Calculabilité et Complexité: CM6

Florian Bridoux

Université de Caen

2021-2022

Table des matières

Table des matières

Temps non déterministe

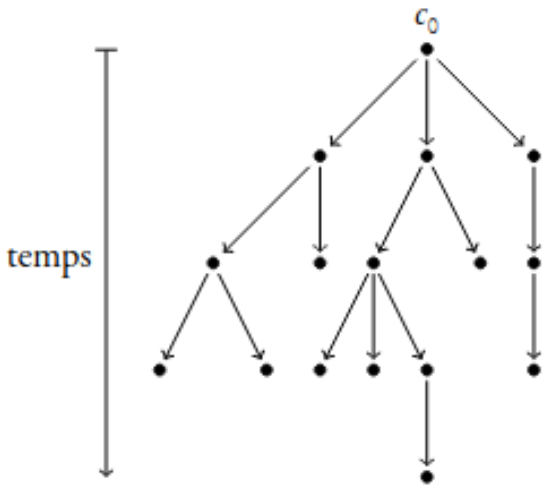
Jusqu'à présent, les machines de Turing que nous avons vues n'avaient aucun choix : la transition qu'elles effectuaient était déterminée uniquement par leur état et les cases lues par la tête. C'est pourquoi on les appelle *machines déterministes*. Si en revanche, à chaque étapes plusieurs transitions sont possibles, la machine a plusieurs exécutions possibles et est appelée *non déterministe*.

Définition: Machine de Turing non déterministe

Une machine de Turing $N = (\Sigma, \Gamma, B, Q, q_0, q_F)$ est dite non déterministe si δ n'est plus une fonction mais une relation:
$\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{\leftarrow, \rightarrow\})$.
À chaque étape, plusieurs transitions sont possibles : tout élément $(q, \Gamma_i), (q', \Gamma_j, d) \in \delta$ signifie que de l'état q, en lisant la lettre Γ_i, la machine N peut aller dans un état q' écrire Γ_j et se déplacer dans la direction d.

Temps non déterministe

Plutôt qu'une suite de configurations, le calcul d'une machine de Turing non déterministe est un arbre de configurations puisqu'une configuration a plusieurs successeurs possibles. Dans cet arbre, certaines configurations peuvent éventuellement apparaître plusieurs fois si elles sont atteintes par différents chemins.



Arbre de calcul d'une machine non déterministe

Temps non déterministe

Définition

Une exécution d'une machine non déterministe N est une suite de configurations compatible avec la relation de transition σ, d'une configuration initiale à une configuration finale.

- Une exécution de N est aussi appelée chemin, car il s'agit d'un chemin dans l'arbre de calcul, de la racine à une feuille.
- Le temps de calcul d'une machine de Turing non déterministe est le temps maximal d'une de ses exécutions, c'est-à-dire la hauteur de son arbre de calcul.
- De même, l'espace utilisé par une machine de Turing non déterministe est l'espace maximal utilisé par l'une de ses exécutions

Temps non déterministe

Définition

Le langage reconnu par une machine non déterministe N sur l'alphabet Σ est l'ensemble des mots $x \in \sigma^*$ tels qu'il existe un chemin acceptant dans le calcul $N(x)$.

Ainsi, pour accepter un mot x, il faut et il suffit qu'il y ait au moins une exécution qui mène à un état acceptant dans l'arbre de calcul de $N(x)$. En d'autres termes, un mot x est rejeté ssi aucun chemin n'est acceptant dans l'arbre de calcul de $N(x)$.

Exercice application

Écrire un programme haut-niveau qui résout le problème suivant en temps polynomial en utilisant du non-déterminisme.

COMPOSÉ
Entrée: Un entier x donné en binaire
Question: x est-t-il composé (non-premier)?

Temps non déterministe

Définition

Pour une fonction $t : \mathbb{N} \to \mathbb{N}$, la classe **NTIME**$(t)$ est l'ensemble des langages reconnus par une machine de Turing non déterministe N telle qu'il existe une constante α pour laquelle, sur toute entrée x, $N(x)$ fonctionne en temps $\leq \alpha t(|x|)$ (c'est-à-dire que toute branche de son arbre de calcul sur l'entrée x est de taille majorée par $\alpha t(|x|)$). Si T est un ensemble de fonctions, alors **NTIME**(T) désigne $\bigcup_{t \in T}$ **NTIME**(t).

