DS 4

Option informatique, deuxième année

Julien Reichert

Partie 1 : Logique

Vous participez à un concours de mathématiques comportant une partie de raisonnement logique. Plusieurs orateurs font des déclarations et vous devez répondre à des questions en vous appuyant sur des informations déduites de ces déclarations. La règle suivante s'applique : « Les orateurs sont de trois natures : les véridiques, les menteurs et les changeants. Les véridiques disent toujours la vérité, les menteurs mentent toujours, et les changeants disent en alternance une vérité et un mensonge (c'est-à-dire, soit une vérité, puis un mensonge, puis une vérité, etc. ; soit un mensonge, puis une vérité, puis un mensonge, etc.). Pendant tout le concours, les orateurs ne peuvent pas changer de nature. »

Les épreuves comportent deux phases :

- Les différents orateurs font plusieurs déclarations dont l'analyse permet de déterminer la nature de chaque orateur (véridique, menteur, changeant commençant par dire la vérité, ou changeant commençant par dire un mensonge).
- Les orateurs font une seconde série de déclarations. Puis, vous devez répondre à des questions en exploitant les informations contenues dans ces déclarations.

Question 1.1 : Dans la première phase, quel est le nombre minimum de déclarations que doit faire chaque orateur pour qu'il soit possible de déterminer sa nature ? Justifier.

Question 1.2 : Soit un orateur A qui fait une suite de n déclarations A_i. Proposer des formules du calcul des propositions A_V, A_M, A_{CV} et A_{CM} qui permettent de caractériser la nature de A (respectivement véridique, menteur, changeant commençant par dire la vérité, ou changeant commençant par dire un mensonge).

Vous participez à une première épreuve avec un orateur A qui fait les déclarations suivantes :

- J'aime le rouge mais pas le bleu.
- Soit j'aime le rouge, soit j'aime le vert.
- Si j'aime le rouge et le vert, alors j'aime le bleu.

Nous noterons R, V et B les variables propositionnelles associées au fait que l'orateur aime le rouge, le vert ou le bleu. Nous noterons A_1 , A_2 et A_3 les formules propositionnelles associées aux déclarations de A.

Question 1.3 : Représenter les déclarations de l'orateur sous la forme de formules du calcul des propositions A_1, A_2 et A_3 dépendant des variables R, V et B.

Question 1.4 : Appliquer les formules permettant de caractériser la nature des orateurs proposées pour la question 2.2 pour l'orateur A dépendant des variables A_1, A_2 et A_3.

Question 1.5 : En utilisant le calcul des propositions (résolution avec les tables de vérité, par exemple), déterminer la nature de l'orateur A. Quelles sont les couleurs qu'aime A ?

Vous participez à une seconde épreuve avec trois orateurs G, H et I. Vous avez déterminé dans la première phase avec succès que G est un menteur, que H est un véridique et que I est un changeant sans savoir s'il doit dire la vérité ou un mensonge pour sa déclaration suivante. Ceux-ci font les déclarations :

— I : Le losange est visible
— G : Le cercle n'est visible que si le losange est visible.
— I : Le triangle n'est pas visible.
— H : Soit le cercle est visible, soit le triangle est visible.

Nous noterons G_1, H_1, I_1 et I_2 les formules propositionnelles associées aux déclarations des orateurs dans cette épreuve. Nous noterons C, L et T les variables propositionnelles associées au fait que le cercle, le losange ou le triangle soit visible.

Question 1.6 : Représenter les déclarations des orateurs sous la forme de formules du calcul des propositions G_1, H_1, I_1 et I_2 dépendant des variables C, L et T.

Question 1.7 : Représenter les informations sur la nature des orateurs sous la forme d'une formule du calcul des propositions dépendant des variables G_1, H_1, I_1 et I_2.

Question 1.8 : En utilisant le calcul des propositions (résolution avec les formules de De Morgan, par exemple), déterminer quelle est (ou quelles sont) la (ou les) figure(s) visible(s) ainsi que la nature exacte de l'orateur changeant I.

Partie 2 : Homomorphismes de langages

Le but de cet exercice est de prouver qu'un homomorphisme de langage préserve la structure de langage régulier : l'image et l'image réciproque d'un langage régulier par un homomorphisme de langage sont des langages réguliers.

On suppose les définitions de base sur les automates, les langages et les expressions régulières connues. Soient deux alphabets Σ et Σ_2, un homomorphisme de langage h est une application de Σ^* dans Σ_2^* telle que $h(\varepsilon) = \varepsilon$ et pour $w_1, w_2 \in \Sigma^*$, on a $h(w_1 w_2) = h(w_1)h(w_2)$. Il est dit ε-libre si aucun mot non vide n'a pour image le mot vide.

