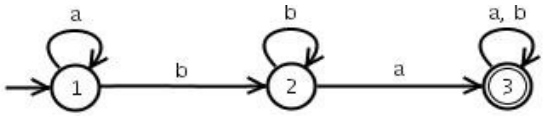
**Langages formels**
**Corrigé – Laboratoire 1**

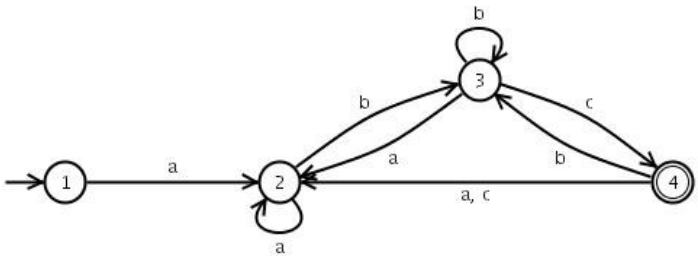
## Exercice 1

a)



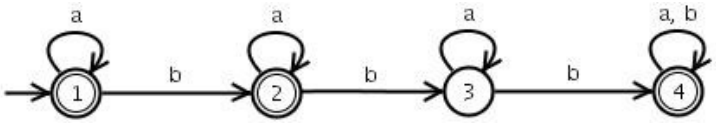

ER : (a+b)*ba(a+b)*

b)



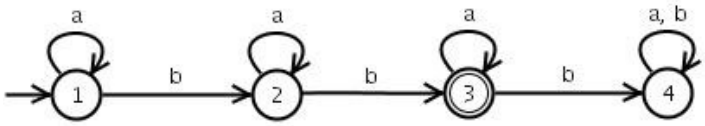

ER: a(a+b+c)*bc

c)




ER: a* + a*ba* + a*ba*ba*b(a+b)*

d)



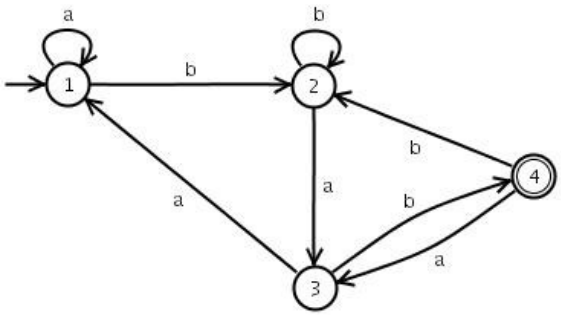

ER: a*ba*ba*

e) Lorsqu'il faut construire un automate pour un langage L qui est décrit comme l'union, l'intersection ou la différence de deux autres langages (nommons les $L_1$ et $L_2$), une méthode formelle permet de trouver aisément l'automate résultant à partir des automates représentants $L_1$ et $L_2$.

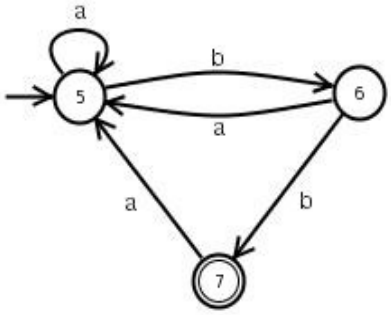
Ici, on peut décrire $L_1$ et $L_2$ ainsi:

- $L_1$ = { m | m ε {a, b}$^*$ et m se termine par ‘bab’ }
- $L_2$ = { m | m ε {a, b}$^*$ et m se termine par ‘bb’ }

L'automate $A_1$ correspondant à $L_1$ est:




L'automate $A_2$ correspondant à $L_2$ est:




Maintenant que l'on connait les 2 automates correspondants aux langages $L_1$ et $L_2$, on est prêt à construire l'automate A correspondant au langage L. Essentiellement, chaque état de A correspondra à un état de $A_1$ et à un état de $A_2$.

