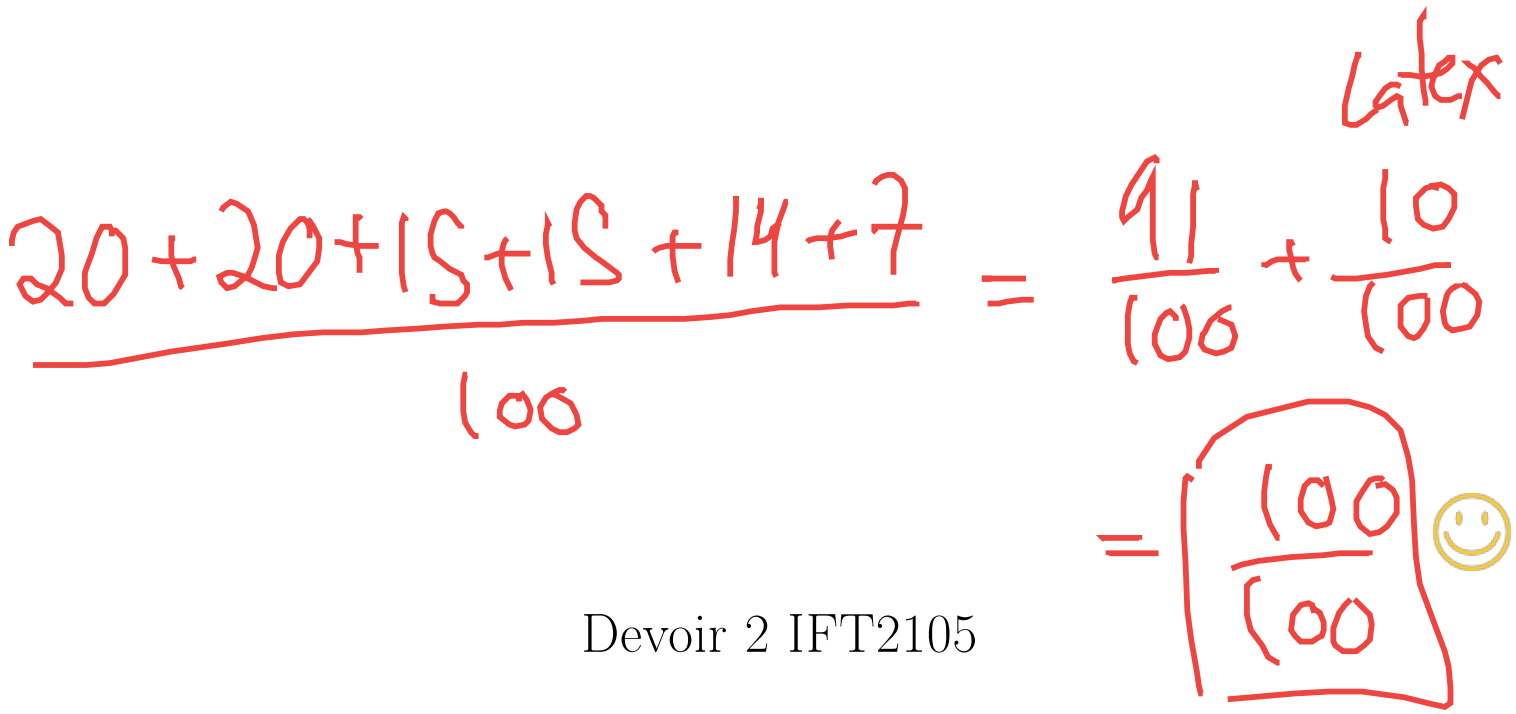


Devoir 2 IFT2105

Catherine Larivière 0955948

Dominique Vigeant 20129080

21 juin 2021

Très bien 15/15 → 20/20

1. Donnez une machine de Turing qui simule l'instruction $r_0 \leftarrow r_1$ en supposant que la tête de lecture/écriture est initialement sur le premier bit de $\langle r_0 \rangle$. À la fin du calcul, la tête de lecture/écriture doit retourner sur le premier bit de $\langle r_0 \rangle$, qui encode maintenant la valeur r_1. Votre machine doit laisser inchangées les valeurs de tous les autres registres. Elle devrait donc produire la chaîne suivante sur le ruban :

$$\ldots \sqcup \sqcup \langle r_1 \rangle \# \langle r_1 \rangle \# \langle r_2 \rangle \# \ldots \# \langle r_n \rangle \sqcup \sqcup \ldots$$

avec la tête de lecture/écriture positionnée sur le premier bit à gauche. Donnez votre machine de Turing sous la forme d'un graphe tel que vu en démo. Supposez que $\Sigma = \{0, 1, \#\}$ et donnez explicitement l'alphabet de ruban Γ dont vous avez besoin pour le fonctionnement de votre machine.

