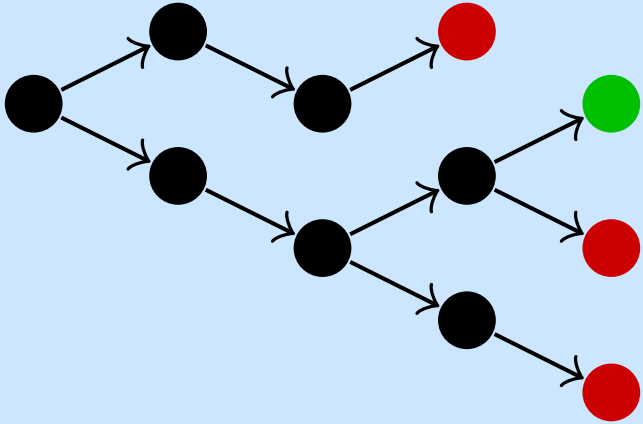
Calculabilité & Complexité

Complexité (3/4) : non déterminisme

Nicolas Ollinger (LIFO, Université d'Orléans)

M1 info, Université d'Orléans — S2 2024/2025



1. Dans l’épisode précédent...

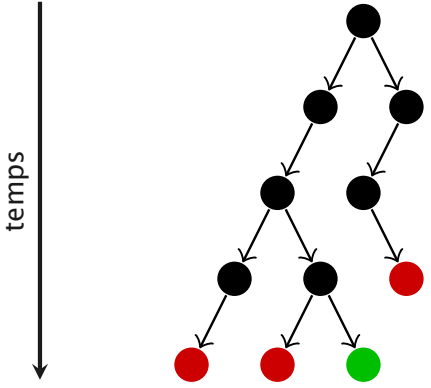
Formellement

Définition Une **MTND à k-rubans** est un tuple $(Q, \Gamma, \Sigma, \delta, q_0, B, q_a, q_r)$ où

- Q est l'ensemble fini des **états** ;
- Γ est l'**alphabet** fini de **travail** ;
- $\Sigma \subseteq \Gamma$ est l'**alphabet** fini **d'entrée** ;
- $\delta \subseteq \left((Q \setminus \{q_a, q_r\}) \times \Gamma^{k-1}\right) \times \left(Q \times \Gamma^{k-1} \times \{\blacktriangleleft, \blacktriangledown, \blacktriangleright\}^k\right)$ est la **relation de transition** ;
- $q_0 \in Q$ est l'**état initial** ;
- $B \in \Gamma \setminus \Sigma$ est le **symbole blanc** ;
- $q_a \in Q$ est l'**état d'acceptation** de la machine ;
- $q_r \in Q$ est l'**état de rejet** de la machine.

Arbre de configurations

Définition Un **calcul** d'une MTND sur une entrée $u \in \Sigma^*$ est un **arbre de configurations** obtenus en calculant toutes les transitions accessibles depuis la **configuration initiale** $c_0(u)$.



Langage reconnu

Définition Une **exécution** d'une MTND est un **chemin** d'un arbre de configurations : une suite de configurations compatible avec la relation de transition, d'une **configuration initiale** à une **configuration terminale**.

Temps de calcul MTND
$t_{\mathcal{M}}(u)$ = hauteur de l'arbre des configurations sur l'entrée u

$$t_{\mathcal{M}}(n) = \max_{u \in \Sigma^n} t_{\mathcal{M}}(u) \qquad \forall n \in \mathbb{N}$$

Définition Le **langage $L_{\mathcal{M}}$ associé** à une MTND $\mathcal{M}$ est l'ensemble des entrées pour lesquelles $\mathcal{M}$ possède une **exécution acceptante**.

