

2: Le test fonctionnel

2.1: Le test de conformité de systèmes réactifs

Types d'application

Typologie du test

Théorie du test de conformité

Test d'interopérabilité

Application composée de plusieurs entités communicantes

- Système embarqué
- Système à base de composants
- Protocole
- Peut être temporisé
- .../...

- **Test de Conformité**

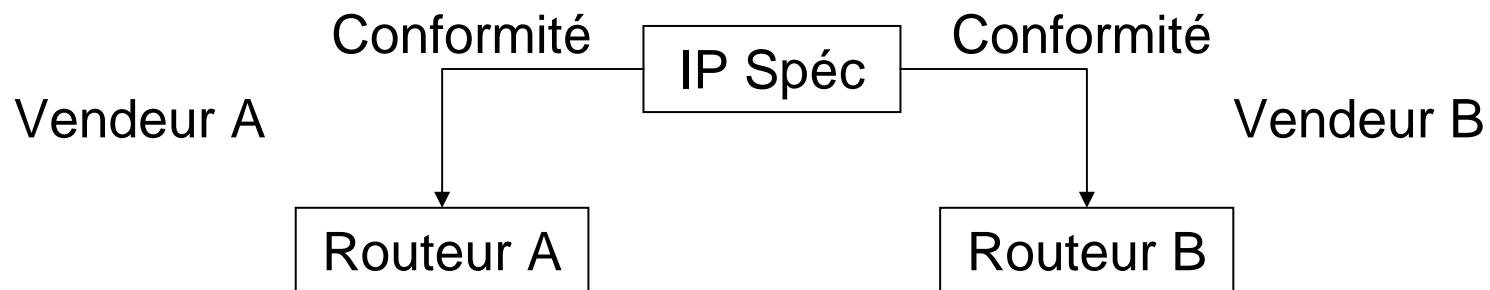
- Une seule entité est testée.

Objectif : déterminer si une implémentation est conforme à sa spécification.

- **Test d'Interopérabilité**

- Deux ou plusieurs entités sont testées.

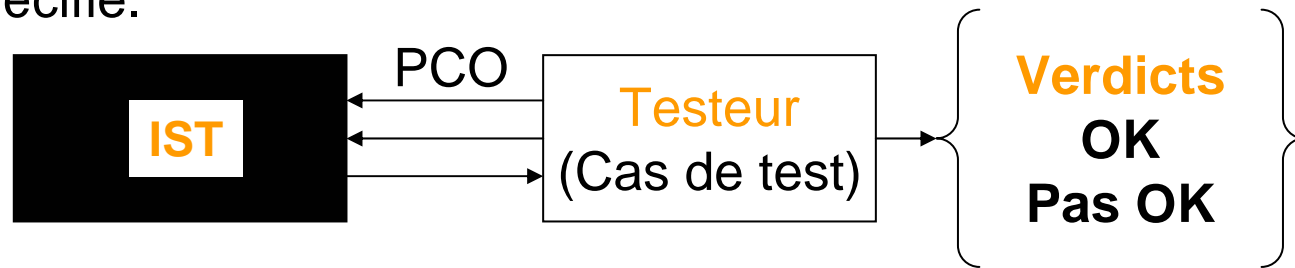
Objectif : déterminer si ces implémentations peuvent interagir ensemble comme prévu par la spécification.



Ces routeurs interagissent-ils correctement ?

Test de conformité

- But : être sûr qu'une implémentation fait ce que le standard spécifie.



IST (implémentation Sous Test) :

- Architecture boîte noire
- PCO (Points de contrôle et d'observation) : contrôle restreint et observation avec certaines interfaces

Testeur :

- Sortie : événements pour contrôler l'IST (cas de test),
- Entrée : observation de l'IST

Verdict :

- Résultat d'un cas de test (OK – Pas OK – Inconcluant)
- Un même cas de test peut conduire à différents verdicts

Test d'Interopérabilité:

- But final dans le développement d'un produit de systèmes communicants (exemple : routeur)
- Grand champ d'application : systèmes distribués (systèmes réactifs, systèmes embarqués, systèmes à base de composants, protocoles...)

Des implémentations peuvent être considérées conformes à leur spécification, mais peuvent ne pas interopérer.

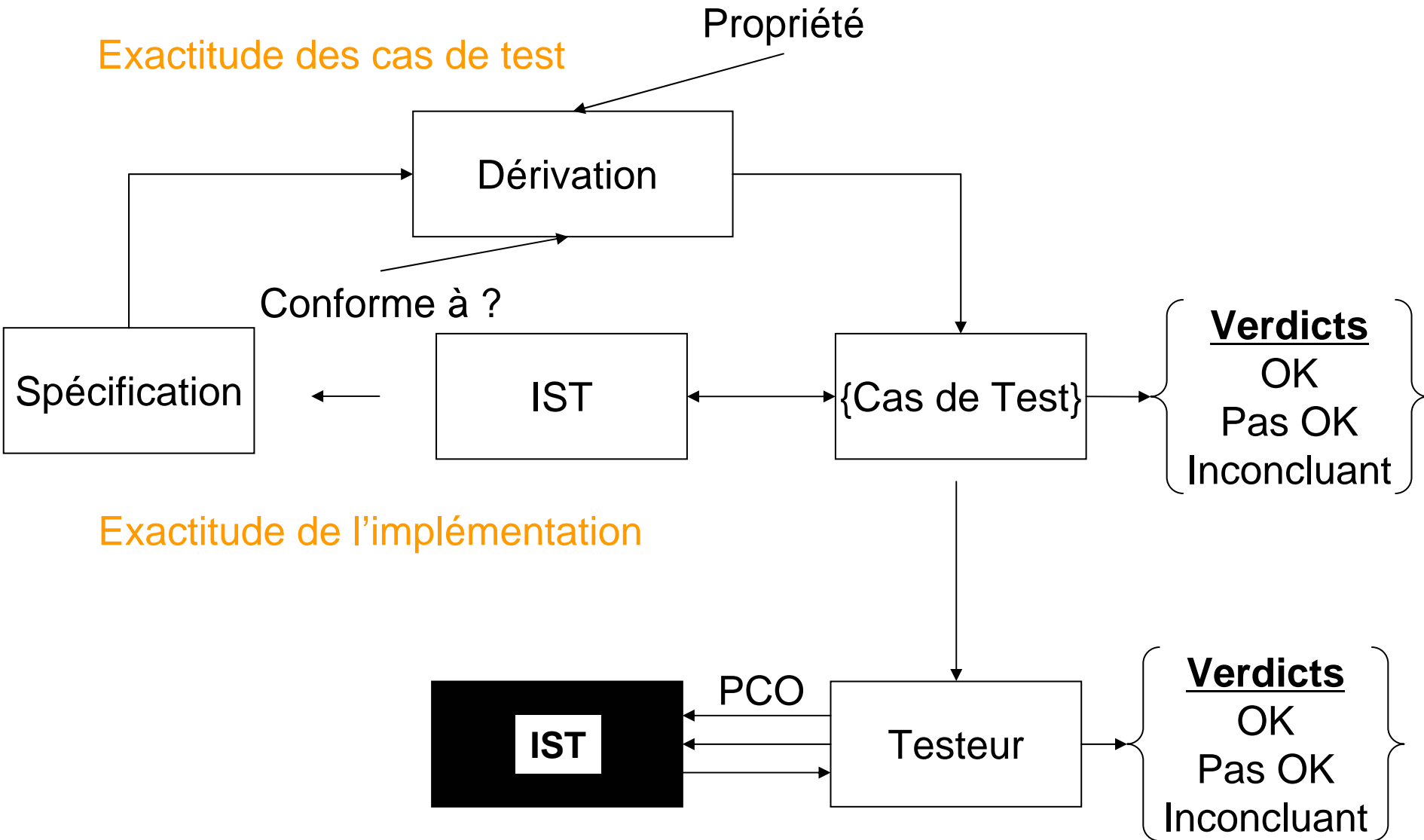
Contrairement à la conformité, peu de travaux théoriques sur le test d'interopérabilité (définitions, méthodologies, algorithmes...)

