Space Hierarchy Theorem

L'algorithme suivant décide un langage en $O(t(n))$ non décidable en $o(t(n))$

D = sur l'entrée w :

1) soit n = longueur de w
2) Calcul $f(n)$ par la space constructibilité.
3) Si w n'est pas de forme $\langle M \rangle$ où M est une machine de Turing alors rejeter
4) Simuler M sur w (entrée en encre) de M utilise trop de case on prend plus de $2^{f(n)}$ alors rejeter sinon renvoyer le résultat de M.

(On le calcule en encre et on marque la case où on arrive)

Si M accepte en encre, sinon accepte

L'étape 4 est l'étape cruciale. Si elle s'exécute en $O(f(n))$ espace alors notre machine aussi.

Pour utiliser le calcul de la machine simulée on a besoin de seulement $O\left(\frac{1}{n}\right)$ case.

Donc la simulation n'utilise que n cases $O(f(n))$ espace

D décide notre langage car elle ne s'exécute qu'un nombre limité.

A ce langage

A est décidable en $O(f(n))$

Si M' décide A en $o(f(n))$ alors D simule M en $o(f(n))$ d'où D simule M à partir de n_0.

$\langle M \rangle 10^{n_0}$ fini et donc D connaît l'issue de M dans M ne peut pas décider A.

$\{\langle G, s, t\rangle$ | $\exists$ un chemin entre s et t dans G ?

orienté

PATH et NL - complet

$A \in NL$

M machine de T décide A en $O(\log n)$ espace.

Pour x on construit $\langle G, s, t\rangle$ en log espace où G est le graphe orienté qui a un chemin de s à $t \Leftrightarrow$ M accepte x

Les nœud de G sont les configurations de M sur x.

$\forall (c_1, c_2)$ deux configurations

$(c_1, c_2) \in E$ si il y a une transition possible entre les deux.

Explication

s est la configuration de départ et t la configuration finale.

On a bien $\langle G, s, t\rangle \Leftrightarrow$ M accepte x

G liste ses nœuds et arêtes

Chaque config de M sur x peut être représenté en $c \log n$

Test les config et imprime les nœuds et arêtes possibles