Complexité - Nicolas Ollinger

Alexandre Masson

15 Janvier 2013

Table des matières

1 non-déterminisme

$P = \cup_{k>0} DTIME(n^k)$ la classe des problèmes qu'on peut résoudre en temps polynomial.
$NP = \cup_{k>0} NTIME(n^k)$ la classe des problèmes vérifiables en temps polynomial

Rétrospective DFA : automate déterminisme , NFA non déterminisme.
ex : $(a+b)^*baba$ mot finissant par baba. Tout NFA non déterminisme, se code en DFA, mais au prix d'une complexité plus importante en nombre d'état.

Définition une MTND (MT non déterministe), est une MT avec comme table de transition :

$$T \subseteq (Qx\Gamma^k)^{input} x (Qx\Gamma^k x\{<-, v, ->\})^{output}$$

on redéfinie la relation de transition :

$c \vdash c'$ si T contient une transition applicable sur c qui produit c'

Une exécution d'une MTND partant d'une entrée u est une suite de transition valides de la configuration initiale associée à u à une configuration d'arrêt.

$$c_0 \vdash c_1 \vdash ... \vdash c_n$$

Une MTND M accepte un mot $u \in \Sigma^*$ s'il existe une exécution acceptée pour u.

	MT	MTND
Rejet	l'exécution doit s'arrêter sur q_{non}	toutes les exécutions doivent s'arrêter sur q_{non}
Acceptation	l'exécution s'arrête sur q_{oui}	un seul état à oui renvoie oui

remarque A un ralentissement linéaire près, on peut supposer que les choix non déterministes sont au plus binaires : pour tout configuration c :
- ou bien c n'as pas d'image (arrêt)
- ou bien $c \vdash c'$ unique.
- ou bien $c \vdash c' \& c \vdash c''$

exemple le 3-color, entrée : un graphe G=(V,E), sortie : G est il 3-colorable 3Color est reconnu par une MTND dont les exécutions s'arrêtent en temps polynomial en la taille de l'instance.
la machine fonctionne en deux temps.
- Choix non déterministe d'une couleur par sommet (O(|v|)
- Vérification déterministe que le coloriage choisit est valide : si oui accepte, sinon rejette.

Définition NTIME(f(n)) est la classe des langages reconnus par une MTND dont toutes les exécutions sont de longueur $\leq$ f(n) sur les entrées de taille n.

proposition Si $f(n) \leq g(n)$ alors $NTIME(f(n)) \subseteq NTIME(g(n))$
$DTIME(f(n)) \subseteq NTIME(f(n))$
$NTIME(f(n)) \subseteq DTIME(2^{O(f(n))})$
Idée : Il suffit de simuler les choix de la MTND par du backtracking

def $NP = \cup_{k>0} NTIME(n^k)$
$NEXP = \cup_{k>0} NTIME(2^{n^k})$
Question : Est ce que P NP, **rq :** $P \subseteq NP$
prop NP ets clos par union et intersection.
def $co\Im = \{!L | L \in \Im\}$ ou $\Im$ et une classe de langages.

Un problème B est un **certificateur** pour un problème A si $\forall x \in \Sigma^*, x \in A <=> \exists y \in \Sigma^* < x, y > \in B$ on dit que y est un certificat pour x.

thm Np la classe des langages qui possèdent un certificateur dans P avec certificats de longueur polynomiale en la longueur de l'instance.
ex 3Color $\in$ NP , cf exemple antérieur. certificateur pour 3COLOR.
sur l'entrée <G,C> avec colorisation des sommets vérifier que les arêtes ont des couleurs distinctes à chaque extrémité. Ce certificat est dans P(car il ne fais que parcourir les arêtes pour vérifier une propriété). la taille du certificat est polynomiale.

Démonstration

i (dans ce sens, on connait A de P , et B le certificateur, on essaie de construire un chemin qui accepte x dnas la machine M, ce chemin s'appelle y , si l'on suis le chemin y pour arrive en qoui ave x en entrée, alros y est certificat et le certif existe , donc B est certificateur.)
Soit A un langage qui possède un certificateur B $\in$ P avec certificat de longueur $\leq$ p(n) sur les instances de longueur n ou p est polynomiale en n.
montrons que A ets dans NP en construisant une MTND M qui reconnait A comme suit :

- choisir grace au non déterminisme un mot y de longueur $\leq$ p(|x|)
- exécuter une MT qui reconnait B sur l'entrée <x,y>
- accepter si (2) accepte

M s'exécute bien en temps polynomial en |x| et M accepte x ssi $\exists y : |y| \leq p(n)$ et <x,y> $\in$ B, si x$\in$ A.