En général : conception manuelle des cas de test à partir d'une spécification informelle

- processus long et répétitif
- coût important
- Aucune assurance sur la correction des cas de test
- Délicate maintenance des cas de test cases

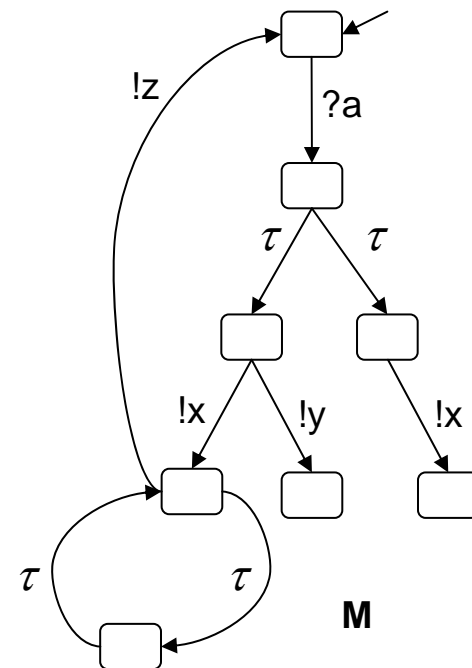
⇒ **La génération automatique des cas de test à partir de la spécifications formelle est très intéressante !**

Théorie du test de conformité



IOLTS $M=(Q^M, \Sigma^M, \rightarrow^M, q_0^M)$, où :

- Q^M un ensemble fini d'états avec q_0^M comme état initial,
- $\Sigma^M = \Sigma^M_i \cup \Sigma^M_o$ un ensemble d'événements observables
 - Σ^M_i un ensemble fini d'entrée (?a),
 - Σ^M_o un ensemble fini de sortie (!a),
- $\rightarrow_M \subseteq Q^M \times (\Sigma^M \cup \{\tau\}) \times Q^M$ la relation de transition
 - τ une action interne (pas dans Σ^M)



Notations

$M = (Q^M, \Sigma^M, \rightarrow^M, q_0^M) : \text{IOLTS}$; $q, q' \in Q^M$; $a \in \Sigma^M$; $\varepsilon, \sigma, \sigma' \in (\Sigma^M)^*$

\Rightarrow décrit des comportements visibles de M :

$q \Rightarrow^\varepsilon q' \equiv q = q' \text{ or } q \rightarrow^{\tau^*} q'$

$q \Rightarrow^a q' \equiv \exists q_1, q_2 \in Q^M / q \Rightarrow^\varepsilon q_1 \rightarrow^a q_2 \Rightarrow^\varepsilon q'$

$q \Rightarrow^{a\sigma'} q' \equiv \exists q_1 \in Q^M / q \Rightarrow^a q_1 \Rightarrow^{\sigma'} q'$

q after σ définit l'ensemble des états accessibles à partir de q par la séquence visible σ :

q after σ $\equiv \{q' \in Q^M \mid q \Rightarrow^\sigma q'\}$

M after σ $\equiv \{q' \in Q^M \mid q_0^M \Rightarrow^\sigma q'\}$

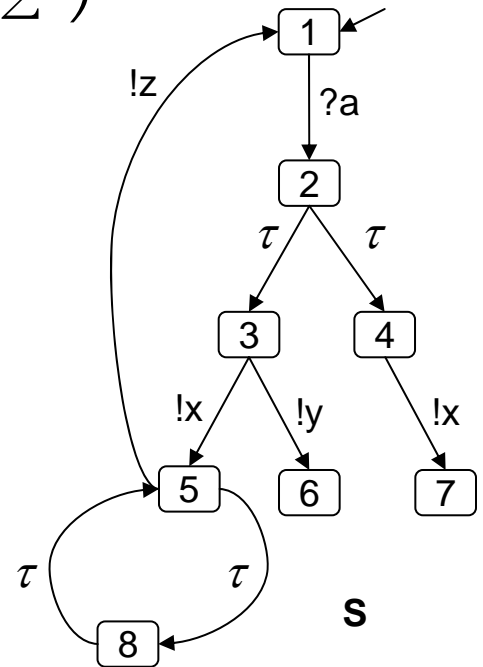
Out_M(q) est l'ensemble des actions de sortie à partir de q :

Out_M(q) $\equiv \{a \in \Sigma_o^M \mid \exists q' \in Q^M, q \rightarrow_M^a q'\}$

Traces(q) décrit les séquences visibles à partir de q :

Traces(q) $\equiv \{\sigma \in (\Sigma^M)^* \mid \exists q' \in Q^M, q \Rightarrow^\sigma q'\}$

Traces(M) décrit les comportements visibles de M : **Traces(M)** $\equiv \text{Trace}(q_0^M)$



Exercices

$M = (Q^M, \Sigma^M, \rightarrow^M, q_0^M) : \text{IOLTS} ; q, q' \in Q^M ; a \in \Sigma^M ; \varepsilon, \sigma, \sigma' \in (\Sigma^M)^*$

$q \Rightarrow^\varepsilon q' \equiv q=q' \text{ or } q \xrightarrow{\tau^*} q'$

$1 \Rightarrow^\varepsilon 1 \equiv ?$

$2 \Rightarrow^\varepsilon 2 \equiv ?$

$3 \Rightarrow^\varepsilon 3 \equiv ?$

$2 \Rightarrow^\varepsilon 3 \equiv ?$

$2 \Rightarrow^\varepsilon 8 \equiv ?$

$5 \Rightarrow^\varepsilon 8 \equiv ?$

$3 \Rightarrow^\varepsilon 8 \equiv ?$

$q \Rightarrow^a q' \equiv \exists q_1, q_2 \in Q^M / q \Rightarrow^\varepsilon q_1 \xrightarrow{a} q_2 \Rightarrow^\varepsilon q'$

