ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES,
ÉCOLES NATIONALES SUPÉRIEURES DE L'AÉRONAUTIQUE ET DE L'ESPACE,
DES TECHNIQUES AVANCÉES, DES TÉLÉCOMMUNICATIONS,
DES MINES DE PARIS, DES MINES DE SAINT-ÉTIENNE, DES MINES DE NANCY,
DES TÉLÉCOMMUNICATIONS DE BRETAGNE,
ÉCOLE POLYTECHNIQUE
(Filière T.S.I.)

CONCOURS D'ADMISSION 2002

ÉPREUVE D'INFORMATIQUE

Filière MP
(Durée de l'épreuve : 3 heures)

Sujet mis à la disposition des concours Cycle International, ENSTIM et TPE-EIVP.

Les candidats et les candidates sont priés de mentionner de façon apparente sur la première page de la copie :
« INFORMATIQUE - Filière MP »

RECOMMANDATIONS AUX CANDIDATS ET CANDIDATES

- **L'énoncé de cette épreuve, y compris cette page de garde, comporte 8 pages.**
- **Si, au cours de l'épreuve, un candidat ou une candidate repère ce qui lui semble être une erreur d'énoncé, il ou elle le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il ou elle a décidé de prendre.**
- **Tout résultat fourni dans l'énoncé peut être utilisé pour les questions ultérieures même s'il n'a pas été démontré.**
- **Il ne faut pas hésiter à formuler les commentaires qui semblent pertinents même lorsque l'énoncé ne les demande pas explicitement.**
- **L'utilisation d'une calculatrice ou d'un ordinateur est interdite.**

COMPOSITION DE L'ÉPREUVE

L'épreuve comprend trois parties indépendantes :

- un exercice de logique des propositions, n° 1 page 2, à résoudre en 30 mn environ ;
- un problème d'algorithmique, n° 2 page 3, à résoudre en 120 mn environ ;
- un exercice sur les automates, n° 3 page 7, à résoudre en 30 mn environ.

1 Exercice de logique des propositions — 30 mn environ

Définitions et notations

On considère un ensemble fini de n variables propositionnelles $\mathcal{V} = \{p_1, \ldots, p_n\}$, et l'ensemble $\mathcal{F}_\mathcal{V}$ des formules construites à partir des variables propositionnelles de $\mathcal{V}$, des connecteurs usuels de conjonction (noté $\wedge$) et de disjonction (noté $\vee$), et de la négation ($\neg$). On considère que la formule sans aucune variable propositionnelle (formule vide), notée T, est un élément de $\mathcal{F}_\mathcal{V}$.

Toutes les formules considérées dans cet exercice sont des formules de $\mathcal{F}_\mathcal{V}$.

Pour toute formule F, $\mathcal{V}_F$ désigne l'ensemble des variables propositionnelles qui apparaissent dans F.

On appelle *littéral* une variable propositionnelle ou bien la négation d'une variable propositionnelle. Le littéral est dit *positif* dans le premier cas, et *négatif* dans le second cas.

On appelle *clause* toute formule de la forme $l_1 \vee \ldots \vee l_q$, où $q \geqslant 1$ et $l_1, \ldots, l_q$ sont des littéraux deux à deux distincts.

On dit qu'une formule est sous *forme normale conjonctive* si elle est sous la forme $C_1 \wedge \ldots \wedge C_m$, où $m \geqslant 0$ et $C_1, \ldots, C_m$ sont des clauses (pour $m = 0$, on obtient T).

Une *formule de Horn* est une formule sous forme normale conjonctive telle que chacune de ses clauses comporte au plus un littéral positif.

Enfin, rappelons qu'une formule est satisfiable s'il existe une valuation à valeurs dans {*vrai*, *faux*} de ses variables propositionnelles qui rende la formule vraie. La formule T est considérée comme étant satisfiable.

Énoncé de l'exercice

1 – Dans les exemples suivants, préciser si les formules sont satisfiables ou non. Quand cela est possible, donner un exemple de valuation des variables propositionnelles qui rende la formule vraie.
   i) $(\neg p_1 \vee p_2) \wedge (p_1 \vee \neg p_2 \vee \neg p_3)$
   ii) $(p_2) \wedge (\neg p_1 \vee \neg p_3) \wedge (\neg p_2) \wedge (p_1 \vee \neg p_3 \vee \neg p_4)$
   iii) $(p_2) \wedge (\neg p_1 \vee \neg p_2) \wedge (p_1 \vee \neg p_2) \wedge (p_1 \vee \neg p_2 \vee \neg p_3)$

2 – Soit H une formule sous forme normale conjonctive telle que chacune de ses clauses contienne au moins un littéral négatif. Montrer que H est satisfiable en exhibant une valuation de $\mathcal{V}$.

