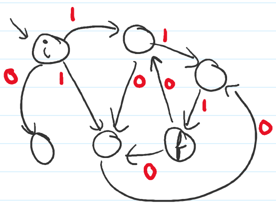
PSPACE (suite)

24 mars 2023 11:02

PSPACE = NPSPACE

- Utile pour montrer $L \in$ PSPACE, car on peut montrer $L \in$ NPSPACE.

- ex: AFN = automate fini non-déterministe



$AFN_{inter} = \{\langle A,B \rangle :$ A, B sont des AFN tels qu'il existe $w \in \Sigma^*$ accepté par A et B $\}$

$AFN_{inter} \in$ PSPACE

idée (marche pas): pour chaque mot w // Pb: $|w|$ pas poly

  si A et B acceptent w
    return true

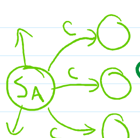
return false

À la place, on cherche w dans l'intersection de façon non-déterministe:

espace — sur entrée $\langle A,B \rangle$:

$O(|A|)$ — $S_A \leftarrow$ état init. de A — // $|A|$ = # états
$O(|B|)$ — $S_B \leftarrow$ état init. de B

while true



$O(1)$ — $c \leftarrow$ symbole de Σ choisi de façon non-déterministe

$O(|A|)$ — $S_A \leftarrow$ état suivant S_A sur caractère c choisi de façon non-dét.

$O(|B|)$ — $S_B \leftarrow$ état suivant S_B sur c, façon non-dét.

$O(|A|+|B|)$ — si S_A et S_B sont acceptants
  accepter $\langle A,B \rangle$

Le langage de cette MT non-dét est $\langle A,B \rangle$ avec intersection non-vide:

- si $\langle A,B \rangle \in AFN_{inter}$, alors il y a w dans l'inter. et on va parcourir les caractères de w par non-dét. et atteindre un état acceptant pour A et B.

- si $\langle A,B \rangle \notin AFN_{inter}$, on ne tombera jamais sur un double-état acceptant pour A et B.

L'espace requis pour cet algo est $O(|A|+|B|)$, et

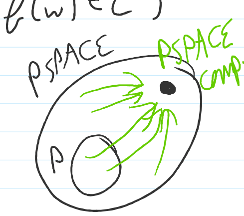
L'espace requis pour cet algo est $O(|A|+|B|)$, et donc AFNinter $\in$ NPSPACE.
Par Savitch, AFNinter $\in$ PSPACE. ∎

Pour $L \in$ PSPACE :

1. Force brute sur les sols possibles (SAT)
2. Algo récursif, avec arbre de récursion de prof. bornée et mémoire bornée à chaque appel. (bornée = polynomial) (NP $\subseteq$ PSPACE) (dpath)
   ↳ faille en prof.
3. $L \in$ NPSPACE (AFNinter)

PSPACE-complétude

- Un langage L est PSPACE-complet si :
  ↳ $L \in$ PSPACE
  ↳ $\forall L' \in$ PSPACE, $L' \leq_p L$ ($\exists f$ en temps poly $w \in L' \Leftrightarrow f(w) \in L$)
- $\leq_p$: si on avait un algo temps poly pour un langage PSPACE-complet, alors on pourrait décider tout langage de PSPACE en temps poly.



TQBF : true quantified boolean formula

Formules avec alternance de quantificateurs :(

ex : $\exists x_1 \forall x_2 \exists x_3 \forall x_4 \; (\underset{T}{x_1} \vee \overline{x_2}) \wedge (\underset{T}{\overline{x_1}} \vee \underset{F}{x_3} \vee \underset{F}{x_4}) \wedge (\underset{F}{x_2} \vee \underset{(F)}{\overline{x_3}})$

J1 : satisfaire la formule
J2 : pas satisfaire

J1 : $x_1 = T$ J2 : $x_2 = F$
J3 : $x_3 = F$ J4 : $x_4 = F$

$x_1 = T$ mène a J2 gagne

J1 : $x_1 = F$ J2 : $x_2 = T$

J1 : $x_1 = F$ J2 : $x_2 = T$ gagne

TQBF = { φ : φ une $\underbrace{\text{formule quantifiée}}_{\exists x_1 \forall x_2 \exists x_3 \cdots \forall x_n \, \psi}$ qui est vraie }

où ψ est une formule booléene

Thm : TQBF est PSPACE-complet

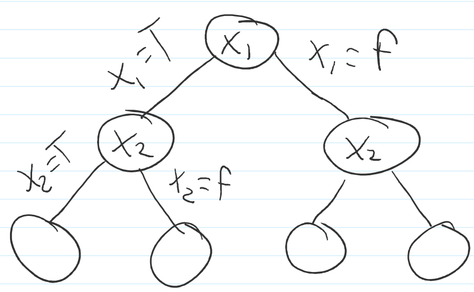
1) TQBF $\in$ PSPACE

2) TQBF est PSPACE difficile [exercice]

Soit φ une instance de TQBF.

Premier / dernier quantif. peuvent être $\exists, \forall$

$\varphi = \exists x_1 \forall x_2 \exists x_3 \cdots \forall x_n \; \psi$ ou $\varphi = \forall x_1 \exists x_2 \cdots \forall x_n \; \psi$



$\exists x_1 \forall x_2 \, (x_1 \vee \bar{x}_2)$

$\varphi_1 = \forall x_2 \, (\text{true} \vee \bar{x}_2)$

$\varphi_2 = \forall x_2 \, (\text{false} \vee \bar{x}_2)$

Cet algo crée un arbre de récursion où

- la profondeur est $O(n)$
  n = #variables

tqbf(φ)

si φ n'a pas de $\exists, \forall$
  retourner la valeur de verité de ψ

si φ commence par $\exists x_1$ [φ']
  soit φ_1 = ce qu'on obtient en posant $x_1 = T$ dans φ'
  res_1 = tqbf(φ_1)
  soit φ_2 = poser $x_1 = F$ dans φ'
  res_2 = tqbf(φ_2)
  return $res_1 \vee res_2$

si φ commence par $\forall x_1$
  φ_1 = poser $x_1 = T$ dans φ'
  φ_2 = poser $x_2 = F$ dans φ'
  res_1 = tqbf(φ_1)
  res_2 = tqbf(φ_2)

- la profondeur est $O(n)$
  $n = \#$variables
- l'espace par appel est $O(n)$
  (copies de φ)
- Espace total: $O(\text{prof} \times \text{appel})$

$res_2 = tqbf(\varphi_2)$

return $res_1 \wedge res_2$

$$O(n \cdot n) = O(n^2) \Rightarrow TQBF \in PSPACE$$