$1 \Rightarrow^{?a} 1 \equiv ?$

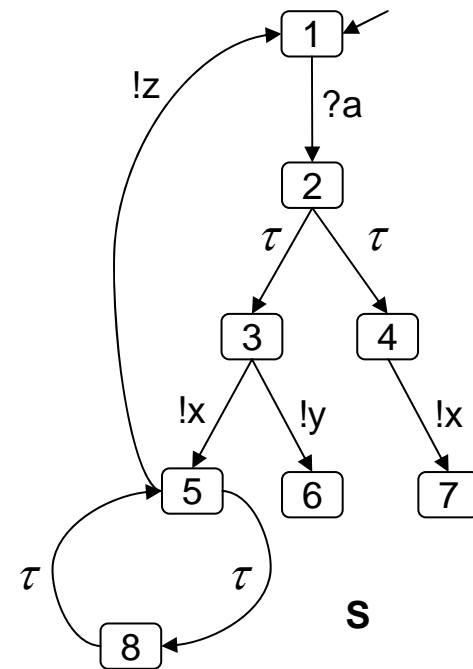
$1 \Rightarrow^{?a} 2 \equiv ?$

$2 \Rightarrow^{!x} 8 \equiv ?$

$q \Rightarrow^{a\sigma'} q' \equiv \exists q_1 \in Q^M / q \Rightarrow^a q_1 \Rightarrow^{\sigma'} q'$

$1 \Rightarrow^{?a!x} 8 \equiv ?$

$5 \Rightarrow^{?a!x!z} 8 \equiv ?$



$2 \Rightarrow^\sigma 8 : \sigma = ?$

$4 \Rightarrow^\sigma 5 : \sigma = ?$

Exercices

$M = (Q^M, \Sigma^M, \rightarrow^M, q_0^M) : \text{IOLTS} ; q, q' \in Q^M ; a \in \Sigma^M ; \varepsilon, \sigma, \sigma' \in (\Sigma^M)^*$

q after σ $\equiv \{q' \in Q^M \mid q \Rightarrow^\sigma q'\}$

1 after ?a.!x \equiv

1 after ?a.!y \equiv

1 after ?a \equiv

1 after !x \equiv

1 after ε \equiv

2 after ε \equiv

3 after ε \equiv

5 after ε \equiv

5 after !z.?a \equiv

M after σ $\equiv \{q' \in Q^M \mid q_0^M \Rightarrow^\sigma q'\}$

M after ?a.!x \equiv

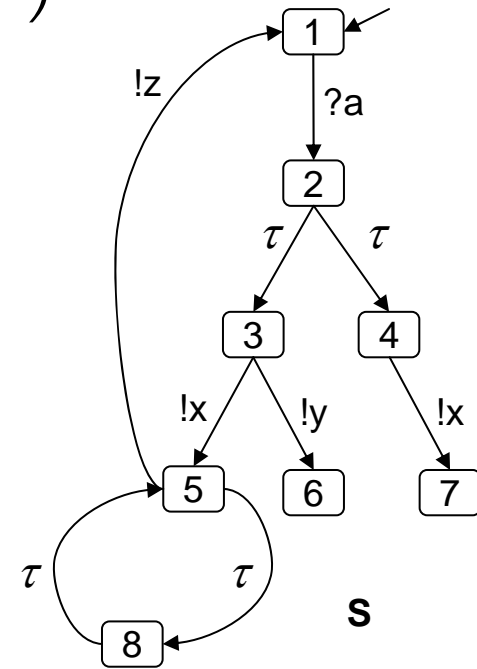
M after ?a.!y \equiv

M after ?a \equiv

M after !x \equiv

M after ε \equiv

M after !z.?a \equiv



Exercices

$M = (Q^M, \Sigma^M, \rightarrow^M, q_0^M) : \text{IOLTS} ; q, q' \in Q^M ; a \in \Sigma^M ; \varepsilon, \sigma, \sigma' \in (\Sigma^M)^*$

$\text{Out}_M(q) \equiv \{a \in \Sigma^M_0 \mid \exists q' \in Q^M, q \rightarrow_M^a q'\}$

$\text{Out}_M(1) \equiv$

$\text{Out}_M(2) \equiv$

$\text{Out}_M(3) \equiv$

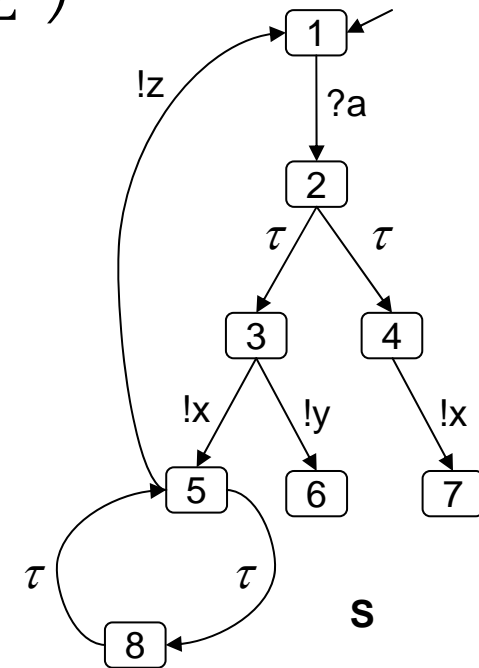
$\text{Out}_M(4) \equiv$

$\text{Out}_M(5) \equiv$

$\text{Out}_M(6) \equiv$

$\text{Out}_M(7) \equiv$

$\text{Out}_M(8) \equiv$



Exercices

$M = (Q^M, \Sigma^M, \rightarrow^M, q_0^M) : \text{IOLTS} ; q, q' \in Q^M ; a \in \Sigma^M ; \varepsilon, \sigma, \sigma' \in (\Sigma^M)^*$

$\text{Traces}(q) \equiv \{ \sigma \in (\Sigma^M)^* \mid \exists q' \in Q^M, q \Rightarrow^\sigma q' \}$

$\text{Traces}(3) \equiv$

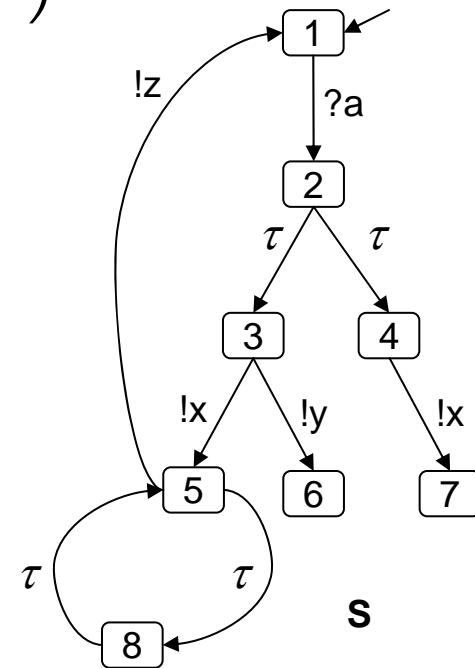
$\text{Traces}(4) \equiv$

$\text{Traces}(6) \equiv$

$\text{Traces}(7) \equiv$

$\text{Traces}(M) \equiv \text{Trace}(q_0^M)$

$\text{Traces}(M) \equiv$



Modèle pour la Spécification : IOLTS $S = (Q^s, \Sigma^s, \rightarrow^s, q_0^s)$

Modèle pour l'Implémentation : IOLTS $IUT = (Q^{IUT}, \Sigma^{IUT}, \rightarrow^{IUT}, q_0^{IUT})$
avec $\Sigma^s_i \subseteq \Sigma^{IUT}_i$ et $\Sigma^s_o \subseteq \Sigma^{IUT}_o$ et input-complet.

