Série 1 – Preuves de correction partielle de programmes

Pour tous les programmes suivants :

1. donnez l'automate de Von Neumann correspondant,
2. donnez la propriété de correction du programme
3. démontrez que le résultat satisfait la propriété annoncée en utilisant la technique de preuve de Floyd-Dijsktra-Hoare,
4. en déduire les conditions d'utilisation du programme,

Exercice 1.1 Multiplication avec l'addition

```
x:=A ; y:=B ; r:=0 ;
while(y>0){ r:=r+x; y:=y-1; }
```

Donnez la propriété de correction qui exprime qu'à la sortie de ce programme `r` contient la valeur de $A \times B$.

Exercice 1.2 Somme des N premiers entiers

```
s:=0 ; r:=0 ;
while(r!=N+1){ s:=s+r; r:=r+1; }
```

Donnez la propriété de correction qui exprime qu'à la sortie de ce programme `s` contient la valeur de Σ_1^N

Exercice 1.3 Somme des N premiers entiers

```
s:=0 ; r:=N ;
while(r!=0){ s:=s+r; r:=r-1; }
```

Donnez la propriété de correction qui exprime qu'à la sortie de ce programme `s` contient la valeur de Σ_1^N

Exercice 1.4 Multiplication rapide

```
x:=A ; y:=B ; r:=0 ;
while(y>0){
  if (y%2=0){ x:=2*x; y:=y/2; }
  else { r:=r+x; y:=y-1; }
}
```

Donnez la propriété de correction qui exprime qu'à la sortie de ce programme `r` contient la valeur de $A \times B$.

Exercice 1.5 Calcul de la racine carrée sans multiplication

```
n=1; s=1;
while(2*s-n<X){ n=n+1; s=s+n; }
```

On prétend qu'à la sortie du programme l'entier n vérifie :

$$n - 1 < \sqrt{X} \leq n$$

Indication L'idée cachée derrière cet algorithme est la suivante

Soit n la partie entière de la racine carré de X, c'est-à-dire $X = n^2 + r$ avec $0 \leq r < 1$

L'algorithme exploite la formule suivante pour approcher X par la somme des n premiers nombres :

$s = \Sigma_{i=1}^{i=n} i = \frac{n(n+1)}{2}$
$2s = n(n+1) = n^2 + n$
$2s - n = n^2$

Exercice 1.6 Exponentiation avec la multiplication

```
x:=A ; y:=N ; r:=1 ;
while(y>0){ r:=r*x; y:=y-1; }
```

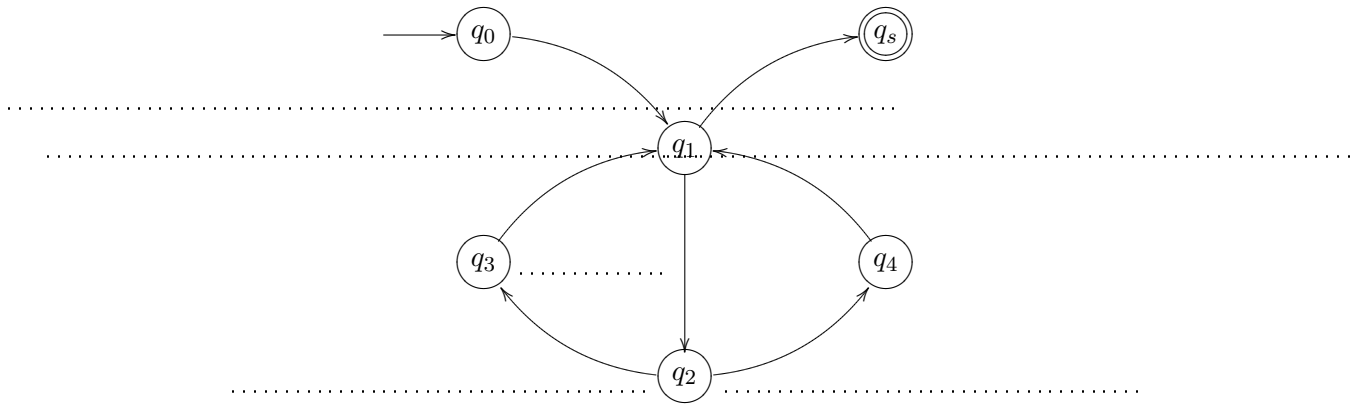
Donnez la propriété de correction qui exprime qu'à la sortie de ce programme `r` contient la valeur de A^N.

Exercice 1.7 (6 pt) Preuve de correction partielle de l'algorithme d'exponentiation rapide

Algorithme

```
x:=A ; y:=N ; r:=1 ;
while(y>0){
  if (y%2=0){ x:=x*x; y:=y/2; }
  else { r:=r*x; y:=y-1; }
}
```

Q1. Donnez les transitions de l'automate pour qu'il corresponde au programme précédent.



Q2. Complétez l'exécution de l'automate en adoptant la présentation ci-dessous.

a) Pour les valeurs $A = 3,\ N = 3$

état	q_0	q_1	q_2	.....	.....	.....	.....	.....	.....	.....	.....	.....
x	..	3	...	...	...	...	...	...	...	...	...	...
y	..	3	...	...	...	...	...	...	...	...	...	...
r	..	...	...	...	...	...	...	...	...	.....	.....	.....

b) Même question pour les valeurs $A = 2,\ N = 0$

Q3. Donnez la propriété de correction qui exprime qu'à la sortie de l'automate la variable `r` contient la valeur de A^N.

Solution (exemple de rédaction) $\psi_s \stackrel{def}{=} r = \ldots\ldots\ldots \wedge \ .. \ = 0$

Q4. (3 pt) **Preuve de correction partielle (soignez la rédaction !)** Donnez et démontrez l'implication associée à chaque transition et donnez les 5 invariants $(\psi_s, \psi_1, \psi_3, \psi_4, \psi_2)$ associés aux états q_s, q_1, q_3, q_4, q_2.

Indication $\psi_1 \stackrel{def}{=} r \ldots\ldots\ldots = A^N \wedge \ .. \ \geq \ ..$

Q5. En déduire les conditions d'utilisations du programme.

Exercice 1.8 Calcul du milieu sans division

```
x=A; y=B;
while(x<y){ x=x+1; y=y-1; }
```

On prétend qu'à la sortie du programme le résultat (x, y) satisfait la propriété suivante : le milieu de A et B est compris entre $y - 1$ et $x + 1$ ce qu'on traduit par la formule

$$y \leq A + B \leq x \wedge 0 \leq x - y \leq 1$$

Q6. Donnez les transitions de l'automate correspondant au programme ci-dessus (q_0 représente le point d'entrée et q_s celui de sortie de l'automate).

$q_0 \xrightarrow{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} q_1$

$q_1 \xrightarrow{\ldots\ldots\ldots\ldots} q_s$

$q_1 \xrightarrow{\ldots\ldots\ldots\ldots} q_2$

$q_2 \xrightarrow{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} q_1$

Recherche d'invariants Faîtes quelques exécutions pour des valeurs de A et B afin d'en déduire un invariant sur $x + y$. Il vous sera utile pour la preuve.

Q7. Que vaut $x + y$ au cours du programme ? $x + y = \ldots\ldots\ldots\ldots\ldots\ldots$

Q8. Preuve de correction partielle Rédigez la preuve (en suivant la méthode de Floyd-Dijsktra-Hoare) qu'à la sortie du programme le résultat (x, y) satisfait la propriété

$$2(y-1) \leq A + B \leq 2(x+1)$$

Q9. En déduire les conditions d'utilisation du programme.

Exercice 1.9 Calcul des termes de la suite de Fibonacci

```
a=1; b=0; i=0; r=1;
while(i!=N){ r=a+b; b=a; a=r; i=i+1; }
```

On prétend qu'à la sortie du programme le résultat r satisfait la propriété $r = fib(N)$ où

$$\begin{cases} fib(-1) = \ldots \textit{(on peut choisir la valeur librement puisque la suite n'est pas définie pour -1)} \\ fib(0) = 1 \\ fib(1) = 1 \\ fib(i) = fib(i-1) + fib(i-2) \end{cases}$$

Q10. Donnez les transitions de l'automate correspondant au programme ci-dessus (q_0 représente le point d'entrée et q_s celui de sortie de l'automate).

$$q_0 \xrightarrow{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} q_1$$

$$q_1 \xrightarrow{\ldots\ldots\ldots\ldots\ldots} q_s$$

$$q_1 \xrightarrow{\ldots\ldots\ldots\ldots\ldots} q_2$$

$$q_2 \xrightarrow{\ldots := \ldots\ldots\ldots\ldots} q_3$$

$$q_3 \xrightarrow{\ldots := \ldots} q_4$$

$$q_4 \xrightarrow{\ldots\ldots\ldots\ldots\ldots ; \ldots\ldots\ldots\ldots\ldots\ldots\ldots} q_1$$

Q11. Preuve de correction partielle Rédigez la preuve (en suivant la méthode de Floyd-Dijsktra-Hoare) qu'à la sortie du programme $r = fib(N)$. Vous prendrez pour invariant en q_1 une propriété de la forme : $r = \ldots\ldots\ldots\ldots \wedge a = \ldots\ldots\ldots\ldots \wedge b = \ldots\ldots\ldots\ldots\ldots\ldots$

Q12. En déduire les conditions d'utilisation du programme.

Exercice 1.10 Calcul du pgcd de deux entiers naturels

```
 a=A; b=B;
 while(a!=b){
  if (a>b)
  then a:=a-b;
  else b:=b-a;
}
```

On prétend qu'à la sortie du programme le résultat r satisfait la propriété $a = b = pgcd(A, B)$

Q13. Donnez les transitions de l'automate correspondant au programme ci-dessus (q_0 représente le point d'entrée et q_s celui de sortie de l'automate).

Propriétés du pgcd On note $x|y$ le fait que x divise y, c'est-à-dire que y est un multiple de x, soit encore $\exists k \in \mathbb{N}^*, y = k.x$ et on rappelle les propriétés du $pgcd$ nécessaires pour prouver la correction de ce programme.

