TD 2

Ex 2

1) $DTIME(f(n)) \subset NTIME(f(n))$

(ensemble de Pb de décision)

Soit Π un pb de décision de $DTIME(f(n))$, montrons que $\Pi \in NTIME(f(n))$

$\exists$ une machine M qui résout Π en temps $O(f(n))$ déterministe

D'après le Lemme ci-après, $\exists N$ non det qui résout Π en temps $O(f(n))$

$\rightarrow \Pi \in NTIME(f(n))$

Une autre manière de résoudre l'exercice est de changer les définitions du cours. Actuellement, on sépare distinctement machine déterministe et machine non déterministe. On pourrait très bien dire qu'une machine déterministe est un cas particulier de machine non déterministe (mais, sémantiquement, c'est étrange à dire).

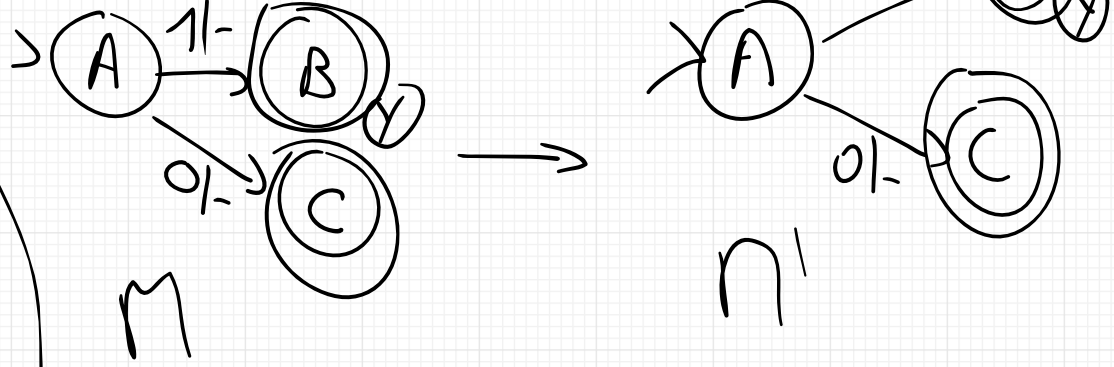
La démo DTIME(f(n)) C DSPACE(f(n)) est similaire.

Lemme : Soit Π une machine det qui résout Π en temps $O(f(n))$
$\rightarrow \exists$ une machine Π' non det qui résout Π en temps $O(f(n))$

Idée : on peut transformer Π en Π' en regardant un choix non déterministe inutile :

Par exemple :

