Cours de mathématiques

TOME X

INFORMATIQUE

Mathématiques générales

France $\sim$ 2024

Écrit et réalisé par Louis Lascaud

Table des matières

Chapitre 1

Analyse numérique

Résumé

L'analyse numérique étudie les méthodes programmables en algorithmique pour résoudre des problèmes mathématiques. C'est donc une discipline bel et bien mathématique à l'interface avec l'informatique théorique. Parmi les problèmes les plus étudiés, citons :

- ⋆ la réduction des systèmes linéaires ;
- ⋆ la convergence des méthodes de calcul d'une valeur d'une fonction en un point ;
- ⋆ l'interpolation polynomiale ;
- ⋆ les méthodes d'approximation géométriques ;
- ⋆ le calcul intégral informatique ;
- ⋆ la résolution d'équations différentielles par discrétisation ;
- ⋆ l'optimisation, linéaire ou non linéaire.

1.1 Résolution de systèmes linéaires

Théorème. ***(Caractérisation de la solvabilité par l'orthogonal)***

Soit A une matrice de taille $m \times n$. L'équation $Ax = b$ admet au moins une solution si et seulement si b est orthogonal à $\mathrm{Ker}(A^*)$.

Chapitre 2

Théorie de la complexité

Résumé

Dans un premier temps, on définit les concepts de base du traitement algorithmique informatique : machines de Turing, automates, etc. Dans un second temps, on introduit l'outil des circuits booléens qui facilite nombre de preuves de la section suivante. Dans un troisième temps, on introduit plusieurs classes de complexité et l'on explicite les liens entre elles, dont certains énoncés constituent des questions encore ouvertes à ce jour, parmi lesquels l'un des plus fameux problèmes du millénaire.

2.1 Machines de Turing

2.1.1 Définition et premières propriétés

En informatique théorique, une machine de Turing est un modèle abstrait du fonctionnement des appareils mécaniques de calcul, tel un ordinateur. Ce modèle a été imaginé par Alan Turing en 1936, en vue de donner une définition précise au concept d'algorithme ou de « procédure mécanique ». Il est toujours largement utilisé en informatique théorique, en particulier dans les domaines de la complexité algorithmique et de la calculabilité.

À l'origine, le concept de machine de Turing, inventé avant l'ordinateur, était censé représenter une personne virtuelle exécutant une procédure bien définie, en changeant le contenu des cases d'un ruban infini, en choisissant ce contenu parmi un ensemble fini de symboles. D'autre part, à chaque étape de la procédure, la personne doit se placer dans un état particulier parmi un ensemble fini d'états. La procédure est formulée en termes d'étapes élémentaires du type : « si vous êtes dans l'état 42 et que le symbole contenu sur la case que vous regardez est « 0 », alors remplacer ce symbole par un « 1 », passer dans l'état 17, et regarder maintenant la case adjacente à droite ».

Définition. ***(Machine de Turing)***

Une *machine de Turing* ou *machine de Turing déterministe* ou *MT* est

Reformulation pratique. ***(Machine de Turing)***

Une machine de Turing est un concept abstrait composé des éléments suivants :

(i) un *ruban infini* dans les deux sens ou *bande*, éventuellement plusieurs rubans dont l'un choisi par défaut est dit *ruban de travail*, divisé en *cases* ;

(ii) un alphabet $\mathcal{A}$, dit *alphabet de travail*, souvent $\{0,1\}$, de sorte que les cases sont remplies des symboles de l'alphabet, à raison d'une lettre par case ; on suppose que $\mathcal{A} \ni B$ est non vide où B fixé est appelé *symbole blanc*, aussi souvent noté 0 ou $\#$;

(iii) une *tête de lecture*, qui *est* dans un état (on dit, mais cela n'a pas de sens précis, qu'il existe un *registre d'état* qui mémorise l'*état courant* de la machine) ;

(iv) un ensemble fini Σ d'états.

À chaque *étape* d'un *calcul*, la tête est devant une case, lit son contenu et décide de trois choses : le contenu de ladite case à la prochaine étape, son propre état pour la prochaine étape, et si elle se déplace à droite ou à gauche. Autrement dit, la tête, ou *fonction de transition*, est une application

$$\delta : A \times \Sigma \to A \times \Sigma \times \{G,D\}.$$

On note aussi $G = \leftarrow$ et $D = \rightarrow$. Si l'on n'identifie pas la tête à δ, on l'appelle *table d'actions* ou *partie de contrôle* qui est une entité non réalisée.

En particulier, le choix de l'état suivant par la tête est indépendant de sa position, mais dépend a priori et du symbole de la case lue et de son propre état actuel. On suppose qu'il existe un état $q_0 \in \Sigma$ dit *état initial* dans lequel la machine se trouve au départ, *i.e.* à la première étape. De plus, on supose qu'il existe un état $h \in \Sigma$ dit *halte* de sorte que si la machine arrive à cet état, elle s'*arrête*. La suite des étapes peut être finie ou infinie.