3 – Soit H une formule de Horn telle qu'une de ses clauses soit restreinte à un littéral positif p_k (où k est dans $\{1, \ldots, n\}$), et qu'aucune autre de ses clauses ne soit restreinte à $\neg p_k$. Montrer que l'on peut construire à partir de H une formule de Horn H' telle que :
   - $\mathcal{V}_{H'} \subseteq \mathcal{V}_H \backslash \{p_k\}$,
   - H est satisfiable si et seulement si H' est satisfiable.

4 – Déduire des deux questions précédentes un algorithme permettant de déterminer si une formule de Horn H est satisfiable ; la complexité dans le pire des cas de cet algorithme doit être majorée par un polynôme en n et m (où n et m désignent respectivement le nombre de variables propositionnelles et le nombre de clauses de H), propriété que l'on justifiera. Cet algorithme sera explicité sans utiliser un langage de programmation.

5 – Appliquer l'algorithme de la question 4 à l'exemple iii) de la question 1.

FIN DE L'EXERCICE DE LOGIQUE DES PROPOSITIONS

2 Problème d'algorithmique — 120 mn environ

Préliminaire. Il faudra écrire des programmes à l'aide d'un langage de programmation qui pourra être soit **Pascal**, soit **Caml**, tout autre langage étant exclu. **Indiquer le langage de programmation choisi**. **Attention !** Il est interdit de modifier ce choix au cours de l'épreuve.

Le problème du sac à dos

On considère le problème suivant : Madame X doit partir en voyage ; elle emporte un sac à dos qui peut supporter au maximum un poids Q, poids qui est un entier strictement positif. Par ailleurs, elle dispose de n objets numérotés par 0, 1, ... , $n-1$. Pour tout i appartenant à 0, 1, ... , $n-1$, l'objet de numéro i possède deux caractéristiques : son utilité notée u_i et son poids noté p_i qui sont des entiers strictement positifs. Madame X désire emporter certains de ces objets dans son sac à dos ; le poids total des objets emportés ne doit pas dépasser Q ; elle cherche à maximiser la somme des utilités des objets qu'elle emporte parmi les contenus possibles ne dépassant pas au total le poids Q.

Attention : pour la programmation que vous aurez à faire ultérieurement :

- le nombre n est mémorisé dans une variable entière nommée aussi n ;
- les quantités u_i et p_i sont mémorisées dans des tableaux d'entiers nommés respectivement u et p et indicés de 0 à $n-1$;
- la quantité Q est mémorisée dans une variable entière nommée aussi Q.

Quand on écrira une fonction (ou une procédure) dans le langage de programmation choisi, les tableaux u et p ainsi que les variables Q et n seront supposés déjà définis comme variables globales et initialisés avec les données du problème. Après l'initialisation, à l'indice i du tableau u (resp. p) se trouve la valeur u_i (resp. p_i).

Notation. Si x est un nombre réel, $\lfloor x \rfloor$ désigne la partie entière par défaut de x (c'est-à-dire le plus grand entier inférieur ou égal à x) tandis que $\lceil x \rceil$ désigne sa partie entière par excès (c'est-à-dire le plus petit entier supérieur ou égal à x).

Première Partie
Sac à dos fractionnaire

Dans cette partie, les objets sont fractionnables. On note x_i la quantité de l'objet de numéro i emportée par Madame X. Le problème s'écrit :

$$\begin{cases} \text{Maximiser } z = \displaystyle\sum_{i=0}^{n-1} u_i x_i \\ \text{avec les contraintes :} \\ \quad \displaystyle\sum_{i=0}^{n-1} p_i x_i \leqslant Q \\ \quad \text{et pour } i \in \{0,1,\ldots,n-1\},\ x_i \in [0,1] \end{cases}$$

Tournez la page S.V.P.