1) $pgcd(x, x) = x$
2) $p|a \wedge p|b \Longrightarrow p|pgcd(a, b)$
3) $p|a \wedge p|b \Longrightarrow p|a - b$
3′) $p|a \wedge p|b \Longrightarrow p|a + b$
4) $p|q \wedge q|p \Longrightarrow q = p$

Ces propriétés permettent de démontrer que $pgcd(a - b, b) = pgcd(a, b)$ et $pgcd(a, b - a) = pgcd(a, b)$.

Q14. Complétez la preuve ci-dessous.

Solution (exemple de rédaction) *Soit* $p \stackrel{def}{=} pgcd(a, b)$ *et* $q \stackrel{def}{=} pgcd(a - b, b)$.

Puisque $p = pgcd(a, b)$ *alors* $p|a$ *et* ..... *et donc* $p|$..... *(d'après ..) mais alors* $p|pgcd($....., ..*)*

(d'après ..) et donc $p|$ .. *(par définition de ..).*

Puisque $q \stackrel{def}{=} pgcd(a - b, b)$ *alors* $q|$..... *et* $q|b$ *et donc* $q|a$ *(d'après ....) mais alors* $q|pgcd($ ..... *) (d'après ..) et donc* $q|$ .. *(par définition de ..).*

Conclusion : $q = p$ *(d'après ..), autrement dit* $pgcd($ ..... $) = pgcd($ ..... $)$. *On démontre de la même manière que* $pgcd(a, b - a) = pgcd(a, b)$.

Q15. Preuve de correction partielle Rédigez la preuve (en suivant la méthode de Floyd-Dijsktra-Hoare) qu'à la sortie du programme $a = pgcd(A, B)$. Vous prendrez pour invariant en q_1 une propriété de la forme : $pgcd(\ldots, ..) = pgcd(A, B)$

Q16. En déduire les conditions d'utilisation du programme.

Série 2 – Exécution, produit, complémentaire d'automates déterministes

Exercice 2.1 Automates reconnaisseurs – complémentaire, produit, accessibilité, minimisation

On considère l'alphabet $\Sigma = \{a, b\}$.

Q17. Donnez un automate A qui reconnaît les mots de la forme $?^*aba$, c'est-à-dire les mots qui se terminent par aba.

Solution (exemple de rédaction)

catégories	ω	ωa	.....	.....
A	0	..	..	3
a	..	..	..	..
b	..	..	..	..

Q18. Donnez un automate B qui reconnaît les mots qui ne commencent pas par aba, c'est-à-dire les mots qui ne sont pas de la forme $aba?^*$.

Solution (exemple de rédaction)

catégories	ϵ	a	.....	$aba\omega$	*échec*
A	$0'$	$1'$	$2'$	$3'$	$4'$
a	....	....	....	....	....
b	....	....	....	....	....

Q19. À partir des automates A et B construire un automate qui reconnaît le langage $L_c \stackrel{def}{=} \mathscr{L}(\Sigma^* aba) \cap \overline{\mathscr{L}(aba\Sigma^*)}$

Complétez la table de transitions ci-après.

Solution (exemple de rédaction)

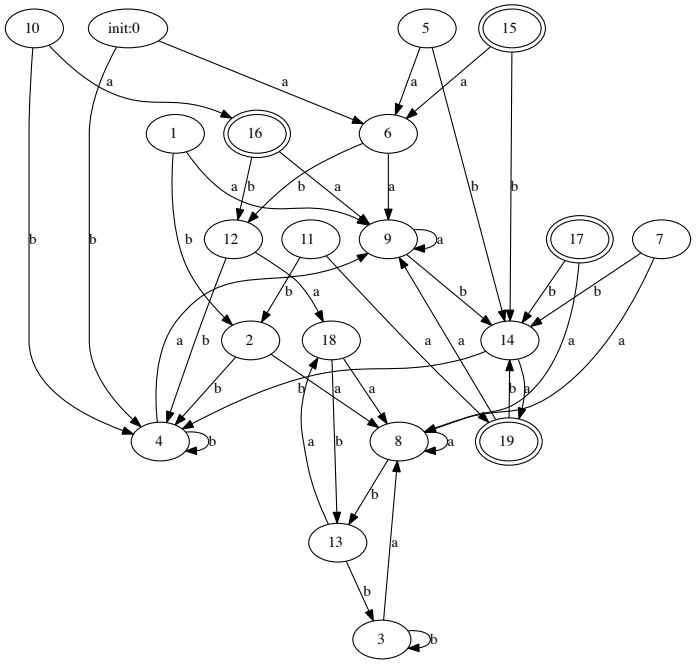
	$(0,0')$	$(0,1')$	$(0,2')$	$(0,3')$	$(0,4')$
$A \times \overline{B}$	0	1	2	3	4
a	....	(1,4′) 9	(1,3′) 8	....	....
b	....	(0,2′) 2	(0,4′) 4	....	....

	$(1,0')$	$(1,1')$	$(1,2')$	$(1,3')$	$(1,4')$
	5	6	7	8	9
a	$^{(1,1')}$ 6	. . . .	$^{(1,3')}$ 8	. . . .	. . . .
b	$^{(2,4')}$ 14	. . . .	$^{(2,4')}$ 14	. . . .	. . . .

	$(2,0')$	$(2,1')$	$(2,2')$	$(2,3')$	$(2,4')$
	10	11	12	13	14
a	$^{(3,1')}$ 16	$^{(3,4')}$ 19	. . . .	. . . .	. . . .
b	$^{(0,4')}$ 4	$^{(0,2')}$ 2	. . . .	. . . .	. . . .

	$(3^a,0'^a)$	$(3^a,1'^a)$	$(3^a,2'^a)$	$(3^a,3')$	$(3^a,4'^a)$
	15	16	17	18	19
a	$^{(1,1')}$ 6	$^{(1,4')}$ 9	$^{(1,3')}$ 8	. . . .	. . . .
b	$^{(2,4')}$ 14	$^{(2,2')}$ 12	$^{(2,4')}$ 14	. . . .	. . . .

Q20. De nombreux états de cet automate ne sont pas accessibles depuis l'état initial. Il est inutile de les construire. Marquez sur la représentation graphique les états qui ne sont pas accessibles depuis l'état initial.



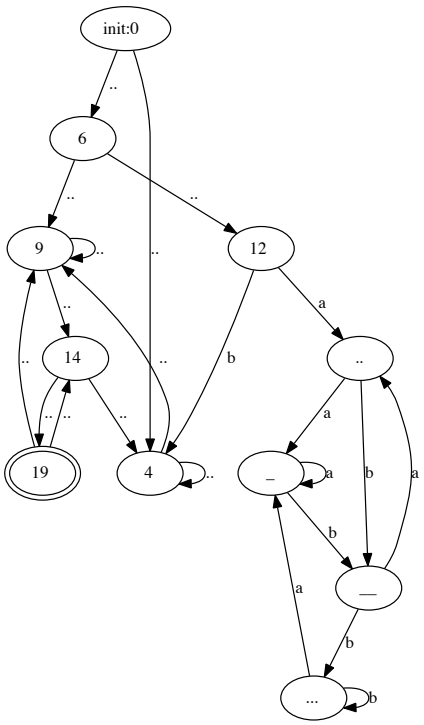
Q21. Pour éviter de construire ces états inaccessible, on part de l'état initial $(0, 0')$ et on ajoute uniquement dans la table les états que l'on atteint au cours de la construction. On obtient un automate à 10 états accessibles au lieu des 20 états précédents.

Solution (exemple de rédaction)

	$_i(0,0')$	$(1,1')$	$(0,4')$	$(1,4')$	$(2,2')$	$(2,4')$
$A \times \overline{B}$	$_i0$	6	4	9	12	14
a	. . . .	. . . .	. . . .	. . . .	. . . .	. . . .
b	. . . .	. . . .	. . . .	. . . .	. . . .	. . . .

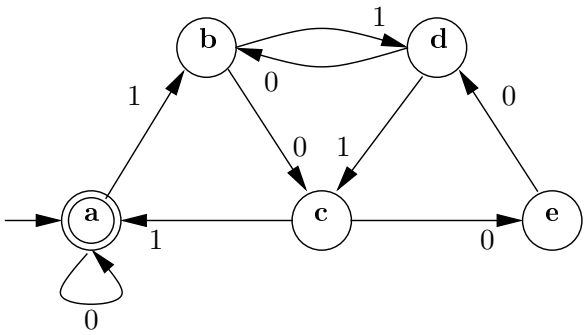
	$(3^a,3')$	$(3^a,4'^a)$	$(1,3')$	$(2,3')$	$(0,3')$
$A \times \overline{B}$	18	19^a	8	13	3
a	. . . .	. . . .	. . . .	. . . .	. . . .
b	. . . .	. . . .	. . . .	. . . .	. . . .

Q22. On obtient l'automate suivant (à compléter). On constate que cet automate n'est pas minimal. Justifiez cette affirmation et proposez un automate équivalent qui comporte moins d'état.



Exercice 2.2 Mots acceptés/rejetés par l'exécution d'un automate déterministe

On considère l'automate A_1 suivant représenté par ce **diagramme de transitions**



Q23. Indiquez :

- Les états de l'automate $Q = \{ \ldots\ldots\ldots\ldots\ldots\ldots \}$
- L'alphabet $\Sigma = \ldots\ldots\ldots\ldots$
- Les états initiaux $I = \{\ldots\}$
- Les états accepteurs $A = \ldots\ldots\ldots$
- Les transitions $\{(a, 0, a), (a, 1, b), \ldots\}$

Q24. Cet automate est-il complet ? Justifiez votre réponse en rappelant le critère qui permet de décider si un automate est complet.

Q25. Que donnent les exécutions de l'automate sur les mots suivants : $101, 00111100, \epsilon, 010001, 1101010$?
Lesquels de ces mots $\in \mathscr{L}(A_1)$?
Lesquels de ces mots $\notin \mathscr{L}(A_1)$?

Q26. Construisez l'automate complémentaire de A_1.

Q27. Lesquels des mots $101, 00111100, \epsilon, 010001, 1101010$ sont acceptés par $\overline{A_1}$?