Dans certains livres, δ est une fonction et non une application et l'image point de $(a,s) \in A \times \Sigma$ hors du domaine de définition est définie dans la définition précédente par (a,h,G) où G est fixé tout à fait arbitrairement.

On peut également fixé $\Sigma \subseteq \mathcal{A}$ un *alphabet d'entrée* constitué des symboles dans les cases à la première étape ne contenant pas nécessairement B, et un ensemble $F \subseteq \Sigma$ d'*états acceptants* ou *finals* ou *finaux* constitué des états qui peuvent précéder une halte selon δ.

⟶ *Convention.* Les cases du ruban dont on ne précise pas quel symbole de l'alphabet elles contiennent, contiennent par défaut le symbole blanc. En particulier toutes les cases sont remplies.

VOC L'*entrée* de la machine est la case où elle *commence*, c'est-à-dire où elle se trouve à la première étape. Elle n'est pas intrinsèque à la machine, ou, selon une convention annexe, le ruban à la première étape : dans ce cas, c'est que le ruban n'a qu'un nombre fini de symbole nons blancs et à la première étape, la tête de la machine se trouve sur le premier symbole non blanc. Parfois, on impose qu'une case contenant B ne peut être une entrée, mais c'est bénin. La *sortie* de

la machine est le symbole devant la tête quand la machine s'arrête, ou, selon une convention annexe, le ruban à la halte. Elle dépend uniquement de l'entrée pour une machine de Turing déterministe.

⟶ *Notation.* δ peut facilement être représentée sous la forme d'un tableau à deux entrées quitte à colorier d'une couleur ou d'une autre les cases selon qu'elles mènent à droite où à gauche.

Remarque. Lorsque le contexte le permet, ce qui est souvent le cas, on se permet quelques abus quant à l'argument ou à la valeur de δ en considérant des projections, par exemple.

Définition. ***(Configuration d'une machine de Turing)***

La *configuration* (d'une machine de Turing à une étape) du calcul consiste en les symboles dans les cases du ruban à cette étape, l'état de la tête à cette étape et la position de la tête à cette étape.

Heuristique

La notion de machine de Turing est trop générale pour être facilement manipulable.

2.1.2 Description

Une machine de Turing peut se voir de nos jours comme l'abstraction d'un ordinateur. Dans une machine de Turing,

- ⋆ la partie de contrôle représente un microprocesseur ;
- ⋆ la bande représente la mémoire interne de l'ordinateur ;
- ⋆ la tête de lecture représente la bus *(et l'on comprend pourquoi différencier abstraitement tête de lecture et partie de contrôle).*

2.1.3 Lecture seule, lecture unique

Définition. ***(Lecture seule)***

Un ruban d'une machine de Turing est dit *en lecture seule* si la machine ne peut pas changer le contenu des cases, autrement dit si $\pi_A \circ \delta = \pi_A$.

Définition. ***(Lecture unique)***

Un ruban d'une machine de Turing est dit *en lecture unique* si après avoir lu une case du ruban, la machine doit se déplacer à droite, autrement dit si $\mathrm{Im}(\pi_{\{G,D\}} \circ \delta) = \{D\}$.

2.1.4 Temps de calcul

⟶ *Notation.* On note $\{0,1\}^* = \bigcup_{n=1}^{\infty} \{0,1\}^n$ l'ensemble des suites finies de 0 et de 1. Si $x \in \{0,1\}^*$, on note $|x|$ la *longueur* de x qui est l'unique $n \in \mathbb{N}^*$ tel que $x \in \{0,1\}^n$.

Définition. ***(Fonction booléenne)***

Une *fonction booléenne* est une fonction $\phi : \{0,1\}^* \to \{0,1\}$. Dans notre contexte, une fonction booléenne représente un *problème*, d'où l'identification terminologique parfois.

Définition. ***(Calcul d'une fonction par une machine)***

Soit ϕ une fonction booléenne. Soit T une machine de Turing. On dit que T *calcule* ϕ si pour chaque $x \in \{0,1\}^*$, si x est l'entrée de T, $\phi(x)$ est la sortie.

VOC On dit aussi qu'on *calcule* $\varphi(x)$ si la machine à entrée x a pour sortie $\varphi(x)$.

Définition. ***(Temps de calcul)***

Soit ϕ une fonction booléenne. Soit T une machine de Turing calculant ϕ. Soit $x \in \{0,1\}^*$. Le *temps de calcul* de $\phi(x)$ par T est le nombre d'étapes de T avant que T ne s'arrête.

Le temps de calcul peut être infini.

2.1.5 Turing-complétude

2.1.6 Machine de Turing non déterministe

Définition. ***(Machine de Turing non déterministe)***

Une *machine de Turing non déterministe* ou *non déterminée* ou *MTND* est une machine avec deux fonctions de transitions, par exemple δ_0 et δ_1.

