Tests de machines à états (boîte grise)

- **Introduction aux FSM**
- Tests de conformité

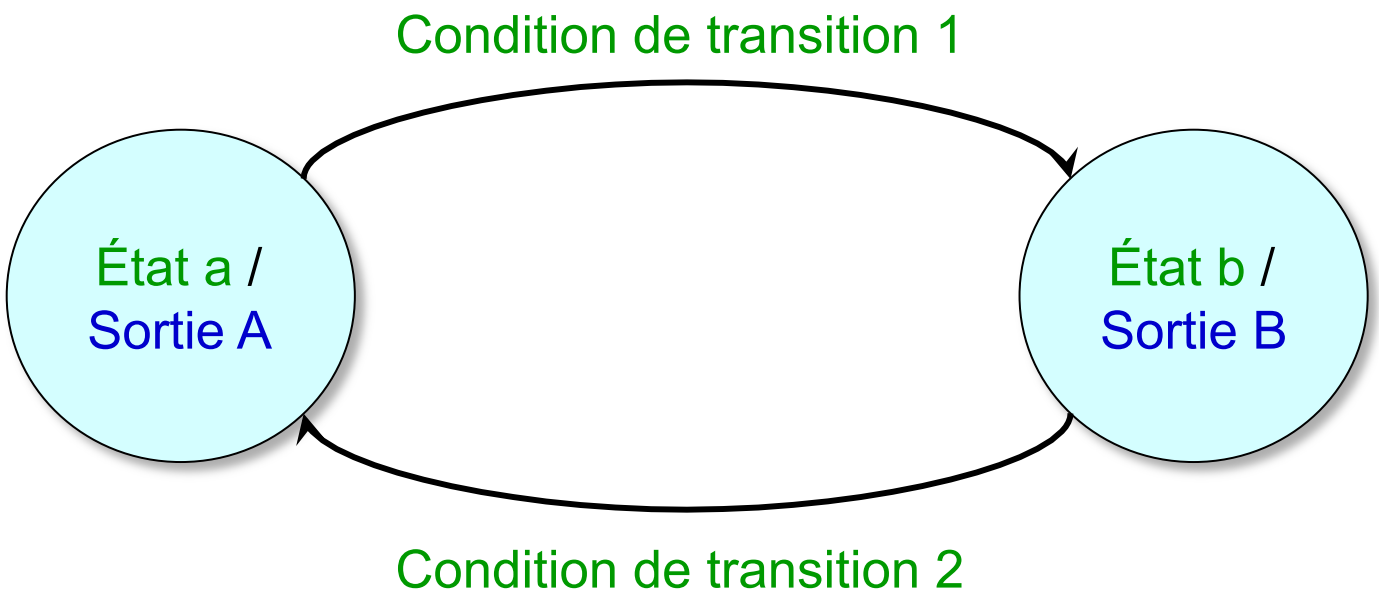
Définition FSM: Deux philosophies …

- Les sorties associées aux états :
  - Machine de Moore

- Les sorties associées aux transitions :
  - Machine Mealy

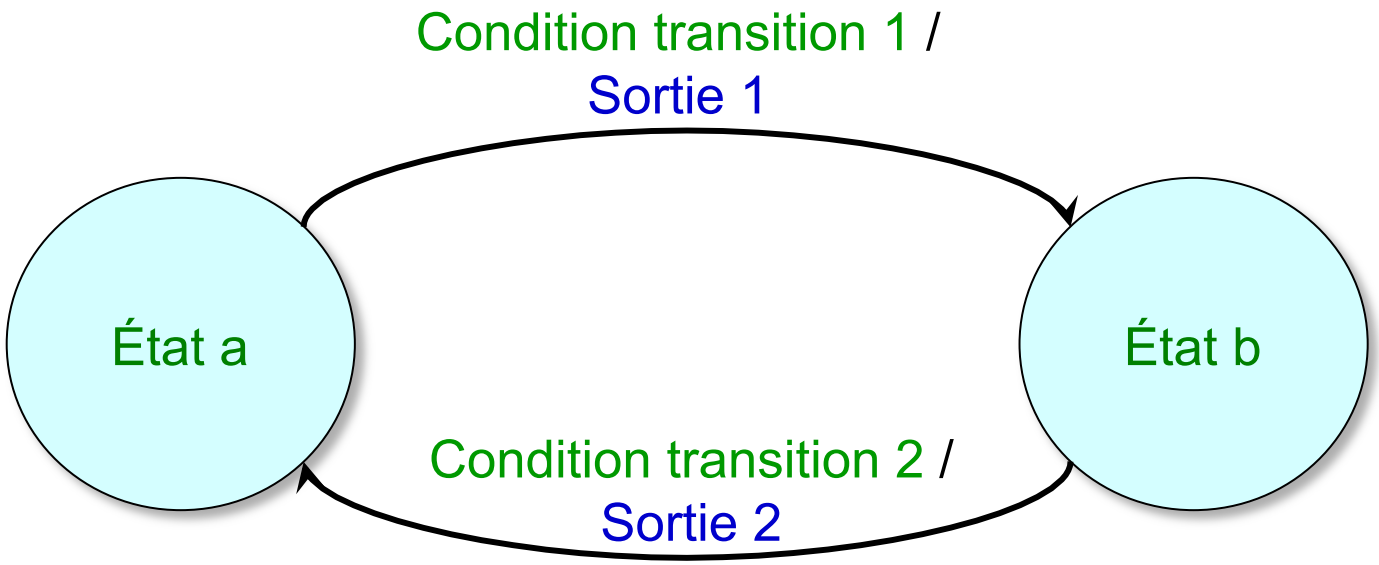
Définitions: Machine de Moore

- Décrit les sorties comme des affirmations concurrentes dépendant de l'état seulement.



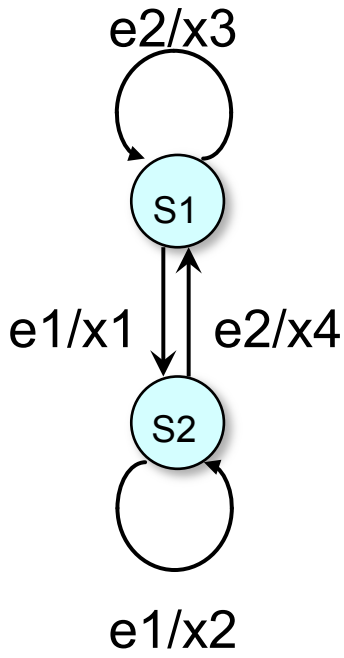
Définitions: Machine de Mealy

- Décrit les sorties comme des affirmations concurrentes dépendant de l'état et des sorties.



***Représentation**: Diagrammes d'état transition*

- Utilisé pour représenter visuellement une FSM.
- L'emphase est sur l'identification des états et des transitions possibles.
- Les cercles représentent les états.
- Les flèches représentent les transitions :
  - e_i sont les entrées
  - x_i sont les sorties



Représentation: Tableaux FSM

- Toutes les entrées, les états, les transitions d'état et les sorties d'une FSM peuvent être listés dans un format tableau.

Événement d'entrées	États courants	Prochains états	Sorties
e1	S1	S2	x1
e1	S2	S2	x2
e2	S1	S1	x3
e2	S2	S1	x4

- Chacune des lignes du tableau est une combinaison unique d'entrées et d'état courant.
- Pour une définition complète de la FSM, toutes les combinaisons entrée/état courant devront être fournies.
- Le tableau est une autre façon pour représenter une FSM avec une emphase sur l'exploration de toutes les combinaisons entrée/état.

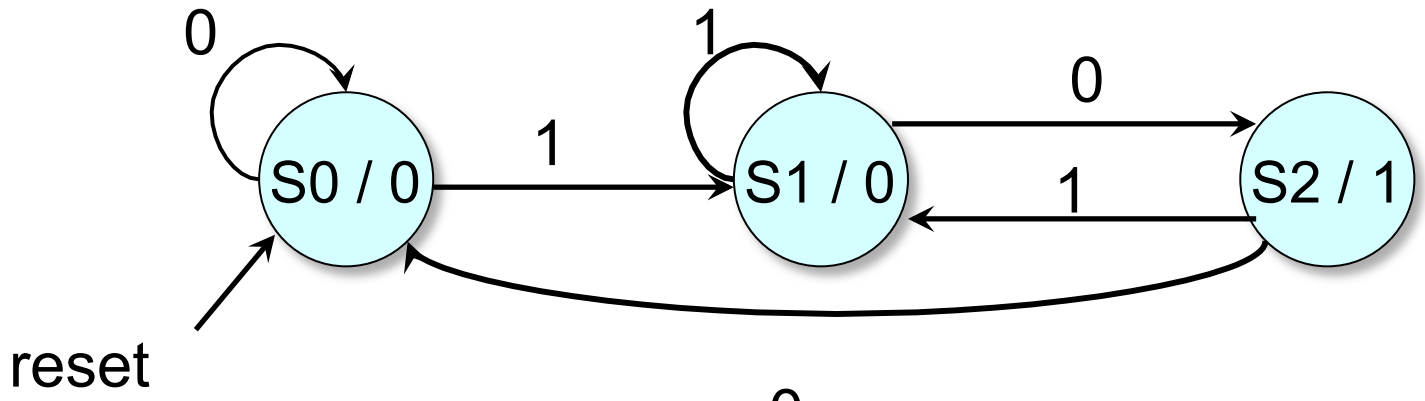
***Représentation**: Tableaux (forme compacte)*

État / entrée	S1	S2
e1	X1, S2	X2, S2
e2	X3, S1	X4, S1

- Ici, la ligne du haut du tableau présente une liste de tous les états alors que la première colonne présente une liste de tous les événements d'entrées.
- La cellule du tableau à l'intersection représente la fonction de transition pour l'état étant donnée une entrée.
- Chacune des cellules contient une liste de sorties et le prochain état.

