MAT115 - Devoir #5

À remettre le **lundi 15 avril avant 23h59**.
Tout retard entraînera jusqu'à 33% de pénalité par jour.

Pondération. Ce travail compte pour 7.5% des points de la session.
Modalités. Considérez les points suivants.

- Le travail peut être fait seul ou en équipe de deux.
- Le devoir doit être remis en format pdf via turnin. La provenance de votre pdf n'a pas d'importance — ce pdf peut provenir d'un export d'un fichier Word ou d'un fichier texte, d'un scan de vos écrits/dessins, etc.

  Il n'y a pas de norme de présentation particulière. Écrivez lisiblement, et assurez-vous que vos noms et CIP apparaissent clairement sur le devoir remis.
- Vous pouvez utiliser les techniques de preuves que vous désirez, les indices ne sont que des suggestions.
- Sauf indication contraire, vous pouvez utiliser les résultats démontrés en classe et dans les solutionnaires d'exercices sans avoir à les démontrer (par exemple, par exemple, vous pouvez utiliser les tables des lois de la logique). Par contre, si vous utilisez de nouveaux résultats, vous devez les démontrer.

Question 1 : création d'automates ($8+8+8+8+8=40$ **points**)

Pour chacun des énoncés ci-bas, donnez un automate dont le langage est l'ensemble de mots spécifié. Vous pouvez remettre un dessin de votre automate ou bien la définition formelle de son quintuplet. Sauf indication contraire, votre automate peut être un AFD ou un AFND.

a. Les codes postaux canadiens, qui sont formés de 6 ou 7 symboles. Quand on lit un code postal de gauche à droite, on doit lire: une lettre, un chiffre, une lettre, un espace *optionnel*, un chiffre, une lettre, puis un chiffre. Par exemple, J1K 2R1 ou J1K2R1 sont des formats corrects.

   On peut représenter l'espace par $\sqcup$. L'alphabet est $[A-Z] \cup [0-9] \cup \{\sqcup\}$.

b. Les mots sur alphabet $\{0,1\}$ qui ont un nombre pair de 0 ou un nombre impair de 1.

c. Les mots sur alphabet $\{0,1\}$ qui n'ont *pas* un nombre pair de 0, et qui n'ont *pas* un nombre pair de 1.

d. Les mots sur alphabet $\{0,1\}$ qui contiennent au moins une occurrence de la sous-chaîne 00100. Votre automate doit être déterministe.

e. Une ligne de code C++ qui déclare une variable de type `int`, possiblement initialisée à un nombre entier. Le nom de variable est sur alphabet [a-z]. On peut ajouter un nombre arbitraire d'espaces entre les éléments de la ligne. Voici cinq exemples de lignes valides que vous devez accepter:

```
int x;
int x = 3;
int allo=3;
    int    salut  =    -3    ;
    int    bonjour    ;
```

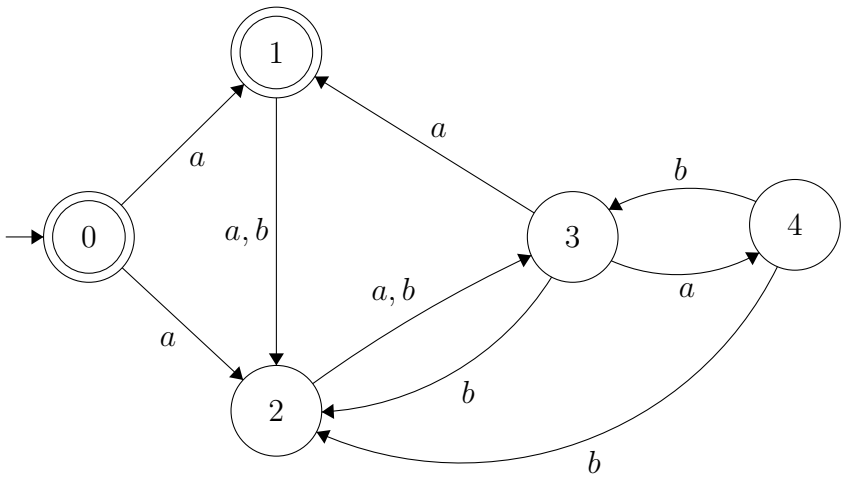
   Notez l'ajout d'espaces dans les deux derniers exemples.