Définition. ***(Calcul d'une fonction par une machine non déterministe)***

Soit ϕ une fonction booléenne. Soit T une machine de Turing non déterministe. On dit que T *calcule* ϕ si pour chaque $x \in \{0,1\}^*$, alors

(i) $\phi(x) = 1$ si et seulement si l'on peut trouver une suite de configurations $C_0,\ldots,C_m, m \in \mathbb{N}$, où C_0 est la configuration initiale avec entrée x, de sorte que l'état de T dans C_m est sa halte, sa sortie est 1, et pour tout $i \in [\![1,m]\!]$, $C_i = \delta_0(C_{i-1})$ ou $\delta_1(C_{i-1})$;

(ii) $\phi(x) = 0$ si et seulement si l'on peut trouver une suite de configurations $C_0,...,C_m, m \in \mathbb{N}$, où C_0 est la configuration initiale avec entrée x, de sorte que l'état de T dans C_m est sa halte, sa sortie est 0, et pour tout $i \in [\![1,m]\!]$, $C_i = \delta_0(C_{i-1})$ ou $\delta_1(C_{i-1})$.

Définition. ***(Temps de calcul par une machine non déterministe)***

Soit ϕ une fonction booléenne. Soit T une machine de Turing non déterministe calculant ϕ. Soit $x \in \{0,1\}^*$. Le *temps de calcul* de $\phi(x)$ par T est la longueur minimale d'une suite de configurations du point (i) de la définition précédente si $\phi(x) = 1$ et du point (ii) si $\phi(x) = 0$.

Remarque. Cette définition est cohérente avec la définition du cas déterministe : si $\delta_0 = \delta_1$, on retrouve le calul d'une fonction booléenne par une machine de Turing.

2.1.7 Machine prenant conseil

2.1.8 Automates

2.1.9 Machine de Turing universelle

2.1.10 Réalisation d'une machine de Turing

2.2 Circuits booléens

2.2.1 Définitions

Définition. ***(Circuit booléen)***

Un *circuit booléen* est un graphe orienté acyclique fini, dont les sommets sont étiquetés par $\wedge$ (une *porte ET*), $\vee$ (une *porte OU*) ou $\neg$ (une *porte NON*), sauf les sommets de degré d'entrée nul au sens des graphes orientés, qui sont les *entrées*. Les sommets de degré de sortie nul sont les *sorties*. On impose de plus qu'un sommet étiqueté par $\neg$ a toujours un unique fils.
La *taille* du circuit booléen est le nombre de sommets, soit de portes et d'entrées réunies.

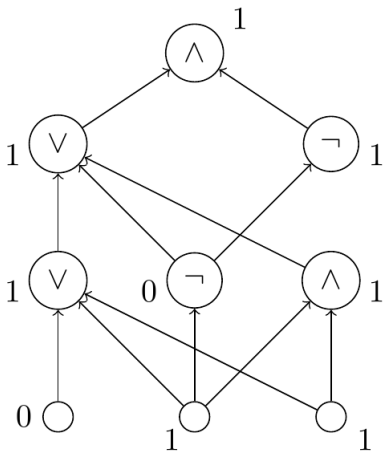

FIGURE 2.2.1 : *Exemple de circuit booléen.* —
On a de plus étiqueté les sommets pour associer au circuit une application booléenne selon la méthode décrite ci-dessous.

Méthode. ***(Application pseudo-booléenne associée à un circuit booléen)***

À un circuit booléen à n entrées rangées dans un certain ordre et m sorties, on peut associer une application $\varphi : \{0,1\}^n \to \{0,1\}^m$ défini récursivement : si les entrées sont $v_1,...,v_n$, pour calculer un certain $\phi(x)$ où $x = (x_1,...,x_n)$, on donne la valeur x_i en v_i, en respectant les règles suivantes.

1. La valeur d'une porte ET est 1 si et seulement si tous ses fils[a] ont la valeur 1.
2. La valeur d'une porte OU est 1 si et seulement s'il a aun moins un fils de valeur 1.
3. Une porte NON a un seul enfant, et sa valeur est l'opposée de la valeur de l'enfant.

En particulier, à un circuit booléen à m sorties, on peut associer une fonction booléenne. Réciproquement, si ϕ est booléenne, on dit qu'un circuit booléen *calcule* ϕ si ϕ est l'application booléenne associée à ce circuit.

[a] On rappelle que dans un graphe orienté $s_1 \to s_2$, s_1 est le fils de s_2.

Lemme

Chaque application $\phi : \{0,1\}^n \to \{0,1\}$ peut être calculée par un circuit booléen de taille inférieure à $2^{n+1} + n + 1$.

⊛ *(Idée de la preuve.)* Soit $X = \{x \in \{0,1\}^n \mid \phi(x) = 1\}$. Pour chaque $x \in X$, il existe un circuit dont l'unique sortie est étiquetée par 1 si et seulement si l'entrée est x. Si les sommets d'entrée sont $v_1,...,v_n$, si $x_i = 0$, on attache une porte NON à v_i, puis l'on attache une porte ET à toutes ses portes NON et aux sommets v_i tel que $x_i = 1$. On fait cela pour chaque $x \in X$, et l'on attache enfin une porte OU à toutes ses portes ET et c'est terminé. ■

2.3 Classes de complexité usuelles

2.3.1 Classes en temps

Principe. ***(Classes de complexité (temporelle))***

Une *classe de complexité* est un sous-ensemble de l'ensemble des fonctions booléennes caractérisé par l'existence d'une machine de Turing (dépendant de la fonction) qui la calcule en un temps de calcul majoré (dépendant de la fonction, indépendant de la machine). Cette majoration doit être en ordre de grandeur indépendante de la fonction.