Exercice 2.3 Complémentaire et non-déterminisme

On considère l'alphabet $\Sigma = \{a\}$ et l'automate non-déterministe suivant :

A	${}_i1$	2^a
a	$1, 2$	

Q28. Dessinez l'automate, décrivez le langage reconnu par A : $\mathscr{L}(A) = \ldots\ldots\ldots\ldots$ et son langage complémentaire $\overline{\mathscr{L}(A)} = \ldots\ldots\ldots$

Q29. Donnez un automate déterministe A^D équivalent à A, c'est-à-dire qui reconnaît le même langage que A.

Q30. Dessinez l'automate A' complémentaire de A, c'est-à-dire complétez l'automate A puis inversez les états accepteurs/non-accepteurs et décrivez le langage reconnu par $A' = \ldots\ldots\ldots\ldots$

Q31. Démontrez que la complétion puis l'inversion des états accepteurs/non-accepteurs d'un automate non-déterministe A ne suffit pas pour obtenir l'automate A^c qui reconnaît pas $\overline{\mathscr{L}(A)}$.

Q32. Construire l'automate complémentaire de A^D. Donnez le langage $\mathscr{L}(A^{DC}) = \{\,..\,\}$.

Conclusion : pour construire le complémentaire d'un automate non-déterministe A il faut

1. le c$\ldots\ldots\ldots\ldots\ldots\ldots$
2. le d$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$
3. i$\ldots\ldots\ldots\ldots$ les états $\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$/$\ldots\ldots\ldots$-$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$

Explication Considérons q_1 un état non accepteur de A et q_2^a un état accepteur de A et imaginons que $\{q_1, q_2\}$ soit une configuration accessible lors d'une exécution A.
Comparons ce qui se passe selon l'ordre des opérations : déterminisation et complémentaire.

1. $A^D := \textbf{\textit{déterminisation}}(A); A' := \textbf{\textit{complémentaire}}(A^D)$
   La configuration $\{q_1, q_2\}$ est un état $\boxed{\{q_1, q_2\}}^a$ de l'automate déterministe A^D et c'est un état accepteur pour A^D puisque q_2 est un état accepteur de A.
   Si on prend le complémentaire de A^D, $\boxed{\{q_1, q_2\}}$ devient un état non-accepteur dans A'
   On n'obtient pas le même automate A' si on effectue les opérations dans l'ordre inverse :
2. $A^C := \textbf{\textit{complémentaire}}(A); A' := \textbf{\textit{déterminisation}}(A^C)$
   En prenant le complémentaire de A l'état q_1 devient accepteur dans A^c et l'état q_2 devient non-accepteur dans A^C. Lors de la déterminisation on rencontre la même configuration $\{q_1, q_2\}$, qui donne un état $\boxed{\{q_1, q_2\}}^a$ accepteur dans A' puisque q_1 est accepteur de A^C.

Exercice 2.4 Automates reconnaisseurs classiques

Dans chacun des cas suivants, donner un automate déterministe reconnaissant le langage sur l'alphabet $\{0, 1\}$:

a) l'ensemble des mots se terminant par 00 ;

b) l'ensemble des mots ayant au moins 3 zéros consécutifs ;

c) donnez une version non-déterministe de l'automate précédent ;

d) l'ensemble des mots dont l'avant-dernier symbole est un 1 ;

Indication Utilisez les catégories $\omega, \omega 1, \omega 11, \omega 10$

e) l'ensemble des mots qui contiennent au plus deux 0 consécutifs et au plus deux 1 consécutifs.

Indication En faisant le produit de deux automates et en remarquant que dans tout examen la dernière question réutilise le résultat d'une question précédente.

Exercice 2.5 Langage reconnu

Q33. Donnez le diagramme de transition de l'automate suivant :

$\delta_{\mathcal{A}}$	${}_i0$	1^a
a	0	1
b	1	0

Q34. Décrivez en une phrase le langage reconnu par cet automate.

Exercice 2.6 Minimisation et dénombrement

On considère l'alphabet $\Sigma = \{0, 1\}$

Q35. Donnez un automate déterministe qui reconnaît le langage L formé des mots tels que tout bloc de trois lettres consécutives contient au moins deux 0.

Exemple : $\epsilon, 01, 010, 0101 \in L$ et $101, 0110 \notin L$

On construit un automate dont les états représentent les différents configurations à envisager, de $\boxed{\ }\boxed{\ }\boxed{\ }$ à $\boxed{1}\boxed{1}\boxed{1}$ et $\boxed{0}\boxed{0}\boxed{0}$ en passant par $\boxed{0}\boxed{\ }\boxed{\ }$, $\boxed{0}\boxed{1}\boxed{\ }$, *etc.*

Q36. Minimisation Complétez l'automate de la Figure 1, indiquez les états accepteurs et minimisez-le.

Q37. Dénombrement

Donnez un automate qui reconnaît le langage L des mots qui se terminent par 010 et comptez le nombre de mots de L de longueur 10. Expliquez votre méthode.

Q38. Généralisation

Donnez une méthode générale qui permet de compter le nombre de mot de longueur n reconnu par un automate A.

Exercice 2.7 Produit et minimalité

L'objectif de cet exercice est de montrer que le produit d'automates ne préserve pas la minimalité. On considère l'alphabet $\Sigma = \{a, b\}$. **Complétez les pointillés et répondez aux questions**.

Q39. Donnez, sous forme graphique, l'automate A_1 déterministe minimal qui reconnaît les mots de la forme $(ab)^*a$. c'est-à-dire les mots formés de ....

....

Q40. Donnez, sous forme graphique, l'automate A_2 déterministe minimal qui recconnaît les mots du langage $\overline{\mathscr{L}(A_1)}$.

Q41. Donnez, sous forme graphique, l'automate déterministe minimal qui recconnaît le langage $\emptyset$.

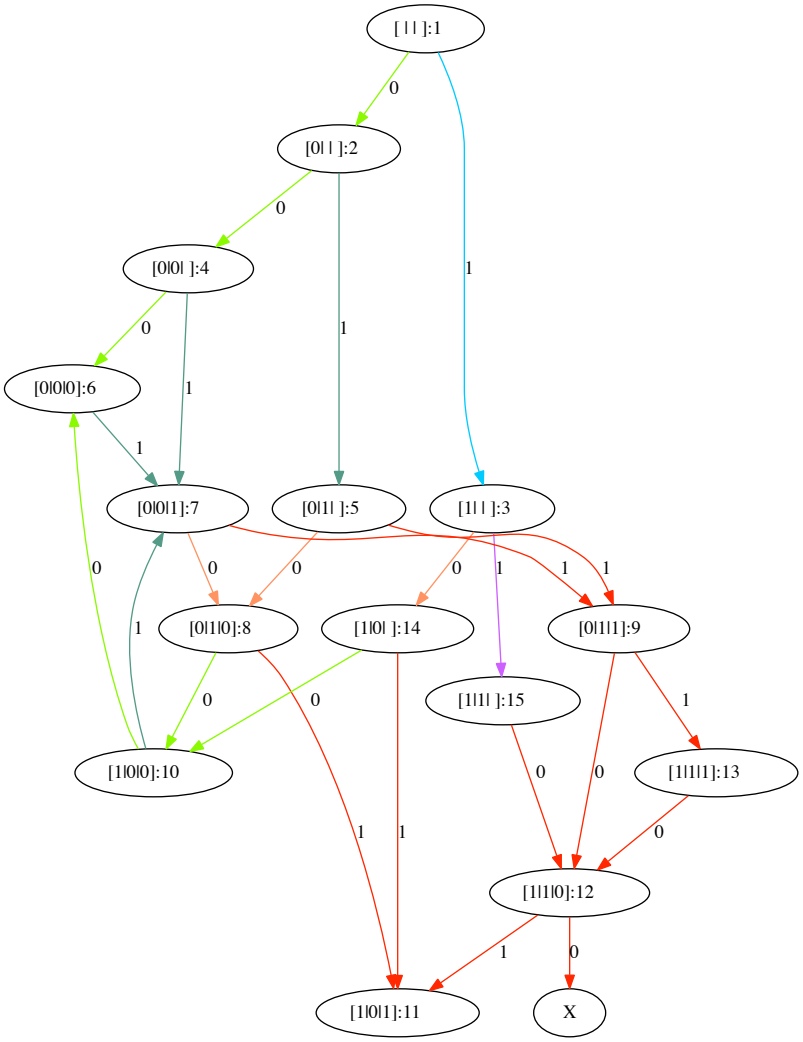


FIG. 1 – Automate de la question Exercice 2.6

On souhaite démontrer que le produit d'automates ne préserve pas la minimalité.

Indication Il faut montrer que l'automate produit de deux automates minimaux ne donne pas nécessairement un automate ..... Pour démontrer cela il suffit de donner un ..... Il faut donc trouver trois automates A , B et C tels que

- A et B sont ..... c'est-à-dire il n'existe pas d'automate avec moins ..... qui reconnaissent $\mathscr{L}(A)$ (resp. $\mathscr{L}(B)$).
- l'automate C reconnaît le ..... l'automate produit $A \times B$ et C a moins ..... que $A \times B$. Ce qui prouvera que le .... n'est pas un automate .....

Les questions précédentes suggèrent un contre-exemple.

Q42. Donnez les automates A, B, C et $A \times B$ sous forme de tableaux et concluez en n'oubliant pas de justifiez pourquoi $\mathscr{L}(C) = \mathscr{L}(A \times B)$.

Série 3 – Déterminisation, minimisation, somme d'automates

Exercice 3.1 Déterminisation d'automates reconnaisseurs

On considère l'alphabet $\Sigma = \{a, b\}$.

Q43. Reconnaisseurs non-déterministes Donnez un ANDEF qui le reconnaît le langage formé des mots :

a) $?^* aba$ (voir Exercice 2.1)

b) dont l'avant dernier symbole est un a (voir Exercice 2.4)

Q44. Déterminisation Pour chacun des ANDEF de la question précédente donnez un ADEF qui reconnaît le même langage.

Q45. Minimisation Pour chacun des ADEF de la question précédente donnez un ADEF minimal qui reconnaît le même langage.