L'image d'un langage $L \subseteq \Sigma^*$ par un homomorphisme h est $h(L) = \{h(w) \mid w \in L\} \subseteq \Sigma_2^*$, et l'image réciproque d'un langage $L \subseteq \Sigma_2^*$ est $h^{-1}(L) = \{w \mid h(w) \in L\} \subseteq \Sigma^*$.

Nous décomposons l'étude de la préservation de la structure de langage régulier par les homomorphismes de langage en plusieurs étapes qui reposent sur une restriction des homomorphismes de langage à un alphabet.

Soient deux alphabets Σ et Σ_2, soit une application $h : \Sigma \to \Sigma_2^*$, nous notons $\hat{h}$ l'extension (dite de Kleene) de h à Σ^* définie par $\hat{h}(\varepsilon) = \varepsilon$ et $\forall \sigma \in \Sigma, w \in \Sigma^*, \hat{h}(\sigma w) = h(\sigma)\hat{h}(w)$.

Question 2.1 : Montrer qu'on définit ainsi un homomorphisme de langage, qui est ε-libre si aucune lettre n'a pour image ε.

Question 2.2 : Montrer que l'extension de Kleene de la restriction à Σ d'un homomorphisme de langage est l'homomorphisme lui-même.

Pour montrer que l'image d'un langage régulier par un homomorphisme de langage est un langage régulier, nous exploitons la définition des langages réguliers sous la forme d'expressions régulières.

Un homomorphisme d'expression régulière s'appuyant sur une fonction h de Σ dans Σ_2^* est une application de domaine l'ensemble des expressions régulières d'alphabet Σ, notée $\tilde{h}$, par laquelle l'image de l'expression vide (à ne pas confondre avec le mot vide) est l'expression vide, l'image de l'expression ε est ε, l'image d'une lettre correspond à son image par h, et, pour deux expressions régulières e_1 et e_2, l'image de $e_1 + e_2$ est $\tilde{h}(e_1) + \tilde{h}(e_2)$, l'image de $e_1.e_2$ est $\tilde{h}(e_1).\tilde{h}(e_2)$ et l'image de e_1^* est $(\tilde{h}(e_1))^*$.

Question 2.3 : Soient les alphabets $\Sigma = \{a, b\}$ et $\Sigma_2 = \{0, 1\}$, soit $h : \Sigma \to \Sigma_2^*$ définie par $h(a) = 10$ et $h(b) = 0$, calculer $\tilde{h}(a(ba + aba)^*)$.

Question 2.4 : Montrer que l'image d'une expression régulière par un homomorphisme d'expression régulière est une expression régulière.

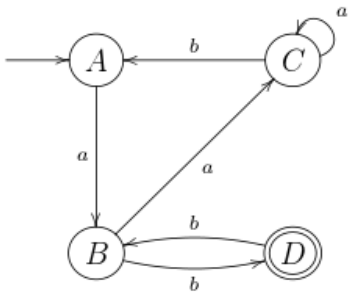
Question 2.5 : Avec les notations précédentes, montrer que $L(\tilde{h}(e)) = \hat{h}(L(e))$.

Question 2.6 : Montrer que l'image par un homomorphisme ε-libre de langage d'un langage régulier est un langage régulier.

Pour montrer que l'image réciproque d'un langage régulier par un homomorphisme de langage est un langage régulier, nous exploitons la définition des langages réguliers sous la forme d'automates finis.

On définit la fermeture réflexive et transitive sur Σ^* de la relation de transition T d'un automate d'alphabet Σ comme la relation T^* contenant les (q, w, q') pour lesquels il existe un chemin de transitions successives sur les lettres de w dans l'ordre entre q et q' dans l'automate.

Question 2.7 : Sans justification, donner une expression régulière représentant le langage sur $\{a, b\}$ reconnu par l'automate représenté ci-après. On pourra aussi donner une expression mathématique caractérisant ce langage.



Question 2.8 : Dans un automate, avec les notations intuitives, montrer que pour tous $q, q' \in Q$ et pour tous $w_1, w_2 \in \Sigma^*$, $(q, w_1 w_2, q') \in T^* \Leftrightarrow \exists q'' \in Q, (q, w_1, q'') \in T^*$ et $(q'', w_2, q') \in T^*$.

L'image réciproque du langage reconnu par un automate fini par un homomorphisme de langage peut être obtenue par transformation de cet automate selon la définition suivante.

Soient Σ et Σ' deux alphabets, soit h une application de Σ dans Σ'^*, soit $\mathcal{A} = (Q, \Sigma', I, F, T)$ un automate fini, l'automate $\mathcal{B} = h^{-1}(\mathcal{A})$, image réciproque de l'automate $\mathcal{A}$, est défini par $\mathcal{B} = (Q, \Sigma, I, F, T_{\hat{h}^{-1}})$, où $\forall \sigma \in \Sigma, \forall q \in Q, \forall q' \in Q, (q, \sigma, q') \in T_{\hat{h}^{-1}}$ ssi $(q', \hat{h}(\sigma), q) \in T^*$.