2.3.1.1 La classe P

Définition. ***(Classe de complexité* P*)***

La *classe de complexité* P est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p tel que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing T qui calcule ϕ avec un temps de calcul de $\phi(x)$ d'au plus $p(|x|)$. Autrement dit, toute entrée de ϕ se calcule par une TM T en temps polynomial indépendant de la machine et de l'entrée.

Exemples. ***(Problèmes de classe* P*)***

1. Étant donné un graphe fini et deux sommets de ce graphe, pour $k \in \mathbb{N}$, le problème d'existence d'un chemin de longueur k sur ce graphe entre ces deux sommets, est encodable par une machine de Turing et se calcule en temps polynomial.

2.3.1.2 La classe NP

Définition. ***(Classe de complexité* NP*)***

La *classe de complexité* NP est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p tel que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing non nécessairement déterministe T qui calcule ϕ avec un temps de calcul de $\phi(x)$ d'au plus $p(|x|)$. Autrement dit, toute entrée de ϕ se calcule par une TM ou une TMND T en temps polynomial indépendant de la machine et de l'entrée.

Exemples. ***(Problèmes de classe* NP*)***

1. Le problème d'existence d'un cycle hamiltonien dans un graphe est de classe NP.

Fait. ***(Inclusion* P $\subseteq$ NP*)***

P $\subseteq$ NP.

Conjecture ?

P = NP.

Heuristique

On croit que la hiérarchie polynomiale est une vraie hiérarchie, mais en général, il est très difficile de montrer qu'une classe de complexité est strictement incluse dans une autre. Au-delà du problème très célèbre de savoir si oui ou non P = NP, on ignore encore si l'égalité est infirmée pour des classes de complexité beaucoup plus faibles.

Théorème. ***(Définition équivalente de la classe*** NP***)***

Une fonction booléenne $\phi \in$ NP si et seulement s'il existe un polynôme $p \in \mathbb{R}[X]$ et une application $\psi \in$ P telle que $\forall x \in \{0,1\}^* \quad \phi(x) = 1 \iff \exists y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$.

▷ Si ϕ peut être calculée en temps polynomial par une machine de Turing non déterministe, pour chaque x on peut choisir une suite $\varepsilon_1,...,\varepsilon_m$ où $m \in \mathbb{N}, m \leqslant p(|x|)$ telle que si l'on choisit les fonctions de transition $\delta_{\varepsilon_1},...,\delta_{\varepsilon_m}$, on calcule $\phi(x)$. Alors, soit ψ l'application qui calcule $\phi(x)$ étant donnés x et $\varepsilon_1,...,\varepsilon_m$. Il est clair que $\psi \in P$.

Réciproquement, si une telle ψ existe, on peut utiliser une machine de Turing non déterministe pour écrire y et calculer $\psi(x,y)$. ■

La quantification introduite dans cette caractérisation rend intuitive la description de plusieurs autres classes de complexité. Par exemple la

Définition. ***(Classe de complexité*** co-NP***)***

Une fonction booléenne $\phi \in$ co-NP si et seulement s'il existe un polynôme $p \in \mathbb{R}[X]$ et une application $\psi \in$ P telle que $\forall x \in \{0,1\}^* \quad \phi(x) = 1 \iff \forall y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$.

Reformulation pratique. ***(Calcul de*** co-NP***)***

co-NP $= 1 - NP = \{1 - \phi, \phi \in \text{NP}\}$.

Cette autre définition n'est donc pas du tout symétrique à la première !

Conjecture ?

NP = co-NP.

2.3.1.3 La classe PH

On souhaite généraliser les trois constructions précédentes P, NP et co-NP. Pour définir la classe PH, on a besoin de sous-classes définies de manière imbriquées, qui sont les classes $(\Sigma_n)_{n\in\mathbb{N}}$ et $(\Pi_n)_{n\in\mathbb{N}}$.

On définit par récurrence :

- ⋆ $\Sigma_0 = \Pi_0 = \mathrm{P}$;
- ⋆ $\Sigma_1 = \mathrm{NP}$, $\Pi_1 = \text{co-NP}$;
- ⋆ soit $i \in \mathbb{N}$. On caractérise : $\phi : \Sigma_{i+1}$ s'il existe $\psi \in \Pi_i$ telle qu'il existe p polynomiale telle que $\forall x \quad \phi(x) = 1 \iff \exists y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$; de même, $\phi \in \Pi_{i+1}$ s'il existe $\psi \in \Sigma_i$ telle qu'il existe p polynomiale telle que $\forall x \quad \phi(x) = 1 \iff \forall y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$.