Solution (exemple de rédaction) *Construction de l'automate minimal A_a^M de la question a) par minimisation de l'automate déterministe A_a^D*

Partition0 : $\underbrace{\{1, 2, 3, 4^a\}}_{Q_0}$

$4 \ldots Acc(A)$
$1, 2, 3 \ldots Acc(A)$
} *donc on partitionne le groupe Q_0 selon Acc en Q_1 et Q_2*

Partition1 : $\underbrace{\{1, 2, 3\}}_{Q_1}\ \underbrace{\{4^a\}}_{Q_2}$

$1 \xrightarrow{a} 2 \Rightarrow 1 \xrightarrow{a} Q$ ..
$2 \xrightarrow{a} 2 \Rightarrow 2 \xrightarrow{a} Q$ ..
$3 \xrightarrow{a} 4 \Rightarrow 3 \xrightarrow{a} Q$ ..
} *donc on partitionne le groupe Q_1 selon a en Q_3 et Q_4*

Partition2 : $\underbrace{\{1, 2\}}_{Q_4}\ \underbrace{\{3\}}_{Q_3}\ \underbrace{\{4^a\}}_{Q_2}$

$1 \xrightarrow{..} 1 \Rightarrow 1 \xrightarrow{..} Q$ ..
$2 \xrightarrow{..} 3 \Rightarrow 2 \xrightarrow{..} Q$ ..
} *donc on partitionne le groupe Q_4 selon b en Q_5 et Q_6*

Partition3 : $\underbrace{\{1\}}_{Q_5}\ \underbrace{\{2\}}_{Q_6}\ \underbrace{\{3\}}_{Q_3}\ \underbrace{\{4^a\}}_{Q_2}$

Conclusion : L'algorithme de minimisation par partitionnement successif donne la partition (compatible avec la relation de transition de l'automate) qui regroupe le plus d'états. Il se trouve qu'elle isole chaque état, donc l'automate de départ était minimal.

Q46. Comparez les ANDEF, ADEF, ADEF minimaux avec l'automate que vous aviez construit intuitivement (Exercice 2.1 et Exercice 2.4).

Exercice 3.2 Construction d'automates reconnaisseurs

On considère un alphabet Σ contenant au moins les symboles a, b, c.
Donnez un automate qui reconnaît le langage

Q47. des mots qui se terminent par b s'ils contiennent au moins un a.

$$L_a \stackrel{def}{=} \{\omega \mid \big(\exists \omega_1, \omega_2 \in \Sigma^*,\ \ \omega = \omega_1 a \omega_2\big) \Rightarrow \big(\ldots \omega_3 \in \Sigma^*\ \omega = \ldots\ldots\ldots\big)\}$$

Indication

- Rappels de logique : $P \Rightarrow Q$ équivaut à $\neg P \vee Q$ $\quad \neg\big(\forall x,\ P(x)\big)$ équivaut à $\exists x,\ \neg\big(P(x)\big)$
- Rappel sur les ensembles :

$$\begin{aligned} \{x \mid P(x) \Rightarrow Q(x)\} &= \{x \mid \neg P(x) \vee Q(x)\} = \{x \mid \neg P(x)\} \ldots \{x \mid Q(x)\} \\ &= \overline{\{x \mid \ldots\ldots\ldots\ldots\}} \ldots \{x \mid Q(x)\} \end{aligned}$$

Solution (exemple de rédaction)

$$\begin{aligned} L_a &= \{\omega \mid \big(\exists \omega_1, \omega_2,\ \omega = \omega_1 a \omega_2\big) \Rightarrow \big(\exists \omega_3,\ \omega = \omega_3 b\big)\} \\ &= \{\omega \mid \ \ldots \big(\exists \omega_1, \omega_2,\ \omega = \omega_1 a \omega_2\big) \ldots . \big(\exists \omega_3,\ \omega = \omega_3 b\big)\} \\ &= \overline{\{\omega \mid \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}} \ldots \{\omega \mid \exists w_3,\ \omega = \omega_3 b\} \end{aligned}$$

Exercice 3.3 Construction d'automates reconnaisseurs (révision)

Pour chaque automate donnez sa représentation graphique et sous forme de table de transitions.

Q48. Donnez un automate déterministe à nombre d'états finis (ADNEF) qui reconnaît les mots qui se terminent par ba.

Q49. Donnez un ADNEF qui reconnaît les mots qui commencent par ab

Q50. Donnez un ADNEF qui reconnaît les mots qui contiennent $abaa$.

Q51. Complémentaire Donnez un ADNEF qui reconnaît les mots qui ne contiennent pas $abaa$.

Q52. Intersection de langage, produit d'automate Donnez un ADNEF qui reconnaît le langage formé des mots qui commencent par ab et se terminent par ab.

Q53. Donnez un ADNEF qui reconnaît le langage formé des mots qui contiennent $abaa$ et ne se terminent pas par ba.

Q54. Union de langages Donnez un automate A_g (non déterministe) à nombre d'états finis (ANEF) qui reconnaît les mots qui contiennent $abaa$ ou commencent par ab ou terminent par ab

Q55. Élimination des ϵ-transitions Donnez un automate A_h équivalent à A_g qui n'ait plus d'ϵ-transitions.

Q56. Déterminisation Donnez un automate A_i déterministe équivalent à l'automate A_h.

Q57. Donnez un automate A_j déterministe équivalent à l'automate A_g en effectuant simultanément la déterminisation et l'élimination des ϵ-transitions.

Exercice 3.4 Révision des définitions sur les langages

Soit a un symbole, L, L_1, L_2 des langages et Σ un alphabet.

Q58. Complétez

- L'élément neutre de la concaténation de mot est ...
- Σ^* est l'ensemble des mots finis formés de 0 ou plus de symboles de l'alphabet Σ. En particulier, le mot .. de longueur 0 ..... Σ^* et l'ensemble .... des mots de longueur 1 ..... Σ^*.
- $L_1 \cdot L_2 = \{\omega_1.\omega_2 \mid \ldots\ldots\ldots\ldots \wedge \ldots\ldots\ldots\ldots\}$
- Si l'alphabet Σ est vide, alors $\Sigma^* = \ldots\ldots\ldots$
- $L \cdot \{\} = \ldots\ldots$
- $\emptyset \cdot L = \ldots$
- $\{a\} \cdot \{\}^* = \ldots\ldots\ldots\ldots\ldots\ldots = \ldots\ldots\ldots$
- $\emptyset^* \cdot L = \ldots\ldots$
- $\emptyset^* \cdot \emptyset = \ldots$
- Σ^+ est l'ensemble des mots de Σ^* de longueur ≥ 1, c'est-à-dire $\Sigma^+ \stackrel{def}{=} \{\omega \in \ldots\ldots\ldots \mid \ldots\ldots\ldots\ldots\ldots\}$
  $\Sigma^+ = \ldots\ldots \cdot \ldots\ldots\ldots = \ldots\ldots\ldots \cdot \ldots\ldots$
  $\Sigma^+ = \ldots\ldots\ldots \setminus \ldots\ldots\ldots$
- L un langage sur l'alphabet Σ signifie $L \ldots\ldots \Sigma^*$.

Q59. Rédigez un raisonnement Donnez des conditions nécessaires et suffisantes pour avoir $\{a\} \cdot \Sigma^* \subseteq \Sigma^+$

Indication Considérez les cas $\Sigma = \emptyset$, $\Sigma \neq \emptyset$, $a \in \Sigma$ et $a \notin \Sigma$.

Solution (exemple de rédaction)

- *Dans le cas $\Sigma = \emptyset$, forcément $\Sigma^* = \ldots\ldots\ldots$ et $\Sigma^+ = \ldots\ldots \cdot \Sigma^* = \ldots\ldots \cdot \ldots\ldots\ldots = \ldots\ldots$ donc $\{a\} \cdot \Sigma^* = \{a\} \cdot \ldots\ldots\ldots = \ldots\ldots\ldots$, et donc l'inclusion est .....*
- *Dans le cas $\Sigma \neq \emptyset$, distinguons le cas $a \in \Sigma$ et le cas $a \notin \Sigma$*
  - *si $a \in \Sigma$ alors si on ajoute la lettre a devant un mot de Σ^* on obtient un mot de ..... donc l'inclusion est .....*
  - *si $a \notin \Sigma$, alors les mots du langage $\{a\} \cdot \Sigma^*$, c'est-à-dire les mots ..... par a ne sont pas des mots de ..... et donc l'inclusion est .....*

Conclusion : l'inclusion est vraie si et seulement si .....

Q60. Rédigez un raisonnement Même question avec $\Sigma^+ \subseteq \{a\} \cdot \Sigma^*$

Solution (exemple de rédaction)

- *Dans le cas $\Sigma = \emptyset$, $\Sigma^+ = \ldots \subseteq \ldots\ldots = \{a\} \cdot \Sigma^*$ et donc l'inclustion est .....*
- *Dans le cas $\Sigma \neq \emptyset$,*

  L'inclusion est vraie si et seulement si ..... mot de ..... est un mot de .....,

  c'est-à-dire si les mots de ..... sont tous des mots .....

  Il faut donc nécessairement que $\Sigma =$ .....
  ***En effet**, supposons que l'alphabet Σ contienne une autre lettre, par exemple b, alors le langage ..... contient le mot ..... qui ..... On obtient une contradiction ; ce qui confirme que la seule possibilité pour que l'inclusion soit vraie c'est que Σ contienne ..... la lettre a.*

Q61. Rédigez un raisonnement En exploitant les résultats des questions b et c, donnez les conditions sur Σ nécessaires et suffisantes pour avoir $\Sigma^+ = \{a\} \cdot \Sigma^*$.

Q62. Est-il vrai que $\Sigma^* \cdot \Sigma^* \subseteq \Sigma^*$?

Solution (exemple de rédaction) *Par définition, Σ^* contient tous les mots construits par ..... d'un nombre ..... de ..... de Σ*
Considérons deux mots ω et ω' de Σ^* et concaténons les.
$\omega = l_1 \ldots l_n$ pour un certain n et des symboles $l_i \in \Sigma$
$\omega' = l'_1 \ldots l'_p$ pour un certain p et des symboles $l'_i \in \Sigma$

donc $\omega\omega' =$ ..... est un mot formé de la concaténation de .....

symboles de Σ donc $\omega\omega'$ .....

