Licence Sciences et Technologies
Université Joseph Fourier

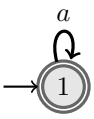
Examen final
INF 232: Langages et Automates
L2, 2013/2014

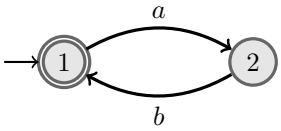
Rappel des consignes et quelques conseils/remarques

- Durée : 2 heures (09h00 → 11h00).
- Pas de sortie avant 30 minutes. Pas d'entrée après 30 minutes.
- Tout document du cours ou du TD est autorisé. Tout autre document est interdit.
- Tout dispositif électronique est interdit (calculette ; téléphone, tablette, etc.).
- **Le soin de votre copie sera pris en compte (-1 point si copie non soignée).**
- Les exercices sont indépendants.
- Le barème est donné à titre indicatif.

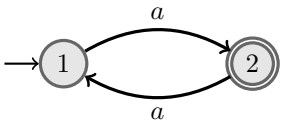
Solution de l'exercice 1

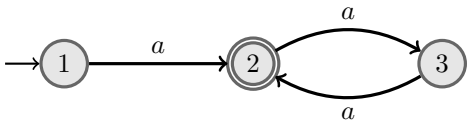
1. Vrai. C'est le cas par exemple d'un automate déterministe et complet qui n'a pas d'état accepteur.
2. Faux. La fermeture de Kleene d'un langage L est l'ensemble des mots qui peuvent se définir comme une concaténation finie de mots de L. Pour un langage L tel que $w \in L$ mais $w \cdot w \notin L$, on a $w \cdot w \in L^*$.
   Autre contre-exemple : n'importe quel langage tel que $\epsilon \notin L$ avec $\epsilon \in L^*$.
3. Faux. Par exemple, la fermeture de Kleene du langage dénoté par l'expression régulière a^* est égale à ce langage.
4. Faux. Les automates ne font pas forcément référence à un langage unique. Par exemple, les deux automates suivants sont minimaux et reconnaissent les langages a^* et $(a \cdot b)^*$.





5. Faux. Un automate non minimal et son minimisé sont équivalents mais n'ont pas le même nombre d'états. Par exemple, nous pouvons considérer les deux automates suivants :





6. Vrai. Les automates déterministes ont le même pouvoir expressif que les expressions régulières (de part l'équivalence entre expressions régulières et automates non-déterministes avec ϵ-transitions et entre automates non-déterministes avec ϵ-transitions et automates déterministes). De plus, tout automate déterministe admet un automate minimal, c'est-à-dire un automate qui reconnaît le même langage et qui est tel que si on enlève un état, alors l'automate reconnaît un langage différent. Il suffit donc d'utiliser les traductions entre les formalismes.
7. Faux. La correction partielle considère (et utilise) uniquement les exécutions qui terminent. Elle ne permet pas de déduire la terminaison du programme (qui fait généralement l'objet d'une preuve séparée).
8. Vrai. La propriété dit que pour chaque entier relatif, on peut trouver un entier naturel tel que la somme des deux entiers est strictement positive. En effet, soit $x \in \mathbb{Z}$, prenons $y = 1 - x$, nous avons alors $x + y = 1$.

Solution de l'exercice 2

1. L'algorithme est implémenté par la fonction check_determinisme(). L'algorithme prend en entrée un

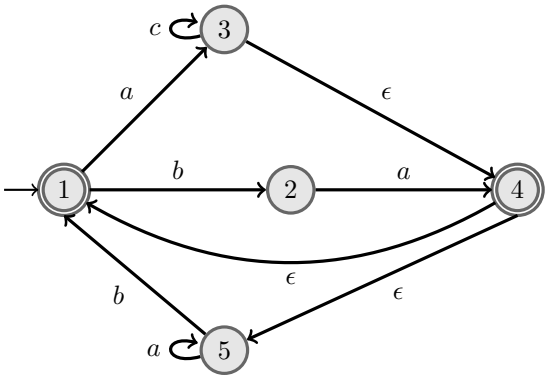


FIGURE 1 – Automate pour l'expression régulière $\big((a \cdot c^* + b \cdot a) \cdot (a^* \cdot b + \epsilon)\big)^*$

automate $(Q, q_{init}, \Sigma, \delta, F)$. (La solution n'a pas vocation à être efficace mais simple).

```
1 deterministic := true ;
2 for each q in Q do
3 |   for each σ in Σ do
4 |   |   deterministic := deterministic ∧ |{(q, σ, q') ∈ δ | q' ∈ Q}| ≤ 1;
5 return updated
```