Autre idée
Rajouter un
état initial inutile

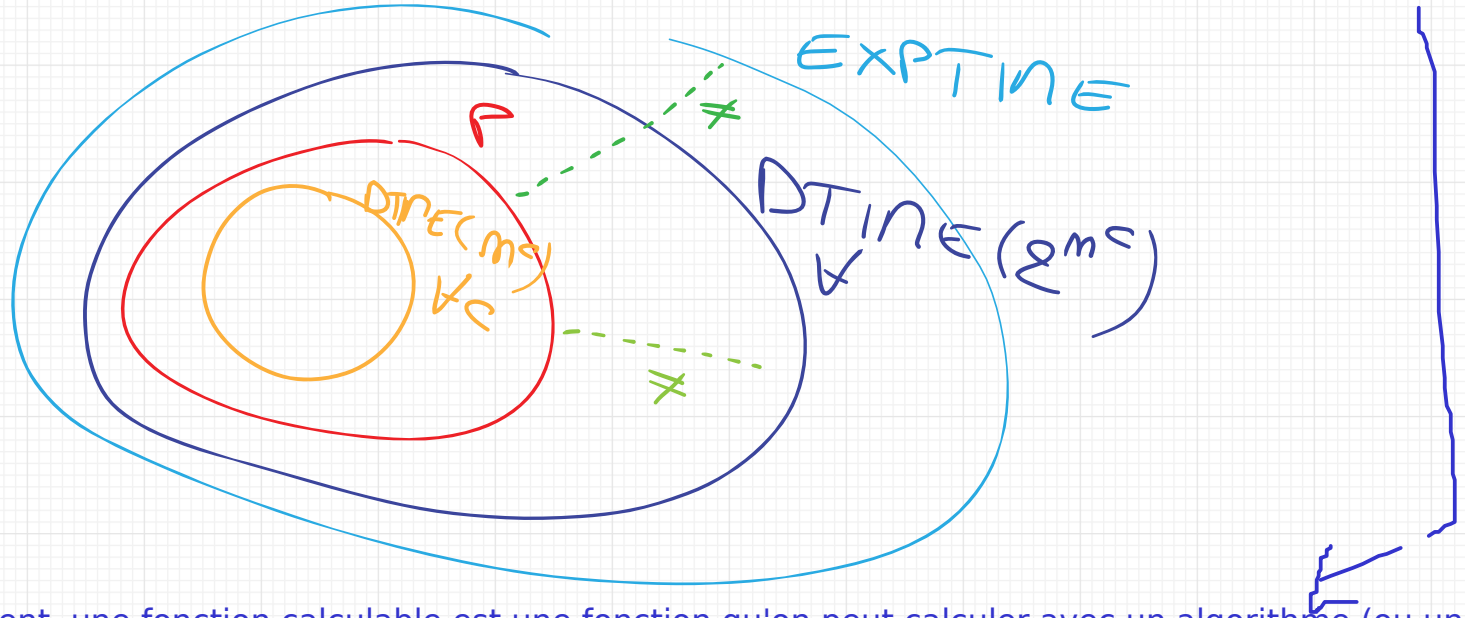


(Une machine qui a
2 états initiaux est
non déterministe.)

M' peut faire plein de choix inutiles mais, pour atteindre B elle n'a besoin que d'une itération.
D'après la définition de la complexité d'une machine non déterministe du cours,
sa complexité est constante.
De manière générale, M' a bien la même complexité que M.

6) Mq $\forall g$ $DTIME(g(n)) \subsetneq DTIME(g^2(n))$
$\Rightarrow P \subsetneq EXPTIME$

Comme expliqué pendant la séance, cette hypothèse est vraie pour la plupart des fonctions usuelles, notamment les polynomes et les exponentielles.



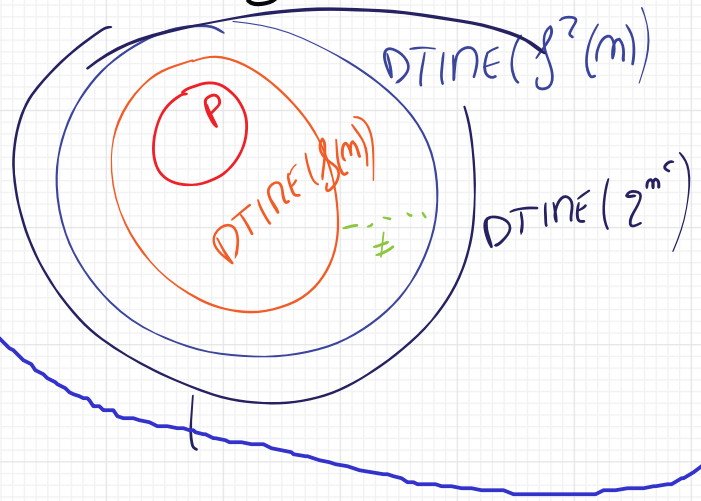
Plus exactement, une fonction calculable est une fonction qu'on peut calculer avec un algorithme (ou une machine de Turing). On lui donne n, l'algo renvoie f(n).
Ici, on demande en plus que la complexité de l'algo soit au pire O(f(n)). Donc f se calcule en un temps qui ne la dépasse pas. Par exemple O(n^p), où p est une constante, peut se calculer en O(log(n)) en faisant p multiplications d'un entier contenant environ log(n) bits : ok.
Autre exemple O(2^n) peut se calculer en temps O(n) en faisant n multiplications.

Enfin, pour que le résultat soit vrai, il faut que f(n) = o(f^2(n)) sinon ça ne fonctionne pas. C'est bien le cas pour les polynomes et mes exponentielles.

Je n'ai pas insisté sur ces points ici, simplement car ce n'est pas l'objet de l'exercice. J'ai juste mis une hypothèse très forte.

My $\exists \Pi \in EXPTIME$ et $\Pi \notin P$ (ce quon veut montrer)

- Idée : my $\exists c \mid \exists \Pi \in DTIME(2^{n^c})$ et $\Pi \notin P$ (si on montre ça on prouve la ligne du dessus)
- $\forall f \exists \Pi \in DTIME(f^2(n))$ et $\Pi \notin DTIME(f(n))$ (c'est l'hypothèse)



IDée : trouver f tel que
P C DTIME(f(n)
et DTIME(f^2(n)) C DTIME(2^(n^c))

On prouve ainsi que P est strictement inclu dans DTIME(2^(n^c))

Si : $f(n) = 2^{(\frac{1}{2}n^c)}$

alors $f^2(n) = 2^{n^c}$

$\forall$ polynome $n^d = o(2^{\frac{1}{2}n^c})$

$\Rightarrow P \subset DTIME(2^{\frac{1}{2}n^c}) \subsetneq DTIME(2^{n^c}) \subset EXPTIME$

Il faut bien noter la puissance de l'hypothèse: elle est vraie pour toute fonction f. On peut donc prendre n'importe quelle fonction. Il faut juste faire son choix.