Définition. ***(Classe de complexité*** PH***)***

On définit $\mathrm{PH} = \bigcup_{n=0}^{\infty} (\Sigma_n \cup \Pi_n)$.

Fait. ***(La classe*** PH ***est volumineuse)***

$\mathrm{P} \subseteq \mathrm{PH}$, $\mathrm{NP} \subseteq \mathrm{PH}$ et $\text{co-NP} \subseteq \mathrm{PH}$.

2.3.1.4 Les classes EXPTIME et NEXPTIME

Définition. ***(Classe de complexité*** EXPTIME***)***

La *classe de complexité* EXPTIME est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p tel que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing T qui calcule ϕ avec un temps de calcul de $\phi(x)$ d'au plus $\exp(p(|x|))$.

Définition. ***(Classe de complexité*** NEXPTIME***)***

La *classe de complexité* NEXPTIME est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p tel que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing non nécessairement déterministe T qui calcule ϕ avec un temps de calcul de $\phi(x)$ d'au plus $\exp(p(|x|))$.

2.3.1.5 La classe P/poly

Définition. ***(Classe de complexité*** P/poly***)***

Une fonction booléenne $\phi \in \mathrm{P/poly}$ si et seulement s'il existe un polynôme $p \in \mathbb{R}[X]$ et pour chaque $n \in \mathbb{N}$, une suite finie $(y_i) \in \{0,1\}^{p(n)}$ et une application $\psi_n \in \mathrm{P}$ telle que $\forall x \in \{0,1\}^* \quad \phi(x) = \psi(x, y_n)$. On appelle une telle $(y_i)_{i \in [\![1, p(n)]\!]}$ une *suite de conseil*.

2.3.2 Classes en espace

2.3.2.1 La classe PSPACE

Définition. ***(Classe de complexité*** PSPACE***)***

La *classe de complexité* PSPACE est l'ensemble des fonctions booléennes telles qu'il existe un polynôme en une variable p et une machine de Turing T qui calcule ϕ tels que pour tout $x \in \{0,1\}^*$, la position de T pendant le calcul est comprise entre $-p(|x|)$ et $p(|x|)$ à translation près du ruban, ou ce qui est équivalent (pourquoi ?), dans un intervalle de cases de longueur $p(|x|)$.

Exemple fondamental. ***(Problème de classe*** PSPACE***)***

Le problème de détermination du gagnant au jeu de go, si les joueurs utilisent des stratégies optimales, n'est pas PH mais est PSPACE.

2.3.2.2 La classe L

On veut définir la classe des problèmes n'utilisant que des ressources « en quantité logarithmique ». Problème, une telle restriction telle quelle est évidemment trop forte : simplement l'écriture de l'entrée dans un ruban n'est pas un problème résolu en temps logarithmique, mais linéaire ! Il faut donc y aller plus subtilement.

Définition. ***(Classe de complexité*** L***)***

La *classe de complexité* L est l'ensemble des fonctions booléennes telles que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing T à deux rubans dont le premier est à lecture seule, contient pour entrée x, et le deuxième est le ruban de travail, de sorte que pour chaque x, la machine ne regarde que $\mathcal{O}(\log(|x|))$ de cases du ruban de travail et calcule $\phi(x)$.

Exercice 1

On dit qu'une fonction booléenne $\phi \in \mathrm{LP}$ si pour tout $x \in \{0,1\}^*$, il existe une machine de Turing à deux rubans dont le premier est à lecture seule, contient pour entrée x, et le deuxième est de travail et tel que pour chaque x, la machine ne regarde que $\mathcal{O}(\log(p(|x|)))$ de cases du ruban de travail et calcule $\phi(x)$. Montrer que $\mathrm{LP} = \mathrm{L}$.

Heuristique

C'est la complexité minimale raisonnable.

L'inclusion extrêmement grossière suivante est peut-être une égalité, car on ne sait pas si

Conjecture ?

$\mathrm{L} \subsetneq \mathrm{PH}$.

Mnémonik : on ne peut pas prouver que les miracles n'existent pas...

Définition. ***(Classe de complexité*** NL***)***

La *classe de complexité* NL est l'ensemble des fonctions booléennes telles que pour tout $x \in \{0,1\}^*$, il existe une machine de Turing non déterministe T à deux rubans dont le premier est à lecture seule, contient pour entrée x, et le deuxième est le ruban de travail, de sorte que pour chaque x, la machine ne regarde que $\mathcal{O}(\log(|x|))$ de cases du ruban de travail et calcule $\phi(x)$.

On voudrait dire : *"(une fonction booléenne $\phi \in$ NP si et seulement s'il existe un polynôme $p \in \mathbb{R}[X]$ et une application $\psi \in$ L telle que $\phi(x) = 1 \iff \exists y \in \{0,1\}^{p(|x|)} \quad \psi(x,y) = 1$)*, mais ce n'est pas le cas.

Exercice 2

Démontrer l'affirmation précédente.

On peut bel et bien caractériser, de même qu'avec P et NP, la classe NL à partir de L, mais il faut être plus fin.