Série 4 – Automates $\simeq$ Équations d'Arden $\simeq$ Expressions régulières

Notions abordées dans ce TD

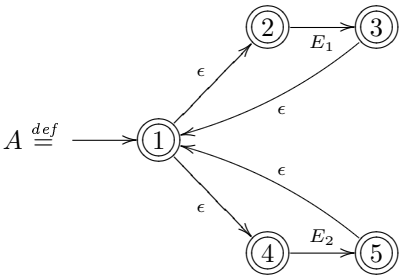
- *construction de l'automate associé à une expression régulière par la méthode de Thompson*
- *élimination des ϵ-transitions, déterminisation, minimisation*
- *construction du système d'équations correspondant à un automate*
- *résolution d'équations entre langages à l'aide du lemme d'Arden*
- *Manipulation des différentes représentations de langages pour montrer des équivalences entre langages*

Des expressions régulières aux automates

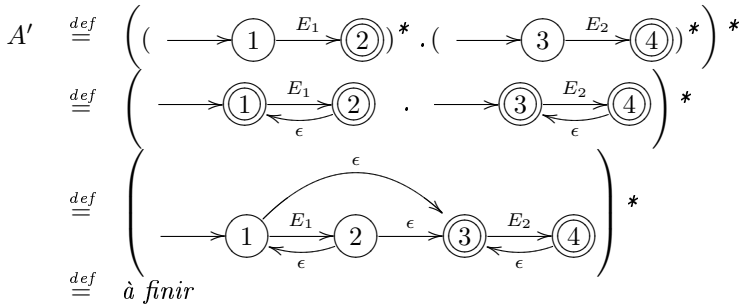
Exercice 4.1 Expressions régulières équivalentes

Q63. Soit A_1 l'automate correspondant à l'expression régulière E_1 et A_2 l'automate correspondant à l'expression régulière E_2. Construire les automates non-déterministes A, A' qui reconnaîssent les langages définis par les expressions régulières $(E_1 \mid E_2)^*$ et $({E_1}^*.{E_2}^*)^*$.

Solution (exemple de rédaction)



Les étapes de construction de A'



Q64. Éliminez les ϵ-transitions, déterminisez et minimisez les automates.

Solution (exemple de rédaction)

Élimination des ϵ

$$1 \xrightarrow{\epsilon^*} \begin{cases} 1 \xrightarrow{E_1} \emptyset \\ 2 \xrightarrow{E_1} 3 \xrightarrow{\epsilon^*} 1,2,3,4 \\ 4 \xrightarrow{E_1} \emptyset \end{cases} \quad donc\ 1 \xrightarrow{\epsilon^*.E_1.\epsilon^*} 1,2,3,4$$

$$2 \xrightarrow{\epsilon^*} 2 \xrightarrow{E_1} 3 \xrightarrow{\epsilon^*} 1,2,3,4\ donc\ 2 \xrightarrow{\epsilon^*.E_1.\epsilon^*} 1,2,3,4$$

$$3 \xrightarrow{\epsilon^*} \begin{cases} 1 \xrightarrow{E_1} \emptyset \\ 2 \xrightarrow{E_1} 3 \xrightarrow{\epsilon^*} 1,2,3,4 \\ 3 \xrightarrow{E_1} \emptyset \\ 4 \xrightarrow{E_1} \emptyset \end{cases} \quad donc\ 3 \xrightarrow{\epsilon^*.E_1.\epsilon^*} 1,2,3,4$$

$$4 \xrightarrow{\epsilon^*} 4 \xrightarrow{E_1} \emptyset\ donc\ 4 \xrightarrow{\epsilon^*.E_1.\epsilon^*} \emptyset$$

$$5 \xrightarrow{\epsilon^*} \begin{cases} 1 \xrightarrow{E_1} \emptyset \\ 2 \xrightarrow{E_1} 3 \xrightarrow{\epsilon^*} 1,2,3,4 \\ 4 \xrightarrow{E_1} \emptyset \\ 5 \xrightarrow{E_1} \emptyset \end{cases} \quad donc\ 5 \xrightarrow{\epsilon^*.E_1.\epsilon^*} 1,2,3,4$$

Appliquez le même principe pour les transitions $\epsilon^*.E_2.\epsilon^*$

$A^{\not\epsilon}$	${}_i1^a$	2^a	3^a	4^a	5^a
ϵ	$2,4$		1		1
ϵ^*	$1,2,4$	2	$3,1,2,4$	4	$5,1,2,4$
E_1		3			
$\epsilon^*.E_1.\epsilon^*$	$1,2,3,4$	$1,2,3,4$	$1,2,3,4$		$1,2,3,4$
E_2				5	
$\epsilon^*.E_2.\epsilon^*$	$1,2,4,5$		$1,2,4,5$	$1,2,4,5$	$1,2,4,5$

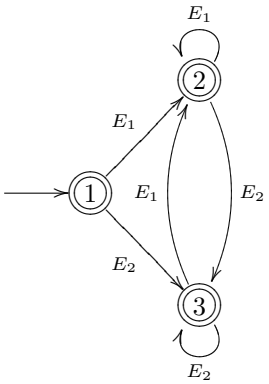
Déterminisation

A^D	${}_i\{1\}^a$	$\{1,2,3,4\}^a$	$\{1,2,4,5\}^a$
E_1	$\{1,2,3,4\}$	$\{1,2,3,4\}$	$\{1,2,3,4\}$
E_2	$\{1,2,4,5\}$	$\{1,2,4,5\}$	$\{1,2,4,5\}$

$=$

A^D	${}_i1^a$	2^a	3^a
E_1	2	2	2
E_2	3	3	3

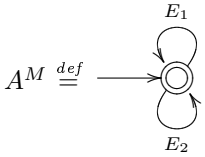
$=$



Minimisation

$$partition_0 \stackrel{def}{=} \underbrace{\{1^a, 2^a, 3^a\}}_{les\ états\ accepteurs} \sqcup \underbrace{\{\}}_{les\ états\ non\ accepteurs}$$

Il n'y a aucune raison de rafiner cette partition puisque aucun des états accepteurs ne peut aller dans le groupe des états non accepteurs. La minimisation est donc terminée et l'automate minimal possède un seul état ${}_i\{1,2,3\}^a$.



Appliquez la même méthode à l'automate A'**.**

Q65. En déduire que les expressions régulières sont équivalentes. Rédigez votre réponse.

Solution (exemple de rédaction) *Les automates qui reconnaissent le langage défini par les expressions régulières* E_1 *et* E_2 *sont* ..... *donc* E_1 *et* E_2 *définissent* ..... *langage, donc elles sont* .....

Exercice 4.2 Expressions régulières équivalentes (similaire à Exercice 4.1)

Q66. Construire les automates A_1, A_2, A_3 qui reconnaissent les langages défini par E^*, $E + \epsilon^*$ et E^{**}

Q67. Éliminez les ϵ-transitions, déterminisez et minimisez ces automates

Q68. En déduire que les expressions régulières E^*, $E + \epsilon^*$ et E^{**} sont équivalentes.

Exercice 4.3 Vrai ou Faux ?

Indiquez si les expressions régulières suivantes sont équivalentes ou non. Justifiez votre réponse par une preuve ou un contre-exemeple

a) $(E_1 \mid E_2)^* = {E_1}^* \mid {E_2}^*$: .....

b) $(R^* \cdot S^*)^* = (S^* \cdot R^*)^*$: Vrai

c) $(E_1 \,.\, E_2 \mid E_1)^* \,.\, E_1 \,.\, E_2 = (E_1 \,.\, {E_1}^* \,.\, E_2)^*$: .....

d) $(E_1 \,.\, E_2)^* \,.\, E_1 = E_1 \,.(E_2 \,.\, E_1)^*$: vrai, on le montre en construisant les automates minimaux des expressions régulières droite et gauche de l'égalité.

e) $(A \,.\, B \mid A)^* \,.\, A = A \,.(B \,.\, A \mid A)^*$: vrai, démontrez le en appliquant le résultat précédent.

f) $(E^*)^* = E^*$

g) $({E_1}^* \,.\, {E_2}^*)^* = (E_2 \mid E_1)^*$: .....

h) $(E \mid \epsilon)^* = E^*$: .....

i) $(E_1 \,.\, E_2)^* = {E_1}^* \,.\, {E_2}^*$: .....

Des équations d'Arden aux expressions régulières

Exercice 4.4 Langage(s) solution(s) d'une équation d'Arden

Étant donné deux expressions régulières A et B, on cherche le langage L solution de l'équation :

- **égalité entre langages**

  $$L = \big(\mathscr{L}(A)\big)^* \cdot L \;\cup\; \mathscr{L}(B)$$

- **équivalence entre expression régulières**

  $$L = A^*.L \;+\; B$$

Q69. Complétez : Si l'ensemble de mots L est solution de l'équation alors :

- Forcément B L
- Si $\omega \in L$ alors d'après l'équation les mots du langage $A \cdot \omega$ .... L mais alors les mots du langage $A \cdot$ ..... $\subseteq$ ...., *etc.* Finalement les mots du langage ..... $\subseteq L$.

Q70. Donnez des langages solutions de l'équation d'Arden dans les cas suivants
Donnez en particulier la plus petite et la plus grande solution.

a) $\mathscr{L}(A) = \{a\},\ \mathscr{L}(B) = \{\epsilon\}$

b) $\mathscr{L}(A) = \{a\},\ \mathscr{L}(B) = \emptyset$

c) $A = a,\ B = b$

d) $A = a + \epsilon,\ B = b$

Solution (exemple de rédaction)

D'après l'équation $L = (a+\epsilon)L \cup B = a\cdot \ldots \cup \ldots \cdot L \cup B = a\cdot \ldots \cup \ldots \cup B$ *donc tout langage* L *contenant* B *et tel que* $\omega \in L \Rightarrow a^*\omega \in L$ *est solution. En particulier, le langage universel* ..... *est solution et tout langage de la forme* $B \cup \{\omega, a^*\omega\}$ *est solution.*

Q71. À chercher On suppose que $\epsilon \in \mathscr{L}(A)$. Indiquez si les langages suivants sont solutions de l'équation d'Arden $L = A \cdot L \cup B$. Justifiez votre réponse.

a) $L = \Sigma$

b) $L = \mathscr{L}(A^* \cdot B)$

c) $L = \mathscr{L}(A^*)$

d) $L = \mathscr{L}(B)$

e) $L = \mathscr{L}(B) \cup L' \cup \mathscr{L}(A^* \cdot L')$ pour un langage L' quelconque

Exercice 4.5 Résolution d'un système d'équations à l'aide du lemme d'Arden

Q72. Automate Donnez un ANDEF (avec le moins de transitions possible) qui reconnaît les mots sur l'alphabet $\{a, b\}$ dont l'avant dernier symbole est un a.