Complexité en temps non déterministe

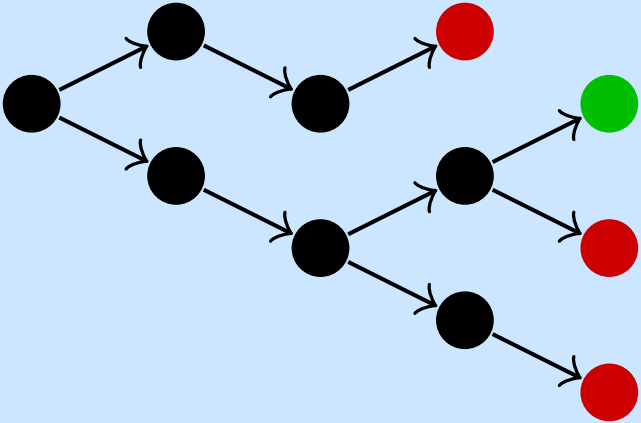
Définition Pour toute fonction $f : \mathbb{N} \to \mathbb{N}$, la classe $\mathrm{NTIME}(f(n))$ est l'ensemble des langages reconnus par une MTND $\mathcal{M}$ en temps borné par $\alpha f(n)$ pour un certain α.

$$\forall u \in \Sigma^* \qquad t_{\mathcal{M}}(u) \leqslant \alpha f(|u|)$$

Définition Si $\mathcal{F}$ est un ensemble de fonctions on note

$$\mathrm{NTIME}(\mathcal{F}) = \bigcup_{f \in \mathcal{F}} \mathrm{NTIME}(f(n))$$

$$\mathrm{NP} = \mathrm{NTIME}\left(\left\{n^k \,\middle|\, k \in \mathbb{N}\right\}\right)$$



2. Temps non-déterministe

Théorème de hiérarchie

Définition Une fonction $f : \mathbb{N} \to \mathbb{N}$ est **constructible en temps** s'il existe une constante α et une MT qui sur l'entrée 1^n (l'entier n en unaire) renvoie $1^{f(n)}$ (l'entier $f(n)$ en unaire) en temps $\leqslant \alpha f(n)$.

Théorème Pour toutes fonctions $f, g : \mathbb{N} \to \mathbb{N}$ avec $f(n) \neq 0$ et g croissante et **constructible en temps** avec $f(n+1) = o(g(n))$. Alors

$$\mathrm{NTIME}(f(n)) \subsetneq \mathrm{NTIME}(g(n))$$

Lemme Si $f(n) \leqslant g(n)$ pour tout n, $\mathrm{NTIME}(f(n)) \subseteq \mathrm{NTIME}(g(n))$.

Première séparation

$$\begin{aligned} \mathrm{NP} &= \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(n^k) \\ \mathrm{NEXP} &= \bigcup_{k\in\mathbb{N}} \mathrm{NTIME}(2^{n^k}) \end{aligned}$$

Corollaire $\mathrm{NP} \subsetneq \mathrm{NEXP}$

Idée $\mathrm{NP} \subseteq \mathrm{NTIME}\left(2^n\right) \subsetneq \mathrm{NTIME}\left(2^{n^2}\right) \subseteq \mathrm{NEXP}$

Exemples de problèmes dans NP

CLIQUE

entrée : *un graphe non orienté G et un entier k*
question : *le graphe G contient-il une clique de taille k ?*

SOMME PARTIELLE

entrée : *une liste d'entiers $a_1, \ldots, a_m$ et un entier cible s*
question : *existe-t-il une sous-liste dont la somme vaut s, c'est-à-dire $S \subseteq \{1, \ldots, m\}$ telle que $\sum_{i \in S} a_i = s$?*

Exemples de problèmes dans NEXP

ARRÊT NEXP

entrée : *une MTND $\mathcal{N}$ et un entier k en binaire*
question : *est-ce que $\mathcal{N}$ s'arrête sur l'entrée vide en temps $\leqslant k$?*

Certificats

Proposition Un langage A est dans NP si et seulement s'il existe un polynôme $p(n)$ et un langage $B \in \mathrm{P}$ tels que

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

On dit que y est un **certificat** (ou une **preuve**) pour x et que B est un **certificateur** pour A.

Proposition Un langage A est dans NEXP si et seulement s'il existe un polynôme $p(n)$ et un langage $B \in \mathrm{EXP}$ tels que

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

Interprétation

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**.