1 – On suppose que, pour tout i appartenant à $\{1, \ldots, n-1\}$:

$$\frac{u_{i-1}}{p_{i-1}} \geqslant \frac{u_i}{p_i} .$$

On définit un entier i^* compris entre 0 et n par :

$$\begin{cases} \text{si } \displaystyle\sum_{i=0}^{n-1} p_i < Q, \text{ alors } i^* = n \\ \text{si } p_0 \geqslant Q, \text{ alors } i^* = 0 \\ \text{si } p_0 < Q \text{ et } \displaystyle\sum_{i=0}^{n-1} p_i \geqslant Q, \text{ alors } i^* \text{ vérifie } \sum_{i=0}^{i^*-1} p_i < Q \text{ et } \sum_{i=0}^{i^*} p_i \geqslant Q \end{cases}$$

a – Montrer qu'une solution maximale pour le problème est donnée par :

$$\begin{cases} \text{pour } i \text{ vérifiant } 0 \leqslant i \leqslant i^* - 1, \ x_i = 1 \\ \text{pour } i \text{ vérifiant } i^* + 1 \leqslant i \leqslant n-1, \ x_i = 0 \\ \text{si } i^* \neq n, \ x_{i^*} = \dfrac{Q - \sum_{i=0}^{i^*-1} p_i}{p_{i^*}} \end{cases}$$

b – Résoudre le problème de sac à dos fractionnaire suivant :

$$\begin{cases} \text{Maximiser } z = 16x_0 + 21x_1 + 19x_2 + 15x_3 + 13x_4 + 7x_5 \\ \text{avec les contraintes :} \\ \qquad 15x_0 + 22x_1 + 20x_2 + 17x_3 + 15x_4 + 9x_5 \leqslant 51 \\ \qquad \text{et pour } i \in \{0,1,2,3,4,5\}, \ x_i \in [0,1] \end{cases}$$

On donnera la valeur du maximum de la fonction z et les valeurs de x_0, x_1, x_2, x_3, x_4 et x_5 permettant d'atteindre ce maximum.

c – On revient au problème général du sac à dos fractionnaire.
Dans le langage de programmation choisi, écrire une fonction ou une procédure qui examine si les inégalités :

$$\frac{u_{i-1}}{p_{i-1}} \geqslant \frac{u_i}{p_i}$$

sont bien vérifiées pour tout i appartenant à $\{1, 2, \ldots, n-1\}$.

d – En supposant que, pour tout i appartenant à $\{1, 2, \ldots, n-1\}$, les inégalités $\dfrac{u_{i-1}}{p_{i-1}} \geqslant \dfrac{u_i}{p_i}$ sont vérifiées, écrire, dans le langage de programmation choisi, une fonction qui calcule i^*.

2 – *Remarque* : le reste du problème ne dépend pas de cette question 2.

a – On cherche à déterminer la solution du problème du sac à dos fractionnaire **sans faire l'hypothèse** que, pour tout i appartenant à $\{1, \ldots, n-1\}$:

$$\frac{u_{i-1}}{p_{i-1}} \geqslant \frac{u_i}{p_i} .$$

En revanche, on suppose que l'on a $\sum_{i=0}^{n-1} p_i \geqslant Q$.

On cherche à déterminer i^* appartenant à $\{0,1,\ldots,n-1\}$ et une partition de l'ensemble $\{0,1,\ldots,n-1\} \setminus \{i^*\}$ en deux sous-ensembles I' et I'' de façon à avoir simultanément :

$$\begin{cases} \text{pour tout } i \in I',\ \dfrac{u_i}{p_i} \geqslant \dfrac{u_{i^*}}{p_{i^*}} \\ \text{pour tout } j \in I'',\ \dfrac{u_{i^*}}{p_{i^*}} \geqslant \dfrac{u_j}{p_j} \\ \displaystyle\sum_{i \in I'} p_i < Q \\ p_{i^*} + \displaystyle\sum_{i \in I'} p_i \geqslant Q \end{cases}$$

On dit qu'un algorithme est *linéaire* en une variable v liée aux données traitées s'il existe une fonction linéaire de v majorant le nombre d'opérations élémentaires (opérations arithmétiques, comparaisons...) effectuées par cet algorithme pour traiter ces données.

On admet qu'on dispose d'un algorithme $\mathcal{A}$ qui, étant donné un ensemble E de m éléments ayant chacun une valeur, partitionne cet ensemble en deux sous-ensembles E' et E'' de cardinaux respectifs $\lfloor \frac{m}{2} \rfloor$ et $\lceil \frac{m}{2} \rceil$ de telle sorte que toutes les valeurs des éléments de E' soient supérieures ou égales à toutes les valeurs des éléments de E'' ; on suppose de plus que $\mathcal{A}$ est linéaire en m.