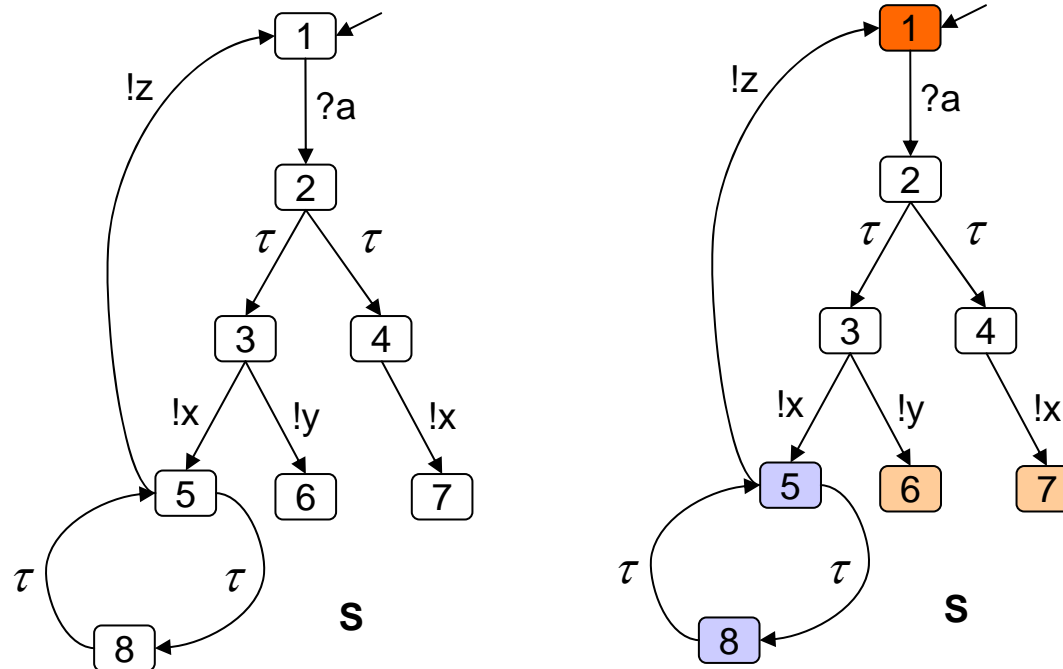
Un IOLTS M est dit **input-complet** (input-enabled) si M accepte toutes les entrées dans tous ses états. Formellement,

$$\forall q \in Q^M, \forall a \in \Sigma^M_i, \exists q' \in Q^M, q \xrightarrow{a} q'$$

Comportement visible : traces + silences

Un testeur observe des traces de l'IST, mais aussi les **silences**.

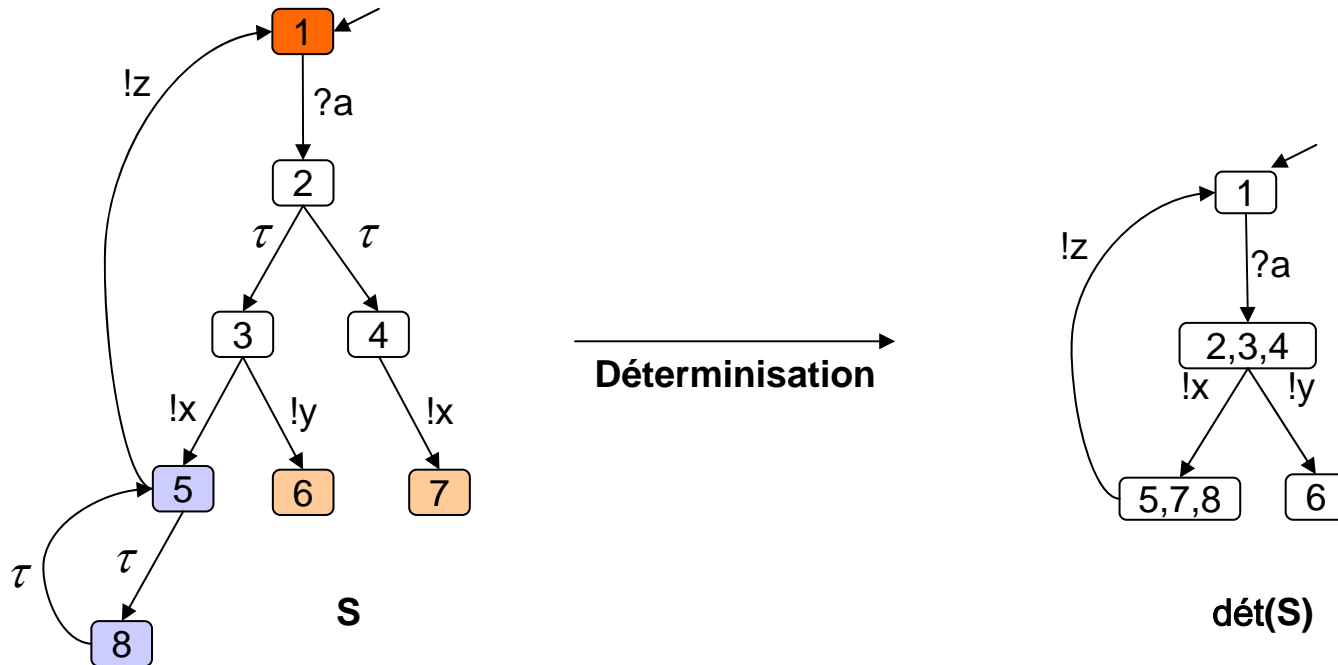
- Silence: absence de comportement 'visible'
- 3 types de silence : **deadlock**, **livelock** et **input-lock**.



Caractérisation des traces d'un IOLTS

Deux séquences avec la même trace ne peuvent être distinguées.

=> Nous considérons le IOLTS déterministe $\text{dét}(S)$ qui a les mêmes traces que S .

