Test fonctionnel de Protocoles

Test actif

IUT : Implementation under test

PCO : ~~ou~~ Point of control and observation

PDU : paquets permettant de définir des protocoles

ASP : primitives de service abstraites

SEKUNUM = ?

IUT Lower/upper

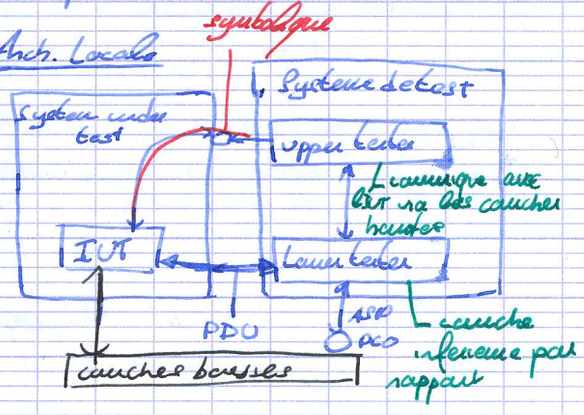
Arch. de test

ISO 9646 dans ce [debut] de cours

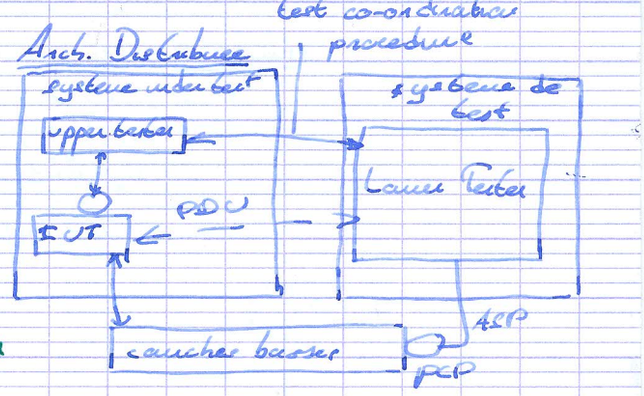
Principe : le testeur est relié à l'~~UET~~ IUT etc

cf IMBUE : modèle de premier

Arch. Locale

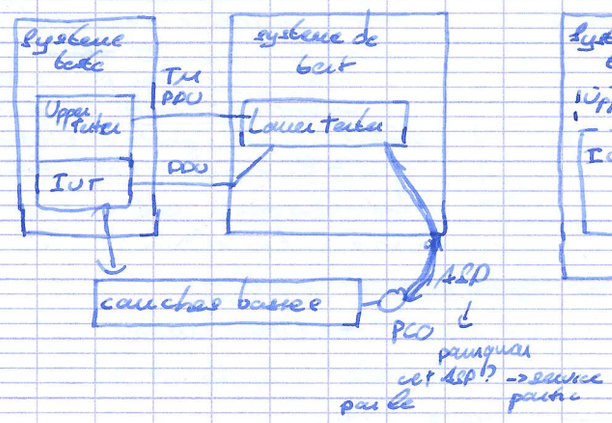


Arch. Distribuée



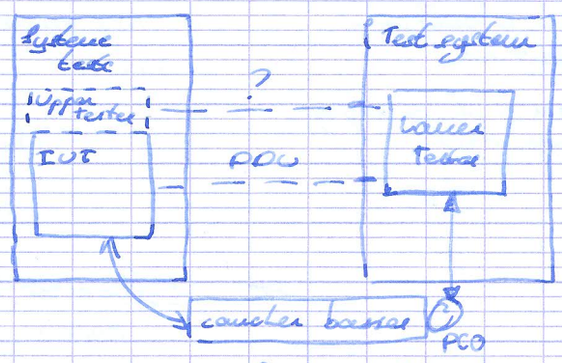
SEKULIX

Arch. Coordonnée



L'upper est intégré au

Arch. distribuée



Pourquoi nouvel heure Sekulix

par faiblement beaucoup d'infos sur le upper tester

L'utilise dans deux cas : pas de couches plus haute, ou alors quand on veut tester quelque chose dont on est parmi de ce qui va servir