Question 2.9 : Soient deux alphabets $\Sigma = \{a, b\}$ et $\Sigma' = \{0, 1\}$, soit $h : \Sigma' \to \Sigma$ telle que $h(0) = ab$ et $h(1) = b$, construire l'automate $\hat{h}^{-1}(\mathcal{A})$, où $\mathcal{A}$ est l'automate décrit dans la figure ci-avant.

Question 2.10 : Caractériser le langage reconnu par $\hat{h}^{-1}(\mathcal{A})$ par une expression régulière ou une caractérisation ensembliste. Comparer le langage reconnu par $\hat{h}(\hat{h}^{-1}(\mathcal{A}))$ avec le langage reconnu par $\mathcal{A}$.

Question 2.11 : Soit un automate d'alphabet Σ, avec les notations intuitives, soit $h : \Sigma \to \Sigma'$, montrer que : $\forall w \in \Sigma^*, \forall q, q' \in Q, (q, w, q') \in T^*_{\hat{h}^{-1}} \Leftrightarrow (q, \hat{h}(w), q') \in T^*$.

Question 2.12 : Quelle relation liant les langages reconnus par un automate et par son image réciproque (à la manière de ce qui a été défini ci-avant) peut-on déduire des questions précédentes ?

Question 2.13 : Soient deux alphabets Σ et Σ', soit $h : \Sigma \to \Sigma'^*$ un homomorphisme de langage ε-libre, soit $L_{\Sigma'}$ un langage régulier sur Σ', montrer que $h^{-1}(L_{\Sigma'})$ est un langage régulier sur Σ.

Partie 3 : Programmation

Toutes les questions sont indépendantes.

Exercice P.1 : Avec les types suivants pour les automates, écrire une fonction qui crée à partir d'un automate le graphe induit par retrait des transitions (ce qui peut servir de première étape à la vérification, pour laquelle on ne demande pas ici d'écrire de fonction, que le langage d'un automate est non vide...).

```
type automate = { etats : string list ; alphabet : char list ;
initiaux : string list ; finaux : string list ;
transitions : (string * char * string) list ; };;

type graphe1 = { sommets : string list; arcs : (string * string) list; };;
```

Exercice P.2 : Écrire une fonction qui prend en entrée un tableau et qui détermine la plus grande somme d'éléments consécutifs dans ce tableau. Si la complexité n'est pas linéaire, **aucun point** ne sera accordé. Indication : il est utile de connaître la plus petite somme d'éléments consécutifs du début à n'importe quel indice.

Exercice P.3 : Implémenter l'algorithme de Thomson.

Pour rappel, et notamment car il est hors-programme et fait intervenir la notion hors-programme d'epsilon-transitions, l'algorithme de Thomson permet de créer un automate avec epsilon-transitions à partir d'une expression rationnelle, l'automate obtenu devant être par la suite transformé en un automate-déterministe. Cet algorithme procède par induction sur l'expression rationnelle, en l'occurrence :

- Tous les automates auront un état initial q_0, un état final q_f et un « corps » avec d'éventuels états.
- L'automate pour l'expression vide n'a pas de transition.
- L'automate pour l'expression « mot vide » a une epsilon-transition entre q_0 et q_f.
- L'automate pour une lettre a une transition étiquetée par cette lettre entre q_0 et q_f.
- Le corps de l'automate pour l'expression $e_1 + e_2$ est formé de la réunion des automates pour e_1 et e_2 avec des epsilon-transitions de q_0 vers les deux états initiaux des automates et des deux états finaux des automates vers q_f.
- Le corps de l'automate pour l'expression $e_1 e_2$ est formé de la réunion des automates pour e_1 et e_2 avec des epsilon-transitions de q_0 vers l'état initial de l'automate pour e_1, de l'état final de l'automate pour e_1 vers l'état initial de l'automate pour e_2 et de l'état final de l'automate pour e_2 vers q_f.
- Le corps de l'automate pour l'expression e^* est formé de l'automate pour e avec des epsilon-transitions de q_0 vers l'état initial de l'automate pour e, de l'état final de l'automate pour e vers l'état initial de l'automate pour e, de l'état final de l'automate pour e vers q_f et de q_0 vers q_f.

On utilisera les types suivants pour les expressions rationnelles et pour les automates avec epsilon-transitions :

```
type erat = Epsilon | Lettre of char | Somme of erat * erat
| Concat of erat * erat | Etoile of erat;;

type expression_rationnelle = Vide | Expr of erat;;

type automate_epsilon_transitions = { etats : string list ;
alphabet : char list ; initiaux : string list ; finaux : string list ;
transitions : (string * string * string) list ; };;
(* élément du milieu dans les transitions : chaîne vide ou de taille 1 *)
```