ii (dans ce sens, on sais que le Y sera un certificat, puisqu'on chosiit un chemin acceptant de x dans l'exécution de M, donc si B repond ok , vu que le Y ets ok , cela veux dir que B est bon)
Soit $A \in NP$ On va construire un certificat $B \in P$ pour A avec certificats polynomiaux.
Soit M un MTND qui reconnais A.
On utilise les exécutions acceptantes de M comme certificats et B se content de

vérifier l'exécution. Sur l'entrée <x,y>B vérifie que Y est une exécution acceptable pour M sur l'entrée x. B est bien un certificateur dans P avec certificat polynomiaux pour A.
Np est la classe des problèmes vérifiables en temps polynomial.

3Color : E : graphe G, S : est il 3Colorable, certif : coloration, Verif : pas d'arete monochrome.

Hamilton : E/S idem : certif : une permutation des sommets, i.e le cycle, verif : les aretes existent bien.

CLIQUE(sous graphe complet du graphe) Entrée : un graphe et un entier k : certif : la clique, verfi : les aretes existe =elles

SAT (clause satisfiables) entrée : formule Φ en CNF, sortie : Φ est elle satisfaisable ?
variable : x,y,z, littéraux : var ou neg.var, clause : disjonction de littéraux, formules : conjonction de clauses, **certificat :** une affectation qui satisfait Φ, vérif : évaluer Φ **3SAT** SAT où Φ ont des clause d'ua moins 3 littéraux.

COMPOSITE entrée : entier n en binaire, question : existe il p et q > 1 tels que n=pq ?, certif : la paire <p,q> en binaire. vérif : multiplier p par q et comparer à n.

2 19 Mars 2013

2.1 NP-Completude

Au menu : théorème de Cook-Levin SAT$\in$ P <=> P=NP
remarque sens <= trivial car SAT $\in$ NP.
Définition $A \leq_p B$ si $\exists f : \Sigma^* - > \Sigma^*$ totale calculable en temps polynomiale telle que :$\forall x \in \Sigma^*, X \in A <=> f(x) \in B$

prop si $A \leq_p B$ et B $\in$ P alors A $\in$ P. Que B soit dans P implique qu'il existe un algo poly qui résout B,

def on dit que A est NP-difficile si quelquesoit le problème que je prend dans NP , ce problème est plus simple que A.($\forall B \in NP, B \leq_p A$
A est NPComplet si
A est NP-difficile **ET** A$\in$ NP.(**NE PAS L'OUBLIER**)

Remarque : si A est NP-complet, et A $\in$ P, alors P=NP. En effet soit B $\in$ NP, comme A est NP-difficile, d'après la def, $B \leq_p A$ et d'après la proposition ci dessus, B∈P . Donc $NP \subseteq P$, i.e P=NP.

thm Cook-Levin SAT est NP-complet. On sais que SAT ∈ NP, on va montrer que $\forall A \in NP, A \leq_p SAT$
soit $A \in NP$.soit M une MTND totale qui reconnaît A en temps polynomial p(n).
On peux supposer sans perte de généralité que M à un seul ruban, à choix binaire, à ruban semi infini, avec arrêt sur la case 0, exactement au temps p(n).
A l'instant 0 , le ruban contient le mot de taille N, si le temps d'arrêt est p(n), alors la ruban ne peux etre de taille plus grand que p(n) , car on parcours une case par étape , donc pour un mot donnée, on av obtenir des q_{oui} ou des q_{non} ,
$x \in A$ ssi ∃ un digne-espace temps de M sur l'entrée X, avec q_{oui} en (0,p(n)).

Idée : Pour montrer que $A \leq_p SAT$ on construit f, fonction de réduction , qui a un mot $x \in \Sigma^*$ associe une formule CNF $\Phi_x^{M,p}$ qui code les diagrammes espace-temps de M sur l'entrée x de sorte que $\Phi_x^{M,p} \in SAT$ ssi le diagramme possède des choix non-déterministes acceptant , c'est a dire avec q_{oui} en (0,p(x)).