L'algorithme permettant de fusionner $A_1$ et $A_2$ afin de créer A est le suivant :

- Créer un premier état de A correspondant à l'état initial de $A_1$ et à l'état initial de $A_2$. Cet état sera l'état initial de A.
- Tant qu'il existe des états de A pour lesquels on n'a pas cherché de transition :
  - Soit E, le prochain état de A qui n'a pas ses transitions.
  - Soit $E_1$, l'état correspondant à E dans l'automate $A_1$.
  - Soit $E_2$, l'état correspondant à E dans l'automate $A_2$.
  - Pour chaque symbole du vocabulaire :
    - Soit s le symbole suivant du vocabulaire.
    - Créer la transition t(E, s) = E', où E' correspond à l'état d'arrivée de la transition $t(E_1, s)$ et de la transition $t(E_2, s)$.
    - Si E' n'existe pas dans A, ajouter E' à A.
    - Ajouter à A la transition t(E, s) = E'.

Cet algorithme écrit de manière formelle peu sembler un peu complexe, mais l'application est en fait assez simple:

1. L'état initial de $A_1$ est l'état '1' et l'état initial de $A_2$ est l'état '5'. On crée donc l'état '1 5' qu'on ajoute au nouvel automate A. L'état '1 5' est l'état initial de A.
2. On a toujours pas trouvé les transitions partant de l'état '1 5'. C'est donc la prochaine étape. Il s'agit d'ajouter une transition pour chaque

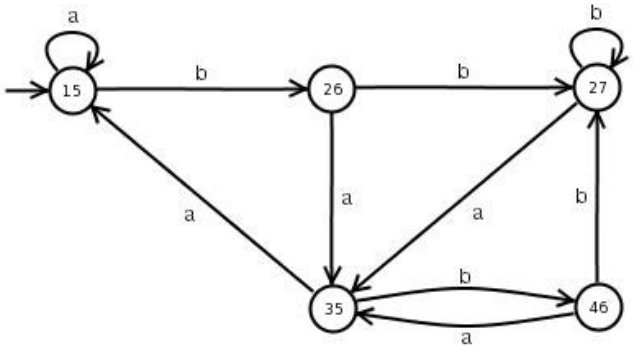
symbole du vocabulaire, c'est-à-dire sur les symboles 'a' et 'b'. Pour ce faire il faut se demander: “Quel est l'état d'arrivée de la transition sur le symbole 'a' à partir de l'état '1' de l'automate $A_1$?” et “Quel est l'état d'arrivée de la transition sur le symbole 'a' à partir de l'état '5' de l'automate $A_2$?”. La réponse à ces deux questions permettra de trouver l'état d'arrivée de la transition sur le symbole 'a' à partir de l'état '1 5' de l'automate A. Dans notre cas, la réponse à la première question est '1' et la réponse à la deuxième question est '5'. L'état d'arrivée de la transition sur 'a' à partir de l'état '1 5' de A est l'état '1 5'. Notez que cette transition boucle donc sur un même état.

Maintenant, il faut aussi trouver l'état d'arrivée de la transition sur le symbole 'b'. On se pose à nouveau les mêmes questions, mais en utilisant le symbole 'b'. Cette fois, les réponses sont '2' et '6'. L'état d'arrivée de la transition sur 'b' à partir de l'état '1 5' est donc l'état '2 6'. L'état '2 6' n'existe pas dans A. Il faut donc l'ajouter, en plus de la transition t('1 5', b) = '2 6'.

3. Comme on vient de créer un nouvel état '2 6' pour lequel on a toujours pas cherché les transitions sortantes, il faut refaire le même procédé et se poser les mêmes questions, mais en utilisant l'état '2' de $A_1$ et l'état '6' de $A_2$.