Description de chaque état ("_" représente un "$\sqcup$") :

— q0 : État initial. Le but est d'aller rejoindre le # qui marque la fin de r_0.

— q1 : On va par la droite jusqu'à ce qu'on trouve un X ou un Y, ou bien le # marquant la fin de r_1.

— q2 : Si on se retrouve sur un 0, on marque un X pour dire qu'on a lu un 0. Si on se retrouve sur un 1, on marque un Y pour dire qu'on a lu un 1. Si on se retrouve sur un X ou un Y, on continue d'aller à gauche jusqu'à ce qu'on trouve un 0, 1, ou #. Si on se retrouve sur un #, cela veut dire qu'on a fini de lire r_1.

— q3 : Dans r_1, on a lu un 0. On transporte X à gauche jusqu'au # marquant la fin de r_0.

— q4 : Dans r_1, on a lu un 1. On transporte Y à gauche jusqu'au # marquant la fin de r_0.

— q5 : Tant qu'on lit un X ou un Y, on continue d'aller à gauche jusqu'à ce qu'on trouve un 0, 1, ou _. Quand on a trouvé, on remplace le symbole par X pour dire que dans r_1 on a lu un 0.

— q6 : Tant qu'on lit un X ou un Y, on continue d'aller à gauche jusqu'à ce qu'on trouve un 0, 1, ou _. Quand on a trouvé, on remplace le symbole par Y pour dire que dans r_1 on a lu un 1.

— q7 : On a fini de lire r_1, notre but est de retourner au début de r_0. Si r_0 était plus long que r_1, ça veut dire qu'il reste des 0/1 au début du registre. On les change en _ jusqu'à ce qu'on rencontre un _.

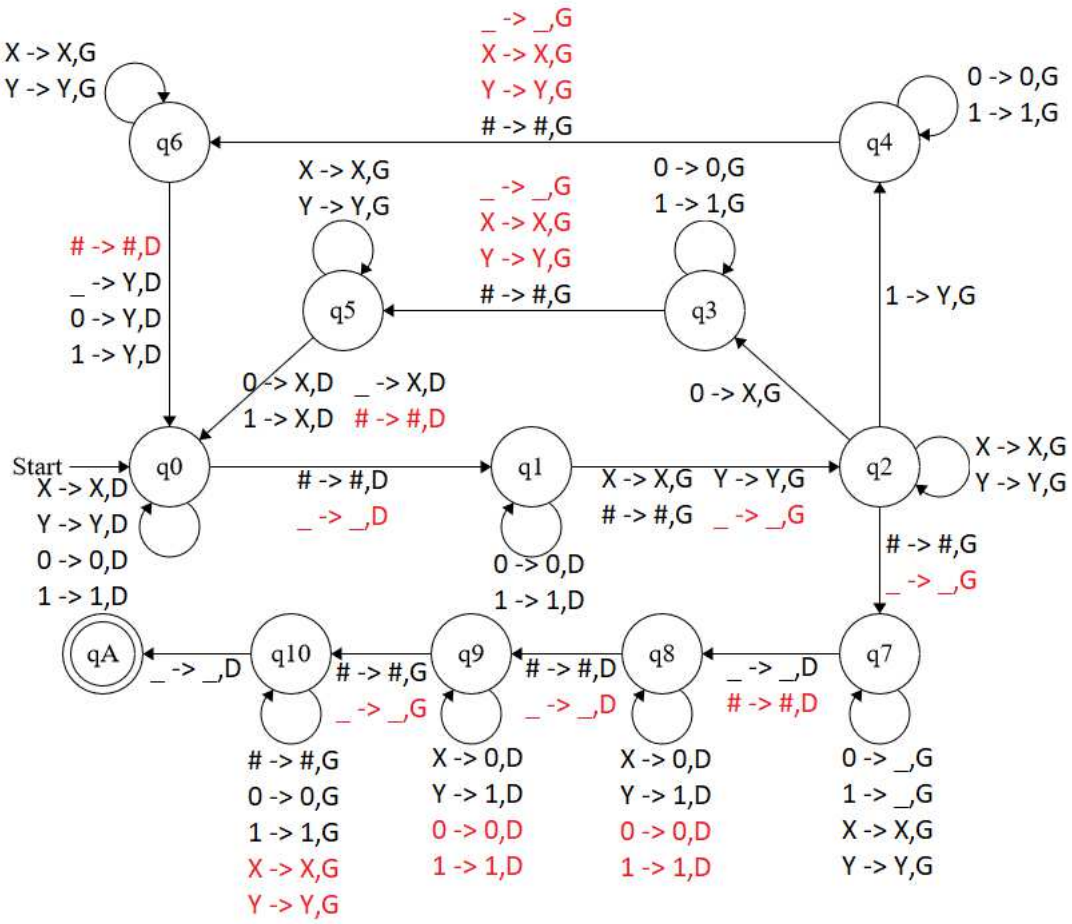
— q8 : On est revenu au début de r_0, maintenant notre but est de changer les X en 0 et les Y en 1 jusqu'au # marquant la fin de r_0.