Expliciter, sans utiliser de langage de programmation, un algorithme récursif (qu'il ne sera pas nécessaire de prouver) utilisant l'algorithme $\mathcal{A}$ qui détermine i^*, I' et I'' ; cet algorithme devra être linéaire en n.

b – Prouver la linéarité en n de l'algorithme proposé ci-dessus. Pour simplifier cette preuve, on pourra se restreindre aux valeurs de n qui sont des puissances de 2 ; on admettra alors que l'algorithme est aussi linéaire lorsque n est quelconque.

c – Déduire des questions 1 - a, 2 - a et 2 - b qu'on peut résoudre le problème du sac à dos fractionnaire portant sur n objets en un temps majoré par une fonction linéaire de n même si les objets ne sont pas au préalable classés pour que les rapports $\frac{u_i}{p_i}$ ($i \in \{0,1,\ldots,n-1\}$) soient décroissants.

SECONDE PARTIE
Sac à dos en 0-1

Dans cette partie, les objets ne sont plus fractionnables : chaque objet est pris ou laissé. Le problème s'écrit :

$$\begin{cases} \text{Maximiser } z = \displaystyle\sum_{i=0}^{n-1} u_i x_i \\ \text{avec les contraintes :} \\ \quad \displaystyle\sum_{i=0}^{n-1} p_i x_i \leqslant Q \\ \quad \text{et pour } i \in \{0,1,\ldots,n-1\},\ x_i \in \{0,1\} \end{cases}$$

On dit que $\overline{x} = (\overline{x_0}, \overline{x_1}, \ldots, \overline{x_{n-1}}) \in \{0,1\}^n$ est *réalisable* si on a $\sum_{i=0}^{n-1} p_i \overline{x_i} \leqslant Q$. On appelle *valeur* d'une telle solution la valeur correspondante de la fonction z ou, autrement dit, la quantité $\sum_{i=0}^{n-1} u_i \overline{x_i}$. Le

problème consiste donc à déterminer la meilleure solution réalisable, c'est-à-dire celle qui a la plus grande valeur.

3 – Montrer que le maximum du problème du sac à dos en 0-1 est inférieur ou égal au maximum du problème de sac à dos fractionnaire ayant les mêmes données numériques.

4 – **Méthode par séparation**

Écrire, dans le langage de programmation choisi, un algorithme fondé sur la stratégie *diviser pour régner* qui exhibe tour à tour toutes les solutions réalisables et mémorise la meilleure.

Cet algorithme utilisera le principe esquissé ci-après : la meilleure solution réalisable peut être cherchée successivement parmi les solutions réalisables avec $x_0 = 1$ (s'il en existe) puis parmi celles avec $x_0 = 0$. La meilleure solution réalisable pour laquelle $x_0 = 1$ peut être cherchée successivement parmi les solutions réalisables avec $x_1 = 1$ (s'il en existe) puis parmi celles avec $x_1 = 0$, etc.

Avant d'écrire l'algorithme en langage de programmation :

- on donnera la signification des différentes variables utilisées (autres que celles introduites par l'énoncé) ;
- on précisera le rôle des fonctions ou procédures utilisées.

5 – Indiquer la complexité dans le pire des cas de la méthode par séparation.

6 – **Méthode par séparation et évaluation**

Remarque : cette question peut être traitée sans avoir résolu les deux questions précédentes.

La méthode par séparation et évaluation améliore la méthode par séparation en utilisant la résolution du problème de sac à dos fractionnaire ; elle se place dans l'hypothèse où, pour tout i appartenant à $\{1, \ldots, n-1\}$, on a : $\dfrac{u_{i-1}}{p_{i-1}} \geqslant \dfrac{u_i}{p_i}$.

On ne fera qu'esquisser cette méthode à travers un exemple.

On associe au problème de sac à dos en 0-1 un arbre binaire décrit ci-dessous.

La racine de l'arbre représente l'ensemble de toutes les solutions réalisables.