On va découper $\Phi_x^{M,p}$ en plusieurs parties :
$\Phi_{sim}^{M,p(|x|)}(calculdelaMT) \wedge \Phi_{input}^{M,P,x}(entr\acute{e}e) \wedge \Phi_{accept}^{M,P(|x|)}(sortie)$

Pour chaque case du diagramme, elle sont codées par les variables booléennes, $x_1^{ij}, ..., x_k^{ij}$, en binaire.
les codes valides sont vérifiés par une formule CNF $\Phi_{val}(x_1^{ij}, ..., x_k^{ij})$
$\Phi_{sim}^{M,p(|x|)} = \wedge_{i=0}^{N} \wedge_{j=0}^{N} \Phi_{val}(x_1^{ij}, ..., x_k^{ij}) \wedge \Phi_{trans}^{M,N}$
la variable C_t code le choix non-det à l'instant t. (si le contenu est loin de la tête , il n'a aucune chance de changer entre t et t+1.
$\Phi_{trans}^{M,N} = \wedge_{i=0}^{N} \wedge_{j=0}^{N} \Phi_{delta}^{M,N}(x_1^{i-1j}, ..., x_k^{i-1j}, x_1^{ij}, ..., x_k^{ij}, x_1^{i+1j}, ..., x_k^{i+1j}, x_1^{i-1j+1}, ..., x_k^{i-1j+1}, x_1^{ij+1}, ..., x_k^{ij+1}, x_1^{i+1j+1}, ..., x_k^{i+1j+1})$
où Φ_{delta}^{M} vérifie que que le rectangle 3x2 cellules communiqués est une transition valide pour M

On obtient une formule CNF $\Phi_{sim}^{M,N}$ à kN^2 variables et de taille O(N^2). On est tous convaincu de savoir produire $\Phi_{sim}^{M,N}$ en temps poly.

$\Phi_{input}^{M,P,x}$ vérifie que la ligne 0 code via la config initial
$\Phi_{input}^{M,P,x} = \Phi_{case}^{M}(q_0, x_0), x_1^{00}, ..., x_1^{00} \wedge \wedge_{i=1}^{|x|-1} \Phi_{case}^{M}(x_i, x_1^{i0}, ..., x_k^{i0}) \wedge \wedge_{i=|x|}^{p(|x|)}(B, x_1^{i0}, ..., x_k^{i0})$
formule de ligne O(n)$^i = |x|$ est calculable en temps polynomial

$\Phi_{accept}^{M,N}$ vérifie que la case (0,n] contient bien q_{oui}
la formule $\Phi_x^{M,N}$ est donc calculable en temps polynomial en |x| et est satisfiable ssi l existe des choix non déterministes pour lequels M accepte x, i.e x in A.

Si A est NP-diificile et $A \leq_p B$ alors B est NP difficile, car B est plus compliqué que A., pour montrer que BLOP est np complet, on montre que blop in NP et blip $\leq_p$ BLOP , comme blop plus compliqué que blip et blip np , alors blop np
prop 3SAT et variante est NP-complet,
une formule CNF avec jsuqu'a 3 littéraux par clause
idem mais avec exactement
idem mais avec 3 littéraux distincts

démonstration
3SAT in NP : cf cours précédent
3SAT est NP-difficile, montrons $SAT \leq_p 3SAT$
Exhibons F calculable en temps ploynomial telle que
$\forall \Phi, \Phi \in SAT <=> f(\Phi) \in 3SAT$
no doit trouver un moyen de transformer les littéraux de tailel n en littéraux de taille 3 avec des conjonctions entre eux, pour respecter CNF.

F remplace chaque clause $(l_1 v l_2 v ... v l_k)$ par la formule $(l_1 v l_2 v \alpha_2) \wedge (\bar{\alpha_2} v l_3 n \alpha_3) \wedge ... \wedge (\bar{\alpha_{k-2}} v l_{k-1} v l_k)$ où $\alpha_2, ..., \alpha_{k-2}$ sont k-3 nouvelles variables, et cette formule est satisfiable ssi la clause l'est.

3 26 Mars 2013

3.1 NP-complétude suite

Lemme $3_{\neq}SAT \leq_p 3_{ex}SAT \leq_p 3_{\leq}SAT$
Démo Icic à chaque fois $A \leq_m B$ à prouver
avec $A \subseteq B$ et on sait séparer A de B en temps polynomial.
Proposition $3_{\leq}SAT \leq_p 3_{\neq}SAT$
Proposition $SAT \leq_p 3_{\leq}SAT$
Corollaire 3SAT est NP-Complet
Démonstration Astuce : la clause (PvQ) ou P et Q sont des disjonctions de littéraux est satisfiable si et seulement si $(Pvx) \wedge (\bar{x}vQ)$ est satisfiable ou x n'apparaît ni dans P ni dans Q.
=> Supposons (PvQ) satisfiable . Fixons une instanciaition σ qui satisfait (PvQ). ALors on construit une instanciation σ' qui satisfait $(Pvx) \wedge (\bar{x}vQ)$ ainsi
- si $\sigma(P)$ ets vraie on pose $\sigma'(x)$ = faux
- si $\sigma(P)$ est faux alors $\sigma(Q)$ ets vrai on pose $\sigma'(x)$ = vrai
- $\forall y, y \neq x$, $\sigma'(y) = \sigma(y)$

<= Supposons $(Pvx) \wedge (\bar{x}vQ)$ satisfait par ine instanciation σ alors σ satisfait aussi (P $\wedge$ Q). En effet si $\sigma(X)$ ets vraie alors $\sigma(Q)$ ets vraie, et si $\sigma(X)$ ets faux alors $\sigma(P)$ ets vraie.