— q9 : On est rendu au début de r_1, maintenant notre but est de changer les X en 0 et les Y en 1 jusqu'au # marquant la fin de r_1. On a besoin de deux états pour cela puisque sinon on ne saurait pas où arrêter notre "traduction".

— q10 : On est rendu à la fin de r_1 et tous les X/Y ont été reconvertis en 0/1. Il faut maintenant retourner au début de r_0, donc on va à gauche jusqu'à ce qu'on trouve un _ et on se déplace une fois à droite.

— q_A : On est revenu au premier bit de r_0 et on a fini l'instruction $r_0 \leftarrow r_1$.

$$\Gamma = \{1, 0, X, Y, \#, \_\}$$


X -> X,G
Y -> Y,G
q6
_ -> _,G
X -> X,G
Y -> Y,G
-> #,G
q4
0 -> 0,G
1 -> 1,G
X -> X,G
Y -> Y,G
q5
_ -> _,G
X -> X,G
Y -> Y,G
-> #,G
q3
0 -> 0,G
1 -> 1,G
1 -> Y,G
-> #,D
_ -> Y,D
0 -> Y,D
1 -> Y,D
0 -> X,D
1 -> X,D
_ -> X,D
-> #,D
0 -> X,G
Start
q0
X -> X,D
Y -> Y,D
0 -> 0,D
1 -> 1,D
-> #,D
_ -> _,D
q1
0 -> 0,D
1 -> 1,D
X -> X,G
-> #,G
Y -> Y,G
_ -> _,G
q2
X -> X,G
Y -> Y,G
-> #,G
_ -> _,G
qA
_ -> _,D
q10
-> #,G
0 -> 0,G
1 -> 1,G
X -> X,G
Y -> Y,G
-> #,G
_ -> _,G
q9
X -> 0,D
Y -> 1,D
0 -> 0,D
1 -> 1,D
-> #,D
_ -> _,D
q8
X -> 0,D
Y -> 1,D
0 -> 0,D
1 -> 1,D
_ -> _,D
-> #,D
q7
0 -> _,G
1 -> _,G
X -> X,G
Y -> Y,G

