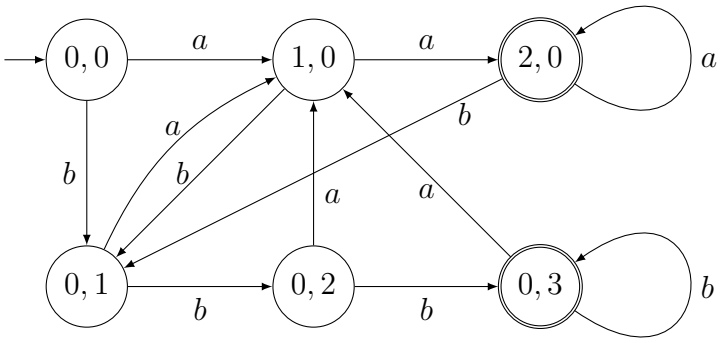
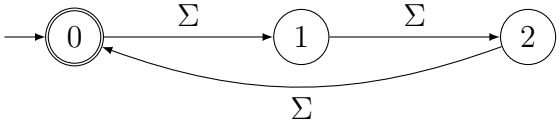
1 Automates finis déterministes

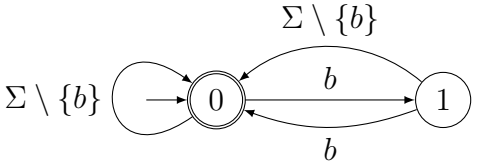
1) a) mots terminant par a^2 ou b^3, pour s'aider, on code les dernières lettres lues (nbre de a,nbre de b) :



b) mots de longueur un multiple de 3 :



c) mots terminant par un nombre pair de b :



2) a)

```
let a = {alphabet = ['a';'b'] ;
         nbre_etats = 3;
         transitions = (let t i x = match i,x with
                                |0,'a' -> 1
                                |0, 'b' -> 2
                                |1, 'a' -> -1
                                |1, 'b' -> 0
                                |2, 'a' -> 1
                                |2, 'b' -> -1
                                |_ -> -1
                        in t);
         etats_acceptation = [1;2]};;
```

b) Pour lire un mot dans l'automate on parcourt le mot et on regarde l'état courant. Si on peut lire le mot sans blocage, on regarde alors si l'état est dans la liste des états acceptants.

```
let rec appartient l x =
 match l with
  | [] -> false
  | t :: q -> (t = x) || appartient q x;;


let lecture a l =
 let rec avance li et = match li with
   | [] -> appartient a.etats_acceptation et
   | x :: q -> let newet = a.transitions et x
     in (newet <> (- 1)) && (avance q newet)
 in
 avance l 0;;
```

c) Pour compléter un automate on ajoute un état puit.

```
let completude a =
let n = a.nbre_etats in
let newtrans i x =
 if (i = n) || (a.transitions i x = - 1) then n else a.transitions i x
in
{alphabet = a.alphabet; nbre_etats = a.nbre_etats + 1;
 transitions = newtrans; etats_acceptation = a.etats_acceptation};;
```

d) Pour déterminer le complémentaire, **on complete d'abord l'automate**. ll suffit ensuite de créer la liste des états qui n'était pas acceptants.

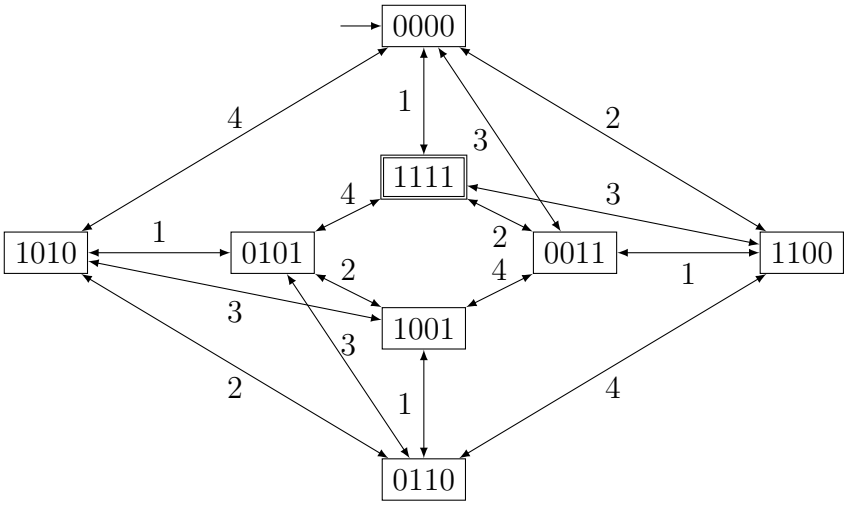
```
let complementaire a =
 let ac = completude a in
 let l = ref [] in
 for i = 0 to (ac.nbre_etats - 1) do
  if not (appartient ac.etats_acceptation i) then l := i :: !l
 done;
 {alphabet = ac.alphabet; nbre_etats = ac.nbre_etats;
  transitions = ac.transitions; etats_acceptation = !l};;
```