4. On continue ainsi jusqu'à obtenir un automate A pour lequel tous les états ont leur transitions sur chaque symbole.

L'automate que l'on obtient est le suivant:




Cet automate est presque complet, seulement il lui manque ses états finaux. Pour trouver les états finaux, il suffit d'appliquer la règle suivante:

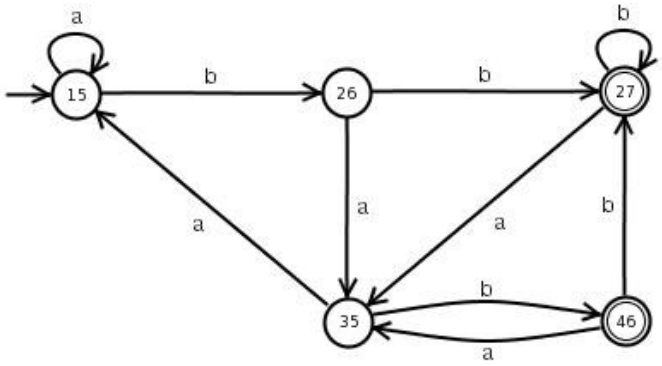
- Si l'on cherche à créer un automate acceptant les mots de $L_1$ ou $L_2$ (c-a-d $L_1 \cup L_2$), chaque état de A correspondant à un état final de $A_1$ ou à un état final de $A_2$ devient un état final. Ici, l'état '2 7' et l'état '4 6' serait dans cette situation.

- Si l'on cherche à créer un automate acceptant les mots de $L_1$ et $L_2$ (c-a-d $L_1 \cap L_2$), chaque état de A correspondant à un état final de $A_1$ et à un état final de $A_2$ devient un état final. Ici, aucun état n'est dans cette situation. (Note: s'il y avait un état '4 7', il serait final).

- Si l'on cherche à créer un automate acceptant les mots de $L_1$ sans les mots de $L_2$ (c-a-d $L_1 - L_2$), chaque état de A correspondant à la fois à un état final de $A_1$ et à un état non-final de

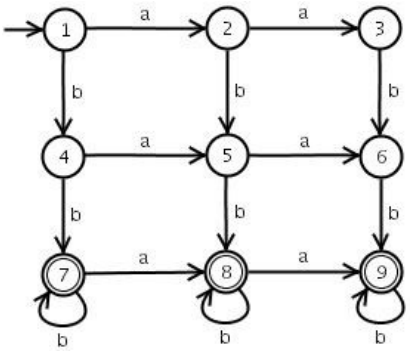
$A_2$ devient un état final. Ici, seul l'état '4 6' serait dans cette situation.

Comme l'on cherche à trouver l'automate correspondant à $L_1 \cup L_2$, l'automate final recherché est le suivant:




ER : (a+b)*bab + (a+b)*bb = (a+b)*(bab + bb)

f)




ER : L'expression régulière pour cet automate est assez complexe et ne sera pas donnée. Cependant, une méthode possible pour trouver l'expression régulière serait de donner l'expression régulière pour chaque chemin menant à un état final, i.e.: bbb* + bbb*ab* + ...

**Exercice 2**

La déterminisation consiste à transformer un automate fini non-déterministe en une version déterministe équivalente (c-a-d un AFD qui accèpte le même langage). Un automate est considéré non-déterministe s'il possède l'une ou l'autre des propriétés suivantes:

- Contient au moins une transition sur ε
- Contient au moins un état à partir duquel sort 2 transitions sur le même symbole.

On donne l'appelation « non-déterministe » à ce type d'automate, car ces 2 propriétés font en sorte que dans certaines circonstances, on a un choix arbitraire à prendre sur la prochaine action à entreprendre.

Pour l'automate de l'exercice #1, celui-ci contient 2 transitions sur ε et 2 transitions sur le symbole 'a' à partir de l'état 3. Il est donc non-déterministe.

Comme un automate non-déterministe permet de faire certains choix sur les transitions à prendre, il est donc possible, pour une même séquence de symboles d'entrée, de se retrouver dans divers états différents. Le processus de déterminisation consiste à prendre chaque ensemble d'état qu'il est possible d'atteindre à un moment précis dans l'automate non-déterministe et de créer pour chacun d'entre eux, un état qui les représente dans l'automate déterministe.

La première étape du processus de déterminisation consiste donc à trouver quels sont les états qu'il est possible d'atteindre avant même lire un premier symbole. Cet ensemble contient nécessairement l'état initial *i* de l'AFN, ainsi que tous les états atteignables grâce à des transitions sur ε à partir de i. Cet ensemble d'état devient l'état initial de l'AFD que nous cherchons à construire. Nous ajoutons cet état à la liste E' des états de l'AFD.

