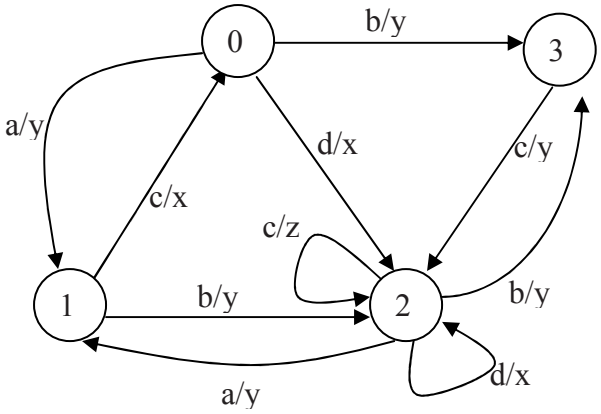
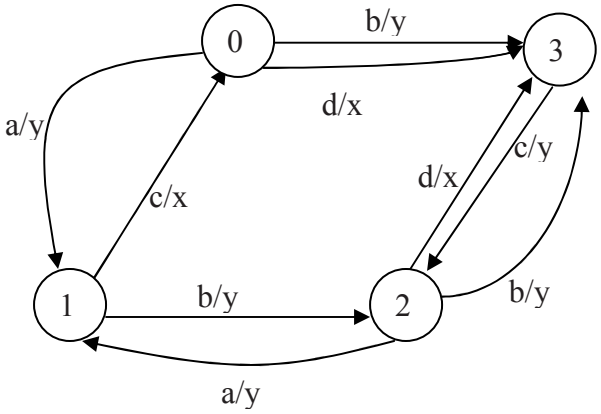


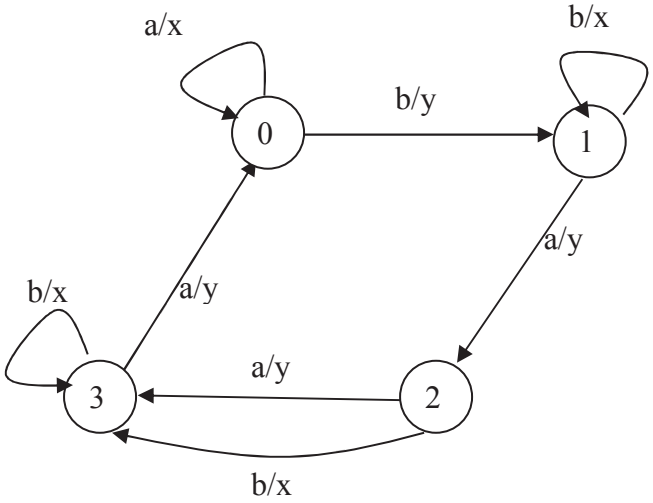
We consider an FSM with '0' as the initial state, the input set ={a,b,c,d} and output set={x,y,z}.

Using the (P)UIO strategy, write a test sequence for the transition (1,c/x,0).



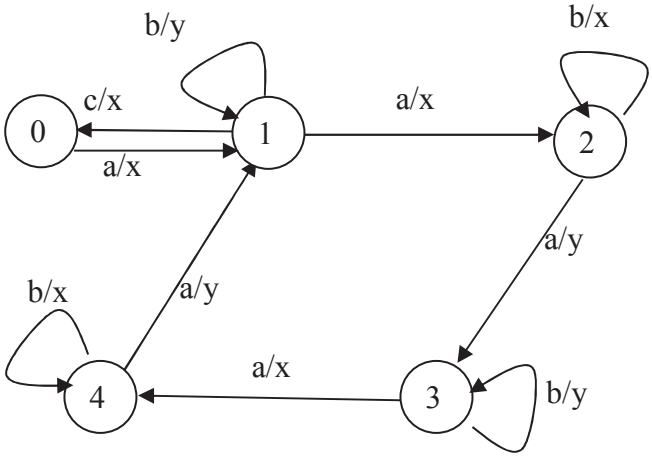
We consider an FSM with '0' as the initial state, the input set ={a,b,c,d} and output set={x,y}.

Using the UIO strategy, write a test sequence for the transition (1,c/x,0).



We consider an FSM with '0' as the initial state, the input set ={a,b} and output set={x,y}.

Using the W strategy, write a test sequence for the transition (3,a/y,0).