Q73. Système d'équations En déduire le système d'équations qui caractérise le langage reconnu par chacun des états. On notera L_i le langage reconnu par l'état i de l'automate.

Solution (exemple de rédaction)

$$\begin{cases} L_1 = \ldots\ldots\ldots + \ldots\ldots\ldots + \ldots\ldots\ldots = (\ldots + \ldots).L_1 + \ldots.L \\ \qquad\qquad \xrightarrow{Arden} L_1 = (\ldots + \ldots)^*.\ldots.L \\ L_2 = \ldots\ldots\ldots + \ldots\ldots\ldots = (\ldots + \ldots) \\ L_3 = \ldots \end{cases}$$

Q74. Expression régulière Résoudre le système d'équations en utilisant le lemme d'Arden afin de déterminer l'expression régulière qui caractérise le langage reconnu par chacun des états de l'automate.

Solution (exemple de rédaction) *Conclusion : L'expression régulière correspondant à l'automate est celle qui caractérise le langage reconnu par l'état initial de l'automate, c'est donc lexpression régulière de* L_1 *:* $(a+b)^*.a.(a+b)$

Exercice 4.6 Transformations d'expressions régulières

Q75. Utilisez la méthode de l'Exercice 4.5 pour trouver l'expression régulière qui reconnaît les mots qui ne sont pas de la forme $a\ .\ a^*\ .\ b$

Q76. Généralisation Soit e une expression régulière classique, c'est-à-dire formée à partir des symboles de l'alphabet Σ, de `$`, `{}` et des opérateurs `.`, `|`, `*`

Expliquez le principe d'un algorithme qui permet de transformer l'expression $\overline{e}$ en une expression régulière classique équivalente de sorte que $\mathscr{L}(\overline{e}) = \overline{\mathscr{L}(e)}$.

Q77. À chercher On souhaite étendre le langage des expressions régulières en ajoutant l'opérateur $(-)$ défini par

$$\mathscr{L}(e_1 - e_2) = \mathscr{L}(e_1) \setminus \mathscr{L}(e_2)$$

où e_1 et e_2 désignent deux expressions régulières classiques.
Montrez qu'on peut construire un automate qui reconnaît le langage défini par l'expression $e_1 - e_2$.

Indication Rappel sur les ensembles : $E \setminus F = \ldots\ldots\ldots\ldots$

Q78. Application Construire l'expression régulière classique équivalente à $a \,.\, a^* \,.\, b - a \,.\, a^* \,.\, b$

Série 5 – Grammaires attribuées (attributs synthétisés uniquement)

Exercice 5.1 Ajout d'un compteur à une grammaire

On ajoute un attribut $n \in \mathbb{N}$ à la grammaire pour qu'en plus de la reconnaissance on calcule le n du mot $a^n b^n$ reconnu.

Q79. Donnez la grammaire du langage $\{a^n b^n \mid n \in \mathbb{N}\}$.

Q80. Donnez, pour votre grammaire, l'arbre de dérivation du mot `"aabb"`

Q81. Ajoutez un attribut donne le nombre n d'un mot reconnu (par exemple $n = 2$ pour `"aabb"`).

Q82. (**) Ajoutez des attributs qui constuisent l'arbre de dérivation du mot reconnu.
Un arbre de dérivation Δ est une donnée de la forme $\textsc{Der}(\textsc{nt}, [\Delta_1; \ldots; \Delta_n])$ qui correspond à l'étape de dérivation

$$\begin{array}{c} \textsc{nt} \\ \downarrow \\ \Delta_1 \ldots . \Delta_n \end{array}$$

où $\textsc{nt}$ est un non-terminal et $\Delta_1, \ldots, \Delta_n$ sont eux-mêmes des arbres qui représentent la suite des dérivations

L'arbre de dérivations associé à la règle $S \to S_1.S_2$ est $\begin{array}{c} S \\ \downarrow \\ \Delta_1.\Delta_2 \end{array}$ en notation graphique, c'est-à-dire $\textsc{Der}(\textsc{s}, [\Delta_1; \Delta_2])$ pour l'implantation, où Δ_1, Δ_2 sont les arbres de dérivation produit par S_1 et S_2.

Série 6 – Grammaires attribuées (attributs hérités et synthétisés)

L'exercice de l'examen sur les grammaires attribuées sera une variante des exercices suivants

Exercice 6.1 Application : calcul de l'entier correspondant à son écriture en base 10

On souhaite écrire une fonction G de reconnaissance des écritures en base 10 qui retourne l'entier correspondant.

```
1   G: string → int
2
3   G ("000123400") = 123400
```

Il s'agit donc de donner une grammaire attribuée qui lise une suite de caractères avec reconnaissance des nombres entiers et calcule de l'entier correspondant.

Q83. Donnez la grammaire sans attribut

Q84. Donnez la grammaire avec attribut

Exercice 6.2 Grammaire des textes

Un texte est une suite de pharses. Une phrase est une suite de mots terminée par un point. Les mots sont toujours précédés d'au moins un espaces. Un mot est composé d'au moins une lettre.

Q85. Donnez la grammaire qui reconnaît les textes

Q86. Ajoutez des attributs synthétisés pour compter le nombre de mot et le nombre de lettres du texte.

Exercice 6.3 Grammaire des expressions arithmétiques

On considère les expressions arithmétiques formées de constantes (dans $\mathbb{Z}$) et des opérateurs binaires $+, \times$.

Q87. Donnez la grammaire G qui décrit les expressions arithmétiques bien formées :

$3 + 2 + 5 \times 10 \in \mathscr{L}(G)$
$+ + \times\ 3\ 4 \notin \mathscr{L}(G)$

Utilisez les non-terminaux N pour les nombres, C pour les chiffres, E pour les expressions.

Q88. Montrez que la grammaire est ambigue en montrant que l'expression $3 + 4 + 5$ admet deux arbres de dérivations

Q89. Grammaire $LL1$ Donnez une version non-ambigue de la grammaire en utilisant des règles telles que $OpE \rightarrow$ `'+'`$.E$ qui permettent de décider quelle règle utiliser en se basant sur le prochain caractères.

Q90. Calculatrice en ligne Ajoutez des attributs afin que la grammaire évalue l'expression arithmétique en même temps qu'elle la reconnaît.

Q91. Hors TD Ajoutez à la grammaire précédente l'opérateur unaire $-$ qui inverse le signe d'une expression.

Exercice 6.4 Grammaire des déclaration de type

Q92. Donnez une grammaire qui définit les déclarations de types du langage C de manière à admettre les déclarations :

```
int x=1;
float y,z;
int t;
float u,v=0;
```

et à rejeter les déclarations :

```
int x, int y;
int x=0,y=1;
```

Q93. Ajoutez un attibut synthétisé de manière à rendre sous la forme d'une chaîne de caractères des déclarations individualisés.

```
int x=1; float y; float z; int t; int u; float v=0;
```

Vous utiliserez l'opérateur $\_\cdot\_$ pour concaténer deux chaînes de caractères.

Q94. Modifiez la gestion des attibuts afin de rendre sous la forme d'une chaîne de caractères un programme avec des déclarations sous la forme :

```
x,t : int ;
y,z,u,v : float ;
x:=1 ;
v:=0 ;
```

c'est-à-dire avec toutes les déclarations de type au début, regroupée par type et toutes les initialisations à la fin.

Exercice 6.5 Grammaire des textes html

Q95. Donnez une grammaire qui définit les textes html formés des balises suivantes

```
<html> ... <ol>  <li>...</li> </ol> ... </html>
```

où
– les pointillés sont des textes (on utilisera le non-terminal T pour faire référence à un texte) de la grammaire de l'exercice 1)
– `<ol>` indique le début d'une liste et `<li>` est un élément de la liste
La grammaire doit autorisées les listes imbriquées :

```
<ol>
  <li> ...
    <ol>
      <li>...</li>
      <li>...</li>
    </ol>
 </li>
</ol>
```

Q96. Ajoutez des attributs afin de compter

1. le nombre d'éléments de la plus longue liste
2. la profondeur maximale d'imbrication : une liste de liste de liste est de profondeur 3

L'exemple précédent retournera le résultat $(2, 2)$

Exercice 6.6 Grammaires attribuées des flottants (3 pt) (15min)

On utilise les notations suivantes pour les non-terminaux F pour Flottant, E pour partie Entière, D pour partie Décimale, C pour Chiffre, P pour le Point qui sépare la partie entière de la partie décimale. On donne la grammaire qui définit les **écritures des nombres flottants** :

$$\begin{cases} F \xrightarrow{(1)} E \ \cdot \ P \ \cdot \ D \\ E \xrightarrow{(2)} C \\ E \xrightarrow{(3)} C \cdot E \\ P \xrightarrow{(4)} \texttt{"."} \\ D \xrightarrow{(5)} C \\ D \xrightarrow{(6)} C \cdot D \\ C \xrightarrow{(7)} \texttt{"0"} \mid \texttt{"1"} \mid \ldots \mid \texttt{"9"} \end{cases}$$

Q97. (0.75 pt) Donnez l'arbre de dérivation qui permet de générer le flottant 20.01 et indiquez à chaque dérivation le numéro de la règle utilisée.

