_i(n) < f_{i+1}(n)$. Alors il existe une fonction récursive s telle que*

$$SPACE\big(s(n)\big) = \bigcup_{i\geq 1} SPACE\big(f_i(n)\big).$$

REMARQUE 21 En particulier $PSPACE$ est une véritable classe de complexité, *i.e.* qu'il existe une fonction dont elle est la classe de complexité ...

7.4 Mesures de complexité abstraites

DÉFINITION 40
Soit (φ_i) un système acceptable de programmation. Une **énumération** (Φ_i) de fonctions Turing-calculables définit une mesure de complexité pour le système $(_v phi_i)$ lorsque :

1. $\forall i, \forall x, \Phi_i(x)$ est définie ssi $\varphi_i(x)$ l'est ;
2. l'ensemble $\{< i, x, y > | \Phi_i(x) \leq y\}$ est récursif.

EXEMPLE 6 (CANONIQUE) *Soit (φ_i) un système acceptable de programmation. Il a une fonction universelle u. On considère l'ensemble $A = \{< i, j > \mid u(< i, j >)$ est défini$\}$. A est récursivement énumérable donc il existe B récursif tel que $< i, j > \in A$ ssi $\exists t, < i, j, t > \in B$.*
Alors $\Phi_i(j) = \mathrm{Min}\{\ t \mid < i, j, t > \in B\}$ est une mesure de complexité sur (φ_i).

THÉORÈME 20 (RELATIVISATION) *Soient* (φ_i) *et* (ψ_i) *deux systèmes acceptables de programmation. Soit* t *une fonction totale récursive qui permet la traduction de l'un dans l'autre (il en existe une bijective). Soient* (Φ_i) *et*(Ψ_i) *des mesures de compelxité associées. Alors il existe une fonction* r *totale récursive telle que pour tout* i *on ait presque partout :*

$$\Phi_i(x) \leq r\big(< x, \Psi_{t(i)}(x) > \big) \text{ et } \Psi_{t(i)}(x) \leq r\big(< x, \Psi_i(x) > \big)$$

REMARQUE 22 Par exemple, on sait comparer polynômialement les machines de Turing et les machines RAM.