e) Pour savoir s'il existe un mot dans le langage, on parcourt l'automate et on regarde si un état acceptant est accessible.

```
let estNonVide a =
 let ac = completude a in
 let b = ref false in
 let visit = Array.make (ac.nbre_etats) false in
 let rec parcours etat =
  if not (visit.(etat)) then
   (visit.(etat) <- true;
    if appartient (ac.etats_acceptation) etat then b := true;
    parcours (ac.transitions etat 'a');
    parcours (ac.transitions etat 'b')
   )
 in
 parcours 0;
 !b;;
```

f) $(a|ba)((ba)^*(bba)^*)^*$

3) On dessine le graphe des états possibles reliés par les arêtes étiquetées par les interrupteurs qui permettent de passer d'un état à l'autre. On peut coder les configurations avec 4 bits : le premier représente l'état des projecteurs dont le numéro est de la forme $3k+1$ et pair, le second bit représente l'état des projecteurs dont le numéro n'est pas de la forme $3k+1$ mais est pair, le troisième bit représente l'état des projecteurs dont le numéro est de la forme $3k+1$ et impair, et le dernier bit représente l'état des projecteurs dont le numéro n'est pas de la forme $3k+1$ et est impair.

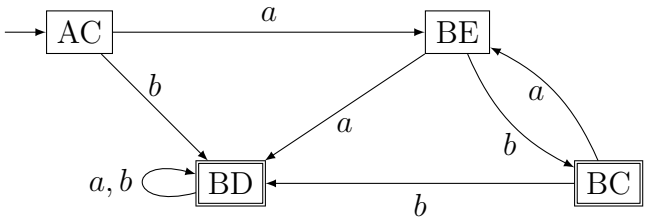


L'ensemble des combinaisons permettant d'éclairer tous les projecteurs en même temps, est exactement le langage reconnu par l'automate que l'on a défini, où l'état initial est l'état 0000 et l'état acceptant est l'état où tout est allumé, c'est-à-dire 1111.

4) a) i) $L_1 = (a|b).(a|b)^*$

ii) $L_2 = (ab)^*a$

b) i)



ii) $(L_1 \setminus L_2) \cup (L_2 \setminus L_1)$

c) Soit (s,t) un état de $\mathcal{A}_1 \oplus \mathcal{A}_2$, alors $(s,t) \in Q_1 \times Q_2$. Soit $a \in X$, alors comme $\mathcal{A}_1$ est complet, il existe $s' \in Q_1$ tq $\delta_1(s,a) = s'$. De même, comme $\mathcal{A}_2$ est complet, il existe $t' \in Q_2$ tq $\delta_2(t,a) = t'$. Par définition de $\mathcal{A}_1 \oplus \mathcal{A}_2$, on a alors $\delta_{\mathcal{A}_1 \oplus \mathcal{A}_2}((s,t),a) = (s',t')$. Il y a donc bien une arête sortante de l'état (s,t) pour la lettre a, et cela pour tout état (s,t) de $\mathcal{A}_1 \oplus \mathcal{A}_2$ et pour toute lettre de l'alphabet X. L'automate $\mathcal{A}_1 \oplus \mathcal{A}_2$ est donc bien complet.

d) Prouvons cette propriété par récurrence sur la longueur du mot m.

(I) Par définition, $\forall q_1 \in Q_1, \delta_1^*(q_1,\varepsilon) = q_1$, $\forall q_2 \in Q_2, \delta_2^*(q_2,\varepsilon) = q_2$ et $\forall q_1 \in Q_1, \forall q_2 \in Q_2, \delta^*((q_1,q_2),\varepsilon) = (q_1,q_2)$. Donc, $\forall q_1 \in Q_1, \forall q_2 \in Q_2$, on a bien $\delta^*((q_1,q_2),\varepsilon) = (q_1,q_2) = \delta_1^*(q_1,\varepsilon), \delta_2^*(q_2,\varepsilon)$.

(H) Soit $n \in \mathbb{N}$, supposons que la propriété est vraie pour tout mot de longueur n. Soit m un mot de longueur $n+1$ alors m est de la forme $m'.x$ où m' est un mot de longueur n et x une lettre de l'alphabet X.
Par définition, $\forall q_1 \in Q_1, \delta_1^*(q_1,\, m'.x) = \delta_1(\delta_1^*(q_1,m'),\, x)$, $\forall q_2 \in Q_2, \delta_2^*(q_2,\, m'.x) = \delta_2(\delta_2^*(q_2,m'),\, x)$ et $\forall q_1 \in Q_1, \forall q_2 \in Q_2, \delta^*((q_1,q_2),\, m'.x) = \delta(\delta^*((q_1,q_2),m'),\, x)$.