Exemples: FSM Moore

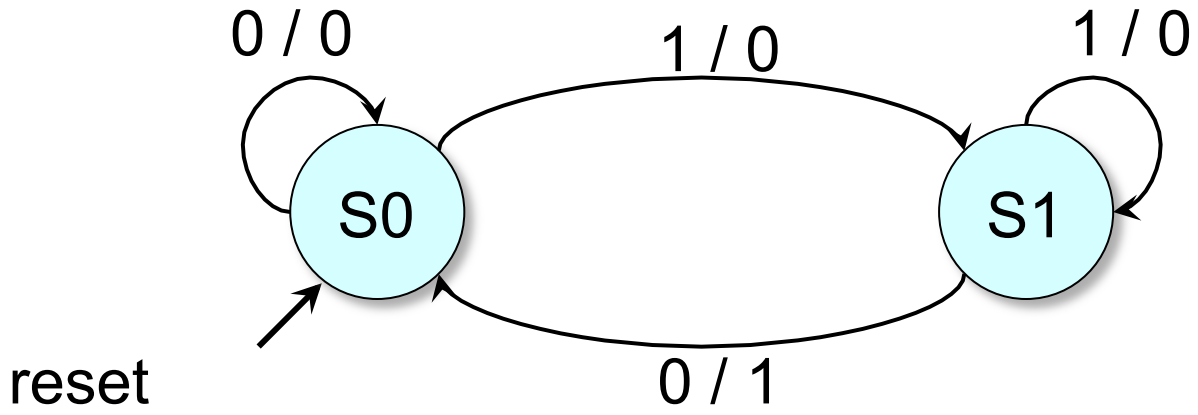
- La FSM Moore qui reconnaît la séquence 10



Signification des états :	S0: Aucun élément de la séquence n'est observé	S1: "1" observé	S2: "10" observé

Exemples: FSM Mealy

- La FSM Mealy FSM qui reconnaît la séquence 10.

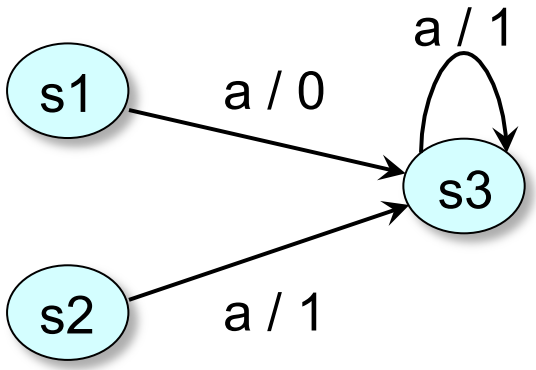


Signification des états :	S0: Aucun élément de la séquence n'est observé	S1: "1" observé

Concepts: Équivalence d'états

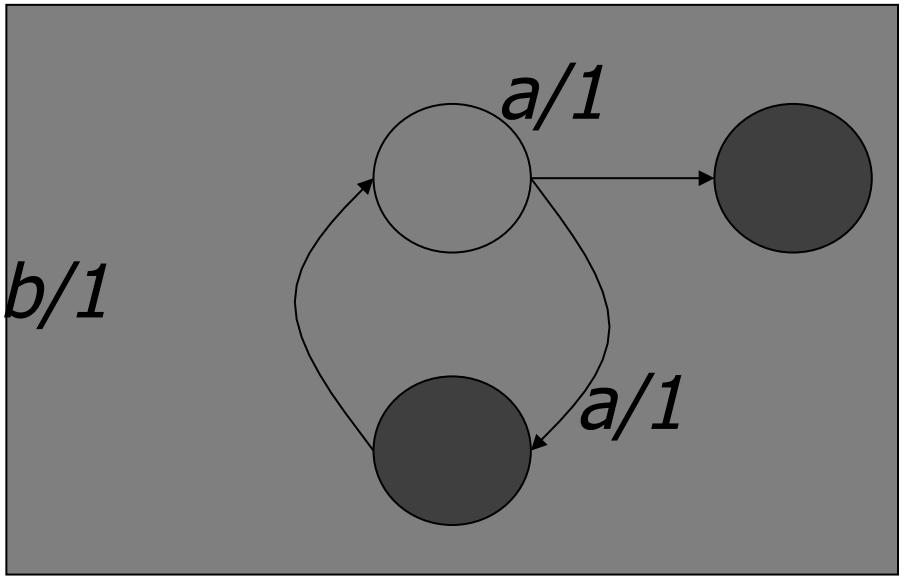
- Deux états sont équivalents si
  - pour n'importe quelle séquence d'entrées, les séquences de sorties des deux états sont les mêmes.

s1 et s2 sont-ils équivalents ?
s2 et s3 sont-ils équivalents ?



- Un automate qui n'a pas deux états équivalents est réduit ou minimisé.

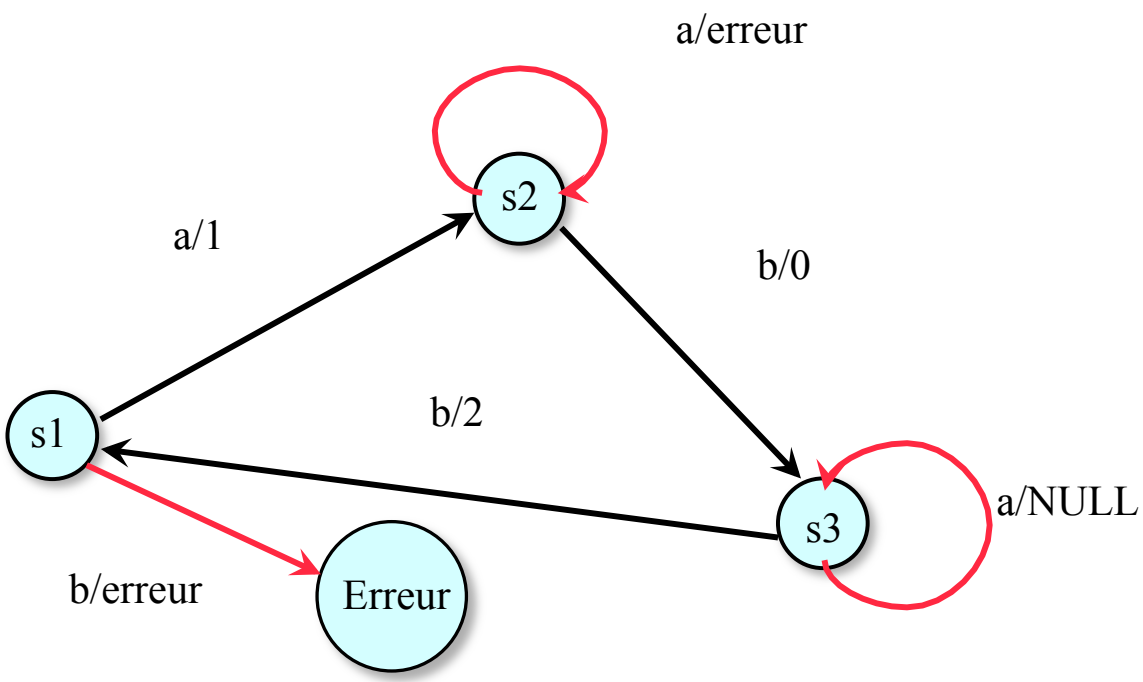
Concepts: *Déterminisme*



Concepts: Spécification partielle (1)

- S'il n'y a pas de transition pour un état *s* et une entrée *i*,
  - Convention d'implémentation Transitions "Implicitement définies"
    - ✓ Ajouter une transition en boucle avec une sortie nulle *(s, i, null, s)*
    - ✓ Ajouter une transition en boucle avec une sortie erronée*(s, i, error, s)*
  - Convention d'implémentation Transitions "Indéfinies par défaut"
    - ✓ Ajouter un ensemble de transitions *{(s, i, o, s') | o ∈ O, s' ∈ S}* *(conformité faible: toutes les options sont permises.)*
  - Convention d'implémentation Transitions "Interdites"
    - ✓ Si l'entrée *i* ne peut être introduite à l'état *s,* ajouter un état *erreur* et une transition de s à *erreur (conformité forte: aucune liberté.)*