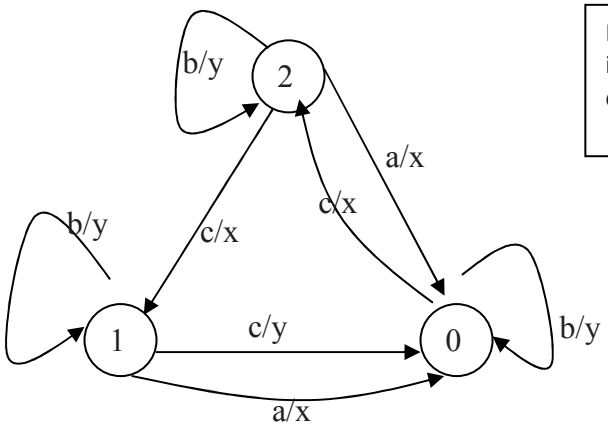


We consider an FSM with '0' as the initial state, the input set ={a,b,c} and output set={x,y}.

Using the (P)UIO strategy, write a test sequence for the transition (3,a/y,0).

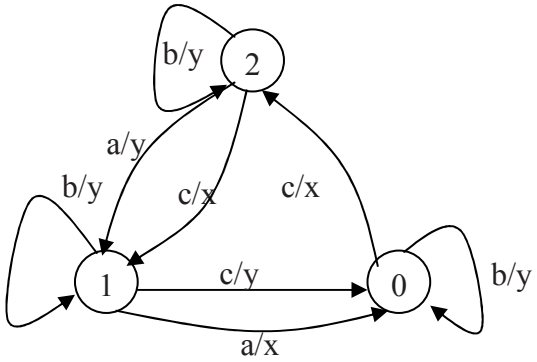
Test Generation of communication protocols

a- Consider the following EFSM representing the specification of a fill-in buffer protocol. Define the W and (P)UIO to differentiate each state.



Initial state: '0'
input set ={a,b,c}
output set={x,y}.

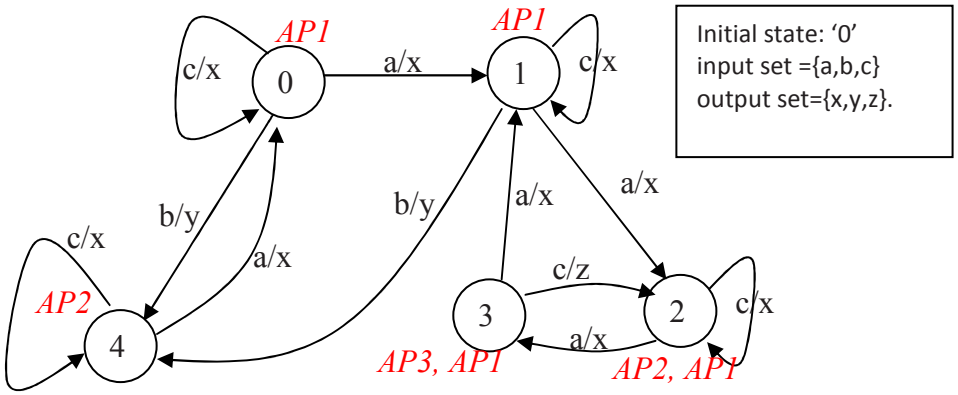
b- An implementation of this protocol has been realized and a reverse engineering method provides the following specification:



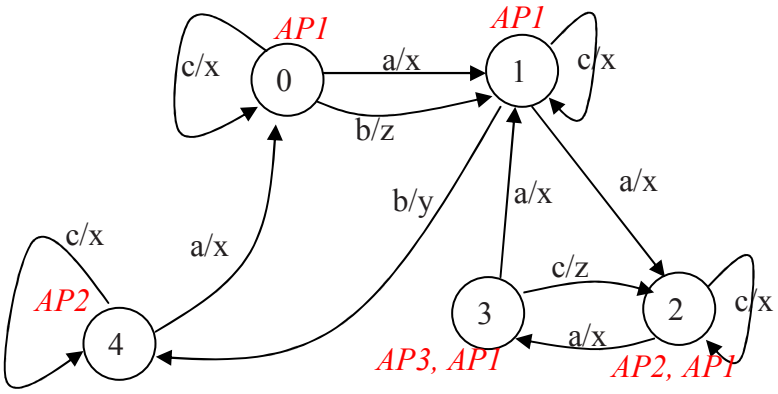
We note that there is an implementation error.
Let propose a method inspired by question a- to detect this implementation error.
A test sequence will be developed and executed on the IUT.