Théorème. ***(Définition équivalente de la classe*** NL***)***

Une fonction booléenne $\phi \in \mathrm{NL}$ si et seulement si pour tout $x \in \{0,1\}^*$, il existe une machine de Turing à trois rubans dont le premier est à lecture seule et contient l'entrée x, le deuxième est à lecture unique et le troisième est un ruban de travail, dont la machine n'utilise qu'une quantité logarithmique de cases, avec : $\phi(x) = 1$ si et seulement s'il existe

une variable y et ψ calculée par une machine de Turing telle que si x et dans le premier ruban et y est dans le deuxième, alors la sortie de T est $\varphi(x)$.

⊛ *(Idée de la preuve.)* C'est un peu plus subtil et laissé aux lecteurs plus aiguisés. ■

2.4 Relations entre classes de complexité

2.4.1 Relations évidentes entre classes de complexité

2.4.2 Relations moins évidentes entre classes de complexité

Pour établir d'autres relations entre des classes de complexité, on aura besoin de la notion de

2.5 Complétude

La complétude est le phénomène bizarre tel que, dans une classe de complexité, si un problème est résoluble en temps raisonnable, alors tous les autres de la classe le sont. Après avoir introduit les problèmes NP-complets, qui sont les plus célèbres, on généralise à d'autres classes de complexité.

2.5.1 Introduction : problèmes NP-complets

Définition. ***(Réduction d'applications booléennes)***

Soient ϕ,ψ deux fonctions booléennes. Soit $\theta : \{0,1\}^* \to \{0,1\}^*$ une *application booléenne à plusieurs sorites*. On dit que θ est un *réduction* de ψ à ϕ si $\psi = \phi \circ \theta$.
Si θ est calculable selon une certaine classe de complexité, la réduction est dite de cette complexité.

VOC Si par exemple θ est calculable en temps polynomial, on dit que θ est *calculable en temps polynomial*.

Définition. ***(NP-complétude)***

Un élement $\phi \in$ NP est dite NP-*complet* si pour tout $\psi \in$ NP, il existe une réduction polynomiale de ψ à ϕ.

On essaie maintenant de voir qu'il existe de tels problèmes.

Exemple fondamental. *(CIRCUITSAT)*

CIRCUITSAT est le problème où l'entrée est un circuit de n entrées est la sortie est 1 si et seulement s'il existe un choix d'entrées du circuit tel que la sortie du circuit égale 1.

Proposition

CIRCUITSAT est NP-complet.

▷ Soit $\psi \in \text{NP}$ et soient $\omega \in cP$ et p un polynôme, tels que $\psi(x) = 1 \iff \exists y \in \{0,1\}^{p(|x|)} \quad \omega(x,y) = 1$. D'après un lemme déjà vu, pour tout n, on peut construire en temps polynomial un circuit C_n qui calcule ω pour les entrées de taille $n + p(n)$. Pour chaque x, on peut donc construire en temps polynomial un circuit C_x qui est la simplification de C_n où les n premières entrées sont fixées à $x_1,...,x_n$. Alors $C_x(y) = 1 \iff \omega(x,y) = 1$. Donc $\psi(x) = 1 \iff \text{CIRCUITSAT}(C_x) = 1$. ■

Définissons un autre problème très pratique pour démontrer que beaucoup de problèmes sont NP-complets.

Définition. ***(Littéraux, clauses, formules, satisfaisabilité)***

Soient $x_1,...,x_n$ des variables booléennes. Un *littéral* est une expression x_i ou $\neg x_i$.
Une *clause* est une disjonction finie de littéraux. La *taille* de la clause est le nombre de littéraux minimal, *i.e.* sans redondance ou vrais, apparaissant.
Une *formule (en forme normale conjonctive)* est une conjonction de clauses. La *taille* d'une formule est la somme des tailles des clauses la constituant.
Une formule est *satisfaisable* s'il existe un choix de valeurs des variables telle que la valeur de la formule égale 1.

Exercice 3 *(Exemple d'une satisfaisabilité uniquement réalisée)*

Montrer que la formule $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3)$ est satisfaisable pour $x_1 = x_2 = x_3 = 1$ et que c'est la seule solution.

Définition. *(nSAT)*

Soit $n \in \mathbb{N}$. nSAT est le problème de déterminer si une formule dont toutes les clauses ont une taille inférieure à n est satisfaisable.

Proposition

3SAT est NP-complet.

▷ Il suffit d'exhiber une réduction polynomiale de CIRCUITSAT à 3SAT. Soit C un circuit booléen de taille n et de degré d'entrée $\leqslant 2$. Pour chaque sommet, on prend une variable donnée. Pour

chaque porte, on trouve une formule de 3SAT qui dit que la valeur du sommet est la valeur dans le circuit :

- ⋆ Si x_i correspond à une porte NON et x_j à son entrée, on prend la formule $(x_i \vee x_j) \wedge (\neg x_j \vee \neg x_i)$.
- ⋆ Si dans le circuit $x_i = x_j \wedge x_k$ ce qui correspond à une porte ET, on veut que $x_i \implies x_j \wedge x_k$ et $x_j \wedge x_k \implies x_i$. On fait correspondre $\neg x_i \vee (x_j \wedge x_k) \rightsquigarrow (\neg x_i \vee x_j) \wedge (\neg x_i \vee x_k)$ et $(\neg x_j \vee \neg x_k \vee x_i)$.
- ⋆ Les portes OU sont traitées de manière similaire.