L'IUT doit être synchrone avec le testeur haut

Testeur haut -> Il bouge, dans ou veut que il soit synchrone

Testeur bas : souvent relié à un réseau qui possède un mécanisme de synchronisation

X-means -> k moyenne mais on donnant une valeur de k

-> Comment partitionner un système en parties testables ? -> choisir les entrées correspondant au plus de classes possibles, qui commente le plus de cas possibles

Principe de test

Test partition (systématique)

-> Idéalement les partitions sont disjointes

La partition définit des classes d'équivalences

Test aléatoire (uniforme)

objectif : couvrir le maximum de verdict

Post-analyse

Démarche de test

Partie Haute/Nous

Concrétisation : Lorsque des valeurs abstraites associées à la spécification aux valeurs concrètes d'une implémentation

↳ Les cas de tests doivent des séquences d'actions abstraites, qui devront être concrétisées à travers le système

Exécution faire l'implémentation sous test

Deux enjeux
→ Création d'un environnement permettant le test de l'IUT
→ Reproduction de la séquence menée → problème difficile, notamment dans le cas de processus concurrents ← non déterministe (même input, output différent)

DSR... → pas déjà dans l'axe manip → intéressant
↳ premier résultat après 2 ans 95% inconclusive
DSRIV ↳ inconclusive 5%

+ tests passifs

→ détection du CV vers la boîte dans laquelle accidentiel

Soft 3M → plus de langage stages possibles

GENERATION DES SUITES DE TEST

Tout basé sur les automates → complets : chaque état accepte tout input
→ déterministe :
→ minimale : aucune paire d'états équivalents
→ fortement connexe : on peut atteindre l'état initial à partir de tout état
→ Pas d'état transitoire (~~état p~~)
(transitions qui se déclenchent sans input, concep. à une ε-transition)

~~Ex~~ : Différentes fautes possibles dans l'IUT
ERREUR DE SORTIE → mauvaise output
——— TRANSFERT → déplacement vers le mauvais état
——— MIXTE → les deux

Erreurs de sortie
Méthode TT (tour de transition) → On tire une séquence de test fausse d'une séquence d'entrée + séquence de sortie correspondante
↳ seule d'input → seule d'output

Sur de sortie ou de transfert
DS (suites distinguées)
W
UIO
→ principe : générer une séquence de test tq ∀(s1, i0, s2) (précondition, état, transition)
→ aller dans s1
Appliquer l'input i et vérifier si l'output o est relevé
Déterminer si l'état de l'IUT est s2

DS : une séquence d'entrée ou une séquence d'input qui distingue tous les états en fonction de l'output, et ce pour tous les états possibles
+ écriture des séquences de test en couvrant les cas que l'on veut couvrir
Si erreur dans la préambule → Verdict inconclusive

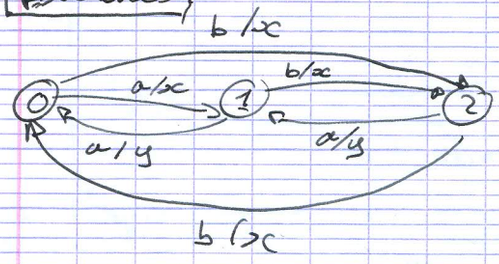
~~problème~~ on il aux

CE WE
→ Rattrapage de tout block
→ TFP - rattrapage

UIO = ?