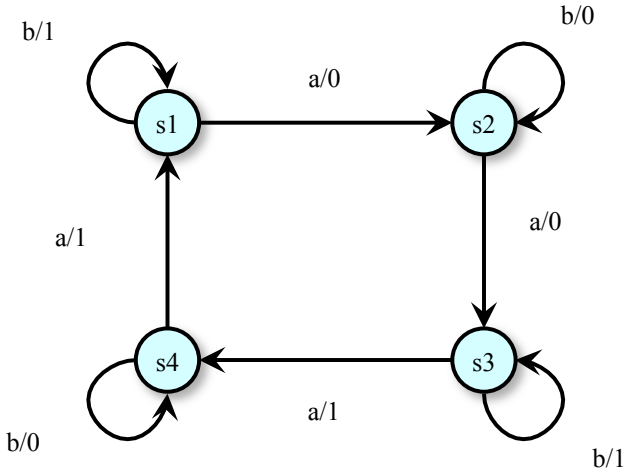
Concepts: Spécification partielle (2)

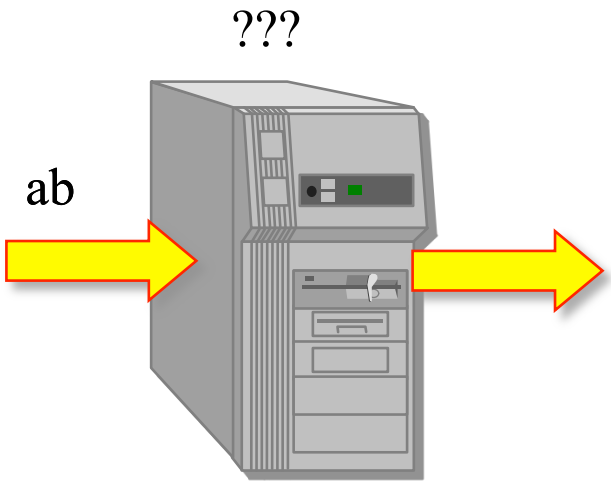


Concepts: *Homing Sequence (1)*

- Soit un automate dont nous connaissons la spécification et l'implémentation.
- Homing Sequence (HS) :
  - La machine est en marche
    - ✓ et nous ne savons pas dans quel état est l'implémentation
  - Une séquence d'événements d'entrée est une HS si :
    - ✓ après l'application de la séquence d'événements d'entrée, nous pouvons déterminer ***l'état d'arrivée*** en observant la séquence d'événements de sortie.
  - En d'autres termes, la séquence de sorties que produit une HS permet d'identifier de manière unique un état d'arrivée.

Concepts: Homing Sequence (2)



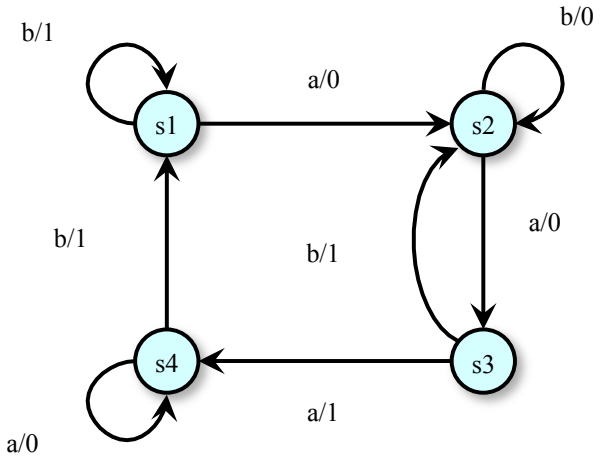


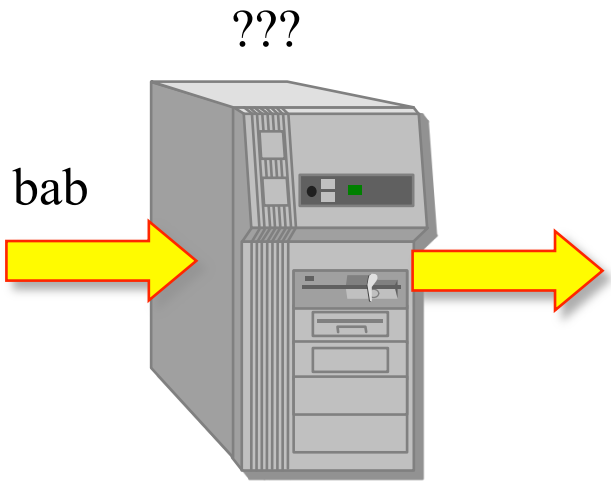
	a	b	
s1	0	0	**s2**
s2	0	1	**s3**
s3	1	0	**s4**
s4	1	1	**s1**

Concepts: *Séquences de synchro. (1)*

- Soit un automate dont nous connaissons la spécification et l'implémentation.
- Séquences de synchronisation :
  - Nous ne savons pas dans quel état est l'implémentation.
  - Une séquence d'événements d'entrée est une **séquence de synchronisation si**
    - ✓ la séquence d'événements d'entrée ***rend toujours l'implémentation au même état d'arrivée quelque soit l'état dans lequel était l'implémentation.***

Concepts: *Séquences de synchro. (2)*





s	b	a	b
s1	s1	s2	**s2**
s2	s2	s3	**s2**
s3	s2	s3	**s2**
s4	s1	s2	**s2**

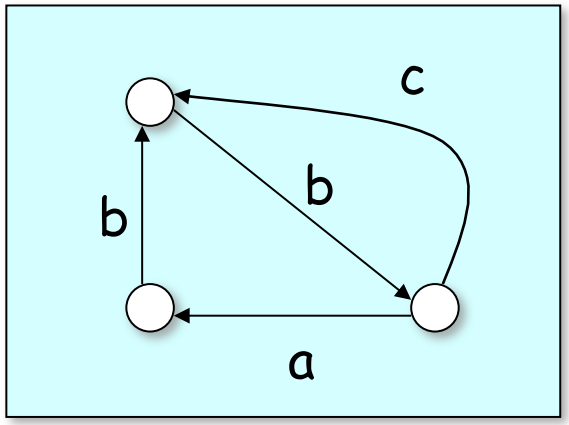
- Introduction aux FSM
- **Tests de conformité**

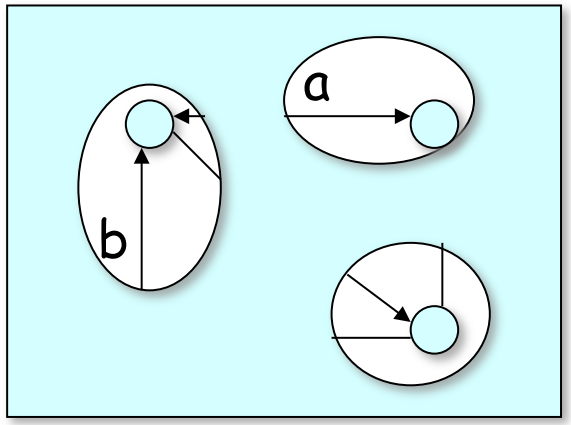
Énoncé

- Soit un automate et sa spécification.
- On ne sait rien de son implémentation mais il nous faut déterminer si l'implémentation est équivalente à la spécification.

La problématique du test

- Une spécification décrit le comportement du système.
- Une implémentation devrait se comporter comme établi par sa spécification.
- En sommes-nous vraiment certains ?