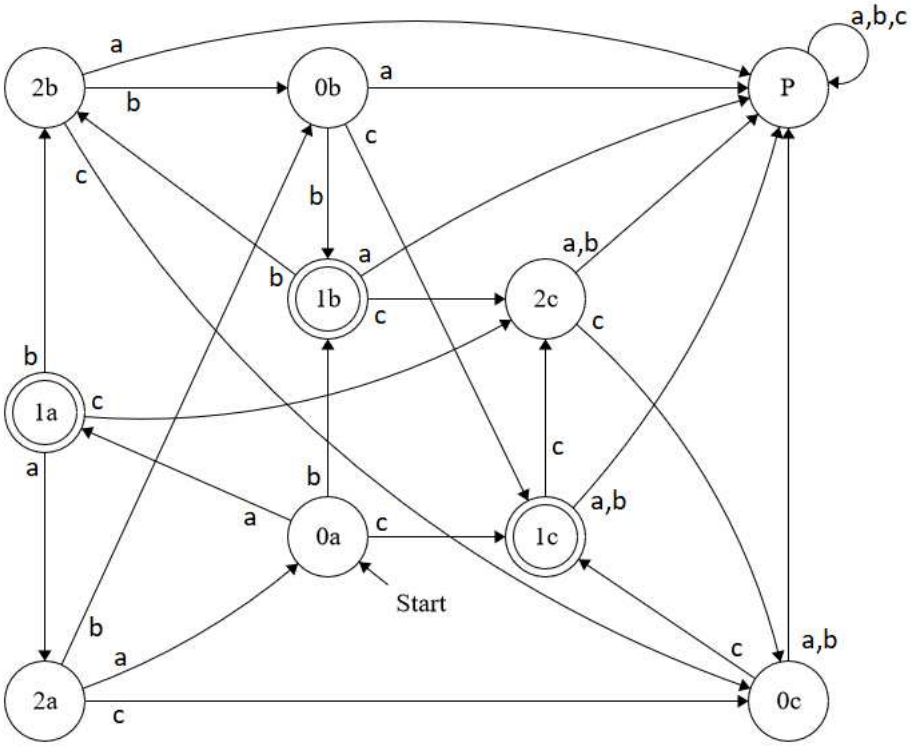
3. Soit L_1 et L_2 deux langages. Nous définissons la différence entre deux langages $L_1 - L_2$ comme $L_1 - L_2 = \{w | w \in L_1 \text{ mais } w \notin L_2\}$. Est-ce que la classe des langages réguliers est fermée pour la différence entre deux langages ? Prouvez votre réponse.

Selon les opérations sur la théorie des ensembles, on peut traduire la soustraction de deux ensembles par $L_1 - L_2 = L_1 \cap L_2^c$. Par le théorème 4.59, la classe REG est fermée pour l'opération complément, et par le théorème 4.60, la classe REG est fermée pour l'intersection. Donc la classe des langages réguliers est fermée pour la différence entre deux langages.

Très bien 15/15

4. Donnez un AFD ou un AFN qui reconnaît $L_4 = \{a^i b^j c^h \mid (i + j + h) \bmod 3 = 1\}$.
   Évidemment, l'alphabet du langage L_4 est $\Sigma = \{a, b, c\}$.

La difficulté ici est qu'on ne peut pas accepter de a après avoir mis un b et on ne peut pas accepter de a ni de b après avoir mis un c. Lorsque cela arrive, on se dirige vers l'état "Poubelle". Le principe est donc de tenir compte du modulo après l'ajout de chaque lettre et surveiller l'ordre a puis b puis c. Notons que i, j, h peuvent avoir 0 comme valeur.



