# Mines Informatique MP 2008 — Corrigé

Ce corrigé est proposé par Marc Mezzarobba (ENS Ulm) ; il a été relu par Benjamin Monmege (ENS Cachan) et Vincent Puyhaubert (Professeur en CPGE).

---

Comme souvent aux Mines, l'énoncé se compose de deux problèmes indépendants, de tailles inégales.

- Le premier, conçu pour occuper un peu moins du quart du temps imparti, est consacré au calcul d'un automate déterministe reconnaissant l'intersection des langages acceptés par deux automates donnés. Proche du cours, il fait appel aux principales opérations sur les automates, comme la déterminisation et la réalisation algorithmique des propriétés de clôture de la classe des langages rationnels. En revanche, il n'utilise pas la notion d'expression rationnelle et ne contient aucune question de programmation.
- Le deuxième problème porte sur le thème classique d'optimisation combinatoire dit du *voyageur de commerce*. C'est principalement un exercice de programmation, centré sur la manipulation de graphes en style impératif. La majorité des questions demandent soit d'écrire des programmes, soit de traiter des exemples à la main ; il faut aussi calculer les complexités de quelques fonctions simples.

  Ce problème est lui-même divisé en quatre parties de difficulté (lentement) croissante. La première se limite à quelques préliminaires d'algorithmique des graphes. Les deux suivantes abordent deux heuristiques pour calculer des solutions approchées au problème du voyageur de commerce : la méthode du plus proche voisin et la méthode de recherche locale « 2-opt ». Enfin, la dernière partie présente un algorithme de recherche par séparation et évaluation pour résoudre le problème de façon exacte.

Dans l'ensemble, il s'agit d'un problème un peu long, d'une difficulté moyenne, que l'on peut recommander pour travailler sur les traits impératifs de Caml (tableaux, références, etc.). Par ailleurs, la méthode *séparation et évaluation*, mieux connue sous son nom anglais *branch and bound*, dont l'algorithme de la quatrième partie donne un exemple, est une technique générale de conception d'algorithmes utile dans de nombreuses situations : il faut au moins en connaître l'idée générale.

# Indications

## Problème I

3 On souhaite qu'un calcul de A d'étiquette $m$ simule l'exécution en parallèle d'un calcul de $A_1$ et d'un calcul de $A_2$, tous deux d'étiquette $m$. Comment choisir T ?

5 Prendre $T' = T$.
On se rappellera aussi qu'un automate déterministe complet a un et un seul calcul étiqueté par chaque mot sur son alphabet d'entrée.

7 Utiliser la question 5.

## Problème II

8 Caractériser les ensembles de permutations qui induisent un même tour.
Pour estimer si la méthode proposée est raisonnable, on peut par exemple comparer le nombre de tours dans un graphe de taille 50 au nombre d'opérations élémentaires qu'un microprocesseur actuel fait en un an.

11 Pour parcourir les sommets appartenant à un ensemble S, il suffit de visiter tous les sommets par une boucle du type `for s = 0 to n-1`, et de tester à chaque itération si $s \in S$ à l'aide du tableau `S` représentant S.

15 Utiliser la fonction `plus_proche` écrite à la question 11, en modifiant convenablement l'ensemble S entre les appels.

16 Utiliser le résultat de la question 12.

19 Observer qu'un vecteur codant le tour $\textit{2-opt}(T, 1, 6)$ s'obtient en « retournant » une partie d'un vecteur codant T.

22 Le tour cherché est de poids 18.

23 Comparer les résultats des deux questions précédentes.

25 Montrer l'inégalité

$$2\sum_{i=p-1}^{n-1} \textit{poids}(\{t_i, t_{i+1}\}) \geqslant \textit{eval1}(G, C, c_0) + \textit{eval1}(G, C, c_{p-1}) + \sum_{x \in U} \textit{eval2}(G, C, x)$$

27 Appeler deux fois `plus_proche`, et modifier le tableau `S` avant et après le second appel.

28 Utiliser les fonctions écrites aux questions 9, 11 et 27 ; introduire un tableau représentant initialement U et le modifier de façon appropriée au cours du calcul, comme à la question 27.

29 Combiner judicieusement les trois méthodes étudiées dans le problème.

# Les conseils du jury

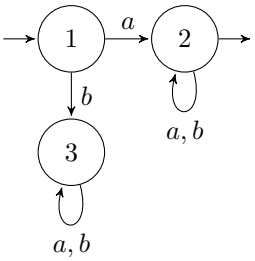
✎ Étant donné que la présentation et la rédaction ont été jugées « bonnes dans l'ensemble », le jury dispense peu de conseils dans son rapport si ce n'est de faire des indentations correctes dans les codes et de réaliser des dessins d'automates que le correcteur puisse interpréter sans difficulté. En outre, le jury regrette que « plusieurs candidats omettent de faire des références explicites aux questions déjà traitées dans le sujet lors de la réponse à une question qui nécessite l'utilisation d'un résultat déjà établi ».

# I. Problème sur les automates

✎ Le rapport du jury déplore de « grosses faiblesses dans le traitement [de ce] problème ». En particulier, les questions 3 et 5 réclamaient des preuves qui sont « rarement bien faite[s] ». Prenez le temps de justifier vos réponses par des démonstrations rigoureuses, à plus forte raison quand l'énoncé le demande explicitement !