5. Vérification et Techniques de Dérivation de Tests de protocoles de communication.

a. Considérez l'EFSM suivante représentant la spécification d'un protocole non défini. Trouvez un ensemble W et les (P)UIO pour différentier chaque état. (NB : les mots en italique étiquetés aux états sont des propositions atomiques utilisées à la question c-).



b. Une implémentation du protocole a été développée et une méthode de reverse engineering utilisée pour fournir le modèle d'implémentation:



Nous notons une faute dans l'implémentation. Proposez deux méthodes inspirées de la question précédente pour mettre en évidence cette faute.
Des séquences de test seront développées et exécutées sur l'IUT. Vous en donnerez les verdicts.

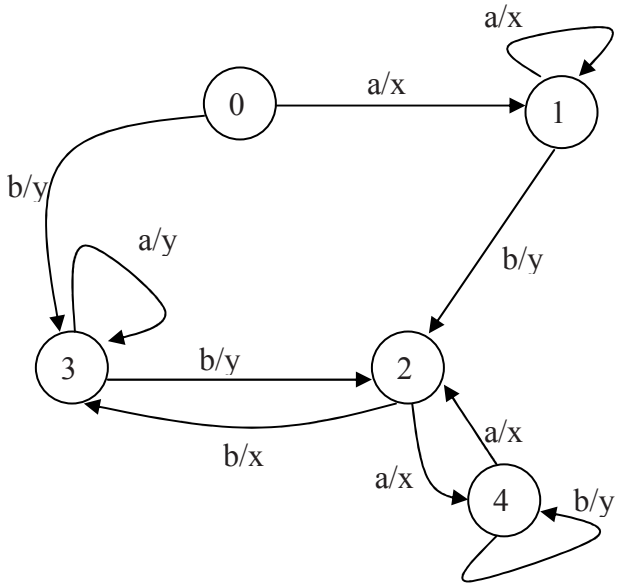
c. Vérification de la spécification. Nous souhaitons maintenant vérifier la présence de propriétés fonctionnelle dans la spécification. Pour cela, des propositions atomiques sont utilisées (en italique dans la spécification).

Voici les propriétés visées:

1. « Si AP2 est vraie, alors dans le futur AP1 sera vraie. »
2. « Si AP1 et AP3 sont vraies, AP3 ne sera plus vraie dans le prochain état. »
3. « AP1 est vraie jusqu'à ce que AP2 soit vraie. »
4. « Dans tous les cas, si AP1 est vraie, AP2 sera vraie dans le futur. »
5. « Dans tous les cas, si AP1 est vraie, AP1 sera vraie dans le prochain état. »

En utilisant le **langage CTL**, exprimer ces propriétés et vérifier, en justifiant, si elles sont vraies ou fausses dans la spécification.

Active testing.

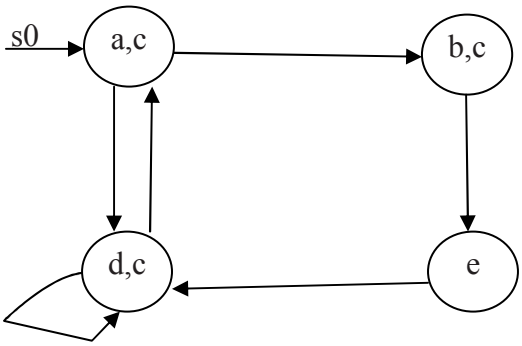


Considérons une FSM avec '0' comme état initial, les inputs sont {a,b} et les outputs {x,y}.

a) En utilisant la stratégie (P)UIO, écrivez une séquence de test pour la transition (1,a/x,1).
b) Supposons que nous appliquons une séquence d'inputs à une IUT et que nous obtenons la séquence d'outputs : NULL,x,x,y,x.
   Cette séquence peut-elle être considérée comme une séquence de test?
   Si oui, que peut-on en déduire ?

Nous considérons l'automate M représenté ci-dessous.
Dites si l'automate satisfait les formules CTL suivantes. Vous devez justifier vos réponses par l'exécution de traces ou un contre exemple.

1) M,s0 |= AGc
2) M,s0 |= EGc
3) M,s0 |= AX(b=>AFd)
4) M,s0 |= EXe
5) M,s0 |= EcUa
6) M,s0 |= AGX(cUe)
7) M,s0 |= E(e=>EXe)



Automaton M