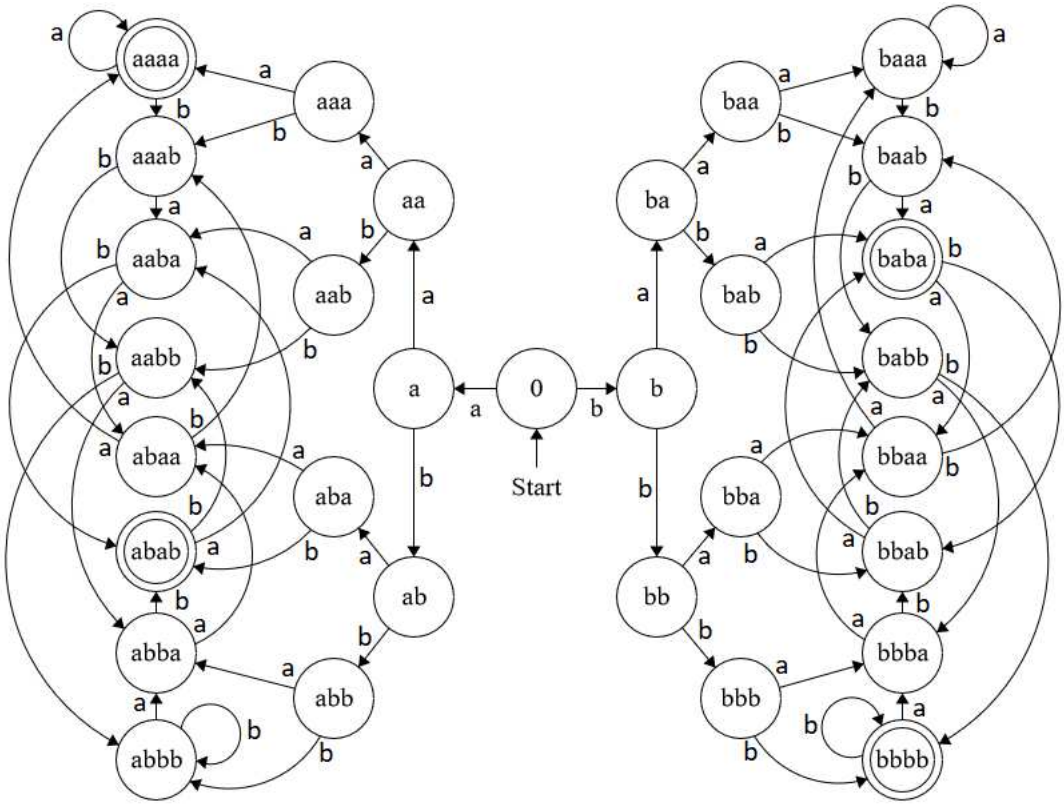
Très bien 15/15

5. Donnez un AFD ou un AFN qui reconnaît

$$L_5 = \{w_1 w_2 ... w_n w_1 w_n \mid n > 1, \text{ et } w_i \in \{a, b\} \text{ pour } 1 \leq i \leq n\}.$$

Évidemment, l'alphabet du langage L_5 est $\Sigma = \{a, b\}$.

On comprend que L_5 contient des mots où le premier symbole est égal à l'avant-dernier et l'avant-avant-dernier symbole est égal au dernier. Entre le premier et l'avant-avant-dernier symbole exclusivement, le symbole n'importe pas. Puisque $n > 1$, le plus petit mot de L_5 a la forme $w_1 w_2 w_1 w_2$ et on ne peut pas accepter de mot de taille < 4. L'important ici est de se "rappeler" du premier symbole, ce qui sépare l'AFD en deux parties. À partir de 4 symboles, si on en ajoute un de plus on peut oublier le deuxième symbole et se concentrer sur les trois derniers symboles. Par exemple, si j'entre les symboles "a", "a", "b", "a" on se retrouve dans l'état *aaba*. Si j'entre ensuite "b", on oublie le deuxième "a" et on s'en va dans l'état *abab*, qui est un état acceptant.



Très bien 14/15

7. Soit $\Sigma = \{0, 1, \#\}$ et dénotons la valeur de $x \in \{0, 1\}^*$ écrite en binaire par $\text{val}(x)$. Par exemple, $\text{val}(101) = 5$ et $\text{val}(00111) = 7$. Soit

$$L_7 = \{x\#y \mid x, y \in \{0, 1\}^* \text{ avec } |x|, |y| > 0 \text{ et } \text{val}(y) = \text{val}(x) + 1\}.$$

Montrez que L_7 n'est pas un langage régulier.

Preuve par contradiction. Soit $p \geq 1$
Considérons le mot $w = 1^p\#10^p$, ce qui crée comme mot $1\#10$, $11\#100$, $111\#1000$, etc. On obtient la décomposition

$$\begin{aligned} x &= 1^k \\ y &= 1^{n-k} \\ z &= 1^{p-n}\#10^p \quad \text{où } k \geq 0 \text{ et } p \geq n \geq 1 \end{aligned}$$

et $k < n$

Nous avons bien $w \in L$ et $|w| = 2p + 2 \geq p$. On obtient clairement que y ne contient que des 1. Si on pompe y, on ajoute des 1 dans notre terme de gauche, ce qui brise notre condition $\text{val}(y) = \text{val}(x) + 1$, car $\text{val}(y) \leq \text{val}(x)$.
Et donc $x \cdot y^2 \cdot z \notin L$, une contradiction avec l'énoncé. Donc $L_7 \notin \text{REG}$.

Ceci aurait pu être explicité

8. Montrer que $L_8 = \{w \in \{a,b\}^* \mid w = \omega \cdot a \cdot \omega' \text{ avec } |\omega| = |\omega'|\}$ n'est pas régulier.

Preuve par contradiction. Soit $p \geq 1$
Considérons le mot $w = b^p a b^p$. Nous avons bien $w \in L$ et $|w| = 2p+1 \geq p$. On décompose pour obtenir :

$$\begin{cases} x = b^{p-1} \\ y = b \\ z = ab^p \end{cases}$$

x et y contiennent seulement des b. Si on pompe y, on débalance la condition $|\omega| = |\omega'|$. Donc $x \cdot y^2 \cdot z \notin L$, une contradiction avec l'énoncé. Ainsi $L_8 \notin$ REG.

Ceci est une décomposition, il faut montrer pour toutes les décompositions