W « Un ensemble d'inputs et un ensemble d'outputs »
→ Tout automate possède un W
sans les transitions, complexité quadratique, difficilement generalisé

Exercices



① Determiner une DS + sequence de test pour 1 → 0

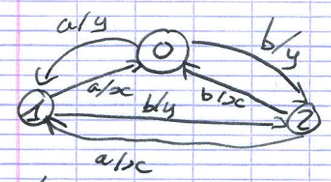
DS "a a"

output a 0: x x
1: y x
2: y y

Sequence de test

→ RN a/x a/y a/y a/y

preambule pour arriver à l'état 1 — sequence testée — verification qu'on est dans le bon



↳ Donner l'ensemble W de ce modele + une sequence UIO pour chaque état

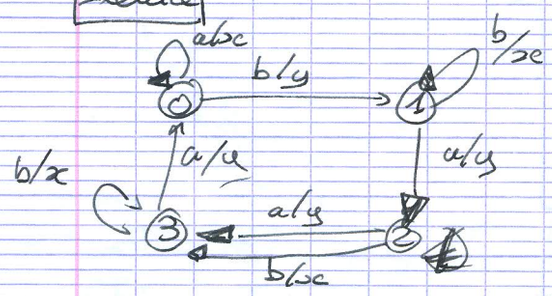
état	I	O
0	a	y
1	a	x
2	a	x

état	input	output
1	b	y
2	b	x

→ sequence W = AB

UIO Sequence unique d'input et d'output
↳ Existe presque toujours, et permet d'avoir des cas de test tres courts

Exercice



→ UIO de cet automate

a/x → état 0
b/y → état 0

état 1: a/y a/y a/y
état 2: a/y a/y a/x
état 3: a/y a/x

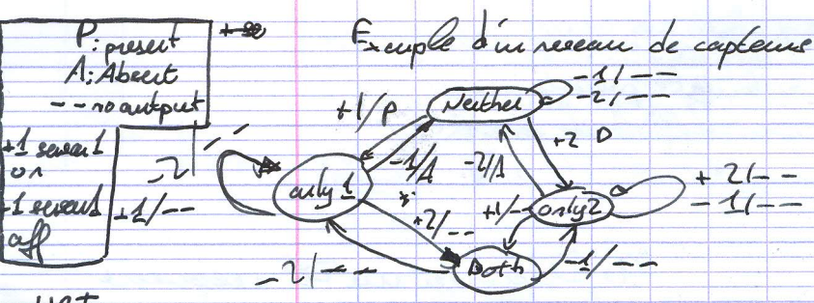
Sequence de test associée à (3 a y 0)

Reset/NUL b/y a/y a/y a/y a x

UIO chercher une transition unique sur le graph

27/10

lundi
mardi
mercredi → Q



→ Pas d'UIO sur ce graphe

HSI

LTS automate

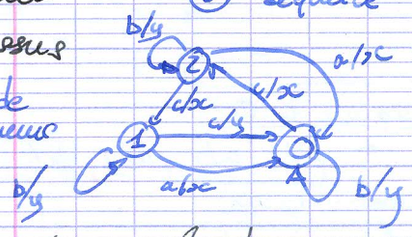
« quelles erreurs dans l'UIT seraient peut tomber dessus

+ quelle séquence de test → chercher des erreurs de comprehension

→ tester de trouver des cycles dans pour le w

EXERCICES

① Sequence W + UIO



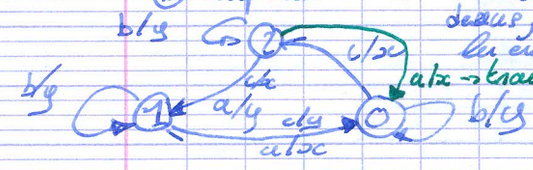
→ automate non complet : pas de sortie ou en zero : pas de problème dans cet exemple

W: {c ; cc}
{cc}
{bc, cc}

UIO
0 : c/x c/x
1 : c/y
2 : c/x c/y

Prendre UIO pour séquence de test

ⓑ Implementation du truc au dessus, trouver les erreurs



Erreur

② (a/x, 0) → erreur

RN c/x a/x c/x c/x

OUT 1 x y y x → erreur
erreur erreur
erreur

→ Commencer par la recherche des W par avant l'UIO

② W

	a	b	c
IN			
1	x	–	
2	|	y	
3	|	x	
4	x	z	
5	y		

problème : on laisse ça tel quel, on peut ne pas avoir de réponse pour l'état 1

Ts (② b/y ②)

