IFT2105–Introduction à l'informatique théorique (Examen Intra – E2021–Durée: 120 minutes)

Louis Salvail

Université de Montréal (DIRO), QC, Canada
salvail@iro.umontreal.ca
Bureau: Pavillon André-Aisenstadt, #3369

IMPORTANT: La remise doit être complétée sur STUDIUM avant 14:15, le mercredi 23 juin 2021. Vous ne remettez qu'un seul fichier PDF avec toutes les réponses aux numéros clairement indiquées. Pour aider la correction, assurez-vous de vous identifier sur la première page de votre remise. Indiquez également votre matricule. Il est très important de répondre clairement. L'association d'une réponse à sa question devrait être évidente, peu importe l'ordre dans lequel vous présentez vos réponses. Vous pouvez évidemment soumettre des réponses manuscrites. Sachez cependant que ma tolérance aux réponses illisibles ou indéchiffrables est très mince.

Rappels et notations

N'oubliez pas que REG désigne la classe des langages réguliers et FINI celle des langages finis. Si Q est une ensemble alors $\mathcal{P}(Q)$ dénote son ensemble puissance.

Par la suite, *AFD* désigne un automate fini déterministe et *AFN* désigne un automate fini non-déterministe. Si $w, w' \in \Sigma^*$, $w \cdot w'$ dénote leur concaténation. Nous notons la concaténation également par ww' pour alléger la notation. Pour $w = w_1 w_2 \dots w_n \in \Sigma^*$, nous dénotons par w^R la chaîne w renversée: $w_n w_{n-1} \dots w_1$. Pour L un langage, $\overline{L}$ dénote son complément. Pour L un langage, L^* désigne le langage qui contient la concaténation d'un nombre fini arbitraire (possiblement 0) de mots de L.

La fonction $B_i : \mathbb{N} \to \mathbb{N}$ est la fonction du chapitre 2 donnée par:

$$B_i(x) = \begin{cases} x+1 & \text{si} \quad i = 0 \\ B_{i-1}^{\langle x+1 \rangle}(1) & \text{si} \quad i > 0. \end{cases}$$

Questions à développement

1. [15PTS] Un langage de programmation est dit *universel* s'il est possible d'écrire un programme dans le langage qui accepte un programme arbitraire du même langage en plus d'un input et simule le programme donné sur l'input donné. Par exemple, il est usuel d'écrire des interpréteurs et des compilateurs en C pour des programmes écrits également en C. Les interpréteurs et les compilateurs écrits en C pour le langage C montrent que C est un langage universel. Est-ce que le langage `RÉPÉTER` est universel? Prouvez formellement votre réponse.

2. [10 PTS] Donnez un argument convaincant (mais informel) qui montre que le langage de programmation `TANTQUE` est universel. Vous pouvez utiliser la thèse de Church-Turing et le théorème 3.12 (les programmes `TANTQUE` peuvent simuler les machines de Turing) pour y parvenir.

3. [15PTS] Donnez un AFD $M_3 = (Q, \Sigma, \delta, q_0, F)$ qui reconnaît le langage

$$L(M_3) = \{w \cdot \texttt{abba} \,|\, w \in \Sigma^*\} \cap \{\texttt{ab} \cdot w \cdot \texttt{ba} \,|\, w \in \Sigma^*\} \ ,$$

sur alphabet $\Sigma = \{\texttt{a}, \texttt{b}\}$. Représentez la fonction de transition δ de M_3 par un diagramme de transition (la représentation graphique de la fonction de transition que nous avons vue en cours). Soyez clair et assurez-vous que le diagramme de transition indique clairement Q, q_0 et F.

4. [10PTS] Soit $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ et $L = \{w \in \Sigma^* \mid w = w^R\}$. Montrez que $L \notin \mathsf{REG}$. Soyez clair et précis.

5. [15PTS] Est-ce que $L = \{z \cdot w | w \in \{\mathtt{0},\mathtt{1}\}^*,$ avec $z = \mathtt{11}$ si $|w|_{\mathtt{0}} > |w|_{\mathtt{1}}$ et $z = \mathtt{00}$ sinon$\} \in$ REG? Prouvez votre réponse clairement sans omettre les détails.