   Pour décrire en détail le format, une telle ligne est constituée, dans l'ordre, de: une série de 0 espaces ou plus (pour l'indentation), suivi du mot `int`, suivi de 1 espace ou plus, suivi du nom de la variable (qui ne contient que des symboles de [a-z]), suivi de 0 espaces ou plus. Ensuite, on peut soit ajouter le symbole ";" pour terminer la déclaration, ou ajouter "=" suivi de 0 espaces ou plus, puis d'un mot représentant un entier positif ou négatif (sans espaces), puis de 0 espaces ou plus, puis de ";". À vous de déterminer l'alphabet.

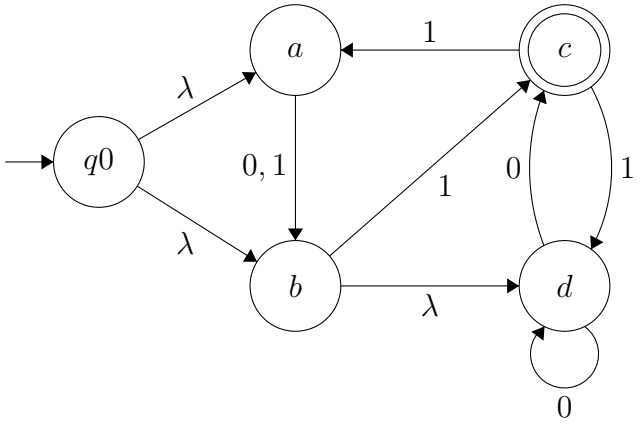
Solution.
Voir autre pdf. □

Question 2 : manipulation d'automates ($10 + 10 + 10 = 30$ **points**)

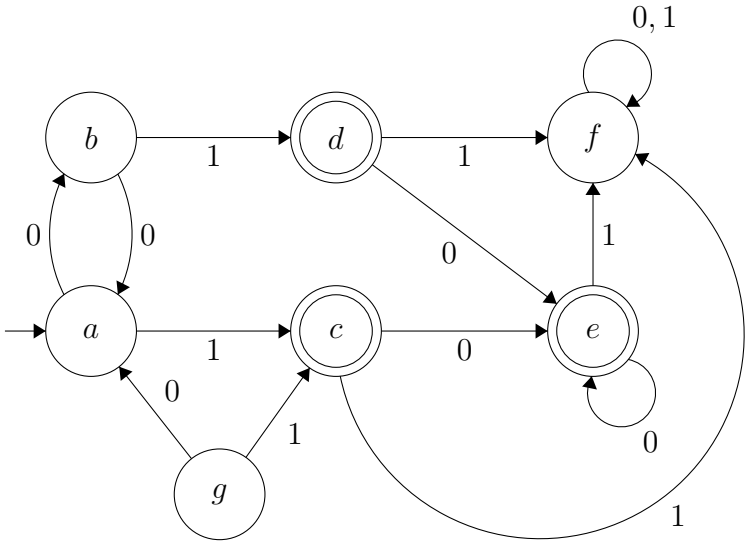
a. Transformez l'AFND suivant en un AFD équivalent.



b. Transformez l'AFND suivant en un AFND équivalent qui n'a pas de transition λ.



c. Minimisez l'AFD suivant. Dites dans quel ordre vous avez déterminé vos partitions en états indistinguables (sans détailler comment vous avez obtenu vos partitions).



<u>Solution</u>.
Voir autre pdf. □

Question 3 : maths et automates (5 + 5 + 10 + 10 points)

a. Soit $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ un AFND. Soit un AFD $A' = \langle Q', \Sigma', \delta', q_0', F'\rangle$ tel que $L(A) = L(A')$ obtenu avec les approches vues en classe. Expliquez pourquoi il est vrai que $|Q'| \leq 2^{|Q|}$.

Solution.
On a vu en classe que la première étape pour transformer A en A' est de poser Q' comme étant l'ensemble d'états $Q' = \mathbb{P}(Q) = \{R | R \subseteq Q\}$. Aussi, on a déjà vu que le nombre de sous-ensembles de Q était $2^{|Q|}$, ce qui justifie la borne donnée dans la question.

□

b. Soit $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ un AFND. Montrez que $|\delta| \leq |Q|^2 \cdot |\Sigma|$.

Solution.
Preuve courte et acceptable: on sait que $\delta \subseteq Q \times \Sigma \times Q$ et que $|Q \times \Sigma \times Q| = |Q| \cdot |\Sigma| \cdot |Q| = |Q|^2|\Sigma|$.

