Complexité du problème d'intersection d'automates

Michael Blondin

Département d'informatique et de recherche opérationnelle
Université de Montréal

11 janvier 2009

Définition

PEP$_k$ (du nom anglais *Product Emptiness Problem*) :

Données : $A_1, \ldots, A_k$, k automates finis déterministes.

Problème : Déterminer si $L(A_1) \cap \cdots \cap L(A_k) = \emptyset$.

Définition

PEP_k (du nom anglais *Product Emptiness Problem*) :

Données : $A_1, \ldots, A_k$, k automates finis déterministes.

Problème : Déterminer si $L(A_1) \cap \cdots \cap L(A_k) = \emptyset$.

Définition

$\mathrm{PEP} = \bigcup_{k \geq 1} \mathrm{PEP}_k$

Variantes :

Nom	**Type d'automates**
NPEP	Non déterministes
uPEP	Alphabet unaire $\{a\}$
SPEP	À balayage
2PEP	Avec tête de lecture bidirectionnelle

$\mathrm{PEP} \in \mathrm{P}$?

PEP $\in$ P ? Non.

Théorème (Kozen, 1977)

PEP est PSPACE-complet.

$\text{PEP}_k \in$ P ?

PEP $\in$ P ? Non.

Théorème (Kozen, 1977)

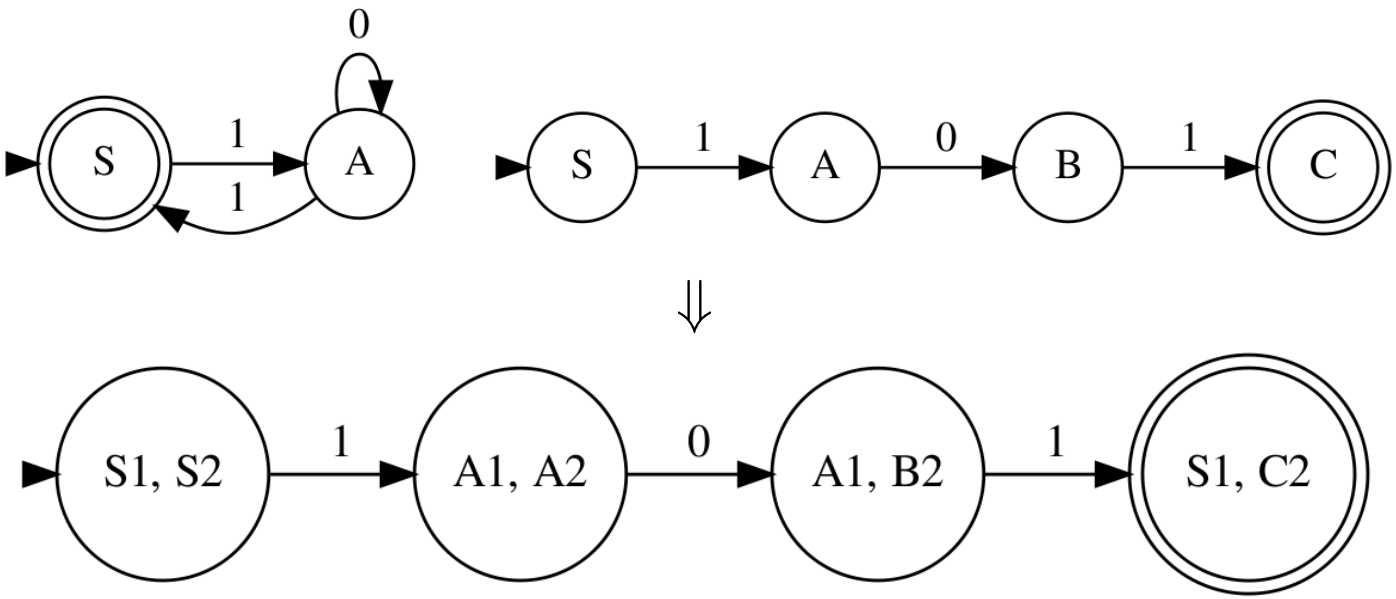
PEP est PSPACE-complet.

$\text{PEP}_k \in$ P ? Oui.

Théorème (Folklore)

PEP_k est résoluble en temps $O(n^k)$.

Comment ? Construire un automate simulant les k automates. Par exemple :



Puis vérifier s'il existe un chemin de l'état initial vers l'état final.

Question

Est-il possible de faire mieux que $O(n^k)$?

Question

Est-il possible de faire mieux que $O(n^k)$?

Réponse (motivation)

Question ouverte. Si oui, alors NL $\neq$ P. Si non, nous avons une borne inférieure non linéaire. [Richard Lipton et al., 2003]

Il serait étonnant de répondre à cette question dans le cadre du cours IFT4055.