RN ax by by cz

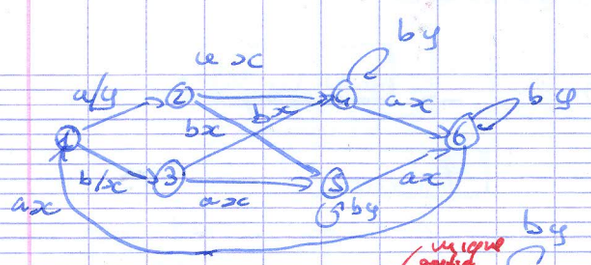
W = {a, b, c}

~~UIO~~ cz
~~UIO~~ cx
~~UIO~~ bx
bz
ay

UIO 1 cz
2 cx
3 bx
4 bz
5 ay

Test fonctionnel de protocole -

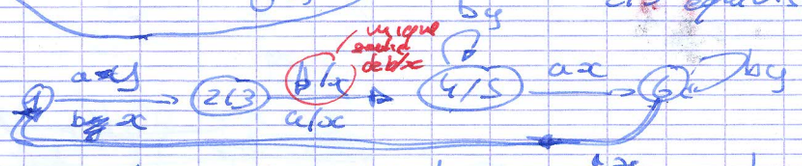
29/10



→ automate pas minimal

4/5 equivalents

2/3 equivalents



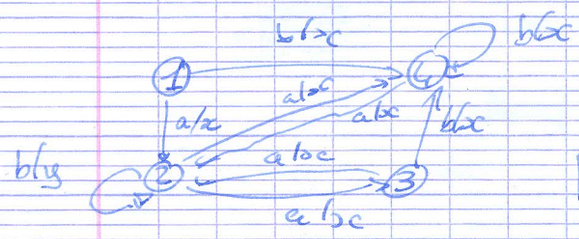
W: ~~b → état 2/3 inconnu~~ ~~ab → état 4/5 inconnu~~ } pas de W bon

W = { a, a, a }

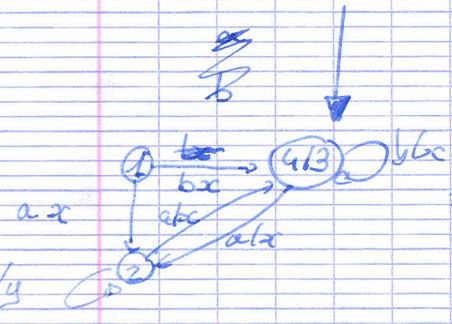
ST_W = (1 a y 2)

RN a g. (a x a x a y x) W(2)

→ Trouver des états equivalents



~~b~~



2/4 et 2/3 sont equivalents car deterministe

Pas forcément connexe

DS

1 equivalent à 3/4

DS = (b)

Après l'exécution de la séquence de test le système répond

→ [N y | y x x]

réponse de RN

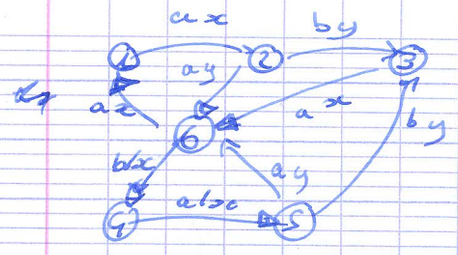
(Ts (2, bq, 2)
RN a x bq bq)

Quel verdict de test ?

→ INCONCLUSIVE, le préambule est faux donc on ne peut pas dire FAIL.