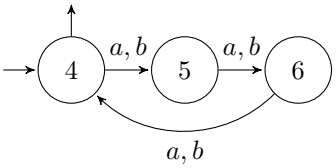
Les autres questions sont « souvent bien traitée[s] ». Les correcteurs relèvent cependant des « difficultés surprenantes » par rapport aux années précédentes. « Donner un automate simple semble au-dessus des compétences de nombreux candidats. Beaucoup ont été perturbés par les définitions (pourtant classiques) d'états accessibles et d'automates déterministes et/ou complets. »

**1** L'automate $A_1\_ex$ représenté ci-dessous reconnaît le langage $L_1\_ex$.




✎ Malgré la simplicité de la question, le rapport du jury constate « un nombre important de mauvaises réponses ».

**2** L'automate $A_2\_ex$ ci-dessous reconnaît le langage $L_2\_ex$.




**3** Adoptons

$$T = \big\{\big((p_1,p_2),x,(q_1,q_2)\big) \in Q \times \Sigma \times Q \;\big|\; (p_1,x,q_1) \in T_1 \text{ et } (p_2,x,q_2) \in T_2\big\}$$

Pour tout mot $m \in \Sigma^*$ et tous états $p_1, q_1 \in Q_1$, $p_2, q_2 \in Q_2$, cette définition entraîne que $(p_1,p_2) \xrightarrow{m} (q_1,q_2)$ est un calcul de A si et seulement si $p_1 \xrightarrow{m} q_1$ et $p_2 \xrightarrow{m} q_2$ sont des calculs, respectivement, de $A_1$ et $A_2$. De plus, au vu des définitions de I et F, le calcul $(p_1,p_2) \xrightarrow{m} (q_1,q_2)$ est réussi (dans A) si et seulement si $p_1 \xrightarrow{m} q_1$ (dans $A_1$) et $p_2 \xrightarrow{m} q_2$ (dans $A_2$) le sont tous les deux. Ainsi un mot $m$ est reconnu par A si et seulement s'il est reconnu à la fois par $A_1$ et par $A_2$, autrement dit l'automate A reconnaît le langage $L_1 \cap L_2$.

En conclusion, on obtient un automate qui reconnaît L en prenant

$$\boxed{T = \big\{\big((p_1,p_2),x,(q_1,q_2)\big) \in Q \times \Sigma \times Q \;\big|\; (p_1,x,q_1) \in T_1 \text{ et } (p_2,x,q_2) \in T_2\big\}}$$

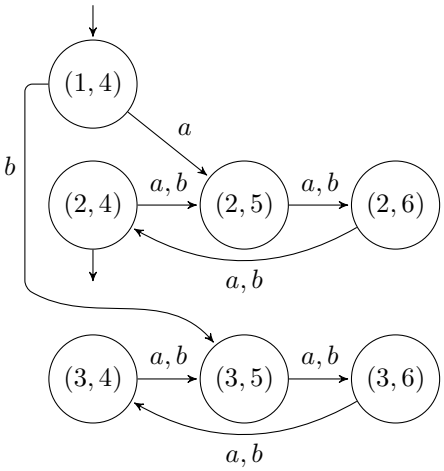
✎ Donner la bonne expression pour T n'est pas suffisant ! Le rapport de jury signale la formule est souvent correcte, mais qu'il y a *deux* inclusions à prouver pour la justifier.

Cette question présente un exemple de construction d'*automate produit* reconnaissant l'intersection de deux langages. Des constructions similaires s'appliquent à diverses autres opérations ensemblistes sur les langages rationnels, et reviennent fréquemment dans les sujets de concours.

**4** On construit les états accessibles de proche en proche, en suivant toutes les transitions possibles à partir de l'état initial. L'automate $\mathrm{A}_3\_ex$ obtenu est représenté ci-dessous.

Informellement, on peut raisonner comme suit : « Au début du calcul, on est dans l'état 1 dans le calcul de $\mathrm{A}_1\_ex$ et dans l'état 4 dans celui $\mathrm{A}_2\_ex$, soit dans l'état $(1,4)$ du produit. De là, en lisant un $a$, on se retrouve dans les états 2 (dans le premier calcul) et 5 (dans le deuxième) ; tandis qu'en lisant un $b$, on aboutit en 3 et en 5 ; donc les transitions de $\mathrm{A}_3\_ex$ issues de $(1,4)$ sont $(1,4) \xrightarrow{a} (2,5)$ et $(1,4) \xrightarrow{b} (3,5)$. Maintenant, depuis l'état 2, on reste sur place quoi qu'on lise, tandis que depuis 5 on va en 6... »

Naturellement, il est aussi possible de commencer par dessiner tous les états, puis toutes les transitions données par la définition, et de supprimer ensuite les états inaccessibles. Mais sur un gros automate, c'est beaucoup moins efficace. Attention si vous répondez à ce genre de question directement au propre, le rapport de jury a relevé « des dessins d'automates difficiles à interpréter ».




$\mathrm{A}_3\_ex$

Les états $(3, q_2)$ de $\mathrm{A}_3\_ex$ ne sont pas coaccessibles mais sont accessibles, il est demandé de les dessiner. On pourrait cependant les remplacer par un unique état puits sans changer le langage reconnu par l'automate.