On a donc construit en temps polynomial une formule qui imite le circuit. Si la variable x_r correspond à la sortie du circuit, on ajout la clause x_r. Cette formule est satisfaisable si et seulement si le circuit est satisfaisable. ■

2.5.2 Définition générale de la complétude

Définition. ***(Classe-complétude)***

Soit C une classe de complexité. Un élement $\phi \in C$ est dite *C-complet* si pour tout $\psi \in C$, il existe une réduction polynomiale de ψ à ϕ.

Pour une classe C en espace, une réduction polynomiale est par définition encore calculée en temps polynomial dans la définition précédente. Heuristiquement, les réductions ont besoin d'être moins puissantes que les classes de complexité considérées pour n'être pas triviales.

2.5.3 Un problème PSPACE-complet

On commence par donner une définition alternative de formule.

Propriété. ***(Définition récursive des formules)***

Une *formule* en n variables booléennes $(x_1,...,x_n)$ est définie par récurrence de la manière suivante : chaque x_i est une formule et si ϕ et ψ sont deux formules, alors $\neg\phi$, $\phi \wedge \psi$ et $\phi \vee \psi$ sont des formules.
De plus, la taille d'une formule est la somme des tailles de ses parties sachant que la taille de la formule x_i est 1.

Définition. ***(Formule quantifiée)***

Une *formule booléenne quantifiée* est une expression de type

$$Q_1 x_1 ... Q_n x_n \quad \phi(x_1,...,x_n)$$

où chaque Q_i représente au choix $\forall$ ou $\exists$, et ϕ est une formule booléenne.

Remarque. Une formule quantifiée peut encore être vraie ou fausse.

Définition. *(FBQV)*

Le problème FBQV pour « formule booléenne quantifiée vraie » est le problème de déterminer si une formule quantifiée est vraie.

Lemme

FBQV $\in$ PSPACE.

▷ Soit f une formule quantifiée $Q_1x_1...Q_nx_ng(x_1,...,x_n)$ où g est une formule booléenne de m variables. Soit $e(m,n)$ l'espace qu'il faut pour déterminer une telle formule est vraie. Pour déterminer si f est vraie, il suffit de déterminer si elle est vraie quand $x_11 = 0$ et si elle est vrai quand $x_1 = 1$. On peut déterminer cela en espace

$$\underbrace{1}_{\text{pour enregistrer le résultat quand } x_1=0} + \underbrace{e(m,n-1)}_{\text{déterminer si la formule simplifiée est vraie}} + \underbrace{\mathcal{O}(m).}_{\text{calculer la formule simplifie}}$$

Par suite, $e(m,n) = \mathcal{O}(mn)$. ■

Proposition

FBQV est PSPACE-complet.

▷ Montrons que pour toute $\phi \in$ PSPACE, il existe une réduction en espace polynomiale de ϕ à FBQV. La preuve ressemble à celle du théorème de Savitch. Soit donc $\phi \in$ PSPACE. Soit T une machine de Turing calculant ϕ en espace polynomial. Pour chaque x, soit G_x le graphe des configurations de T quand l'entrée est x. On a $\phi(x) = 1$ si et seulement s'il existe un chemin orienté de C_{init} à $C_{accepte}$. Il existe une formule f de taille polynomiale telle que pour chaque couple (C,C') de configurations, $f(C,C') = 1$ si et seulement si $\delta_T(C) = C'$. Ainsi $\theta_k(C,C') = 1$ si et seulement s'il existe un chemin de C à C' de longueur inférieure à 2^k. Soit e_k la taille d'une formule quantifiée nécessaire pour calculer θ_k. Alors $\theta_{k+1}(C,C')$ est vraie si et seulement si $\exists C'' \quad \theta_k(C,C'') \wedge \theta_k(C'',C')$.

On ne peut pas utiliser cette formule, car elle est trop grande, mais on peut utiliser la formule équivalente suivante : $\exists C'' \; \forall D_1,D_2 \quad (D_1 = C \wedge D82 = C'') \vee (D_1 = C'' \wedge D_2 = C'))implies\theta_k(D_1,D_2)$. Chaqeu égalité peut être écrite en forme d'une formule booléenne de taille $p(n)$, donc $e_{k+1} \leqslant e_k + p(n)$. La taille de G est $2^{\mathcal{O}(p(n))}$, donc quand $k \geqslant p(n)$, $\theta_(C_{init},C_{accepte}) = 1$ si et seulement si $\phi(x) = 1$ et il y a une formule booléenne quantifiée pour θ_k de taille polynomiale. ■

On peut en déduire

2.5.4 Relation NL = co-NL

Proposition

STCONN ∈ co-NL.