On a pas prouvé cette partie explicitement.
Ca revient à dire que si on a une machine de complexité O(n^d) ; alors puisque n^d = o(2^(1/2 n^c)) ; on a n^d = O(2^(1/2 n^c)) ; donc O(n^d) = O(2^(1/2 n^c)) ; donc la machine a aussi une complexité O(2^(1/2 n^c)).

Lem 3 Soit $\Pi \in$ PSPACE

$\exists c \mid \exists$ M déterministe qui résout Π en complexité en espace $O(n^c)$

Mq $\exists$ M' déterministe qui résout Π en temps $O(2^{n^d})$

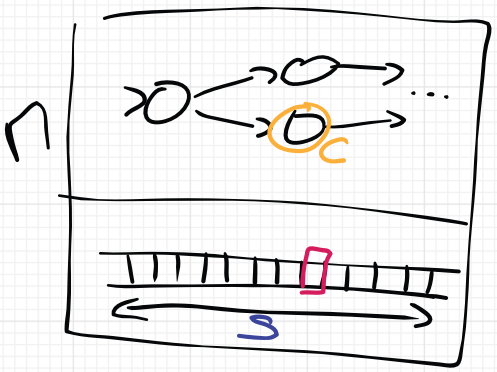
($\Pi \in$ EXPTIME)

- Idée Mq M a une complexité en temps $O(2^{n^d})$ pour un certain d

① M s'arrête (car M résout Π) $\Longrightarrow$ M ne boucle pas à l'infini

② $\exists c$ / M n'écrit pas sur plus $\underbrace{c \cdot n^c}_{O(n^c)}$ cases $\Rightarrow$ S

S'il y a un truc à retenir ici, c'est que cette preuve fonctionne
car on sait que la machine s'arrête et ne boucle pas
parce que c'est dans la définition de "une machine résout un problème de décision".



Ensemble des configurations :

= { (Emplacement de C
Emplacement de H
Ce qui est écrit sur les S cases de la bande) }

Une configuration est un état de la machine.
Mais on ne parle pas d'état car on utilise déjà ce terme pour les états de l'automate (le graphe).

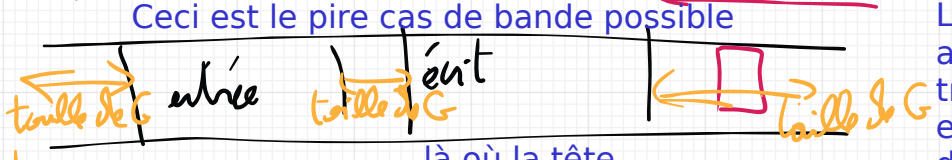
Taille de cet ensemble :

nb de noeuds du graphe = |V|

nb noeud du graphe × 3 + |entrée| + |écrit| = 3|V| + n + S

× 3^S (pour chaque case écrite : 0, 1 ou blanc)

Ceci est le pire cas de bande possible



là où la tête écrit.

La tête ne peut pas aller trop à droite ou trop à gauche, sinon elle est perdue dans le désert de blanc infini et boucle indéfiniement ce qui est impossible puisque la machine s'arrête.

Pour la même raison, la tête ne peut pas écrire une infinité de blancs à côté de l'entrée.

Bien entendu, on peut imaginer des bandes plus courtes, par exemple où on mélange ce qui est écrit et l'entrée.
Il est juste important de noter que la tête ne peut pas se déplacer indéfiniement à droite ou à gauche.
On a borné par |V| le nombre de blancs que la tête peut visiter (un par noeud du graphe de la machine).

Il existe une autre définition plus simple de la complexité en espace nous aurait simplifié le travail : le nombre maximum de cases visitées par la tête. Mais je préfère la définition du cours (le nombre maximum de cases où on écrit) parce qu'elle autorise des complexité O(log(n)) ou O(1) et qu'elle ne change pas les résultats pour les classes classiques (PSPACE, NPSPACE, EXPSPACE, ...).
Avec cette définition, la partie rose est tout simplement bornée par S.

On ne peut être dans la même conf 2 fois sinon on boucle :

car Π est déterministe

donc une configuration implique 1 seul futur possible

"futur"

Si on revient dans la même configuration

"revient"

(on va refaire exactement les même opérations que celle qui nous ont mené à cette configuration.)

on boucle

Attention, ceci n'est pas le graphe de la machine. C'est un schémas représentant les configurations de la machine, son état au cours du temps.

$\Rightarrow \Pi$ ne s'arrête pas

$\Rightarrow$ exclus

La complexité en temps de Π est donc borné par le nb de configurations

$\Rightarrow O(3^{s} \times |V| \times (3|V|+n+s))$

$= O(3^{c \cdot n^{c}} \times |V| \times (3|V|+n+s))$

$\Pi \in EXPTIME$