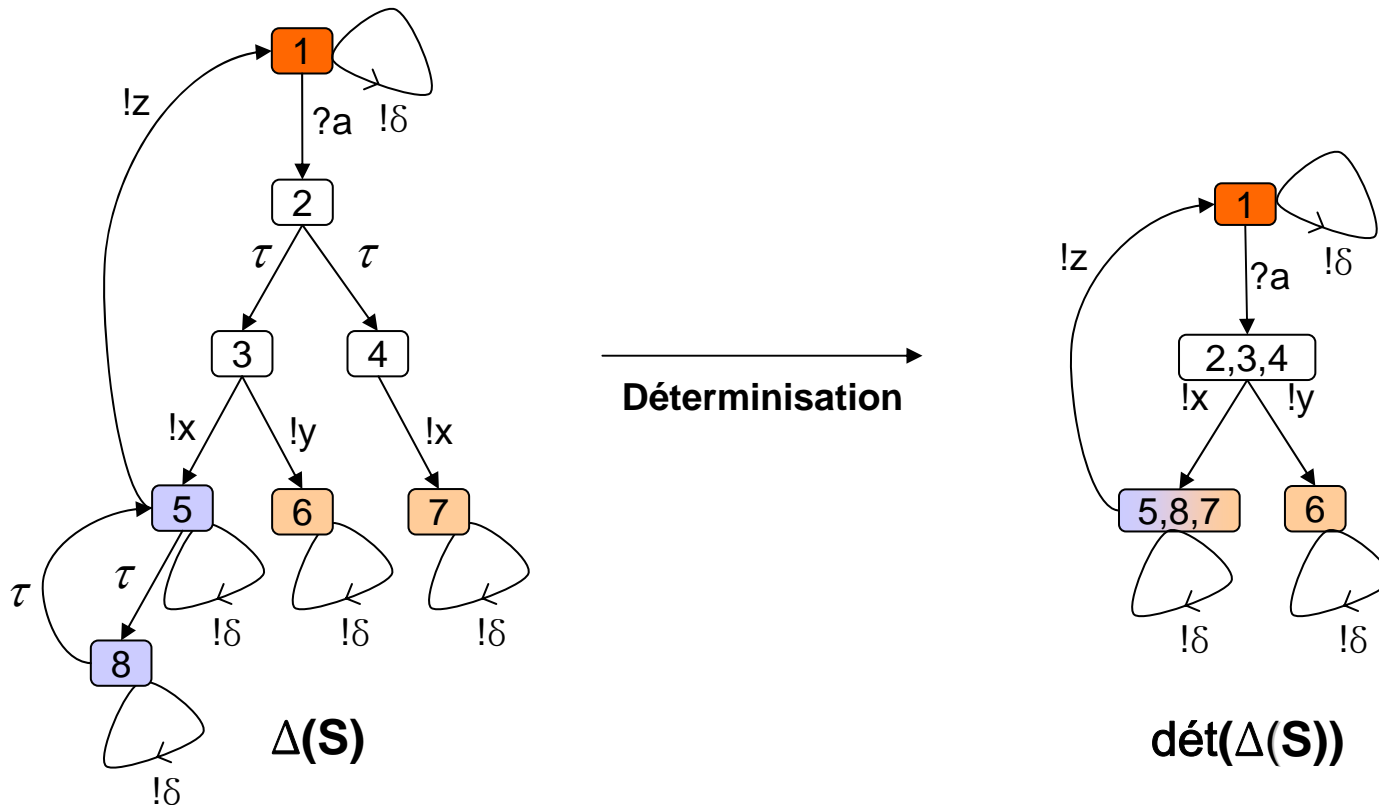


Mais la déterminisation ne préserve pas les silences !

=> Une solution consiste à expliciter les silences [TRE96].

Pour expliciter les silences, l'absence de comportement visible est modélisée par un événement de sortie $!\delta$ (silence) :

- **Automate de suspension $\Delta(S)$** = S + boucle de $!\delta$ sur chaque 'état de silence'
- **Traces Suspendues** de S : **STraces(S)** = Traces($\Delta(S)$).
- $\text{dét}(\Delta(S))$ caractérise les comportements visibles de S.



$IUT \text{ ioco } S \equiv \forall \sigma \in S\text{Traces}(S) : \text{out}(\Delta(IUT) \text{ after } \sigma) \subseteq \text{out}(\Delta(S) \text{ after } \sigma)$

Littéralement :

Pour tout comportement observable σ de S , une possible (observable) sortie de l'IST après σ est aussi une possible (observable) sortie de la spécification après σ .

Intuition :

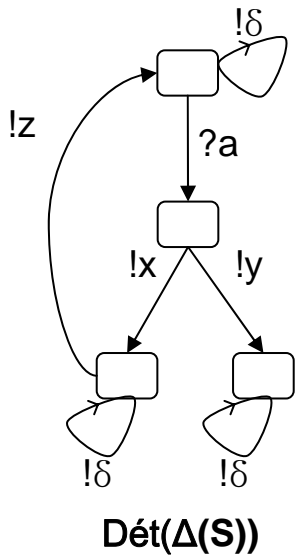
IUT ioco-conforme à S , ssi :

si IUT produit la sortie x après la trace σ ,
alors S peut produire x après σ .

si IUT ne peut pas produire de sortie après la trace σ ,
alors S ne peut pas produire de sortie après σ (silence δ).

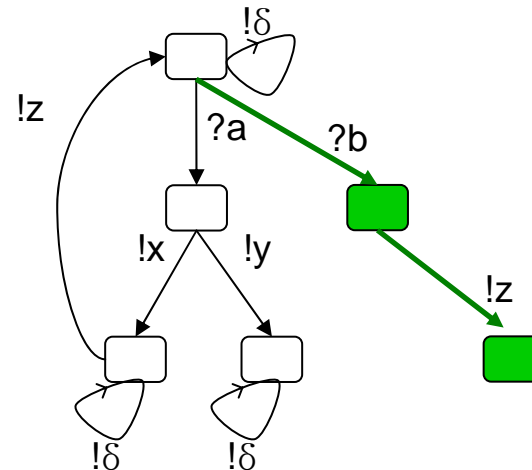
$IUT \text{ ioco } S \equiv \forall \sigma \in STraces(S) : out(\Delta(IUT) \text{ after } \sigma) \subseteq out(\Delta(S) \text{ after } \sigma)$

Pour tout comportement observable σ de S , une possible (observable) sortie de l'IST après σ est aussi une possible (observable) sortie de la spécification après σ .



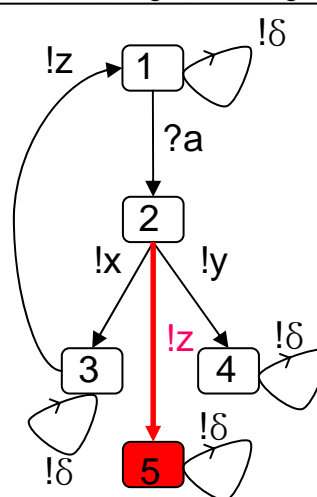
$IUT1 \text{ ioco } S$

$\Delta(IUT1)$



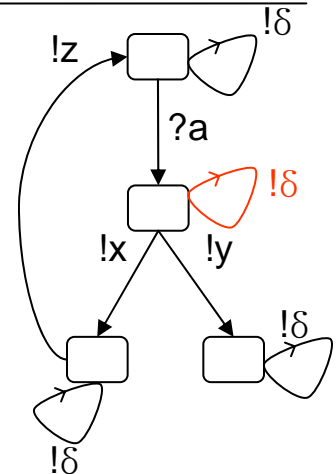
$\neg(IUT2 \text{ ioco } S)$

$\Delta(IUT2)$



$\neg(IUT3 \text{ ioco } S)$

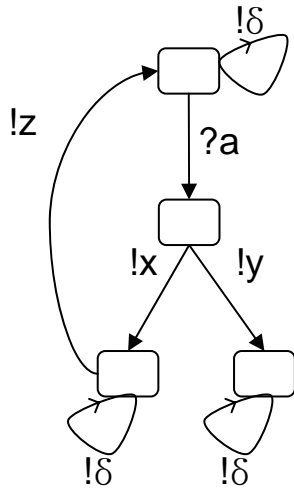
$\Delta(IUT3)$



Relation de Conformité

$IUT \text{ ioconf } S \equiv \forall \sigma \in \text{Traces}(S) \text{ out}(IUT \text{ after } \sigma) \subseteq \text{out}(S \text{ after } \sigma)$