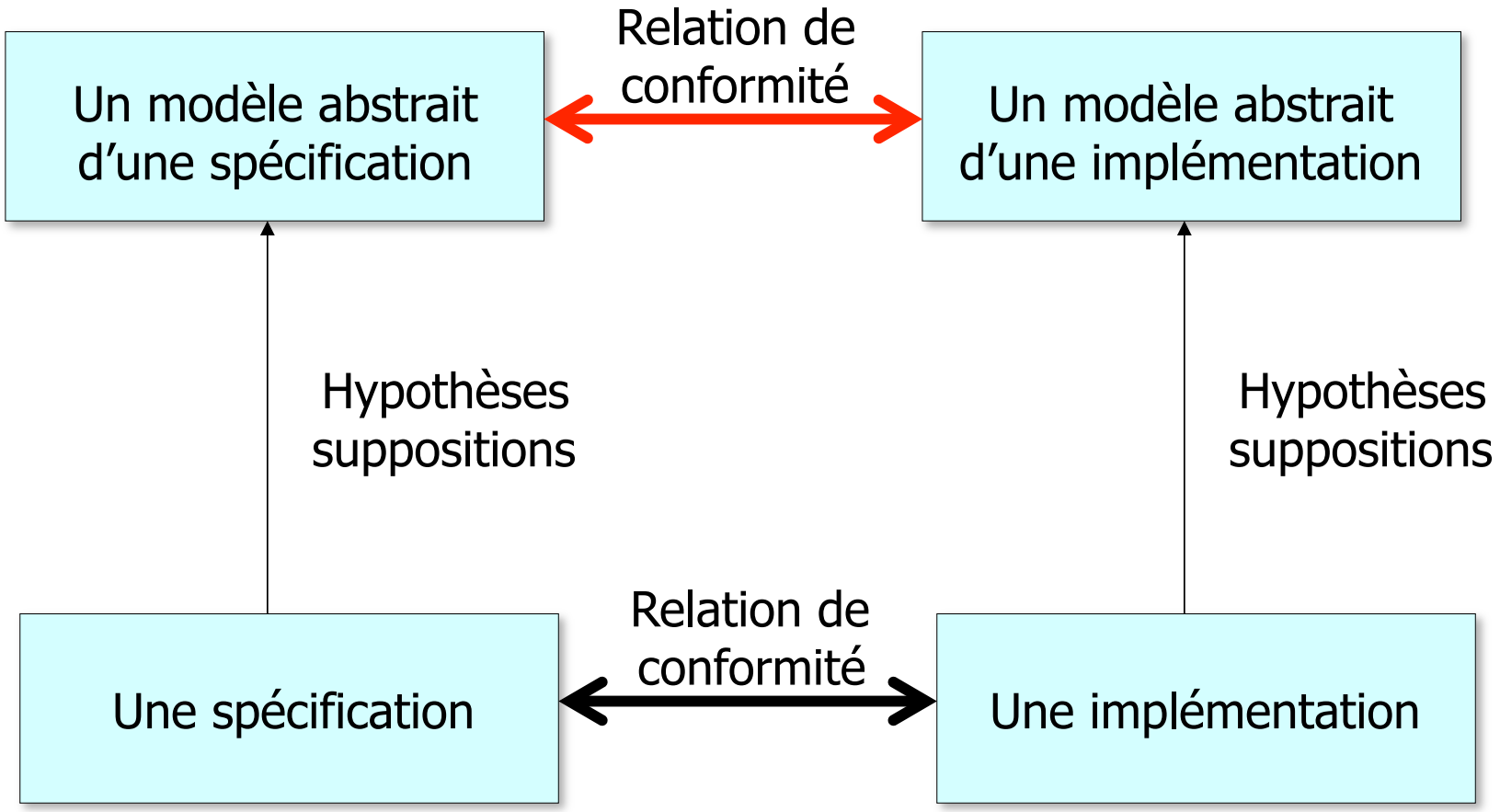




Structure de test

- But du test
  - e.g., identifier un état de l'implémentation.
- Préambule du test
  - diriger l'implémentation vers un état connu.
- Corps du test
  - invoque le comportement correspondant au but du test.
- Vérification des résultats du test.

Test d'hypothèses/ suppositions



Suppositions

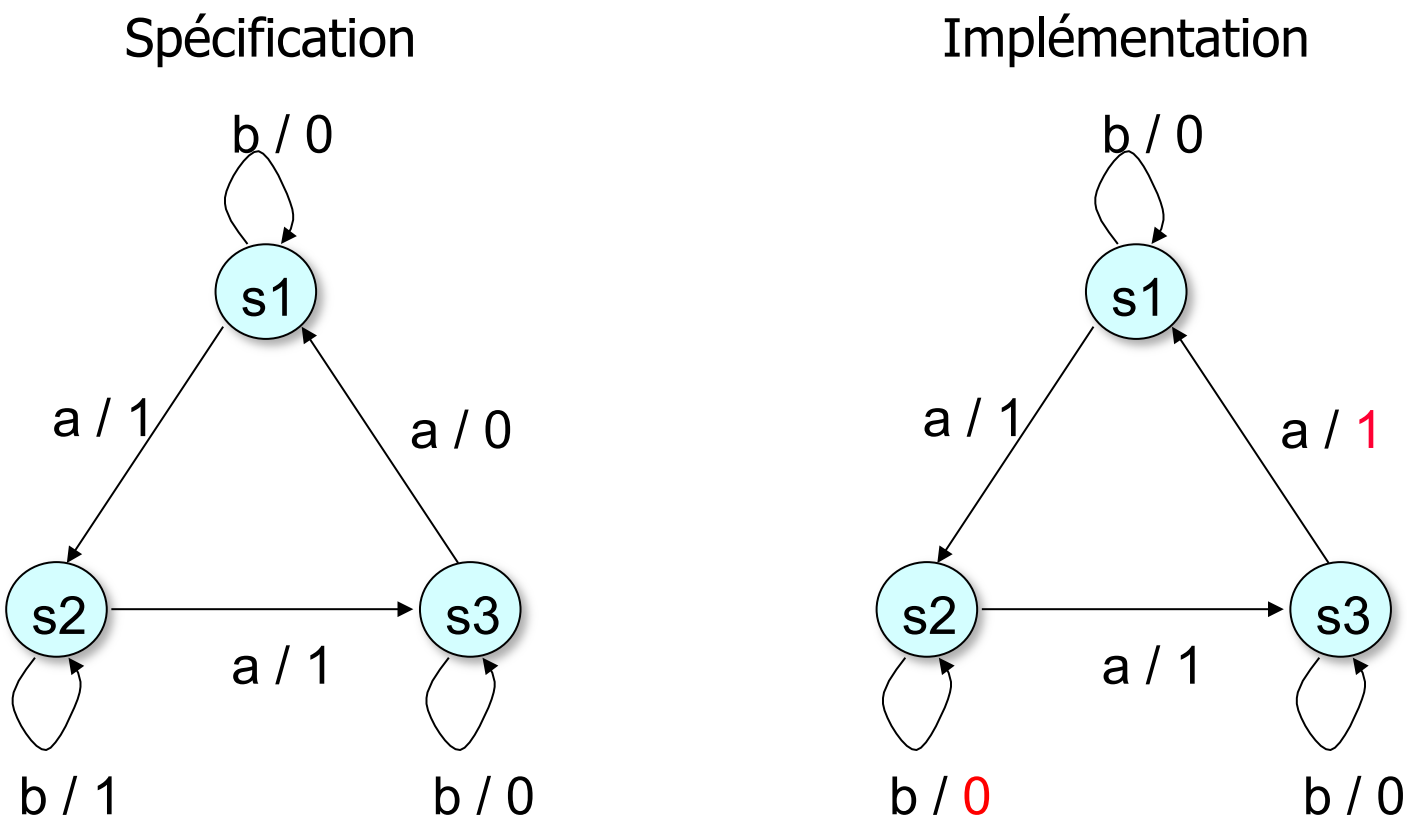
- Suppositions sur les spécifications de la FSM :
  - Réduite (pas d'états équivalents),
  - complètement spécifiée,
  - déterministe,
  - fortement connexe
  - initialisée

- Suppositions sur les implémentations :
  - Utilisation d'un modèle de fautes [un modèle hypothétique afin de déterminer quels types de fautes peuvent se produire dans une implémentation]
  - Le modèle de fautes est normalement basé sur des mutations.

Modèles de fautes FSM

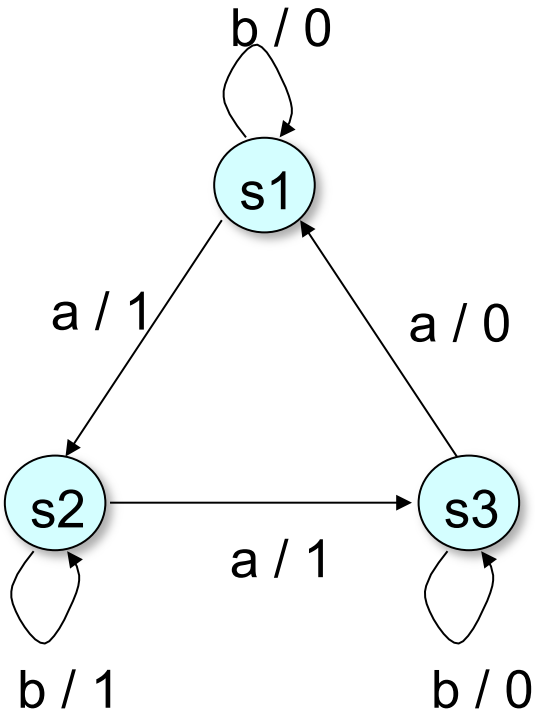
- Modèles de fautes FSM :
  - fautes de sortie,
  - fautes de transfert,
  - fautes de transfert avec états additionnels,
  - transitions additionnelles ou manquantes,
  - états additionnels ou manquants,
  - contrôle et fautes de flots de données.
- Utiles pour
  - Génération d'un jeu de tests pour l'objectif de couverture
  - Formalisation du "but du test"
  - Evaluer et optimiser la couverture de jeux existants
  - Diagnostics