À partir de cet ensemble initial d'états, il faut trouver les ensembles d'états que l'on peut atteindre sur chaque symbole du vocabulaire:

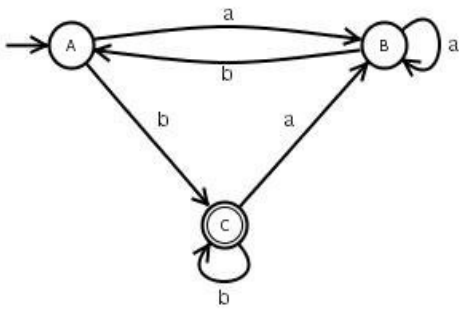
- Soit $E_1$, l'ensemble d'états de départ.
- Pour chaque symbole s du vocabulaire :
  - Soit $E_2$ = {} un ensemble d'états initialement vide.
  - Pour chaque états $e_1$ ε $E_1$ :
    - Pour chaque transition t($e_1$, s) de l'AFN :
      - Soit $e_2$ = t($e_1$, s). //L'état d'arrivée de la transition.
      - Ajouter $e_2$ à $E_2$.
      - Ajouter aussi à $E_2$ tous les états qu'il est possible d'atteindre à l'aide d'une ou plusieurs transitions successives sur ε, à partir de $e_2$.
  - Ajouter $E_2$ à la liste E' des états de l'AFD.
  - Ajouter t($E_1$, s) = $E_2$ à la liste des transitions de l'AFD.

On répète ce processus tant qu'il y a des états dans E' qui n'ont pas été pris en compte. Aussi, n'oubliez pas que par définition, un ensemble ne peut contenir plus d'une fois chaque élément!

Voici l'AFD résultant pour l'AFN du problème, obtenu en utilisant la méthode décrite ci-haut:

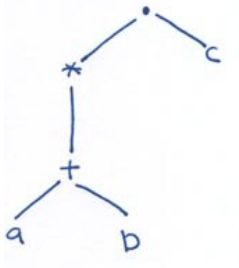
A' = (E', V', i', t', F), où
E' = {A, B, C} *//On ne connait la liste des états qu'une fois le processus terminé*
V' = {a, b}
i' = {1, 3} = A
t' : t'(A, a) = {3, 4} = B
t'(A, b) = {1, 2, 3, 4} = C
t'(B, a) = {3, 4} = B
t'(B, b) = {1, 3} = A
t'(C, a) = {3, 4} = B
t'(C, b) = {1, 2, 3, 4} = C
F = {C} *//Les états finaux sont ceux contenant au moins un état final de l'AFN.*

Graphiquement, cet automate déterministe est représenté ainsi:

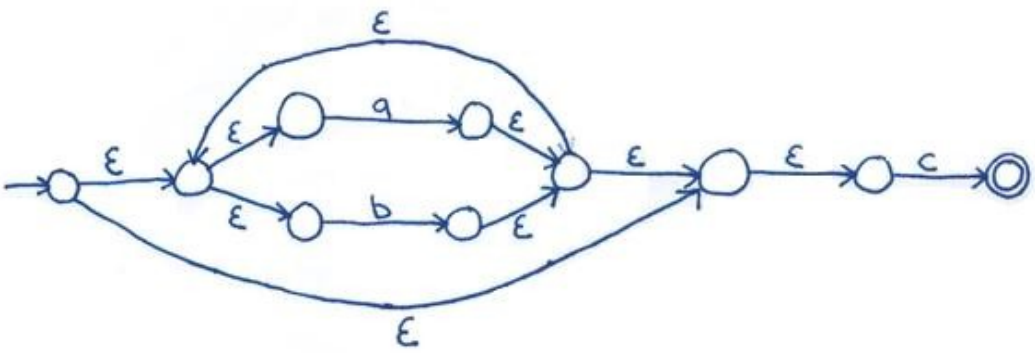



**Exercice 3**

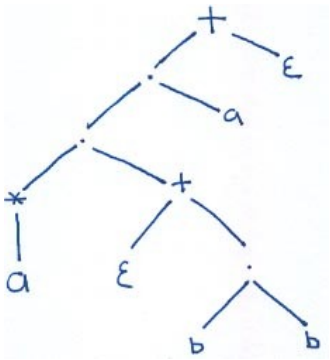
a) Arbre de décomposition :