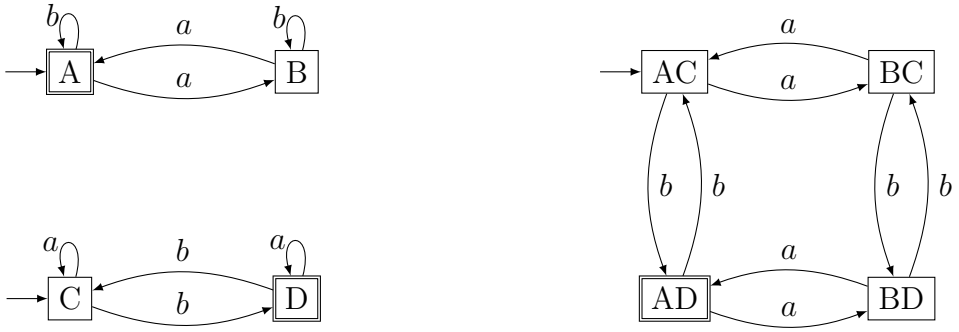
Or, par hypothèse de récurrence, $\delta^*((q_1,q_2),m') = \delta_1^*(q_1,m'),\delta_2^*(q_2,m')$. Ainsi, $\delta(\delta^*((q_1,q_2),m'),\, x) = \delta((\delta_1^*(q_1,m'),\delta_2^*(q_2,m')),\, x)$, et enfin par définition de δ, $\delta((\delta_1^*(q_1,m'),\delta_2^*(q_2,m')),\, x) = (\delta_1(\delta_1^*(q_1,m'),\, x),\delta_2(\delta_2^*(q_2,m'),\, x))$. En conséquence, on a bien $\delta^*((q_1,q_2),m') = (\delta_1^*(q_1,m'.x),\delta_2^*(q_1,m'.x))$.

(C) La propriété est initialisée et héréditaire, donc par le principe de récurrence, elle est vraie pour tout mot m.

e) Un mot m est reconnu par $\mathcal{A}_1$ si et seulement si $\delta_1^*(i_1,m) \in T_1$, de même m est reconnu par $\mathcal{A}_2$ si et seulement si $\delta_2^*(i_2,m) \in T_2$. De plus, m est reconnu par $\mathcal{A}_1 \oplus \mathcal{A}_2$, si et seulement si $\delta^*((i_1,i_2),m) in (T_1 \times (Q_2 \setminus T_2)) \cup ((Q_1 \setminus T_1) \times T_2)$.

Or $\delta^*((i_1,i_2),m) = (\delta_1^*(i_1,m),\delta_2^*(i_2,m))$. Ainsi, m est reconnu par $\mathcal{A}_1 \oplus \mathcal{A}_2$ si et seulement s'il est "reconnu par $\mathcal{A}_1$ mais pas par $\mathcal{A}_2$" ou "reconnu par $\mathcal{A}_2$ mais pas par $\mathcal{A}_1$".

f) On peut construire l'automate $\mathcal{A}_1$ qui reconnaît l'ensemble des mots contenant un nombre pair de a et l'automatque $\mathcal{A}_2$ qui reconnaît l'ensemble des mots contenant un nombre impair de b. On peut alors construire le produit $\mathcal{A}_1 \oplus \mathcal{A}_2$. Pour obtenir un automate reconnaissant l'ensemble des mots souhaité, il suffit de modifier l'ensemble des états acceptants en prenant comme seul état acceptant le couple des deux états acceptants de $\mathcal{A}_1$ et $\mathcal{A}_2$. On a ainsi construit l'intersection des deux ensembles de mots.



5) Appuyons nous sur les notations de l'exercice précédent. Soient $\tilde{\mathcal{A}}_1$ et $\tilde{\mathcal{A}}_2$ les complétés de respectivement $\mathcal{A}_1$ et $\mathcal{A}_2$. Soit $\mathcal{A}$, l'automate $\tilde{\mathcal{A}}_1 \oplus \tilde{\mathcal{A}}_2$ dont les états acceptants sont définis comme suit : $(T_1 \times \tilde{Q_2}) \cup (\tilde{Q_1} \times T_2)$ où $\tilde{Q_1}$ et $\tilde{Q_2}$ sont les ensembles des états de $\tilde{\mathcal{A}}_1$ et $\tilde{\mathcal{A}}_2$

Remarque : on n'introduit pas fait de nouvelle notation pour les ensembles des états acceptants de $\tilde{\mathcal{A}}_1$ et $\tilde{\mathcal{A}}_2$ car l'ensemble des états acceptants n'est pas modifié par l'opération de complétion.

Le nombre d'états est égal à $(n_1+1)*(n_2+1)$, ce qui est bien polynomial par rapport à n_1 et n_2. Les "+1" correspond aux puits ajoutés lors de la complétion.

6) a) Pour décider si $L_1 \subset L_2$, on peut construire à nouveau le produit des automates M_1 et M_2, cette fois, on définit les états acceptants comme suite : $T_1 \times (Q_2 \setminus T_2)$. Le langage de cet automate est vide si et seulement si $L_1 \subset L_2$.

b) $L_1 = L_2$ si et seulement si $L_1 \subset L_2$ et $L_2 \subset L_1$. Mais on peut également tester simplement la vacuité du langage de l'automate $M_1 \oplus M_2$.

Ces deux algorithmes sont bien polynomiaux car l'automate produit est de taille $n_1 * n_2$ où n_1 et n_2 sont le nombre d'états de M_1 et M_2.