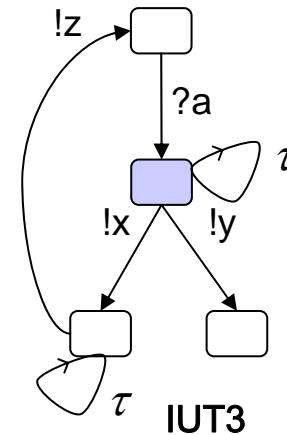
$IUT \text{ ioco } S \equiv \forall \sigma \in \text{STraces}(S) \text{ out}(\Delta(IUT) \text{ after } \sigma) \subseteq \text{out}(\Delta(S) \text{ after } \sigma)$



Det($\Delta(S)$)

$\text{out}(\Delta(IUT3) \text{ after } ?a) = \{!x,!y,!delta\}$

$\text{out}(\Delta(S) \text{ after } ?a) = \{!x,!y\}$



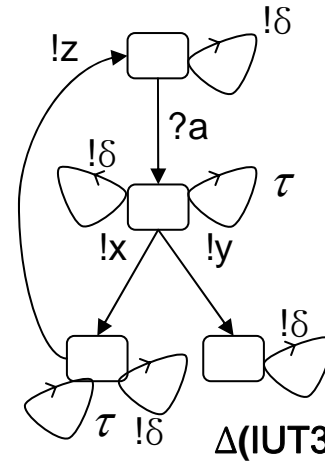
IUT3

IUT3 ioconf S ?

OUI

IUT3 ioco S ?

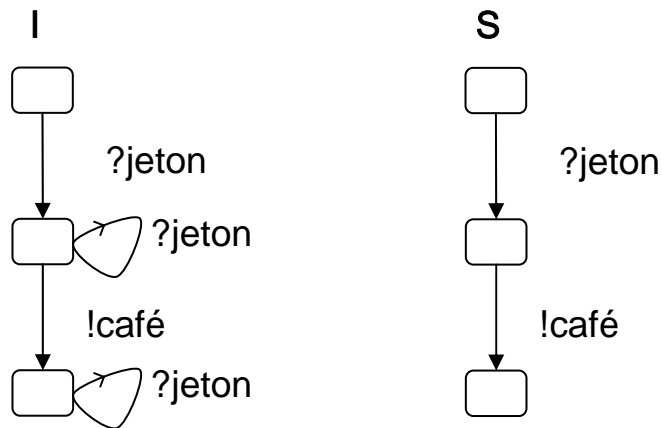
NON



$\Delta(IUT3)$

Exercice

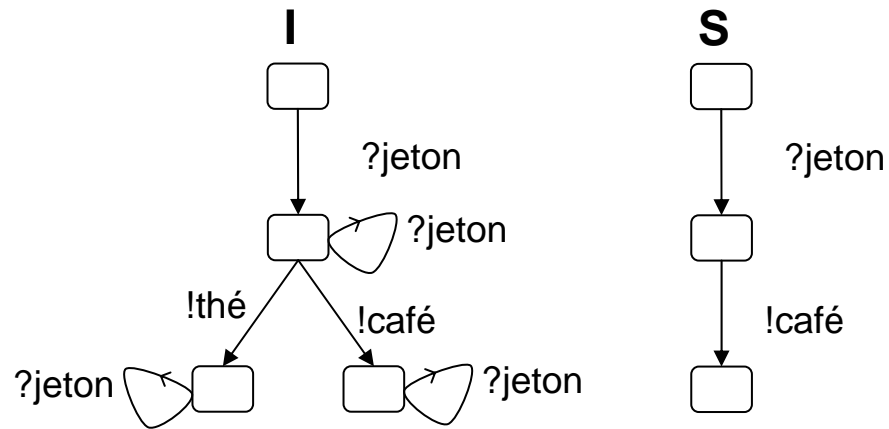
$I \text{ ioco } S \equiv \forall \sigma \in S\text{Traces}(S) : \text{out}(\Delta(I) \text{ after } \sigma) \subseteq \text{out}(\Delta(S) \text{ after } \sigma)$



$I \text{ ioco } S ?$

Exercice

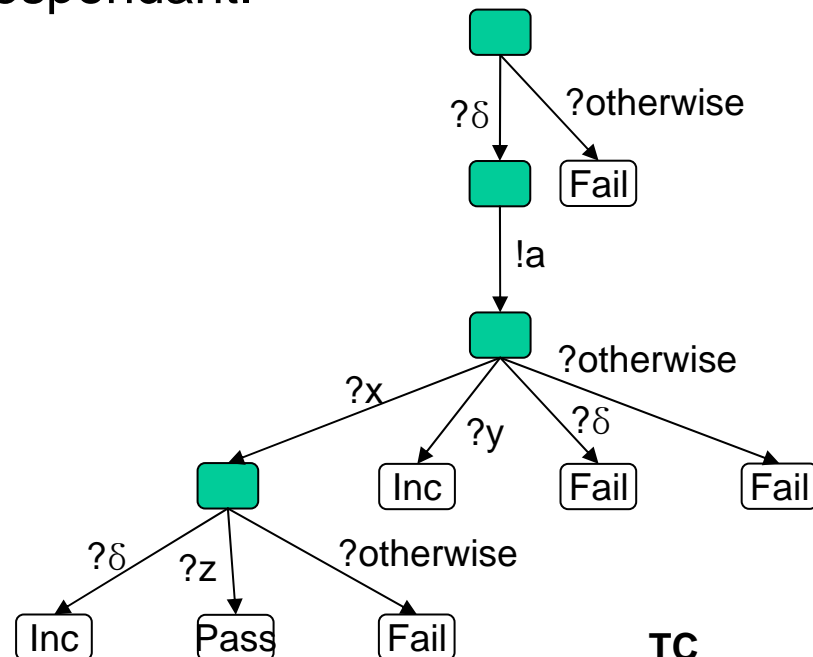
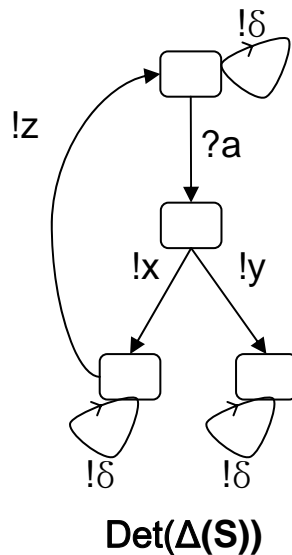
$I \text{ ioco } S \equiv \forall \sigma \in \text{STraces}(S) : \text{out}(\Delta(I) \text{ after } \sigma) \subseteq \text{out}(\Delta(S) \text{ after } \sigma)$



$I \text{ ioco } S ?$

Cas de Test : Exemple

- Un cas de test décrit les interactions entre Testeur et IST (Implémentation).
- Un testeur :
 - exécute un cas de test,
 - observe les réactions/comportements de l'IST,
 - les compare avec les comportements attendus du cas de test et
 - déduit le verdict correspondant.