Question

Quelle est la complexité de PEP (et PEP_k) pour d'autres modèles d'automates ?

Résultats :

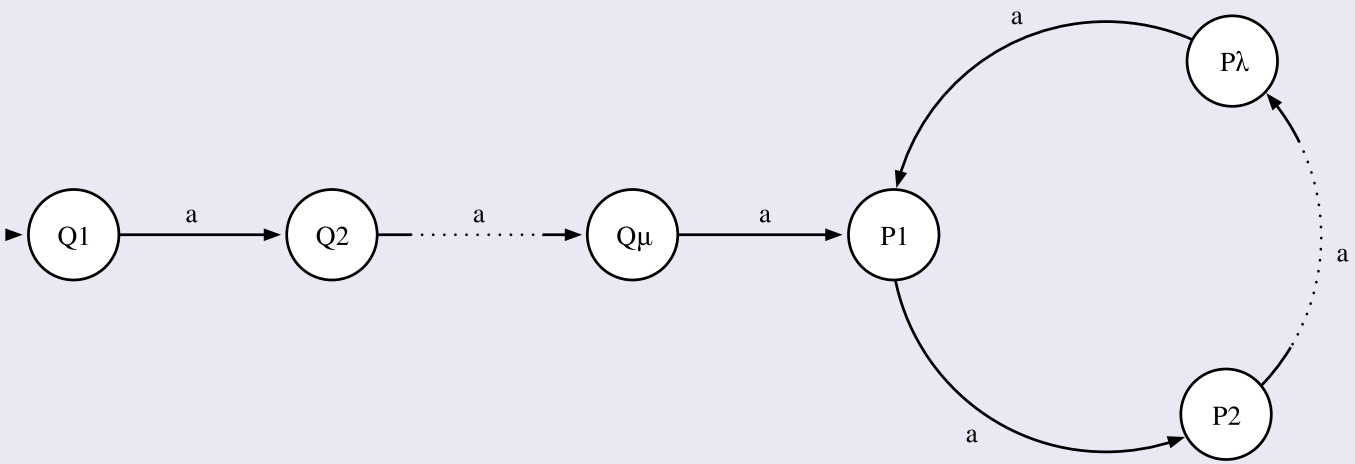
Modèle	PEP_k	PEP
uDFA (1 état final)	L-complet	$NC^3 \subseteq P$
uDFA	L-complet	coNP
DFA	NL-complet	PSPACE-complet
uNFA	NL-complet	PSPACE-complet
NFA	NL-complet	PSPACE-complet
uSDFA	coNP	coNP
u2DFA	coNP	coNP
SDFA	PSPACE-complet	PSPACE-complet
2DFA	PSPACE-complet	PSPACE-complet
uSNFA	coNP	coNP
u2NFA	coNP-complet	coNP-complet
SNFA	PSPACE-complet	PSPACE-complet
2NFA	PSPACE-complet	PSPACE-complet

Théorème

uPEP (1 état final) $\in NC^3 \subseteq P$.

Démonstration.

Automate unaire (uDFA) à 1 état final :



- *Queue* de longueur $\mu \geq 0$
- *Cycle* de longueur $\lambda \geq 1$
- Un état final en position $d \geq 1$

Démonstration.

Étant donné k automates, nous construisons le système S suivant :

$$\begin{pmatrix} 1 & -\lambda_1 & 0 & \cdots & 0 \\ 1 & 0 & -\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & -\lambda_k \end{pmatrix} \begin{pmatrix} m \\ x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \end{pmatrix}$$

La $i^{\text{ème}}$ équation de S est équivalente à $m \equiv d_i \pmod{\lambda_i}$.

Un mot est accepté par les k automates ssi S possède au moins une solution.

Démonstration.

Problème : les solutions doivent être entières.

Vérifier l'existence d'une solution entière pour un système d'équations linéaires, modulo un *petit* entier, peut être effectué dans NC^3. [Cook, McKenzie, 1987]

Solution : vérifier s'il existe une solution au système

$$S' = S \bmod \Lambda,$$

où

$$\Lambda = \lambda_1 \cdots \lambda_k.$$

Démonstration.

Il existe une solution entière pour $S \Leftrightarrow$ il existe une solution entière pour S'.