Exemple de certificats :

- Pour **CLIQUE** on peut prendre la liste des k sommets ;
- Pour **SOMME PARTIELLE** on peut prendre le sous-ensemble.

Padding

Proposition Si $\mathrm{P} = \mathrm{NP}$ alors $\mathrm{EXP} = \mathrm{NEXP}$.

Idée **Bourrer** l'entrée pour disposer de plus de temps de calcul.

Complémentaire

Définition Si C est une classe de complexité, la classe $\mathrm{co}C$ est l'ensemble des complémentaires des langages de C.

$$\mathrm{co}C = \left\{ \overline{A} \,\middle|\, A \in C \right\}$$

Proposition Un langage A est dans coNP si et seulement s'il existe un polynôme $p(n)$ et un langage $B \in \mathrm{P}$ tels que

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

Résumé

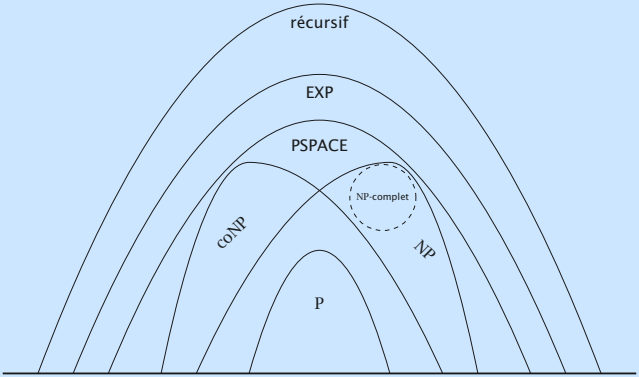
$$P \subseteq NP \subseteq EXP \subseteq NEXP, \qquad P \neq EXP \quad \text{et} \quad NP \neq NEXP$$

$$\text{Si} \quad P = NP \quad \text{alors} \quad EXP = NEXP$$

$$P = coP \quad \text{et} \quad EXP = coEXP$$

$$P \subseteq coNP \subseteq EXP$$

On ne sait pas si $P = NP$!



3. NP-complétude

Réductions many-one

Une technique pour dériver de nombreux problèmes indécidables à partir d'un premier consiste à utiliser des **réductions** : des pré-ordres sur les langages qui **préservent la récursivité**.

Définition Soient $A \subseteq \Sigma^*$ et $B \subseteq \Gamma^*$ deux langages. Une **réduction** (many-one) de A à B est une fonction totale calculable $f : \Sigma^* \to \Gamma^*$ telle que

$$u \in A \Leftrightarrow f(u) \in B \qquad \forall u \in \Sigma^* \quad .$$

Le langage A **se réduit** au langage B, noté $A \leqslant_m B$ s'il existe une réduction de A à B.

Proposition La relation $\leqslant_m$ est une relation de **pré-ordre**.

Réductions many-one polynomiales

En complexité, on s'intéresse à des réductions qui préservent la complexité des problèmes.

Définition Soient $A \subseteq \Sigma^*$ et $B \subseteq \Gamma^*$ deux langages. Une **réduction** (many-one) **polynomiale** de A à B est une fonction totale **calculable en temps polynomial** $f : \Sigma^* \to \Gamma^*$ telle que

$$u \in A \Leftrightarrow f(u) \in B \qquad \forall u \in \Sigma^* \quad .$$

Le langage A **se réduit en temps polynomial** au langage B, noté $A \leqslant_m^P B$ s'il existe une réduction de A à B.

Proposition La relation $\leqslant_m^P$ est une relation de **pré-ordre**.

Complétude

Proposition Les classe P, NP, EXP et NEXP sont **stables** par réductions polynomiales.

Définition Deux languages A et B sont **équivalents pour les réductions polynomiales**, noté $A \equiv_m^P B$, si $A \leqslant_m^P B$ et $B \leqslant_m^P A$.

Définition Un langage A est C**-difficile** pour une classe de complexité C si pour tout $B \in C$ on a $B \leqslant_m^P A$.

Définition Un langage A est C**-complet** pour une classe de complexité C si A est C-difficile et $A \in C$.