Algorithm 1: fonction check_determinisme()

2. Soient $\mathcal{A} = (Q, q_{init}, \Sigma, \delta, F)$ et $\mathcal{A}' = (Q', q'_{init}, \Sigma', \delta', F')$ les deux automates. La solution proposée est donnée dans l'Algorithme 3 et utilise la fonction intermédiaire fermeture_Kleene() donnée dans l'Algorithme 2. Nous utilisons la fonction sont_equivalents() qui indique si les automates donnés en entrée sont équivalents (vue dans le cours sur la minimisation).

```
1 Q^K = Q ∪ {q^K_init};
2 δ^K := δ ∪ {(q^K_init, ϵ, q_init)} ∪ {(f, ϵ, q_init) | f ∈ F};
3 return (Q_K, q^K_init, δ^K, F ∪ {q^K_init})
```

Algorithm 2: fonction fermeture_Kleene()

```
1 kleene := kleene_fermeture(A);
2 return sont_equivalents(A, kleene)
```

Algorithm 3: Algorithme pour la question 2

Solution de l'exercice 3

1. Observons tout d'abord que l'expression régulière proposée peut se simplifier en $\big((a \cdot c^* + b \cdot a) \cdot (a^* \cdot b + \epsilon)\big)^*$. Nous construisons l'automate de la Figure 1 pour cette dernière expression.
2. Supprimons les ϵ-transitions, nous obtenons l'automate représenté sur la Figure 2 où les transitions en pointillés bleus sont celles ajoutées.
   Appliquons l'algorithme de déterminisation sur l'automate obtenu.

	1∗	1345∗	2	12∗	145∗
a	1345	1345	145	1345	1345
b	2	12		2	12
c		1345			

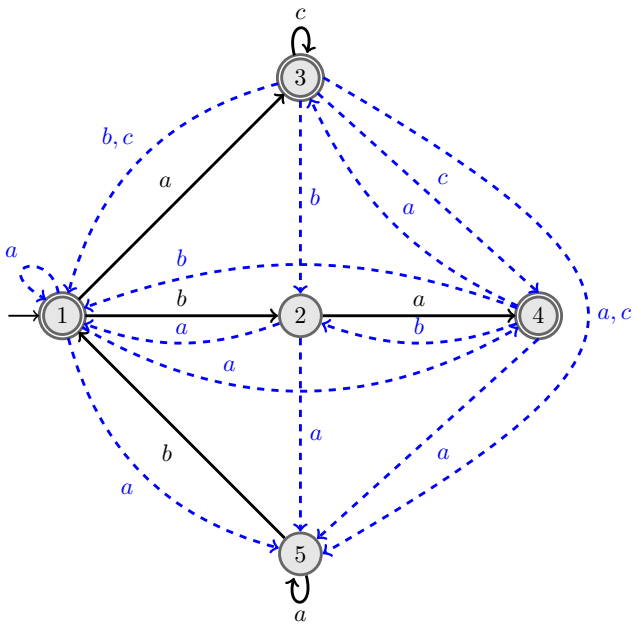


FIGURE 2 – Suppression des ϵ-transitions de l'automate de la Figure 1

Renommons les états, nous obtenons :

	1∗	2∗	3	4∗	5∗
a	2	2	5	2	2
b	3	4		3	4
c		2			

3. Appliquons l'algorithme de minimisation.

$\equiv_0$	$\equiv_1$	$\equiv_2$
1	1	1
2	4	4
4	2	2
5	5	5
3	3	3

La classe d'équivalence $\{1, 4\}$ peut être réduite à un état. Nous obtenons :