Temps non déterministe

Les classes **NTIME**sont closes par union et intersection.

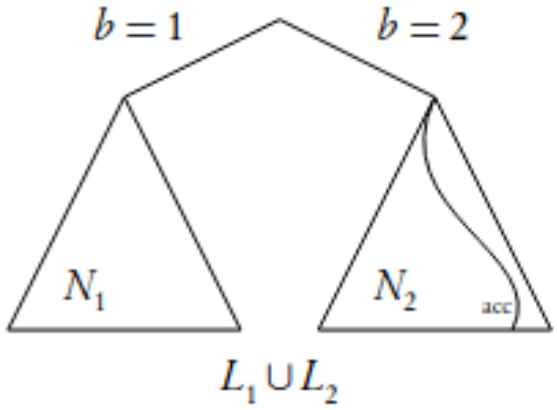
Proposition

Si $L_1, L_2 \in \mathbf{NTIME}(t)$ alors $L_1 \cup L_2 \in \mathbf{NTIME}(t)$ et $L_1 \cap L_2 \in \mathbf{NTIME}(t)$.

Preuve:

- Supposons que N_1, N_2 reconnaissent L_1, L_2 en temps t.

Temps non déterministe

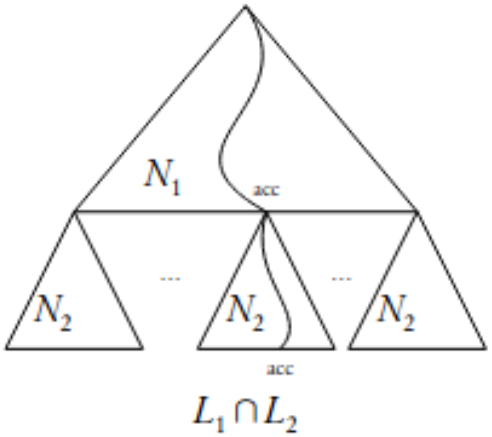


Union pour des machines non déterministes.

Pour reconnaître $L_1 \cup L_2$, on considère la machine $N_\cup$ suivante sur l'entrée x :

- deviner $b \in \{1, 2\}$;
- exécuter $N_b(x)$;
- accepter ssi le chemin simulé de $N_b(x)$ accepte.

Temps non déterministe



Intersection pour des machines non déterministes.

Pour reconnaître $L_1 \cap L_2$, on considère la machine $N_\cap$ suivante sur l'entrée x :

- Exécuter $N_1(x)$;
- Exécuter $N_2(x)$;
- accepter ssi les chemins simulés de $N_1(x)$ et $N_2(x)$ acceptent tous les deux.

Temps non déterministe

Question

Considérons $L \in \mathbf{NTIME}(t)$. Est-ce que son complémentaire $\overline{L}$ est dans $\mathbf{NTIME}(t)$ aussi? Autrement dit, est-ce que $\mathbf{NTIME}(t)$ est clos par complémentation comme $\mathbf{DTIME}(t)$ l'est?

Temps non déterministe

Question

Considérons $L \in \mathbf{NTIME}(t)$. Est-ce que son complémentaire $\overline{L}$ est dans $\mathbf{NTIME}(t)$ aussi? Autrement dit, est-ce que $\mathbf{NTIME}(t)$ est clos par complémentation comme $\mathbf{DTIME}(t)$ l'est?

C'est une question très importante dont la réponse n'a pas encore été prouvée mais la plupart des chercheurs en informatique pensent que non. En tout cas, on se convaincra au tableau qu'il ne suffit pas de prendre l'opposé de la réponse de N qui reconnaît L.

Temps non déterministe

Proposition

Pour une fonction $t : \mathbb{N} \to \mathbb{N}$,

$$\mathbf{DTIME}(t(n)) \subseteq \mathbf{NTIME}(t(n)) \subseteq \mathbf{DTIME}(2^{O(t(n))}).$$

Idée de la démonstration: Pour simuler une machine non déterministe fonctionnant en temps t(n) par une machine déterministe, on parcourt tous les chemins de calcul et on détermine si l'un d'entre eux accepte. Puisqu'il y a un nombre exponentiel de chemins, la machine déterministe prend un temps $2^{O(t(n))}$.