Pour montrer $3_{\leq}SAT \leq_p 3_{\neq}SAT$, il faut associer à toute formule ϕ avec jusqu'a 3 lit/clauses, une formule $f(\phi)$ avec exactement 3 lit $\neq$/clause de sorte de ϕ satisfiable <=> $f(\phi)$ sat.
Il faut faire grossir les clauses à ou 2 lit $\neq$ ON applique l'astuce une ou deux fois. Cette transformation se calcule en temps polynomial.
Pour montrer qu e$SAT \leq_p 3_{\leq}SAT$
ϕ quelconque
$f(\phi)$ jusqu'a 3 lit/clause
on applique l'astuce pour faire maigrir les clauses :
$(l_1 v l_2 v ... v l_k)$ on l'applique k-1 fois -> $(l_1 v \alpha_1) \wedge (\bar{\alpha_1} v l_2 v \alpha_2) \wedge ... \wedge (\alpha_{\bar{k-1}} v l_k)$
cette transformation est calculable en temps polynomial donc $SAT \leq_p 3SAT$

```
CIRCUIT-SAT
entrée : un circuit boolean C avec des portes NON,ET,OU, et des entrées et une sortie
question : existe-t-il une affectation des entrées qui force la sortie de C à vrai
```

CIRCUIT-SAT $\in$ NP pour décoder CIRCUIT-SAT, il suffit de deviner une instanciation des entrées et d'évaluer le circuit, en temps polynomial.

Proposition CIRCUIT-SAT est NP-Complet.
Démonstration
1 CIRCUIT-SAT $\in$ NP (axiome)
2 $3SAT \leq_p CIRCUIT - SAT$

On code une formule ϕ de 3SAT par un circuit avec :
- une entrée /variable,
- une porte $\neq$ /littéral négatif
- deux portes OU par clauses
- k-1 portes ET entre les k clauses

Cette transformation est une réduction many-one PTIME de 3SAT en CIRCUIT-SAT.

```
VERTEX-COVER
Entrée : G=(V,E) un graphe et k appartient à N
question : existe-t-il un sous-ensemble U \subseteq V, U de taille k tel que toute arête
```

Proposition VERTEX-COVER est NP-Complet
Prove-it !
i VERTEX-COVER$\in$NP : un certificat est un sous ensemble de sommets qui couvre les arêtes de G
ii Mq 3SAT $\leq_p$ VERTEX-COVER

Soit ϕ une instance de 3SAT on construit $f(\phi)$ à l'aide des gadgets suivants :
- chaque variable x est représentés par : (x)—($\bar{x}$)
- chaque clause $(l_1 v l_2 v l_3)$ est représentée par : un triangle
- on relie chaque sommet des gadgets de clauses aux sommets des gadgets de variables de même label

– k = 2(n+m) avec m = #clause et n=#variables.
=> Si ϕ est satisfiable, on obtient une couverture de $f(\phi)$ de taille k=2m+n) ainsi :
on fixe une instance σ qui satisfait ϕ.

– si $\sigma(x)$ est vrai , on prend le sommet x de l'haltère x sinon le sommet $\bar{x}$
– pour chaque clauses $(l_1vl_2v...vl_n)$ avec $\sigma(l_1) =$ vrai , on prend les sommetl_2etl_3

les aretes internes de chaque gadget sont couverts. une arete d'un littéral vrai est couverte par le littéral lui-meme et une arete d'un littéral faux est couverte par la clause

<= Soit U une k-couverture de $f(\phi)$, elle couvre les haltères donc , comme elle couvre chaque haltère, U contient au moins un sommet par haltere(soit la var et la neg de la var).
elle couvre les triangles, comme elle couvre tous les triangles, alors U contient au moins deux sommets /triangle.
comme #U = k =2m+n la couverture ets de la forme précédente et l'instanciation associée satisfait ϕ

SUBSET-SUM

```
SUBSET-SUM
entrée : des entiers x1,x2,...,xn de N ert un objectif S de N
question: existe-t-il un sous-ensemble des xi qui sommet à S?
peut on prendre un sous ensemble dont la somme des éléments sera S?
```

proposition SUBSET-SUm est NP-Complet
let's Go!!