6. [10PTS] Transformez l'AFN $M_6 = (Q, \Sigma, \delta, q_0, F)$ définie et représenté à la Fig. 1 en AFD qui reconnaît le même langage.
   (a) En premier lieu, éliminez les transitions ε (comme il a été vu en cours) et donnez le diagramme de transition de l'AFN $M_6' = (Q', \Sigma, \delta', q_0', F')$ résultant qui satisfait $L(M_6') = L(M_6)$. Il s'agit d'ajouter des transitions sur les symboles de Σ qui remplacent la fermeture transitive des transitions ε à partir de chaque état. Dans le cas présent, la chose est simple puisqu'il n'y a qu'une seule transition ε et elle n'est pas accessible à partir de l'état initial.
   (b) En second lieu, transformez l'AFN M_6' (sans transition ε) en un AFD $M_6^* = (Q^*, \Sigma, \delta^*, q_0^*, F^*)$ tel que $L(M_6^*) = L(M_6')$, par la méthode décrite dans la preuve du théorème 4.64 (que vous avez vue en cours et en séance de démonstration). N'oubliez pas que votre AFD doit avoir une transition, à partir de chaque état, pour chaque symbole de l'alphabet Σ.

   Rappelez-vous que l'ensemble Q^* des états de M_6^* sont tels $Q^* \subseteq \mathcal{P}(Q')$. Ainsi, notez vos états $q \in Q^*$ par $q \in \mathcal{P}(Q')$ pour aider le correcteur (moi) et montrer que vous avez obtenu M_6^* de la bonne façon. En général, assurez-vous que les diagrammes de transition indiquent clairement Q', q_0', F' pour l'AFN M_6' et Q^*, q_0^*, F^* pour l'AFD M_6^*.

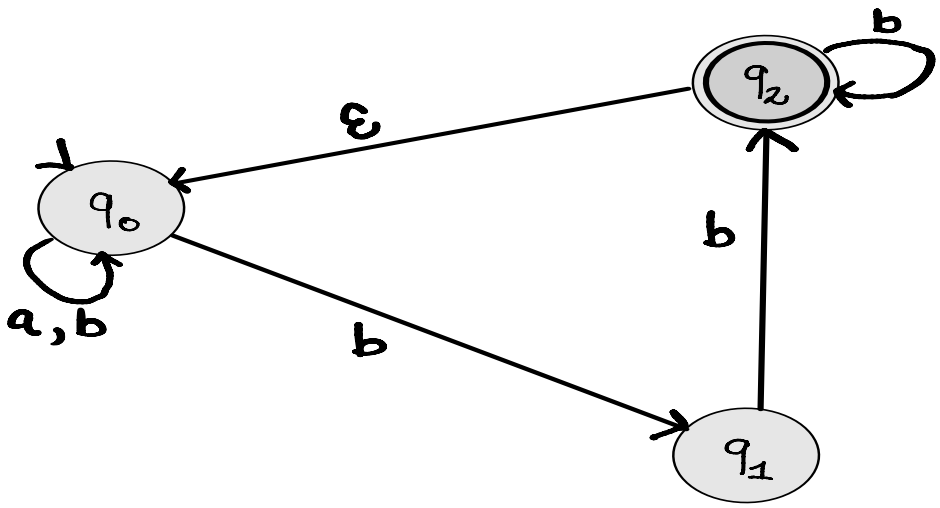

Fig. 1. AFN $M_6 = (Q, \Sigma, \delta, q_0, F)$ avec $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ et $F = \{q_2\}$.

7. [10PTS] Définissez sans ambiguïté un langage $L \in \mathsf{REG}$ qui ne peut pas être reconnu par un AFD qui possède moins de 3 états finaux. Prouvez ensuite que c'est bien le cas.

   Même si le langage donné satisfait la demande, une *preuve* incomplète ou erronée sera évidemment pénalisée. Par *preuve* ici, j'entends n'importe quoi qui réussit à me convaincre.

8. [10PTS] Nous avons vu en cours que les fonctions calculables par une machine de Turing, appelées *fonctions récursives*, sont *également* calculables par un programme `TANTQUE` (théorème 3.12). Nous avons également montré qu'il existe des fonctions récursives qui ne sont pas calculables par un programme `RÉPÉTER` (la fonction d'Ackermann, par exemple). Les fonctions calculables par un programme `RÉPÉTER` sont appelées *primitives récursives*. Considérez maintenant l'ensemble $\mathbf{E}$ des fonctions $f : \{0,1\}^* \to \{0,1\}^*$ qui peuvent être calculées par une machine de Turing, qui sur input $x \in \{0,1\}^n$, produit toujours son output en au plus $2^{B_{17}(n)}$ transitions. La question est la suivante, est-ce que toute fonction $f \in \mathbf{E}$ est primitive récursive? Autrement dit, est-ce que pour chaque fonction $f \in \mathbf{E}$, il existe un programme `RÉPÉTER` qui calcule une fonction $f' : \mathbb{N} \to \mathbb{N}$ telle que $f(x) = y$ implique $f'(x') = y'$, où les entiers x' et y' sont les représentations de Gödel des chaînes x et y respectivement? Répondez par **vrai** ou par **faux** et expliquez votre réponse d'une façon convaincante.

   Une bonne réponse vous donne 3 points. L'explication sera notée sur 7.