ERREUR DANS LE PREAMBULE ⟹ INCONCLUSIVE

→ Determiner W et UIO



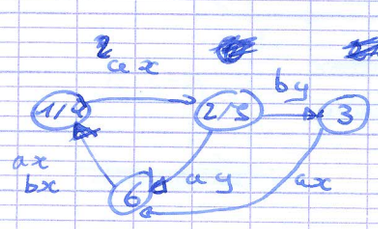
→ non complet

$\boxed{\text{Minimisation}}$:

2/5 equivalent

1/4 equivalent

Graphe minimal



b: permet de diff deux états, mais pas plus mais ne est

3

	a	b	ab
1/4	x a	—	$\boxed{xy}$
$\boxed{2/5}$	$\boxed{y}$		yx
3	x	—	$\boxed{xx}$
$\boxed{6}$	$\boxed{x}$	$\boxed{x}$	—

$W = \{a, b, ab\}$

(de ne pas mettre de b dans la table)

UIO 1/4 = ax ay → mieux car permet d'éviter des redondances

2/5 = ay

3 = ax ax ax

6 = ax ax ay

FAIL plutôt que INCONCLUSIVE du fait de la non-complétude

$\boxed{\text{Séquence de test}}$ (1, abx 2) (2 b/y 3)

$ST_{UIO}(1\ ax\ 3) = (RN\ abx\ a/y)$

$ST_{UIO}(2\ by\ 3)$

= RN abx abx abx a/x

$ST((1\ a/x\ 2) \cdot (2\ b/y\ 3))$

$= ST_{UIO}(1\ ax\ 3) + ST_{UIO}(2\ by\ 3)$ → on peut utiliser un préambule

↳ avec une telle TS, on obtient la séquence suivante

x y x x y x x → différence entre la longueur de la réponse et

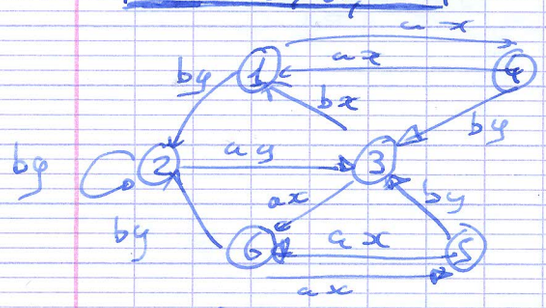
La réponse attendue : la seule solution pour réduire la séquence de test est de réduire la longueur du préambule au des UIO. Dans ce cas, il n'est des B dans les UIO, au risque d'avoir des réponses inconclusives

RN ax ay | RN ax | by | ax ax | ax

a se ax | ax bx | (fin)

↳ préambule

Transcrire DS W, UIO

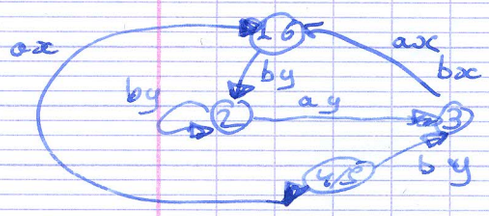


Etats equivalents

→ 1/6

→ 4/5

Nouvel automate



W

IN	a	ba
1/6	x	$\boxed{yy}$
2	$\boxed{y}$	yy
3	x	$\boxed{xx}$
4/5	x	$\boxed{yx}$

$W = \{a, ba\}$

DS Impossible car si on commence par a on va vers 1/6 en allant partant de 3 ou 4/5

b — 2 — 1/6 ou 2

UIO 1/6 → ax by ax

2 → ay

3 → bx ax

4/5 → by ax

TS_{UIO} → 4 abx 1

RN a/x abx ax by ax

autre sequence → en utilisant le m par exemple

→ RN

Incertitude → revenir dans le preambule

→ apres le preambule

+ En utilisant une sequence une de ces sequences, est-il possible de planter l'aut.

→ oui, si l'implementation est mal faite

→ ça ne peut pas planter a coup et pas de problème de communica°

→ completude de l'automate → Correction de l'implementation

→ problème de communica°

→ Inc avec moins d'output qu'attendu → ça peut provoquer un eventuel blocage