Fautes de sortie

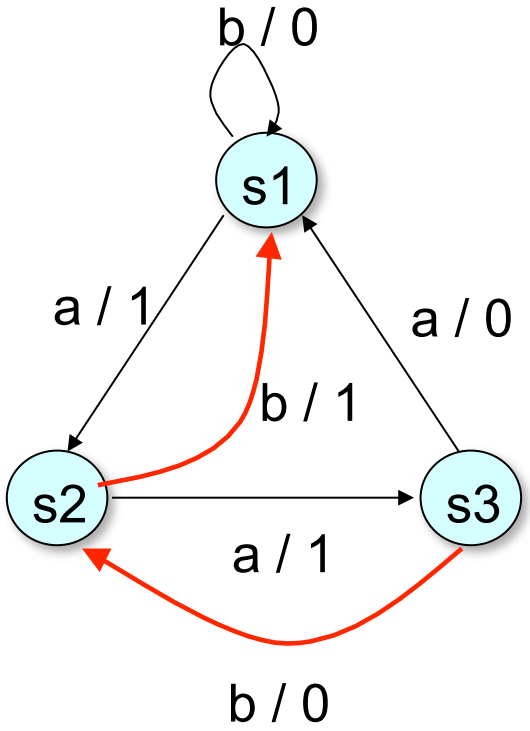


Fautes de transfert

Spécification

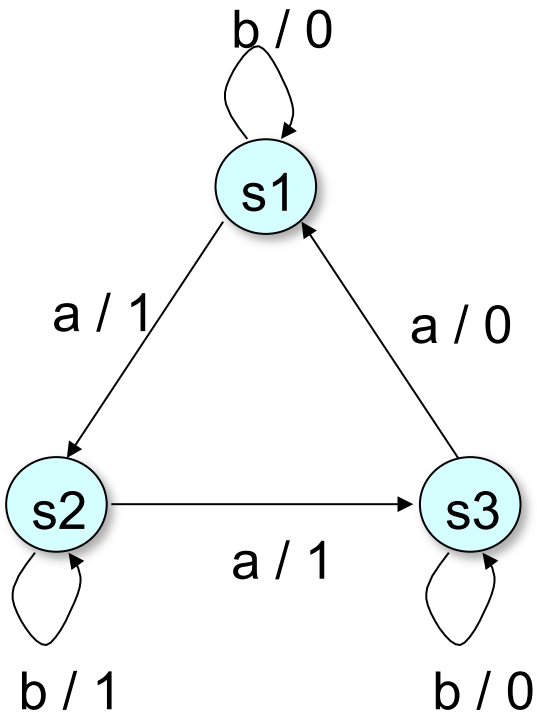


Implémentation

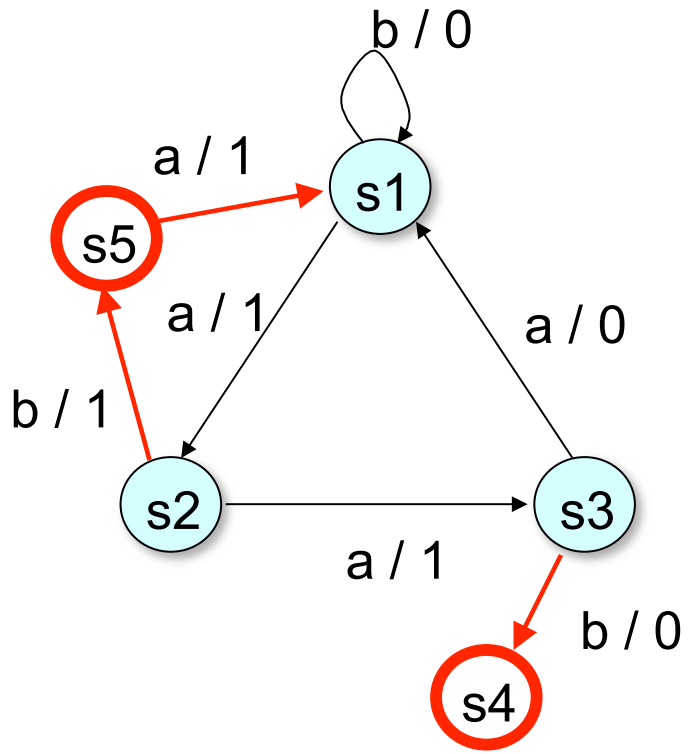


Fautes de transfert avec états additionnels

Spécification



Implémentation



Modèles de fautes pour logiciel

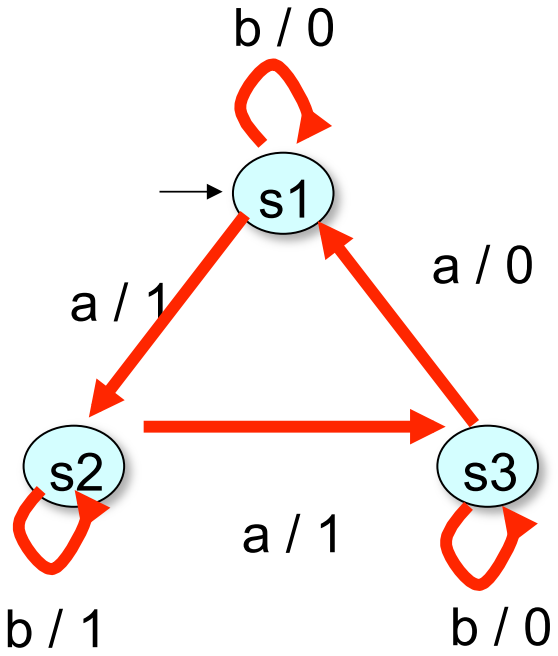
- Le logiciel peut être plutôt complexe et beaucoup de fautes sont possibles :
  - fautes séquentielles,
  - fautes d'arithmétique et de manipulation,
  - mauvais appels de fonctions,
  - mauvaise spécification de type de données,
  - mauvaises valeurs,
  - mauvais nombres de variables,
  - mauvais opérateurs,
  - …

Test basé sur les modèles FSM

- Tester les méthodes en se basant sur le flot de contrôle
  - Méthode du tour de transition(TT) – fautes de sortie
    - ✓ une séquence des entrées simples pour traverser toutes les transitions,
    - ✓ simple mais faible.
  - Méthode de la séquence distinctive (distinguishing sequence (DS)) – fautes de sortie + fautes de transfert
    - ✓ identifie chaque état par les mêmes tests,
    - ✓ fournit une couverture complète, mais peut ne pas exister.
  - Méthode W (ensembles caractérisant) – fautes de sortie + fautes de transfert.
  - Méthode entrée-sortie unique (unique-input-output (UIO)) – fautes de sortie + fautes de transfert
    - ✓ un test pour identifier chaque état dans la spécification.

Méthode TT (1)

- Un tour de transition d'un FSM
  - Un chemin commençant à l'état initial, traverse chaque transition au moins une fois, et retourne à l'état initial.



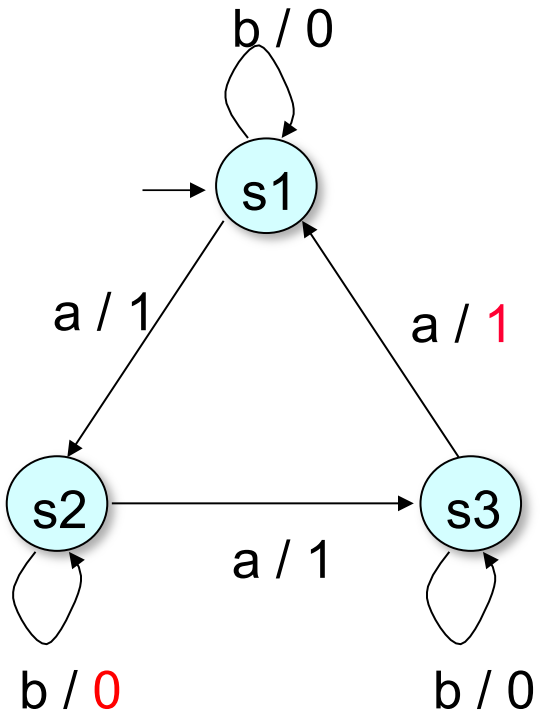
bababa

s1->s1 ->s2->s2->s3->s3->s1

0 1 1 1 0 0

Méthode TT (2)

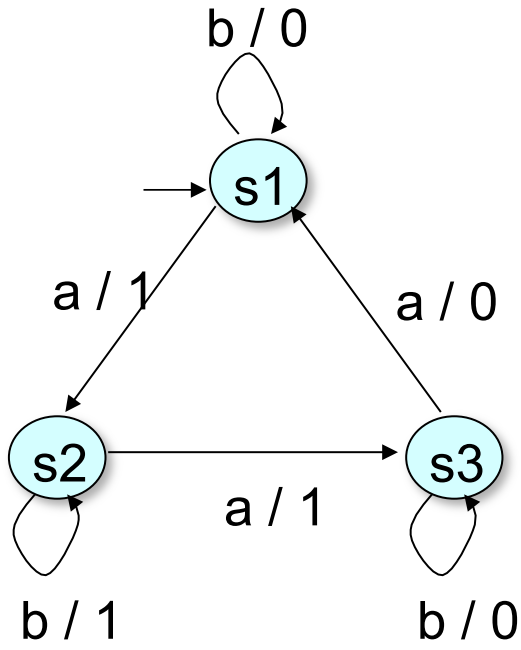
- D'un tour de transition, nous identifions un jeu de tests composé d'une séquence d'entrée et sa séquence de sortie attendue.
  - La séquence d'entrée *bababa* et
  - sa séquence de sortie attendue *011100*
  - *la séquence observée est … ?*



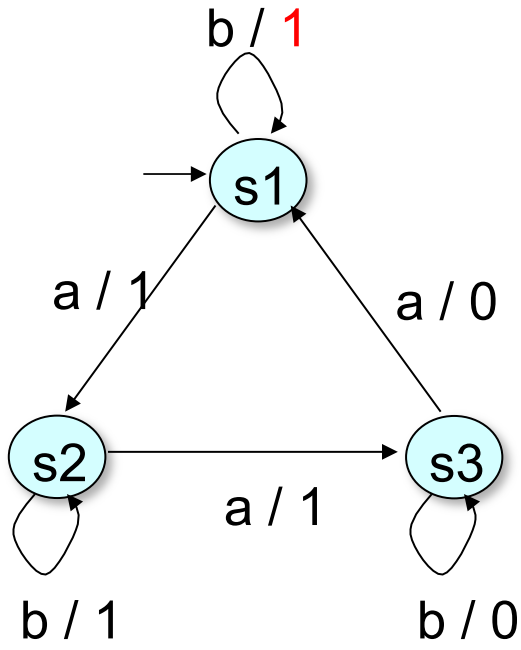
Méthode TT (3)

- Les tours de transition peuvent trouver toutes les fautes de sortie (en l'absence de fautes de transfert).