Cas de Test : Définition

Formellement, un **cas de test** TC est un *IOLTS acyclique* avec une *notion de verdicts* :

$TC = (Q^{TC}, \Sigma^{TC}, \rightarrow^{TC}, q_0^{TC})$ avec $\Sigma^{TC}_o \subseteq \Sigma^s$ et $\Sigma^{TC}_i \subseteq \Sigma^{IUT}_o \cup \{?\delta\}$

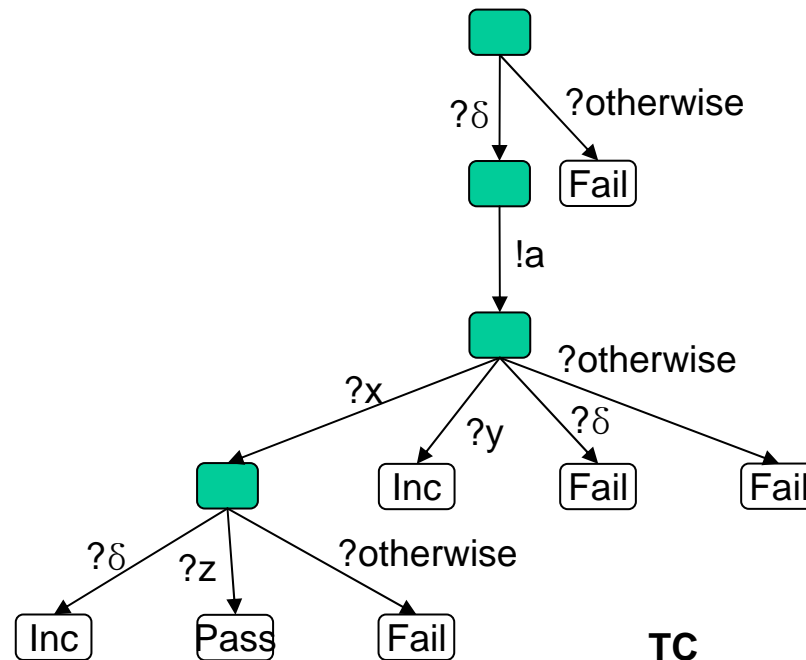
- $Q_{Pass} \subseteq Q^{TC}$ and $Q_{Fail} \subseteq Q^{TC}$ and $Q_{Inconc} \subseteq Q^{TC}$ (verdict states).

[! δ : silence visible / ? δ : expiration du temporisateur]

Cas de Test : Hypothèses

Hypothèses sur les cas de test :

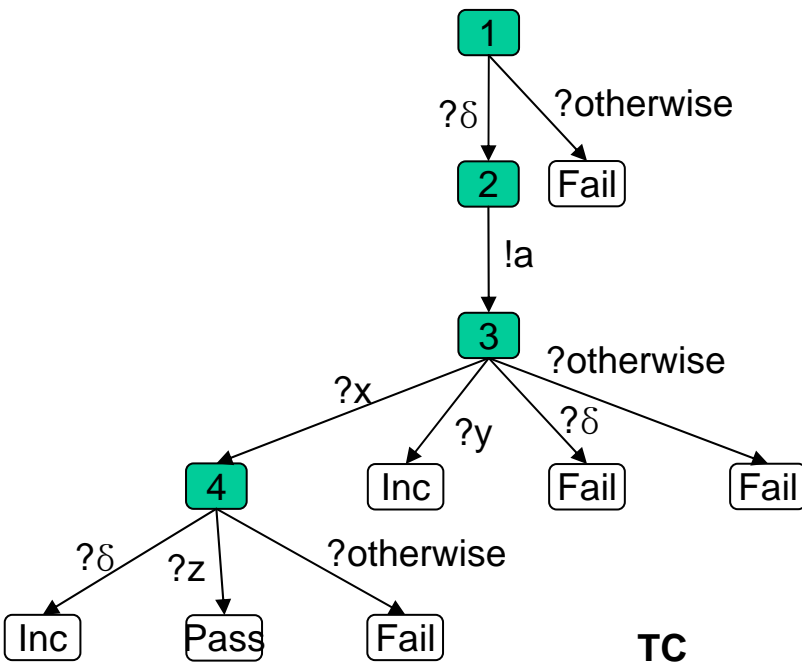
- contrôlable : pas de choix entre une sortie et une autre action,
- Input-complet dans tous les états où une entrée est possible (?otherwise)
- Les états de verdict sont uniquement atteints après des entrées.



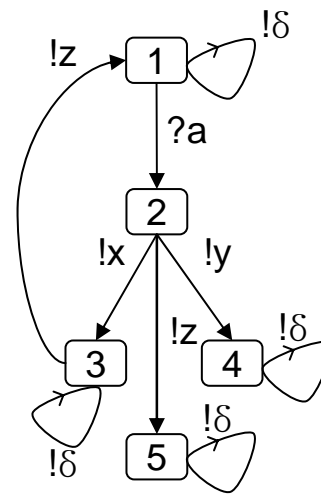
Exécution d'un cas de test

L'**exécution** d'un cas de test TC avec une implémentation IST est modélisée par la **composition parallèle** TC// Δ (IUT) avec une **synchronisation** sur les actions visibles communes (l'événement ?a synchronisé avec !a).

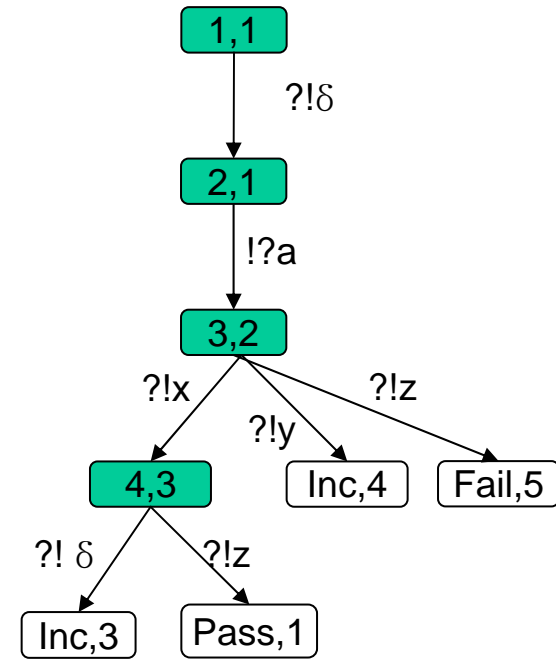
Exemple : Exécution de TC avec IUT2



TC

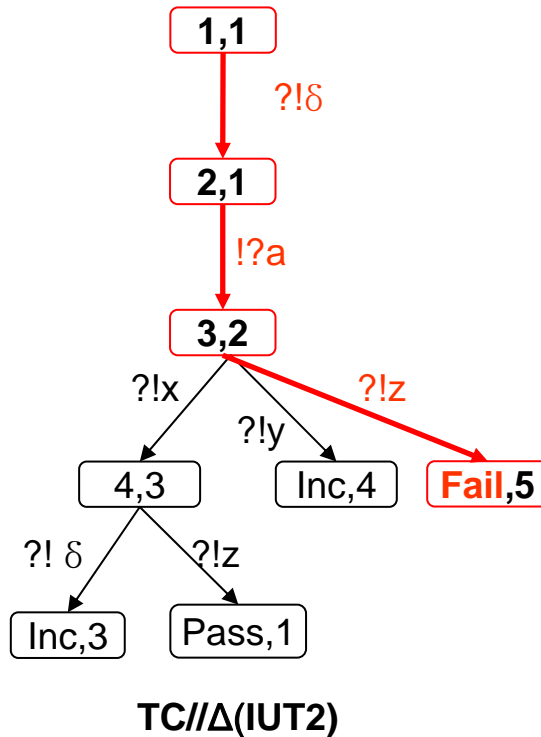


Δ (IUT2)



TC // Δ (IUT2)

Exécution d'un cas de test et verdict



- $TC//\Delta(IUT)$ donne toutes les exécutions possibles.
- $TC//\Delta(IUT)$ s'arrête uniquement dans un état de : $(Q_{Pass} \cup Q_{Fail} \cup Q_{Inc}) \times Q^{IUT}$

Exécution : une **trace maximale** de $TC // \Delta(IUT)$.
Verdict : un état de TC atteint à la fin de l'exécution.