$\Rightarrow$) Facile, la même solution fonctionne.
$\Leftarrow$) Soit $(m, x_1, \ldots, x_k)$ une solution pour S', alors $\forall i \in [1..k]$:

$$\begin{aligned} m' &= m \bmod \Lambda \\ &= d_i + x_i\lambda_i \bmod \Lambda \\ &= d_i + x_i\lambda_i + z_i\Lambda \\ &= d_i + (x_i + z_i\Lambda/\lambda_i)\lambda_i \end{aligned}$$

Posons $y_i = x_i + z_i\Lambda/\lambda_i$, alors $(m', y_1, \ldots, y_k)$ est une solution pour S. □

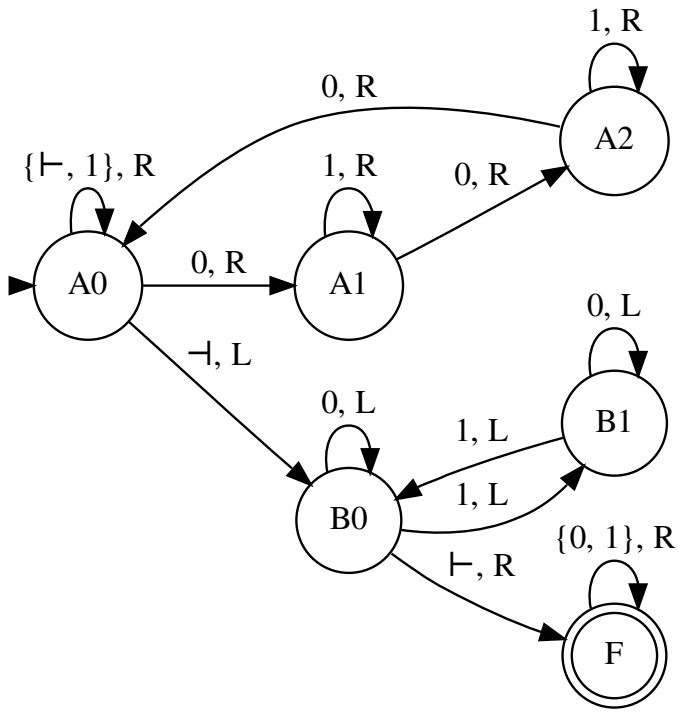
Corollaire

$\overline{uPEP} \in NP$.

Démonstration.

Il suffit de choisir un état final par automate, de façon non déterministe, puis de vérifier si un mot est accepté par tous les automates. □

Automate à tête bidirectionnelle :



Plus précisement : automate à balayage.

Théorème

SPEP $\in$ *PSPACE-complet*

Démonstration.

But : construire un automate à balayage A qui simule plusieurs automates (DFAs).

Comment :

1. A lit l'entrée et simule le premier automate
2. Si A s'arrête sur un état final, le ruban est « rembobiné ».
3. Répéter avec tous les automates
4. Accepter comme le dernier automate.

Donc PEP $\leq$ SPEP$_1$. □

Questions :

- uPEP $\in$ P ? uPEP $\in$ coNP-complet ?
- Existe-t-il une classification de la complexité de PEP en fonction de propriétés algébriques ?
- Quelles sont les restrictions suffisantes (ou nécessaires) pour rendre le problème résoluble efficacement ?

Théorème (Lipton et al., 2003)

Si PEP_k se résoud plus rapidement qu'en temps $O(n^k)$, alors $NL \neq P$.

Esquisse de preuve

But : Simuler une machine M de type NL avec plusieurs automates, puis utiliser le meilleur algorithme pour PEP_k pour obtenir une simulation de M en temps $O(n^2)$.

Esquisse de preuve

Étant donné un mot x et une machine de Turing M pour $S \in \mathsf{NL}$:

1. Découper le ruban de travail de M en k blocs.
2. Construire k automates qui vérifient si une chaîne de calcul est valide pour un certain bloc et acceptent comme M.
3. Construire un automate qui vérifie si x apparaît correctement dans un calcul.
4. Vérifier si l'intersection des $k+1$ automates est vide.

L'intersection est non vide ssi $x \in S$.

Esquisse de preuve

Sous l'hypothèse qu'il existe un meilleur algorithme pour PEP_k, nous obtenons une simulation de M en temps $O(n^{1+\varepsilon})$ pour n'importe quel $\varepsilon > 0$.

Donc $\text{NL} \subseteq \text{DTIME}(n^2)$.

Or, $\text{DTIME}(n^c) \subset \text{DTIME}(n^{c+1})$ et ainsi $\text{NL} \neq \text{P}$. □

Autres conséquences :

Théorème (Lipton et al., 2003)

Il existe un algorithme qui permet de factoriser en temps $O(2^{\varepsilon n})$ pour tout $\varepsilon > 0$.

Théorème (Lipton et al., 2003)

$NTIME(t) \subseteq DTIME(2^{\varepsilon t})$ pour tout $\varepsilon > 0$.

Intéressés ?

- `http://www-etud.iro.umontreal.ca/~blondimi/`
- `http://rjlipton.wordpress.com`

Questions ?

Merci !