Spécification
Séquence d'entrée : *bababa*
Séquence de sortie attendue : *011100*



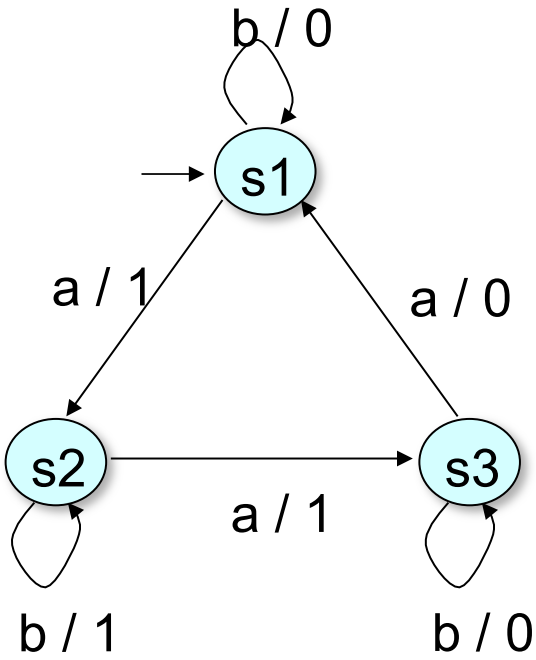
Implémentation
Séquence d'entrée : *bababa*
Séquence de sortie observée : *111100*



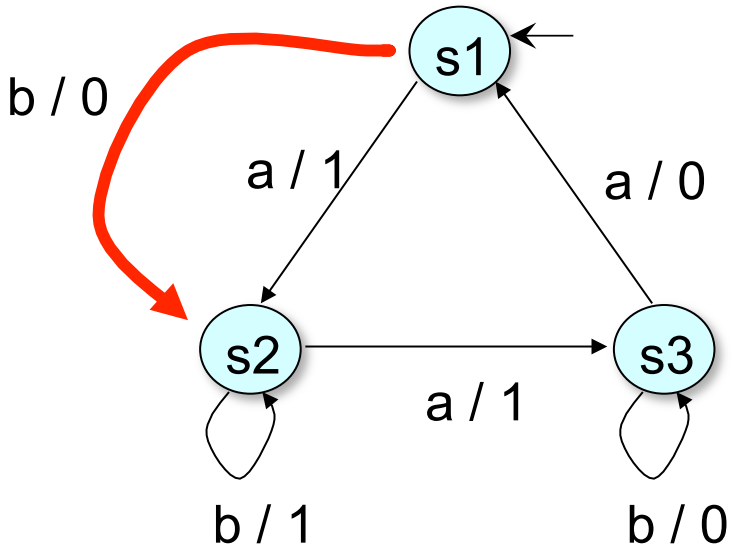
Problème de la méthode TT

- Les tours de transition ne sont pas efficaces pour la détection de fautes de transfert.

Spécification
Séquence d'entrée : *bababa*
Séquence de sortie attendue : *011100*



Implémentation
Séquence d'entrée : *bababa*
Séquence de sortie observée : *010001*



Détecter les fautes de sortie et les fautes de transfert

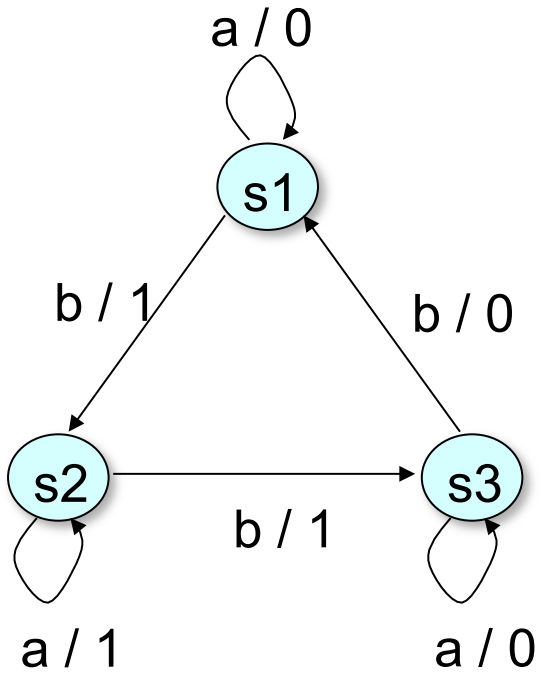
- Méthode DS, méthode W et méthode UIO.
- L'idée principale
  - Génère un jeu de tests tel que, **pour chaque transition** (*s, i, o, s'*)
    - ✓ Étape 1 : Mets l'implémentation dans un état *s* (configuration).
    - ✓ Étape 2 : Applique l'entrée i et vérifie si la sortie actuelle est *o* (faute de sortie).
    - ✓ Étape 3 : Détermine si l'état ciblé de l'implémentation est *s'* (faute de transfert).

Séquence distinctive (DS)

- Une séquence d'entrée est une **séquence distinctive (DS, Distinguishing Sequence)** si
  - après avoir appliqué la séquence d'entrée, nous pouvons déterminer l'état initial en observant la séquence de sortie produite.

Pourquoi …

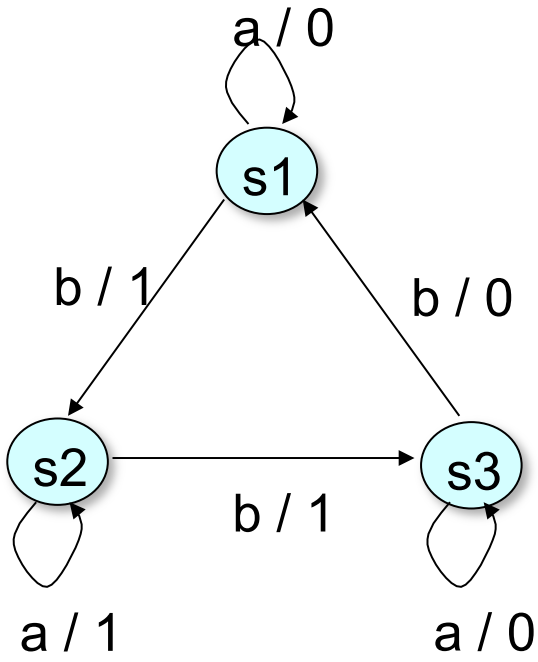
- *a* n'est pas une séquence distinctive …



État initial	Séquence d'entrée	Séquence de la sortie	État final
S1	a	0	S1
S2	a	1	S2
S3	a	0	S3

Pourquoi …

- *ab* est une séquence distinctive.



État initial	Séquence de l'entrée	Séquence de la sortie	État final
S1	ab	01	S2
S2	ab	11	S3
S3	ab	00	S1

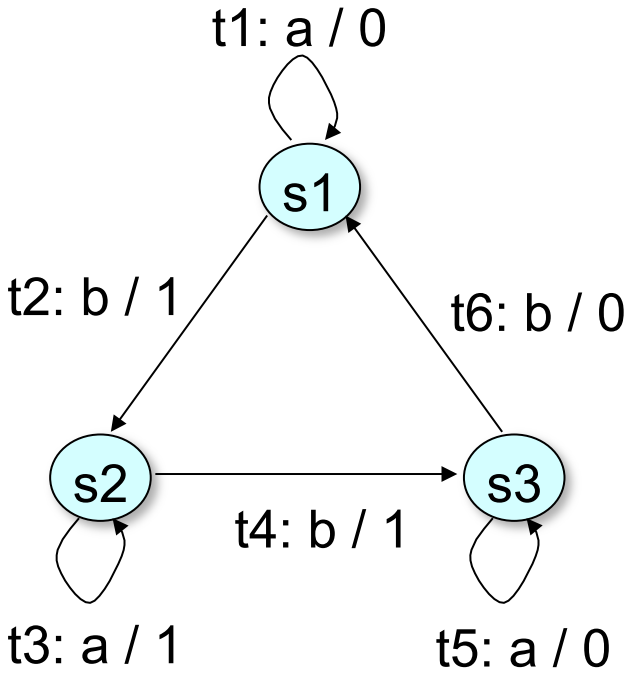
Méthode DS

- Pour chaque transition (*s, i, o, s'*).
  - Étape 1 : Mets l'implémentation en état *s* (configuration).
  - Étape 2 : Applique l'entrée i et vérifie si la sortie actuelle est *o* (faute de sortie)
  - Étape 3 : Détermine si l'état ciblé de l'implémentation est « vraiment » *s'* en utilisant une séquence distinctive (faute de transfert).