	1∗	2∗	3	5∗
a	2	2	5	2
b	3	1		1
c		2		

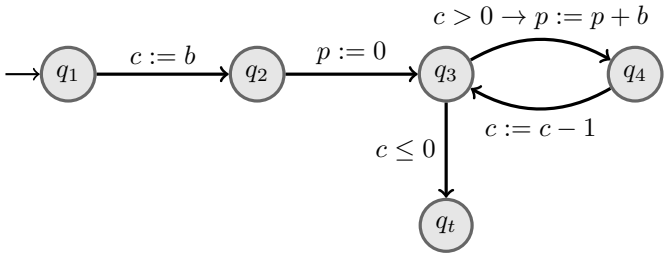
Solution de l'exercice 4

Dans la suite a^i dénote la concaténation de i a's.

1. $a^2 \cdot b^4 \in L$ et $a^3 \cdot b^7 \notin L$.
2. Supposons L régulier. Soit N la constante fournie par le lemme de l'itération. Considérons le mot $w = a^N \cdot b^N$. Nous avons $w \in L$ et $|w| \geq N$. Par le lemme de l'itération, ils existent $x, y, z \in \{a, b\}^*$ tels que :
   – $w = x \cdot y \cdot z$,
   – $|xy| \leq N$
   – $y \neq \epsilon$
   – $\forall k \in \mathbb{N} : x \cdot y^k \cdot z \in L$.
   Comme $|xy| \leq N$, y s'écrit $y = a^i$ avec $i \in \mathbb{N}$. De plus, $y \neq \epsilon$ nous donne $i \neq 0$. (Nous avons $x = N - i - j$ avec $j \geq 0$). Considérons le mot $w' = x \cdot y^{N+1} \cdot z = a^{N-i-j} \cdot a^{(N+1)*i} \cdot b^N = a^{(N+1)*i} \cdot b^N$. D'une part, en utilisant $\forall k \in \mathbb{N} : x \cdot y^k \cdot z \in L$, nous avons $w' \in L$. Mais d'autre part, le fait que $(N+1)*i$ ne divise pas N nous donne $w \notin L$. Ceci est donc une contradiction qui conclut la preuve.

Solution de l'exercice 5

1. Le programme peut être défini par le 5-tuple $(\mathbb{D}, Q, q_1, \mathcal{T}, Q_t)$ tel que :
   – $\mathbb{D} = \{b : \mathbb{Z}, c : \mathbb{Z}, p : \mathbb{Z}\}$,
   – $Q = \{q_1, q_2, q_3, q_4, q_t\}$,
   – $\mathcal{T} = \{(q_1, c := b, q_2), (q_2, p := 0, q_3), (q_3, c > 0 \to p := p + b, q_4), (q_4, c := c - 1, q_3), (q_3, c \leq 0, q_t)\}$,
   – $Q_t = \{q_t\}$.
2. L'automate peut être décrit graphiquement comme suit :



3. Dans cette question, nous reportons la valeur d'une variable uniquement dans l'état initial et lorsque sa valeur change.

   a) L'exécution démarrant dans l'état $[b \mapsto 3, c \mapsto 0, p \mapsto 42]$ est comme suit :

état	b	c	p
q_1	3	0	42
q_2		3	
q_3			0
q_4			3
q_3		2	
q_4			6
q_3		1	
q_4			9
q_3		0	
q_t		0	

   b) L'exécution démarrant dans l'état $[b \mapsto -2, c \mapsto 0, p \mapsto 42]$ est comme suit :

état	b	c	p
q_1	-2	0	42
q_2		-2	
q_3			0
q_t			

   c) L'exécution démarrant dans l'état $[b \mapsto 0, c \mapsto 0, p \mapsto 42]$ est comme suit :

état	b	c	p
q_1	0	0	42
q_2		0	
q_3			0
q_t			

4. Nous proposons d'utiliser les prédicats d'états suivants :
   – $P_{q_1} = b \geq 0$,
   – $P_{q_2} = c \geq 0 \wedge b = c$,
   – $P_{q_3} = c \geq 0 \wedge p = b * (b - c)$,
   – $P_{q_4} = c > 0 \wedge p - b = b * (b - c)$,
   – $P_{q_t} = p = b^2$.