Autre style de preuve acceptable. Rappelons que les éléments de δ ont la forme (q, a, r) où $q, r \in Q$ et $a \in \Sigma$. Le nombre de triplets (q, a, r) possibles est égal au nombre de façons de poser q, fois le nombre de façons de poser a, fois le nombre de façons de poser r. Ceci donne $|Q| \cdot |\Sigma| \cdot |Q| = |Q|^2|\Sigma|$ combinaisons. □

c. Un ensemble de mots $M \subseteq \Sigma^*$ est appelé *régulier* s'il existe un AFD ou un AFND A tel que $L(A) = M$. Montrez que si $M_1, M_2 \subseteq \Sigma^*$ sont deux ensembles de mots réguliers, alors $M_1 \cup M_2$ est régulier.

Suggestion. Je recommande de décrire un automate dont le langage est $M_1 \cup M_2$. Vous devez argumenter que le langage de votre automate est bel et bien le bon.

Solution.
Sachant que M_1 est régulier, il existe un AFD A_1 tel que $L(A_1) = M_1$ (on peut supposer que A_1 est un AFD, car si c'est un AFND il suffit de le déterminiser). De la même façon, il y a un AFD A_2 tel que $L(A_2) = M_2$. Soit q_1 l'état de départ de A_1, et q_2 l'état de départ de A_2.

On peut créer un AFND A pour $M_1 \cup M_2$ comme suit. On ajoute tous les états et transitions de A_1, puis tous les états et transitions de A_2. Les états finaux sont ceux de A_1 et ceux de A_2. On créé un nouvel état de départ q_0. Ensuite, on ajoute une λ-transition de q_0 vers q_1, puis une λ-transition de q_0 vers q_2. Ceci conclut la construction de A.

On argumente que le langage de A est $M_1 \cup M_2$. Il faut argumenter qu'un mot w est accepté s'il est dans $M_1 \cup M_2$, et rejeté sinon. Soit $w \in M_1 \cup M_2$. Si $w \in M_1$, alors A peut aller à q_1 et se faire accepter par A_1. Si $w \in M_2$, alors A peut aller à q_2 et se faire accepter par A_2. Donc tout mot $w \in M_1 \cup M_2$ est accepté.

Maintenant, soit $w \notin M_1 \cup M_2$. Notre AFND A peut faire deux choix comme première transition sur entrée w. Si q_0 va vers q_1, alors w suivra son chemin sur A_1 et ne sera pas accepté car $w \notin M_1$. Si q_0 va vers q_2, alors w suivra son chemin et ne sera pas accepté car $w \notin M_2$. Donc w n'est pas accepté. On conclut que $L(A) = M_1 \cup M_2$.

□

d. Soit $M_1, M_2 \subseteq \Sigma^*$ deux ensembles de mots réguliers. Donnez un AFD ou un AFND dont le langage est $M_1 \setminus M_2$.

Il n'est pas exigé d'argumenter que votre automate est correct - il vous suffit de décrire votre automate. Je recommande de vous inspirer des exercices, où on montre que $M_1 \cap M_2$ est régulier.

Solution.
Soit A_1 et A_2 des AFD pour M_1 et M_2, respectivement.

Soit A un nouvel AFD défini comme suit:

- les états de A sont $Q_1 \times Q_2$, où Q_1 et Q_2 sont les états de A_1 et A_2, respectivement. L'idée est que être dans l'état $(p, q) \in Q_1 \times Q_2$ correspond à "être dans p sur A_1 et être dans q sur A_2";
- l'état initial est (q_{01}, q_{02}), où q_{01} et q_{02} sont les états initiaux de A_1 et A_2, respectivement;
- On met une transition de (p, q) vers (r, s) sur symbole a si et seulement si A_1 contient (p, a, r) et A_2 contient (q, a, s). L'idée est d'imiter le comprtement sur a de A_1 lorsqu'il est dans p et de A_2 lorsqu'il est dans q;
- On met (p, q) comme état acceptant de A si et seulement si p est acceptant sur A_1 et q n'est **pas** acceptant sur A_2.

Ceci serait suffisant comme réponse.

Pour argumenter un peu, on peut voir que si A se retrouve dans l'état (p, q) après avoir lu des symboles, alors A_1 se retrouverait dans p après avoir lu ces mêmes symboles, et A_2 serait dans q.

Si un mot w est dans $M_1 \setminus M_2$, il est accepté par A_1 mais pas par A_2. ALors sur entrée w, A se retrouvera dans un état (p, q) où p accepte et pas q. Dans ce cas, par notre construction A accepte correctement w.

Si w n'est pas dans $M_1 \setminus M_2$, alors soit il est rejeté par A_1 ou bien il est accepté par A_2. Dans le premier cas, A rejette car il termine en (p, q) où p rejette, et dans le deuxième cas, A rejette car il termine dans un état (p, q) où q accepte.

□