- Garantit de trouver toutes les fautes de sortie et de transfert

Méthode DS

- Un jeu de tests



t1: restauration/nul a/0 a/0 b/1
t2: restauration/nul b/1 a/1 b/1
t3: restauration/nul b/1 a/1 a/1 b/1
t4: restauration/nul b/1 b/1 a/0 b/0
t5: restauration/nul b/1 b/1 a/0 a/0 b/0
t6: restauration/nul b/1 b/1 b/0 a/0 b/1

Configuration **Sortie** **Transfert**

Méthode DS – Problèmes

- Peu de FSMs possèdent une DS
- La DS peut être trop longue.
- Dans l'exemple ci-contre, il n'y a pas de DS
  - Une DS ne peut pas commencer par *a*

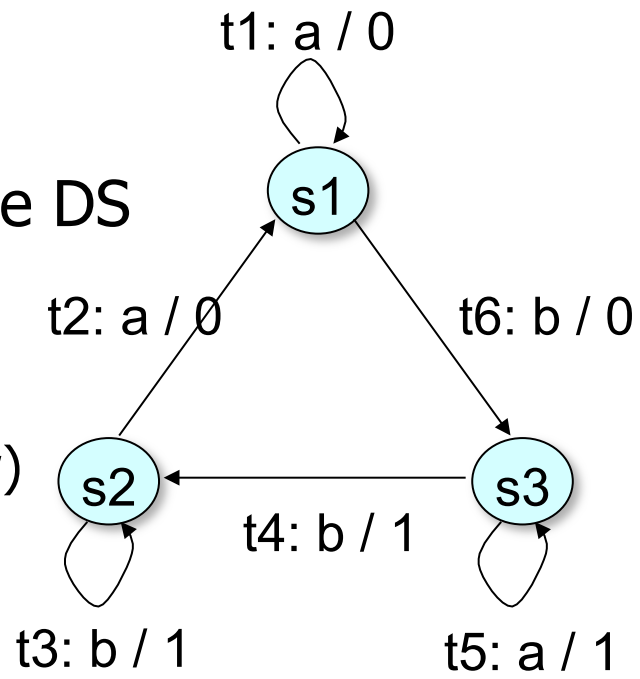
car on ne pourrait distinguer s1 de s2

$\lambda(s1,aw)=0\lambda(s1,w)=\lambda(s2,a)\lambda(s1,w)=\lambda(s2,aw)$

  - Une DS ne peut pas commencer par *b*

car on ne pourrait distinguer s2 de s3.

$\lambda(s2,bw)=1\lambda(s1,w)=\lambda(s3,b)\lambda(s1,w)=\lambda(s3,aw)$



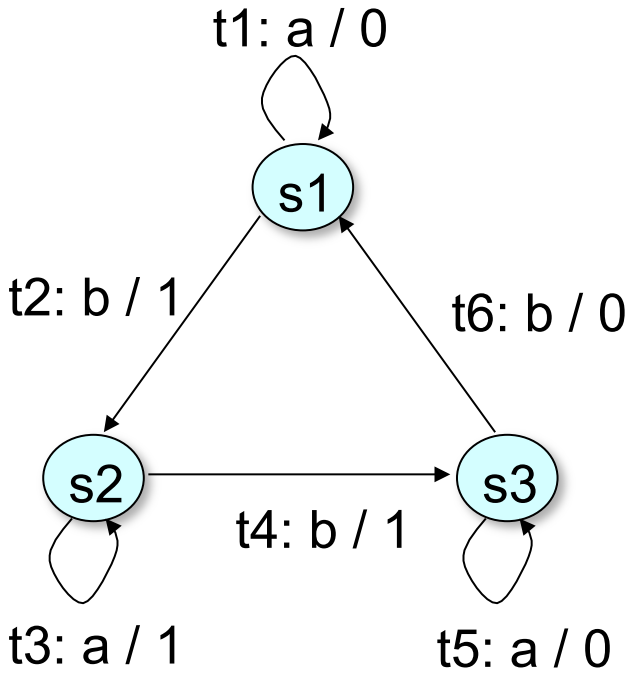
Méthode W

- Un ensemble de séquences d'entrée est un **ensemble caractérisant (characterizing set)** si
  - après avoir appliqué toutes les séquences d'entrée dans l'ensemble, nous pouvons déterminer l'état initial en observant les séquences de sortie.

- Garantit de trouver toutes les fautes d'entrée et de sortie (sous réserve d'un nombre limité d'états du FSM)

Pourquoi …

- {*a,b*} est-il un ensemble caractérisant … ?

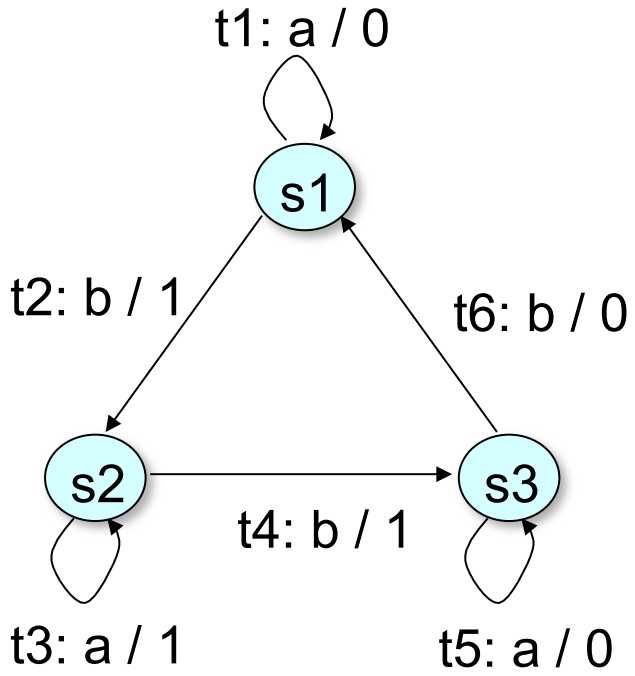


État initial	Séquence de l'entrée	Séquence de la sortie	État final
S1	a	0	S1
S2	a	1	S2
S3	a	0	S3

État initial	Séquence de l'entrée	Séquence de la sortie	État final
S1	b	1	S3
S2	b	1	S2
S3	b	0	S2

Méthode W

- Un jeu de tests.



t1: restauration/nul a/0 a/0
t1: restauration/nul a/0 b/1

t2: restauration/nul b/1 a/1
t2: restauration/nul b/1 b/1

t3: restauration/nul b/1 a/1 a/1
t3: restauration/nul b/1 a/1 b/1

t4: restauration/nul b/1 b/1 a/0
t4: restauration/nul b/1 b/1 b/0

t5: restauration/nul b/1 b/1 a/0 a/0
t5: restauration/nul b/1 b/1 a/0 b/0

t6: restauration/nul b/1 b/1 b/0 a/0
t6: restauration/nul b/1 b/1 b/0 b/1

Méthode W - Problèmes

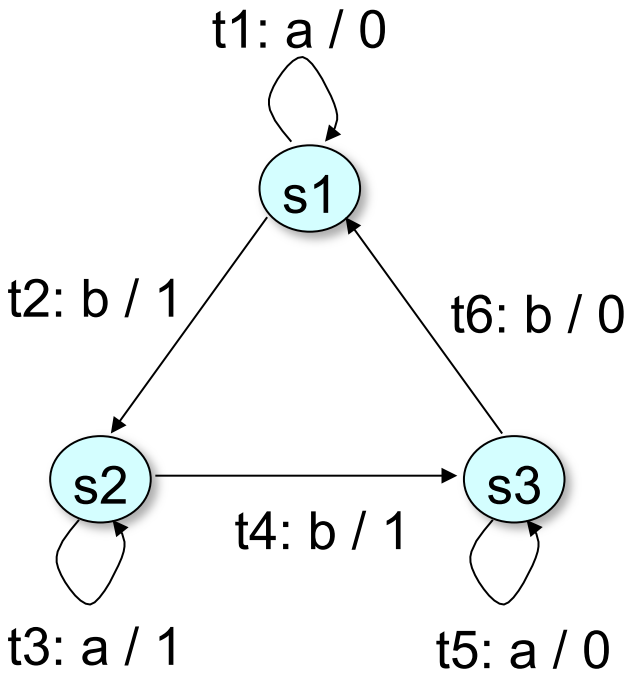
- Bien que chaque FSM ait un ensemble caractérisant, l'ensemble peut avoir trop d'éléments.
- Les séquences distinguantes et les ensembles caractérisants imposent des requis trop forts.
  - Nous sommes seulement intéressés à déterminer si l'état ciblé est un état spécifique.
    - ✓ L'identification de l'état versus la vérification de l'état.