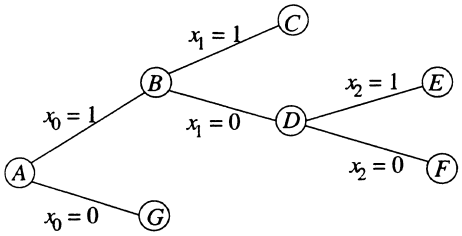
À un sommet (nœud ou feuille) quelconque de l'arbre correspondent un indice i_0 ($0 \leqslant i_0 \leqslant n$) et un i_0-uplet $(\overline{x_0}, \overline{x_1}, \ldots, \overline{x_{i_0-1}}) \in \{0,1\}^{i_0}$; ce sommet représente l'ensemble, qui doit être non vide, des solutions réalisables $(x_0, x_1, \ldots, x_{n-1})$ pour lesquelles on a : $x_0 = \overline{x_0}$, $x_1 = \overline{x_1}$, $\ldots$, $x_{i_0-1} = \overline{x_{i_0-1}}$; ces solutions réalisables seront dites *contenues par ce sommet*. Si $i_0 \neq n$, un tel sommet possède 1 ou 2 fils : un fils correspondant à l'indice $i_0 + 1$ et à $(\overline{x_0}, \overline{x_1}, \ldots, \overline{x_{i_0-1}}, 0)$ et, si $p_{i_0} + \sum_{i=0}^{i_0-1} p_i \overline{x_i} \leqslant Q$, un second fils correspondant à l'indice $i_0 + 1$ et à $(\overline{x_0}, \overline{x_1}, \ldots, \overline{x_{i_0-1}}, 1)$.

On appelle *évaluation d'un sommet* un majorant commun aux valeurs de toutes les solutions réalisables contenues par ce sommet.

On considère dans toute la fin du problème l'exemple suivant, qui sera appelé le problème $(\mathcal{P})$:

$$(\mathcal{P}) \qquad \begin{cases} \text{Maximiser } z = 16x_0 + 21x_1 + 19x_2 + 15x_3 + 13x_4 + 7x_5 \\ \text{avec les contraintes :} \\ \quad 15x_0 + 22x_1 + 20x_2 + 17x_3 + 15x_4 + 9x_5 \leqslant 51 \\ \quad \text{et pour } i \in \{0,1,2,3,4,5\},\ x_i \in \{0,1\} \end{cases}$$

Une partie de l'arbre associé est dessinée ci-dessous :



a – Déterminer directement la meilleure solution réalisable contenue par le sommet C (obtenu en fixant $x_0 = x_1 = 1$).

b – Déterminer directement la meilleure solution réalisable contenue par le sommet E (obtenu en fixant $x_0 = 1, x_1 = 0, x_2 = 1$).

c – Résoudre le problème de sac à dos fractionnaire ci-dessous :

$$\begin{cases} \text{Maximiser } 15x_3 + 13x_4 + 7x_5 \\ \text{avec les contraintes :} \\ \quad 17x_3 + 15x_4 + 9x_5 \leqslant 36 \\ \quad \text{et pour } i \in \{3,4,5\},\ x_i \in [0,1] \end{cases}$$

En déduire une évaluation du sommet F (obtenu en fixant $x_0 = 1$, $x_1 = x_2 = 0$). La solution réalisable optimale du problème $(\mathcal{P})$ peut-elle être contenue dans F ?

d – Montrer, en utilisant la même démarche que dans la question précédente, que la solution réalisable optimale du problème $(\mathcal{P})$ n'est pas contenue dans le sommet G (obtenu en fixant $x_0 = 0$).

e – Déduire des questions précédentes la solution optimale du problème $(\mathcal{P})$.

FIN DU PROBLÈME D'ALGORITHMIQUE

3 Exercice sur les automates finis — 30 mn environ

Rappels pour fixer les notations

Un *alphabet* A est un ensemble fini d'éléments, appelés *lettres*. L'ensemble des suites finies de lettres de A, appelées *mots*, est noté A^*. Le mot vide est noté 1_{A^*}. Un *langage* est une partie de A^*.

Un automate fini $\mathcal{A}$ est un *graphe orienté* et *étiqueté*, noté par un quintuplet $<Q,A,E,I,T>$: A est un alphabet ; Q est l'ensemble (fini) des *états* de $\mathcal{A}$: ce sont les *sommets* du graphe ; I et T, deux sous-ensembles de Q, sont respectivement l'ensemble des *états initiaux* et des *états terminaux* de $\mathcal{A}$. Enfin, E, sous-ensemble de $Q \times A \times Q$, est appelé ensemble des *transitions* de $\mathcal{A}$. Une transition (p,a,q) est un arc du graphe, allant de l'état p à l'état q et étiqueté par a ; on la note également $p \xrightarrow{a} q$.