– SUBSET-SUM : un certificat qui est le choix I des x_i qui somment à S
– 3SAT$\leq_p$SUBSET-SUM

soit ϕ une instance de 3SAT, on construit polynomialement $f(\phi)$ une instance de SUBSET-SUM qui est satisfiable ssi ϕ l'est.

ON code ϕ à n variables et m clauses par des nombre N_i décimaux à m+n chiffres de la manière suivantes :

– on les décomposent en deux zones
– celle de gauche code les variables,
– celle de droite code les clauses
– à chaque variable x_i on associe un entier N_{2i} et N_{2i+1} qui est construit de la manière suivante : que des zéro sauf , un 1 pour le chiffre qui correspond a x_i et un 1 si x_i apparaît dans la clause(ou 1 si x n'apparaît pas pour n_{2i+1} qui code $\bar{x}$
– à chaque clause C_j on associe $N_{2(n+j)}$ et $N_{2(n+j)+1}$identiques et égaux à que des 0 dans la partie variables et un 1 dans le chiffre qui code la clause

=> Si ϕ ets satisfaite par σ, il suffit de choisir N_{2i} si $\sigma(x_i)$ = VRAI, N_{2i+1} sinon.
Et $N_{2(n+j)}$,$N_{2(N+j)+1}$ pour compter le chiffre de C_j si il vaux 1 ou 2.

Proposition Si $f(\phi)$ est satisfait par U. U force pour chaque x_i le choix de N_{2i} ou N_{2i+1} fixant une instanciation σ qui satisfait ϕ.
f est calculable en temps polynomiale donc c'est une réduction
3SAT $\leq_p$ SUBSET-SUM
donc SUBSET-SUM est NP-Complet.

4 3 Avril 2013

4.1 Complexité en espace

DSPACE(f(n)) : ensemble des langages reconnus en espace f(n) par une MT.
NSPACE(f(n)) : essentiellement la même chose mais avec des MTND.

Classes de complexités :
L = DSPACE(log(n)) : reconnu en espace log
NL = NSPACE(log(n)) : idem mais avec MTND
PSPACE = $\cup_{k>1} DSPACE(n^k)$
NPSPACE = $\cup_{k>1} NSPACE(n^k)$

exemple le parcours de graphe $\in$ NL
Idée : Parcourir le graphe en choisissant le prochain sommet au hasard. Borner le parcours en longueur $\leq$ n (# sommets).
prop : pour f(n) $\geq$ log(n)
$NTIME(f(n)) \subseteq_1 DSPACE(f(n)) \subseteq NSPACE(f(n)) \subseteq_2 DTIME(2^{O(f(n))})$
$\subseteq_1$: Simuler toutes les séquences de choix possibles en stockant la séquence courante comme un mot de f(n) bits.
$\subseteq_2$: parcourir le graphe des configurations d'un MTN D en espace f(n) pour trouver un chemin de la configuration initiale à une configuration acceptante.
Ce graphe possède $2^{O(f(n))}$ sommets.
Le parcours se fait en temps polynomial, $DTIME(2^{O(f(n))})$
corollaire ; $L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP \subseteq NEXP$

4.2 QSAT

entrée : une formule quantifiée $\phi = Q_1x_1Q_2X_2...Q_nx_n\Phi(x_1...x_n)$ ou Q_n sont des qualificateur universel ou existentiel
Remarque $\Phi(x_1, ..., x_n) \in SAT$ ssi $\exists x_1 \exists x_2 ... \exists x_n \Phi(x_1, ..., x_n) \in QSAT$

Prop : QSAT est PSPACE-complet , ie QSAT $\in$ PSPACE et $\forall L \in PSPACE$, $L \leq_m^p QSAT$

théorème de Savitch REACHABILITY $\in$ DSPACE(log^2(n))
Cor. Si f(n) $\geq$ log(n) et f(n) constructible en espace, alors NSPACE(f(nàà $\subseteq$ DSPACE(f^2(n))
Cor. PSPACE = NPSPACE
idée de la preuve du THM traitrer le problème récursivement

CHERCHE(s,t,k) :
si k = 0 : on regarde si t voisin de s, oui , sinon NON
sinon pour tout x$\in$V
si CHERCHE(s,x,t-1) et CHERCHE(x,t,k-1) alors oui
finsinon
NON

du coup REACHABILITY(s,t) = CHERCHE(s,t,log(n))
On fait au plus log(n) appels récursifs, la pile stockex de taille (log(n)).
l'espace utilisé est $\leq log^2(n)$
Donc REACHABILITY $\in$ DSPACE($\log^2$(n)).