Q98. (0.75 pt) Complétez la grammaire des chiffres décimaux avec un attribut afin qu'elle retourne le nombre de décimales lues. Par exemple D doit retourner 3 lorsqu'elle reconnaît la partie 007 du flottant 18.007

$$\begin{cases} D \ \ldots\ldots\ldots\ldots\ldots \ \xrightarrow{(4)} \ C \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ D \ \ldots\ldots\ldots\ldots\ldots \ \xrightarrow{(4')} \ C \ \ldots\ldots\ldots\ldots\ldots\ldots \ D \ \ldots\ldots\ldots\ldots\ldots\ldots \end{cases}$$

Q99. (1.5 pt) Complétez la grammaire des flottants avec le calcul des attributs afin que le non-terminal F retourne le flottant f reconnu.

$$
\left\{
\begin{array}{rcl}
F\ \{f\} & \xrightarrow{(1)} & \{..\}\ E\ \{e\}\ \cdot\ P\ \cdot\ \{..\}\ D\ \{(n,d)\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
\{i\}\ E\ \{e\} & \xrightarrow{(2)} & C\ \{..\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
\{i\}\ E\ \{e\} & \xrightarrow{(3)} & C\ \{..\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\ ;\\
 & & \{i'\}\ E\ \{e'\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
P & \xrightarrow{(4)} & \texttt{"."}\\
\{i\}\ D\ \{(n,d)\} & \xrightarrow{(5)} & C\ \{..\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
\{i\}\ D\ \{(n,d)\} & \xrightarrow{(6)} & C\ \{c\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\ ;\\
 & & \{i'\}\ D\ \{(n',d')\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
C\ \{c\} & \xrightarrow{(7)} & \texttt{"0"}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
 & | & \texttt{"1"}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
 & | & \ldots\\
 & | & \texttt{"9"}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}
\end{array}
\right.
$$

Exercice 6.7 Grammaires des listes html (4 pt) (15min)

Q100. (4 pt) Donnez une grammaire qui définit les documents html formés des balises suivantes

```
<html> <ol> <li> </li> </ol> </html>
```

où
– `<html>` indique le début (et `</html>` la fin) d'un **document html**
– `<ol>` indique le début (et `</ol>` la fin) d'une **liste**
– `<li>` indique le début (et `</li>` la fin) d'un **item** (élément de la liste)

La grammaire doit être telle que
– on ne s'occupe pas des espaces, de l'indentation et des sauts de ligne
– un **document html** commence par `<html>` et se termine par `</html>`
– entre les balises `<html>` et `</html>` on peut mettre 0 ou plusieurs **listes**
– entre les balises `<ol>` et `</ol>` il doit y avoir au moins un **item**
– entre les balises `<li>` et `</li>` on peut mettre 0 ou plusieurs **mots** et 0 ou plusieurs **listes**
– on ne peut mettre des **mots** qu'entre les balises `<li>` et `</li>`.
– les **mots** sont formés de (0 ou plusieurs) lettres de l'alphabet et de 0 ou plusieurs espaces

Exemples de documents acceptés par la grammaire

```
<html>
 <ol>
   <li> premier item</li>
   <li> second item
     mots avant la sous liste
     <ol>
       <li></li>
     </ol>
     mots apres la sous liste
   </li>
   <li> dernier item</li>
 </ol>
</html>
```

```
<html>
 <ol>
   <li></li>
 </ol>
 <ol>
    <li></li>
 </ol>
</html>
```

```
<html>
</html>
```

Exemples de documents rejetés par la grammaire

```
<html>
<ol></ol>
</html>
```

```
<html>
<ol> pas de mots ici
 <li> mots </li>
</ol>
</html>
```

```
<html>
pas de mots ici
</html>
```

Série 7 – Recueil d'exercices donnés en examens

Exercice 7.1 Reconnaissance d'un terme par un automate d'arbre (1.5 pt) (5min)

On considère l'alphabet $\Sigma = \{\text{A}, \text{F}^{\#2}, \text{G}^{\#1}\}$ et l'automate d'arbre défini par les transitions suivantes :

$$\begin{cases} \text{A} \xrightarrow{1} \mathbf{Q}_{\text{A}}(\text{A}) \\ \text{G}(\ \mathbf{Q}_{\text{G}}(x)\ ) \xrightarrow{2} \mathbf{Q}_{\text{G}}(\ \text{G}(x)\ ) \\ \text{G}(\ \mathbf{Q}_{\text{A}}(x)\ ) \xrightarrow{3} \mathbf{Q}_{\text{G}}(\ \text{G}(x)\ ) \\ \text{F}(\ \mathbf{Q}_{\text{G}}(x), \mathbf{Q}_{\text{G}}(y)\ ) \xrightarrow{4} \mathbf{Q}_{\text{F}}(\ \text{F}(x,y)\ ) \end{cases}$$

où
- $\mathbf{Q}_{\text{A}}, \mathbf{Q}_{\text{G}}, \mathbf{Q}_{\text{F}}$ représentent les états de l'automate d'arbre,
- $\mathbf{Q}_{\text{F}}$ est l'unique état accepteur
- x et y sont des variables

Q1. (0.5 pt) Donnez un terme qui n'est pas reconnu par l'automate ?

Q2. (1 pt) Donnez l'exécution qui permet de montrer que le terme $\text{F}(\text{G}(\text{A}),\text{G}(\text{A}))$ est reconnu par l'automate

Exercice 7.2 Expressions régulières (1 pt) (5min)

Q1. (0.25 pt) On note L l'ensemble des lettres minuscules et C l'ensemble des chiffres. Donnez l'expression régulière qui décrit les noms de variables formés sur l'alphabet $\Sigma = L \cup C \cup \{\_\}$ sachant qu'un identificateur est formé d'au moins un caractère et doit commencer par un lettre ou un souligné.

$id \stackrel{def}{=}$ .....

Définition de l'opérateur « − » sur les expressions régulières

Q2. (0.25 pt) On note $\mathscr{L}(e)$ le langage associé à l'expression régulière e. Soit e_1 et e_2 deux expressions régulières, définissez l'opérateur « privé de », notée « - »

$\mathscr{L}(e_1 - e_2) \stackrel{def}{=}$ .....

à l'aide des opérateurs ensemblistes $\cup$, $\overline{\ }$ et $\cap$

Q3. (0.25 pt) Soit A_1 l'automate associé à e_1 et A_2 l'automate associé à e_2, expliquez comment construire l'automate associé à $e_1 - e_2$.

$\mathscr{L}(e_1 - e_2)$ = ...

= .....

= .....

Donc l'automate associé à $e_1 - e_2$ est .....

Q4. (0.25 pt) Donnez l'expression régulière qui décrit les noms de variables de la question 1 mais en interdisant les mots-clefs `if,then,else`.

$id \stackrel{def}{=}$ .....

Exercice 7.3 Inclusions de langages (4 pt) (20min)

On considère l'alphabet $\Sigma = \{a, b\}$ et les langages L_1 à L_5 ci-dessous. Comparez les langages 2 à 2 et indiquez les relations qu'ils vérifient : $\subseteq$ (inclusion large), $\subset$ (inclusion stricte), $=$ (égalité), ou bien intersection vide. Justifiez chacunes de vos réponses par un raisonnement ou un contre-exemple.

Les justifications comptent pour $\frac{2}{3}$ des points. Attention, le fait de décrire en français ce que représente l'expression régulière n'est pas considéré comme un raisonnement.

- $L_1 \stackrel{def}{=} \mathscr{L}(\ \Sigma^*\ )$
- $L_2 \stackrel{def}{=} \mathscr{L}(\ (b+a)^*\ )$
- $L_3 \stackrel{def}{=} \mathscr{L}(\ \left((a^*).b + a.(b^*)\right)^*\ )$
- $L_4 \stackrel{def}{=} \mathscr{L}(\ a^* + b^*\ )$
- $L_5 \stackrel{def}{=} \mathscr{L}(\ (ab)^*\ )$

Exercice 7.4 Grammaires attribuées des flottants (3 pt) (15min)

On utilise les notations suivantes pour les non-terminaux F pour Flottant, E pour partie Entière, D pour partie Décimale, C pour Chiffre, P pour le Point qui sépare la partie entière de la partie décimale. On donne la grammaire qui définit les **écritures des nombres flottants** :

$$\begin{cases} F \xrightarrow{(1)} E \cdot P \cdot D \\ E \xrightarrow{(2)} C \\ E \xrightarrow{(3)} C \cdot E \\ P \xrightarrow{(4)} \texttt{"."} \\ D \xrightarrow{(5)} C \\ D \xrightarrow{(6)} C \cdot D \\ C \xrightarrow{(7)} \texttt{"0"} \mid \texttt{"1"} \mid \ldots \mid \texttt{"9"} \end{cases}$$

Q1. (0.75 pt) Donnez l'arbre de dérivation qui permet de générer le flottant 20.01 et indiquez à chaque dérivation le numéro de la règle utilisée.

Q2. (0.75 pt) Complétez la grammaire des chiffres décimaux avec un attribut afin qu'elle retourne le nombre de décimales lues. Par exemple D doit retourner 3 lorsqu'elle reconnaît la partie 007 du flottant 18.007

$$\begin{cases} D \ \ldots\ldots\ldots\ldots\ldots\ldots \xrightarrow{(4)} C \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ D \ \ldots\ldots\ldots\ldots\ldots\ldots \xrightarrow{(4')} C \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ D \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{cases}$$

Q3. (1.5 pt) Complétez la grammaire des flottants avec le calcul des attributs afin que le non-terminal F retourne le flottant f reconnu.

$$
\left\{
\begin{array}{rcl}
F\ \{f\} & \xrightarrow{(1)} & \{..\}\ E\ \{e\}\ \cdot\ P\ \cdot\ \{..\}\ D\ \{(n,d)\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
\{i\}\ E\ \{e\} & \xrightarrow{(2)} & C\ \{..\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
\{i\}\ E\ \{e\} & \xrightarrow{(3)} & C\ \{..\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\ ;\\
 & & \{i'\}\ E\ \{e'\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
P & \xrightarrow{(4)} & \texttt{"."}\\
\{i\}\ D\ \{(n,d)\} & \xrightarrow{(5)} & C\ \{..\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
\{i\}\ D\ \{(n,d)\} & \xrightarrow{(6)} & C\ \{c\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\ ;\\
 & & \{i'\}\ D\ \{(n',d')\}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
C\ \{c\} & \xrightarrow{(7)} & \texttt{"0"}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
 & | & \texttt{"1"}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}\\
 & | & \ldots\\
 & | & \texttt{"9"}\ ;\ \{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\}
\end{array}
\right.
$$