Un *calcul* de $\mathcal{A}$ est un chemin c dans le graphe : $c = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_n} p_n$, où, pour tout i, $0 \leqslant i \leqslant n-1$, $p_i \xrightarrow{a_{i+1}} p_{i+1}$ appartient à E. L'état p_0 est *l'origine* du calcul c, p_n *l'extrémité* de c. La

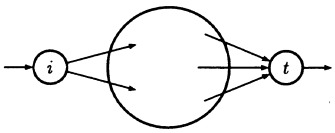
Tournez la page S.V.P.

longueur du calcul est donnée par le nombre n d'arcs de c. L'*étiquette* de c est le mot f formé par la suite des étiquettes des transitions successives du calcul c : $f = a_1 a_2 \dots a_n$. Le calcul c peut aussi être noté $p_0 \xrightarrow[\mathcal{A}]{f} p_n$; il est *réussi* si p_0 est initial et p_n terminal.

Un mot de A^* est *reconnu* par l'automate $\mathcal{A}$ si c'est l'étiquette d'(au moins) un calcul *réussi* de $\mathcal{A}$. Le *langage reconnu* par $\mathcal{A}$, noté $L(\mathcal{A})$, est l'ensemble des mots reconnus par $\mathcal{A}$. Un langage est *reconnaissable* s'il est reconnu par un automate fini.

On dira qu'un automate $\mathcal{A} = <Q,A,E,I,T>$ est *normalisé* si :
i) I est un singleton, $I = \{i\}$, qui n'est l'extrémité d'aucune transition de $\mathcal{A}$;
ii) T est un singleton, $T = \{t\}$, qui n'est l'origine d'aucune transition de $\mathcal{A}$.

Un automate normalisé est donc naturellement représenté par le schéma ci-dessous (noter que certaines transitions peuvent aussi aller de i à t, même si elles n'apparaissent pas sur le schéma).



Énoncé de l'exercice

Si un mot f de A^* se factorise en $f = u\,v$, avec u et v dans A^*, le mot $g = v\,u$ est un *conjugué* de f. Soit Conj : $A^* \to \mathcal{P}(A^*)$ l'application qui à un mot f fait correspondre l'ensemble de ses conjugués :

$$\mathsf{Conj}(f) = \{v\,u \mid (u,v) \in (A^*)^2 \text{ avec } u\,v = f\}.$$

En particulier, pour tout f, $f \in \mathsf{Conj}(f)$ puisque $f = 1_{A^*} f = f\,1_{A^*}$ et, pour tout $f \in A \cup \{1_{A^*}\}$, $\mathsf{Conj}(f) = \{f\}$. L'application s'étend par additivité : si L est un langage, $\mathsf{Conj}(L) = \bigcup_{f \in L} \mathsf{Conj}(f)$.

Dans tout l'exercice, ***L* est un langage reconnu par un automate fini**. L'objet de l'exercice est de montrer que $\mathsf{Conj}(L)$ est reconnaissable.

1 – Montrer que $L \setminus \{1_{A^*}\}$ est reconnu par un automate (fini) normalisé.

2 – Soient $\mathcal{A} = <Q,A,E,\{i\},\{t\}>$ un automate normalisé qui reconnaît le langage $L \setminus \{1_{A^*}\}$ et q un état de $\mathcal{A}$ distinct de i et t. À chaque calcul réussi de $\mathcal{A}$ qui passe par q (et donc de longueur au moins égale à 2) d'étiquette $f = f_1 f_2$: $i \xrightarrow[\mathcal{A}]{f_1} q \xrightarrow[\mathcal{A}]{f_2} t$, on associe le conjugué $g = f_2 f_1$ de f et on note G_q l'ensemble des mots obtenus de cette façon :

$$G_q = \{g = f_2 f_1 \mid i \xrightarrow[\mathcal{A}]{f_1} q \xrightarrow[\mathcal{A}]{f_2} t \text{ est un calcul réussi de } \mathcal{A}\}.$$

Construire un automate qui reconnaît le langage G_q.

3 – Montrer que $\mathsf{Conj}(L)$ est reconnaissable.

FIN DE L'EXERCICE SUR LES AUTOMATES

FIN DE L'ÉPREUVE

page 8/8