Méthode entrée-sortie unique (Unique-input-output (UIO))

- Soit *s* un état.
- Une séquence d'entrée est une **séquence UIO** pour *s* si
  - après avoir appliqué la séquence d'entrée, nous pouvons déterminer que **l'état initial** est *s* ou non, en observant la **séquence de sortie**.

Méthode UIO et états s2, s3

- Montrer que *a* est une séquence UIO pour s2
- Montrer que *b* est une séquence UIO pour s3

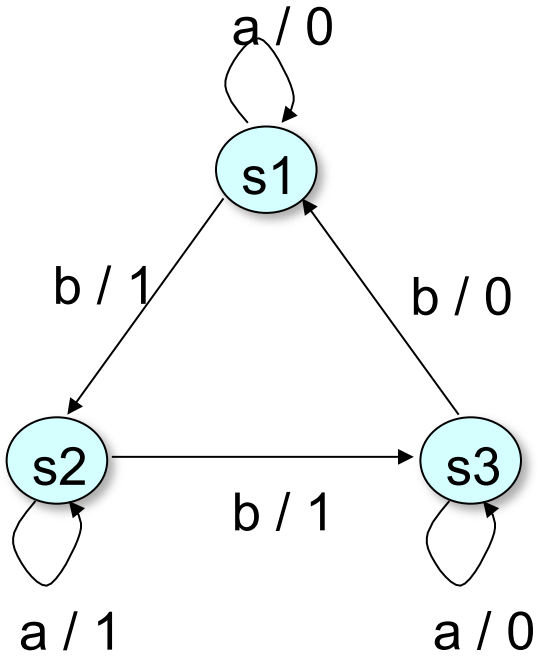


État initial	Séquence de l'entrée	Séquence de la sortie	État final
S1	a	0	S1
S2	a	1	S2
S3	a	0	S3

État initial	Séquence de l'entrée	Séquence de la sortie	État final
S1	b	1	S2
S2	b	1	S3
S3	b	0	S1

Méthode UIO et état s1

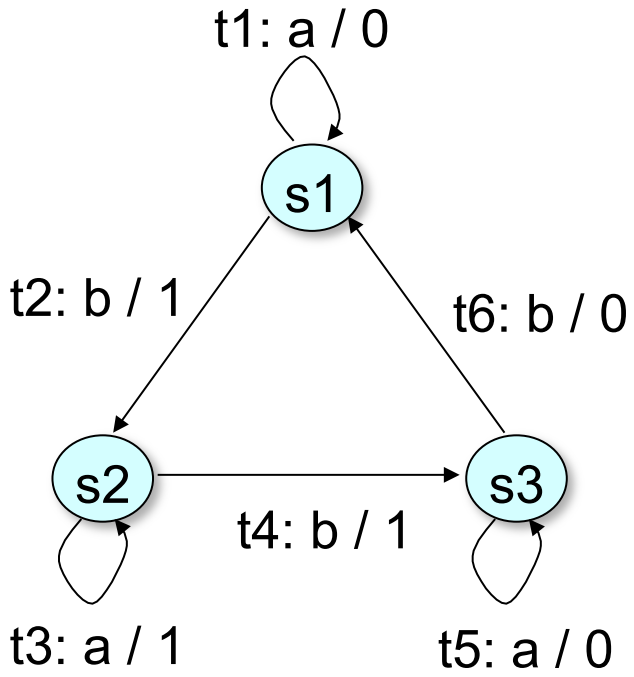
- Montrer que *ab* est une séquence UIO pour s1.



État initial	Séquence de l'entrée	Séquence de la sortie	État final
S1	ab	01	S2
S2	ab	11	S3
S3	ab	00	S1

Méthode UIO – suite de tests

- Un jeu de tests.

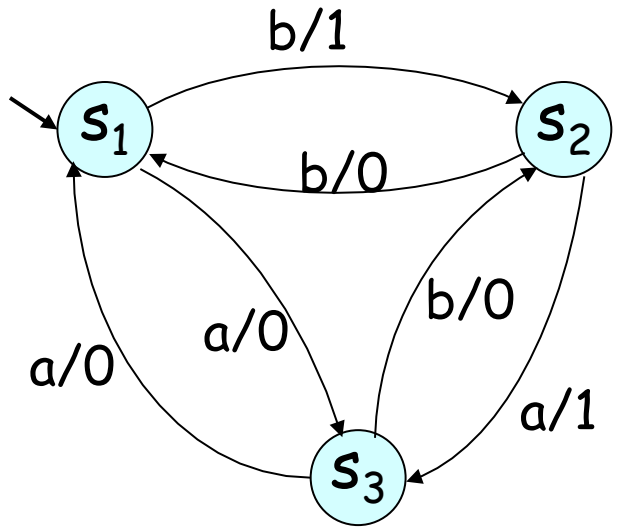


t1: restauration/nul a/0 a/0 b/1
t2: restauration/nul b/1 a/1
t3: restauration/nul b/1 a/1 a/1
t4: restauration/nul b/1 b/1 b/0
t5: restauration/nul b/1 b/1 a/0 b/0
t6: restauration/nul b/1 b/1 b/0 a/0 b/1

Test UIO – Vérification d'états

- Supposition : pour chaque état, il y a une séquence d'entrée qui produit une sortie unique.
- Chaque test vérifie un état.

	préambule	corps
✓ s1		b/1
✓ s2	b/1	a/1
✓ s3	a/0	a/0 b/1



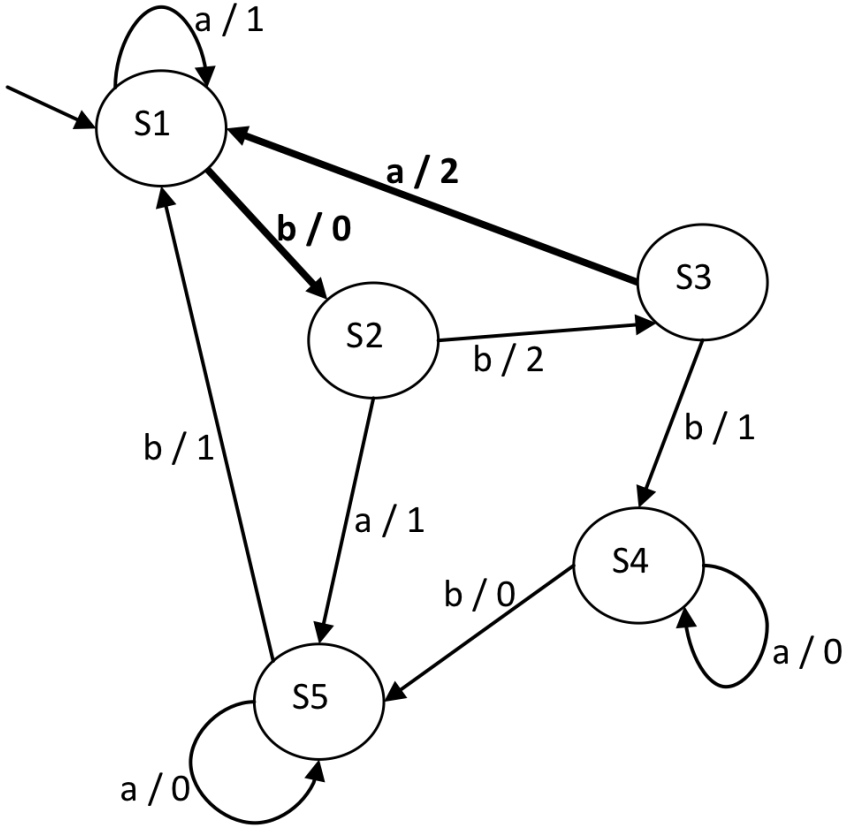
Méthode UIO - pour et contre

- Comparaisons
  - Comparé au DS, les FSMs ont plus souvent des séquences UIO.
  - Les séquences UIO sont normalement courtes.
  - Par contre, cette méthode peut ne pas détecter certaines fautes de transfert ➔ proposition de UIOv [Vuong89]

Exercice

Proposez les séquences les plus courtes pour tester les transitions S1-S2, S3-S1 (en gras) avec, si possible, chacune des méthodes suivantes.

- Transition de tour (ne pas se restreindre aux transitions en gras)
- Séquence distingtive (DS)
- Séquence UIO
- Ensemble caractérisant (W)



Solution

- **TT**: abbabbbababbbab ->10220210001011

- **DS**: ab car S1:10, S2:11, S3:20, S4:00, S5:01
  - **restauration/nul b/0 a/1 b/1**
  - **restauration/nul b/0 b/2 a/2 a/1 b/0**

- **UIO**: S1: ab, S2: b, S3: a, S4: ab, S5:ab
  - **restauration/nul b/0 b/2**
  - **restauration/nul b/0 b/2 a/2 a/1 b/0**

- **W**: {a,b} car S1:{1,0} S2 {1,2} S3{2,1} S4{0,0} S5 {0,1}
  - **restauration/nul b/0 a/1**
  - **restauration/nul b/0 b/2**
  - **restauration/nul b/0 b/2 a/2 a/1**
  - **restauration/nul b/0 b/2 a/2 b/0**