Exercice 7.5 Grammaires des listes html (4 pt) (15min)

Q1. (4 pt) Donnez une grammaire qui définit les documents html formés des balises suivantes

```
<html> <ol> <li> </li> </ol> </html>
```

où
– `<html>` indique le début (et `</html>` la fin) d'un **document html**
– `<ol>` indique le début (et `</ol>` la fin) d'une **liste**
– `<li>` indique le début (et `</li>` la fin) d'un **item** (élément de la liste)

La grammaire doit être telle que
– on ne s'occupe pas des espaces, de l'indentation et des sauts de ligne
– un **document html** commence par `<html>` et se termine par `</html>`
– entre les balises `<html>` et `</html>` on peut mettre 0 ou plusieurs **listes**
– entre les balises `<ol>` et `</ol>` il doit y avoir au moins un **item**
– entre les balises `<li>` et `</li>` on peut mettre 0 ou plusieurs **mots** et 0 ou plusieurs **listes**
– on ne peut mettre des **mots** qu'entre les balises `<li>` et `</li>`.
– les **mots** sont formés de (0 ou plusieurs) lettres de l'alphabet et de 0 ou plusieurs espaces

Exemples de documents acceptés par la grammaire

```
<html>
 <ol>
   <li> premier item</li>
   <li> second item
     mots avant la sous liste
     <ol>
       <li></li>
     </ol>
     mots apres la sous liste
   </li>
   <li> dernier item</li>
 </ol>
</html>
```

```
<html>
 <ol>
   <li></li>
 </ol>
 <ol>
    <li></li>
 </ol>
</html>
```

```
<html>
</html>
```

Exemples de documents rejetés par la grammaire

```
<html>
<ol></ol>
</html>
```

```
<html>
<ol> pas de mots ici
 <li> mots </li>
</ol>
</html>
```

```
<html>
pas de mots ici
</html>
```

Exercice 7.6 Déterminisation d'automate (5 pt) (20min)

On considère l'automate :

A	${}_i1$	2	3	4^a
a	$1,2$	3		4
b	1	3	4	4

Q1. (0.25 pt) Dessinez l'automate.

Q2. (0.75 pt) Donnez une exécution qui montre que le mot `"aabbaa"` est reconnu par l'automate A.

Q3. (2 pt) Donnez, sous forme de tableau, un automate déterministe équivalent à l'automate A.

Q4. (0.25 pt) Que signifie qu'un automate A est équivalent à un automate B ?

Q5. (0.75 pt) Rappelez la condition d'acception d'un mot par un automate non-déterministe.

Q6. (1 pt) Justifiez que le mot `"ababaa"` n'est pas reconnu par l'automate A.

Exercice 7.7 Modélisation de politiques de sécurité (4 pt) (20min)

On considère un réseau constitué de deux parties : un réseau « sûr » car protégé par un pare-feu et l'internet.

Pour contrôler l'usage des machines sur le réseau on installe sur chaque compte utilisateur un moniteur de sécurité qui observe les commandes suivantes :
- `login` qui lance la session principale
- `quit` qui ferme la session principale
- `rlogin` qui permet d'ouvrir une connection sur une machine du réseau protégé
- `ssh` qui permet d'ouvrir une session avec encryption des échanges sur une machine hors du réseau protégé.
- `exit` qui permet de fermer toutes les sessions en cours (sauf la session principale)

Dans cet exercice on considère que les autres commandes que tape l'utilisateur ne sont pas soumises au contrôle de sécurité.

> Principe d'un moniteur de sécurité : Un moniteur de sécurité prend en entrée une politique de sécurité P décrite sous la forme d'une expression régulière ou d'un automate d'états fini ou d'un automate à une pile déterministe. Il autorise uniquement les séquences de commandes qui appartiennent au langage $\mathscr{L}(P)$.

Q1. (0.25 pt) Donnez l'alphabet Σ considérée par les politique de sécurité sur ce réseau.

Q2. (0.75 pt) Donnez 3 exemples de séquences de commandes qui respectent toutes les contraintes de la politique de sécurité n°1 ci-dessous et 3 exemples de séquences de commandes qui ne les respectent pas.

Politique de sécurité n°1 :

1. La session principale commence par un *login* et se termine par *quit*
2. au coeur de la session principale on peut effectuer des sessions *ssh* et des sessions *rlogin*
3. on ne peut ouvrir qu'une session *rlogin* à la fois qu'on doit fermer par *exit*
4. on ne peut ouvrir qu'une session *ssh* à la fois qu'on doit fermer par *exit*
5. on ne peut pas avoir simultanément une session *ssh* et une session *rlogin*

Q3. (1 pt) Donnez une expression régulière P_1 qui décrit la politique de sécurité n°1.

Q4. (1 pt) Donnez un automate à états finis P_2 qui décrit la politique de sécurité n°2.

Politique de sécurité n°2 :

1. La session principale commence par un *login* et se termine par *quit*
2. au coeur de la session principale on peut effectuer soit des sessions *ssh*, soit une session *rlogin*
3. on peut ouvrir au plus une session *rlogin* qu'on doit chacune fermer par *exit*
4. on peut ouvrir au plus deux sessions *ssh* à la fois qu'on doit chacune fermer par *exit*
5. il est interdit d'avoir simultanément des sessions *ssh* et *rlogin*.

Q5. (1 pt) Donnez un automate P_3 qui décrit la politique de sécurité n°3.

Politique de sécurité n°3 :

1. La session principale commence par un *login* et se termine par *quit*
2. au coeur de la session principale on peut effectuer uniquement sessions *ssh*
3. on autorise plusieurs sessions *ssh* simultanées.
4. on peut ouvrir autant de session *ssh* qu'on veut mais on devra fermer chacune des sessions par la commande *exit*. Autrement dit si on fait *ssh.ssh.ssh* il faudra ensuite faire *exit.exit.exit*.

Exercice 7.8 Minimisation d'automate (4 pt) (20min)

On considère l'automate suivant :

A	$_i1$	2^a	3	4^a	5	6	7	8	9	10	11	12^a
a	2	2	4	4	7	11	4	12	9	10	4	12
b	3	3	9	5	4	4	7	8	10	9	11	6

Q1. (0.75 pt) Le langage reconnu par l'automate de la question précédente est-il vide ? **Justifiez votre réponse.**

Q2. (2 pt) Minimisez l'automate en justifiant chaque étape de l'algorithme
Les $\frac{2}{3}$ des points sont attribués aux justifications.

Q3. (0.25 pt) Donnez l'automate minimisé sous forme de tableau.

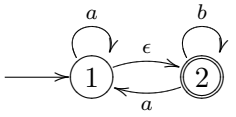
Q4. (1 pt) L'automate de la question précédente reconnaît-il le langage universel $\{a, b\}^*$? **Justifiez votre réponse.**

Exercice 7.9 Questions de cours (7 pt)

Justifiez chacune de vos réponses de façon précise (3 ou 4 lignes suffisent pour démontrez votre réponse). Chaque question est notée de la façon suivante : $\frac{1}{4}$ des points pour la réponse et $\frac{3}{4}$ des points pour la justification.

Élimination des ϵ et déterminisme

Q1. Donnez l'automate obtenu par élimination des ϵ transitions dans l'automate ci-dessous



Q2. L'automate obtenu est-il déterministe ?

Complémentaire, produit et restriction On considère l'alphabet $\Sigma = \{a, b, c\}$

Q3. Qu'est-ce qu'un langage sur l'alphabet Σ ?

Q4. Donnez un automate qui reconnaît le langage Σ

Q5. Quel est le plus grand langage sur l'alphabet Σ ?

Q6. Donnez un automate qui reconnaît le langage complémentaire de Σ

Q7. Complétez la définition de $\Sigma^+ \stackrel{def}{=} \ldots\ldots\ldots\ldots$ et donnez un automate qui reconnaît le langage Σ^+

Q8. Donnez un automate qui reconnaît $\overline{\Sigma^+}$

Q9. Décrivez en une phrase le langage reconnu par l'automate de la question précédente.

Q10. Donnez un automate minimal qui reconnaît ce langage.

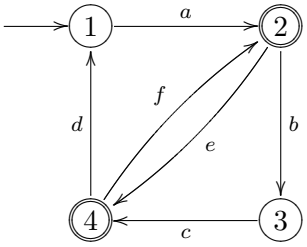
Q11. Donnez un automate qui reconnaît le langage $\Sigma^* \setminus \Sigma$

Q12. Donnez un automate qui reconnaît le complémentaire de ce langage.

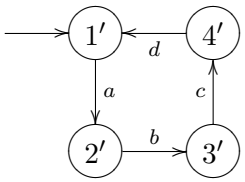
Q13. Décrivez en une phrase le langage reconnu par l'automate de la question précédente.

Q14. Quel est le langage reconnu par le produit des automates A_1 et A_2 ci-dessous.

$A_1 \stackrel{def}{=}$



$A_2 \stackrel{def}{=}$



Grammaires et automates

Q15. Donnez une grammaire de type 3 (c'est-à-dire régulière) qui génère le langage c^*.
On ne demande pas de justification.

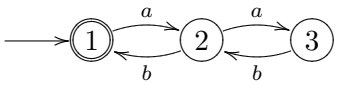
Q16. Donnez une grammaire qui génère le langage $(a + c)^*.b$
On ne demande pas de justification.

Q17. Donnez un automate à pile qui reconnaît le langage des palindromes de la forme $\omega.c^*.R(\omega)$ où $\omega \in \Sigma^*$ avec $\Sigma = \{a, b\}$ et $R(w)$ désigne le mot ω renversé. On ne demande pas de justification.

Q18. Cet automate est-il déterministe ? **Justifiez votre réponse.**

Q19. Donnez une grammaire qui génère le même langage. On ne demande pas de justification.

Q20. Donnez une grammaire qui génère le langage reconnu par l'automate suivant :



Q21. Donnez l'expression régulière correspondant à l'automate précédent. **Détaillez les étapes de résolution.**