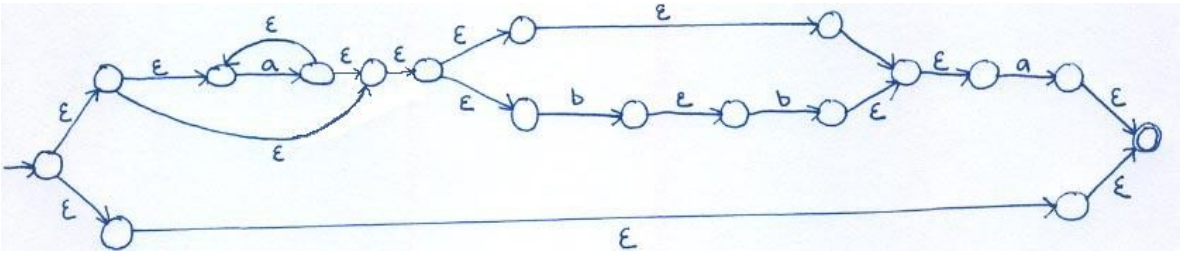

Automate :




b) Arbre de décomposition :



Automate :




**Exercice 4**

Pour minimiser un automate, il faut tout d'abord :

- S'assurer qu'il est déterministe.
- Éliminer les états qui ne sont pas atteignables à partir de l'état initial, ainsi que les transitions adjacentes à ces états.
- Éliminer les états à partir desquels on ne peut pas atteindre un état final, ainsi que les transitions adjacentes à ces états.

Dans le cas de l'automate à minimiser pour cet exercice, celui-ci est déjà déterministe. Les états 10 et 11 sont par contre inatteignables à partir de l'état initial, et les états 12 et 13 ne mènent à aucun état final. Nous pouvons donc éliminer tout de suite ces états.

L'étape suivante est de construire ce tableau, en regroupant les états initiaux dans une classe, et les états finaux dans une seconde classe :

| Classe | tat | a | b |
|---|---|---|---|
| A | 1 | A | {} |
| A | 2 | A | A |
| A | 3 | A | A |
| A | 4 | A | A |
| A | 5 | A | B |
| A | 6 | A | {} |
| A | 7 | A | B |
| B | 8 | A | B |
| B | 9 | A | B |

Comme les états de la classe A n'ont pas tous les mêmes états d'arrivée pour des transitions sur les mêmes symboles, il faut séparer cette classe en 2 nouvelles classes. On continue ainsi jusqu'à ce qu'on obtienne un tableau où chaque classe respecte cette condition :

| Classe | tat | a | b |
|---|---|---|---|
| A | 1 | B | {} |
| A | 6 | C | {} |
| B | 2 | B | A |
| B | 3 | C | B |
| B | 4 | C | B |
| C | 5 | C | D |
| C | 7 | C | D |
| D | 8 | C | D |
| D | 9 | C | D |

| Classe | tat | a | b |
|---|---|---|---|
| A | 1 | B | {} |
| B | 2 | C | E |
| C | 3 | D | C |
| C | 4 | D | C |
| D | 5 | D | F |
| D | 7 | D | F |
| E | 6 | D | {} |
| F | 8 | D | F |
| F | 9 | D | F |

Nous pouvons maintenant créer, à partir du dernier tableau, l'automate minimal acceptant le même langage que l'automate de l'énoncé. Chaque classe correspond à un état de l'automate minimal. L'état minimal est la classe contenant l'état initial de l'automate de départ. L'état final est la classe contenant les états finaux de l'automate initial. Si cette classe avait été séparée en deux ou plusieurs sous-classes durant le processus, alors chacune de ces sous-classes correspondrait à un état final de l'automate minimal.

Automate minimal :