Table des matières

Temps non déterministe polynomial et exponentiel

Définition

La classe NP est l'ensemble des langages reconnus par une machine non déterministe en temps polynomial, c'est-à-dire

$$NP = \mathbf{NTIME}(n^{O(1)}) = \bigcup_{k \in \mathbb{N}} \mathbf{NTIME}(n^k).$$

La classe NEXP (ou NEXPTIME) est l'ensemble des langages reconnus par une machine non déterministe en temps exponentiel, c'est-à-dire

$$NEXP = \mathbf{NTIME}(2^{n^{O(1)}}) = \bigcup_{k \in \mathbb{N}} \mathbf{NTIME}(2^{n^k}).$$

Attention, une erreur souvent commise est de croire que NP signifie "non polynomial" alors que c'est une abréviation pour Nondeterministic Polynomial time.

Exemple problèmes dans NP et NEXP

CLIQUE:
Entrée: un graphe non orienté G et un entier k ;
Question: G a-t-il une clique de taille k ?

Exemple problèmes dans NP et NEXP

CLIQUE:
Entrée: un graphe non orienté G et un entier k ;
Question: G a-t-il une clique de taille k ?

Une clique de taille k est un ensemble de k sommets tous reliés les uns aux autres. Ce problème est dans NP : une machine non déterministe polynomiale pour le reconnaître peut deviner un ensemble de k sommets puis vérifier que ces k sommets sont tous reliés entre eux. Nous verrons que ce problème est " NP- complet ".

Exemple problèmes dans NP et NEXP

ARRÊT NEXP :
Entrée: *le code d'une machine non déterministe N et un entier k en binaire ;*
Question: est-ce que $N(\epsilon)$ s'arrête en $\leq k$ étapes ?

Exemple problèmes dans NP et NEXP

ARRÊT NEXP :
Entrée: le code d'une machine non déterministe N et un entier k en binaire ;
Question: est-ce que $N(\epsilon)$ s'arrête en $\leq k$ étapes ?

Ce problème est dans NEXP car on peut simuler $N(\epsilon)$ pendant k étapes grâce à une machine universelle non déterministe : puisque la taille de l'entrée k est $log(k)$, cela prend un temps exponentiel. Nous n'avons pas encore vu cette notion, mais mentionnons au passage que ce problème est NEXP-complet.

Temps non déterministe polynomial et exponentiel

Proposition (Caractérisation existentielle de NP et NEXP):

- Un langage A est dans NP ssi il existe un polynôme $p(n)$ et un langage $B \in P$ tels que

$$x \in A \Leftrightarrow \exists y \in \{0,1\}^{p(|x|)}, (x,y) \in B.$$

- Un langage A est dans NEXP ssi il existe un polynôme $p(n)$ et un langage $B \in P$ tels que

$$x \in A \Leftrightarrow \exists y \in \{0,1\}^{2^{p(|x|)}}, (x,y) \in B.$$

Le mot y justifiant l'appartenance de x à A est appelé preuve ou certificat.

Idée de la démonstration Le certificat y correspond au chemin acceptant dans la machine non déterministe reconnaissant le langage A.

Temps non déterministe polynomial et exponentiel

Remarque:

Si P est la classe des problèmes pour lesquels on peut trouver une solution efficacement, NP est la classe des problèmes pour lesquels on peut vérifier une solution efficacement. En effet, on nous donne une preuve y et on doit vérifier en temps polynomial que cette preuve est correcte. En d'autres termes, on nous donne une solution potentielle (un chemin acceptant) et on doit vérifier qu'il s'agit en effet d'une solution. On retrouve cela dans l'exemple précédent de problèmes NP : pour CLIQUE, on choisissait (de manière non déterministe) l'ensemble formant la clique et il fallait vérifier qu'il formait bien une clique. Cet exemple illustre un phénomène général : NP est la classe des problèmes "faciles à vérifier".