Pour montrer cela, on a besoin de la définition suivante.

Définition. ***(Calculabilité logarithmique implicite)***

Une application pseudo-booléenne $\phi : \{0,1\}^* \to \{0,1\}^*$ est *implicitement calculable en espace logarithmique (ICEL)* si les deux applications

$$(x,i) \mapsto \phi(x)_i$$

et

$$(x,i) \mapsto \mathbb{1}_{\{i \leqslant |\phi(x)|\}}$$

sont dans L.

Lemme. ***(Stabilité de la calculabilité log implicite par composition)***

Si ϕ et ψ sont deux applications ICEL, alors $\psi \circ \phi$ est ICEL.

▷ Pour claculer $\psi(\phi(x))$ étant données des machines de Turing convenant pour ϕ et ψ respectivement T_ϕ et T_ψ, on commence par suivre les opérations de T_ψ, et chaque fois que T_ψ veut lire une des $\phi(x)_i$, on utilise $_\phi$ pour calculer $\phi(x)_i$. ■

Lemme. ***(Stabilité de la calculabilité log implicite par réduction)***

Si $\phi \in$ L, respectivement NL, et θ est ICEL, alors $\phi \circ \theta \in$ L, respectivement NL.

▷ Si $\phi \in L$ et φ est ICEL, alors $\phi \circ \theta$ est ICEL donc trivialement $\phi \circ \theta \in$ L puisqu'elle n'a qu'une sortie. Si $\phi \in$ NL, on peut imiter la preuve du lemme précédent pour obtenir une machine de Turing non déterministe qui calcule $\phi \circ \theta$. ■

Définition-propriété. ***(Définition équivalente de la* NL*-complétude)***

Un problème ϕ est NL-*complet* si pour tout $\psi \in$ NL, il existe une application θ ICEL telle que $\psi = \phi \circ \theta$.

Proposition

STCONN est NL-complet.

▷ Utilisons la caractérisation précédente. Soit $\psi \in$ NL. Pour chaque x, soit G_x le graphe des configuartions pour une machine de Turing non déterministe T qui calcule ψ en espace logarithmique

quand l'entrée est x. G_x a un nombre polynomial de sommets et G_x est ICEL. De plus, $\phi(x) = 1$ si et seulement s'il existe un chemin L de C_{init} à $C_{accepte}$ dans G_x. Donc $\psi(x) = 1$ si et seulement si STCONN(G_x,C_{init},$C_{accepte}$) = 1. ■

Corollaire

NL = co-NL.

▷ Soit $\psi \in$ NL. Alors il existe θ ICEL telle que $\psi =$ STCONN$\circ\theta$, d'où $1-\psi = (1-$STCONN$)\circ\theta$. Par le théorème d'Immerman-Szelepcsényi, STCONN $\in$ co-NL et donc $1 -$ STCONN $\in$ L, d'où $(1 -$ STCONN$) \circ \theta \in$ NL d'où $1 - \psi \in cNL$, d'où $\psi \in$ co-NL.

Si $\psi \in$ co-NL, alors $1 - \psi \in$ NL d'où $1 - \psi \in$ co-NL d'où $\psi \in$ NL. ■

2.5.5 Applications P-complètes

Les applications P-complètes permettraient sous réserve d'existence dans une certaine classe de montrer des égalités non triviales entre classes.

Définition. ***(Application* P*-complète)***

Une application ϕ est P-complète si toute application ψ dans P a une réduction à ϕ en espace logarithmique.

Proposition

Si ϕ est P-complet et $\phi \in L$, alors $P = L$.

Proposition

Si ϕ est P-complet et $\phi \in$ NC$_{unit}$, alors $P = NC$.

Exemple

Si C est un circuit de n entrées et $x = \{0,1\}^n$, alors CIRCUITEVAL(C,x) est la sortie de C lorsque l'entrée est x.

Chapitre 3

Exercices

Difficulté des exercices :

- ● ○ ○ ○ ○ Question de cours, application directe, exercice purement calculatoire sans réelle difficulté technique
- ● ● ○ ○ ○ Exercice faisable, soit intuitivement, soit en employant des moyens rudimentaires ou des techniques déjà vues
- ● ● ● ○ ○ Exercice relativement difficile et dont la résolution appelle à une réflexion plus importante à cause d'obstacles techniques ou conceptuels, qui cependant devraient être à la portée de la plupart des étudiants bien entraînés
- ● ● ● ● ○ Exercice très exigeant, destiné aux élèves prétendant aux concours les plus difficiles, exercice « classique ».
- ● ● ● ● ● La résolution de l'exercice requiert un raisonnement et des connaissances extrêmement avancés, dépassant les attentes du prérequis. Il est presque impossible de le mener à terme sans indication. Bien qu'exigibles à très peu d'endroits, ces exercices sont très intéressants et présentent souvent des résultats forts.

Appendice

Bibliographie

[1] *Titre du livre*, Auteur du livre, date, maison d'édition

Table des figures

Liste des tableaux