Temps non déterministe polynomial et exponentiel

Corollaire

$P \subseteq NP \subseteq EXP \subseteq NEXP$

Pour rappel, le théorème de la hiérarchie en temps nous donnait le résultat suivant:

Théorème

$P \subset EXP$

Un équivalent non déterministe de cet argument nous donne:

Théorème

$NP \subset NEXP$

Cependant, c'est une question ouverte de savoir si chacune des inclusions $P \subseteq NP$, $NP \subseteq EXP$ et $EXP \subseteq NEXP$ est stricte.

Les problèmes du prix du millénaire

Les problèmes du prix du millénaire sont un ensemble de sept défis mathématiques réputés insurmontables, posés par l'Institut de mathématiques Clay en 2000.
La résolution de chacun des problèmes est dotée d'un prix d'un million de dollars américains offert par l'institut Clay. En 2022, six des sept problèmes demeurent non résolus.
Liste des problèmes:

- Hypothèse de Riemann
- Conjecture de Poincaré - résolue par Grigori Perelman
- Problème ouvert $P = NP$
- Conjecture de Hodge
- Conjecture de Birch et Swinnerton-Dyer
- Équations de Navier-Stokes
- Équations de Yang-Mills

$P = NP$?

"Si quelqu'un prouve P = NP, la première chose qu'il devrait faire, c'est voler les 200 milliards de dollars qui existent en bitcoin. La deuxième, c'est résoudre tous les autres Problèmes du Prix du Millénaire" -Scott Aaronson

"Si P=NP, le monde serait très différent de celui que l'on pense qu'il est. Il n'y aurait [...] aucun écart fondamental entre résoudre un problème et reconnaître la solution une fois qu'elle est trouvée. Tous ceux capables d'apprécier une symphonie seraient Mozart; tous ceux capables de suivre un argument point par point seraient Gauss; tous ceux capables de reconnaître une bonne stratégie d'investissement seraient Warren Buffet" -Scott Aaronson

$P = NP$?

La question de savoir si $P = NP$ est ouverte et centrale en complexité. Pour reprendre la caractérisation précédente, cela revient à savoir si trouver une solution est "aussi simple" que de vérifier une solution. Par exemple:

- Est-il aussi facile de trouver une démonstration d'un énoncé mathématique que de vérifier qu'une démonstration donnée est correcte ?
- Est-il aussi facile de remplir une grille de Sudoku que de vérifier qu'un remplissage donné est bien une solution ?

On pourrait bien sûr multiplier les exemples. Notre expérience semble indiquer qu'il est plus difficile de trouver une solution, car il faut faire preuve d'imagination ou d'intuition : c'est une des raisons pour lesquelles de nombreuses personnes pensent que $P \neq NP$.

Table des matières

co-NP

On va maintenant définir la classe co-NP: l'ensemble des langages dont le complémentaire est dans NP.

Définition

La classe co-NP est l'ensemble des langages A tels qu'il existe une machine de Turing non déterministe N fonctionnant en temps polynomial et satisfaisant : $x \in A$ ssi tous les chemins de calcul de $N(x)$ sont acceptants.

Remarque

Comme on l'a vu auparavant, on ne sait pas décider dans NP le complémentaire de tout langage de NP : cela se traduit par le fait que la question " NP = coNP ? " est ouverte.

Exemple de problème dans co-NP

Le "complémentaire" de n'importe quel problème dans NP:

co-CLIQUE:
Entrée: un graphe non orienté G et un entier k ;
Question: G n'a aucune clique de taille k ?

Si la réponse est non, c'est facile à prouver: il suffit de deviner un ensemble de k sommets et de vérifier que c'est une clique. Si la réponse est oui, on ne sait pas...

Exemple de problème dans co-NP

Le "complémentaire" de n'importe quel problème dans NP:

Proposition (caractérisation universelle de coNP)

Un langage A est dans co-NP ssi il existe un polynôme $p(n)$ et un langage $B \in P$ tels que $x \in A \Leftrightarrow \forall y \in 0,1^{p(|x|)}, (x,y) \in B$.

Idée de la démonstration Le certificat y correspond au chemin refusant dans la machine non déterministe reconnaissant le langage A.