Formellement, nous définissons :

1. Une exécution σ d'un cas de test TC avec l'implémentation IUT est un élément de $MaxTraces(TC//\Delta(IUT))$
2. $verdict(\sigma) = fail$ (resp. *pass*, *inconc*) $\equiv (TC \text{ after } \sigma) \subseteq Q_{Fail}$ (resp. *Pass*, *Inconc*)

Avec :

MaxTraces(M) $\equiv \{\sigma \in (\Sigma M)^* \mid \Gamma(q_0 \text{ after } \sigma) = \emptyset\}$
 $[\Gamma(q)]$ est l'ensemble des actions tirables dans q

Cas de Test et verdicts

Un cas de test peut produire différents verdicts :

Exemple: *TC1* peut produire *Inc* ou *Pass*

Formellement, un rejet possible de l'implémentation IUT par le cas de test *TC* est défini par :

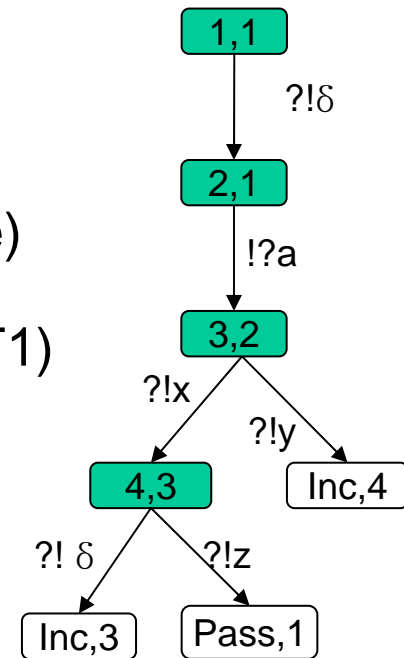
TC may reject IUT

≡

$\exists \sigma \in \text{MaxTraces}(TC // \Delta(\text{IUT})), \text{verdict}(\sigma) = \text{fail}$

('may pass' et 'may inconc' sont définis de la même manière)

Exemple : (*TC1 may pass IUT1*) mais $\neg(\text{TC1 may fail IUT1})$



TC1 // Δ(IUT1)

Propriétés attendues des cas de test

Non biaisé (Soundness) : un cas de test ne doit pas rejeter une implémentation conforme.

Formellement, une suite de test TS est **non biaisée** pour S et $ioco$ ssi :

$$\forall IUT, \forall TC \in TS, TC \text{ may reject } IUT \Rightarrow \neg(IUT \text{ ioco } S)$$

Exhaustivité : une non-conforme implémentation devrait être rejetée par un cas de test.

Formellement, une suite de test TS est **exhaustive** pour S et $ioco$ ssi :

$$\forall IUT, \neg(IUT \text{ ioco } S) \Rightarrow \exists TC \in TS, TC \text{ may reject } IUT$$

Problème:

Etant donnée une spécification S , une relation de conformité R (comme *ioco*), et une propriété P , comment **générer** des cas de test pour une implémentation de S vérifiant P .

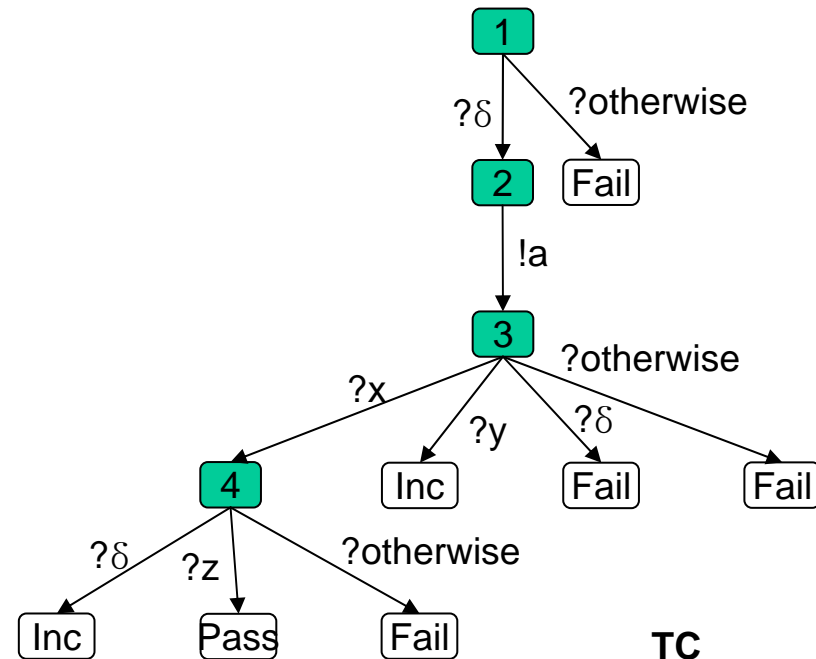
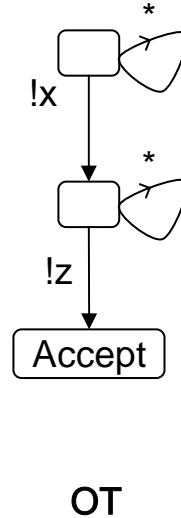
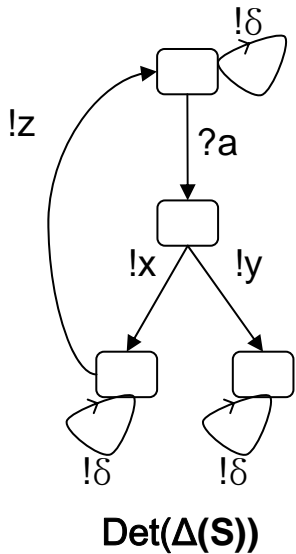
$$\Rightarrow TS = \text{GenTest}(S, R, P).$$

Propriétés peuvent être : *non biaisé*, *exhaustif*, *critère de couverture* (permet de définir une méthode de sélection), .../...

Un exemple de Génération

Spécification: S
 Relation de Conformité: $ioco$
 Propriété: Objectif de Test OT

- Accessibilité de $Det(\Delta(S))$: trace suspendue de S 'acceptée' par OT
- Autres algorithmes de génération



Théorie du test de conformité, en résumé

Modèles pour spécifier :

- Spécification **S**, implémentation **I**, cas de test **TC**, et suite de test **TS**

Formalisation de :

- l'exactitude du test avec une relation (**I ioco S** pour le test de conformité),
 - l'exécution du test (par le testeur) et de ses verdicts (**verdicts(exec(I,TC)), I pass TS...**)
 - Définition de propriétés attendues : **soundness, exhaustively, coverage...**
- ⇒ Définir des méthodes automatiques (algorithme) de dérivation de suite de test pour S avec la bonne propriété P : **TC=gen_test(S,ioco,P)**.

Se ramène à un problème d'accessibilité...

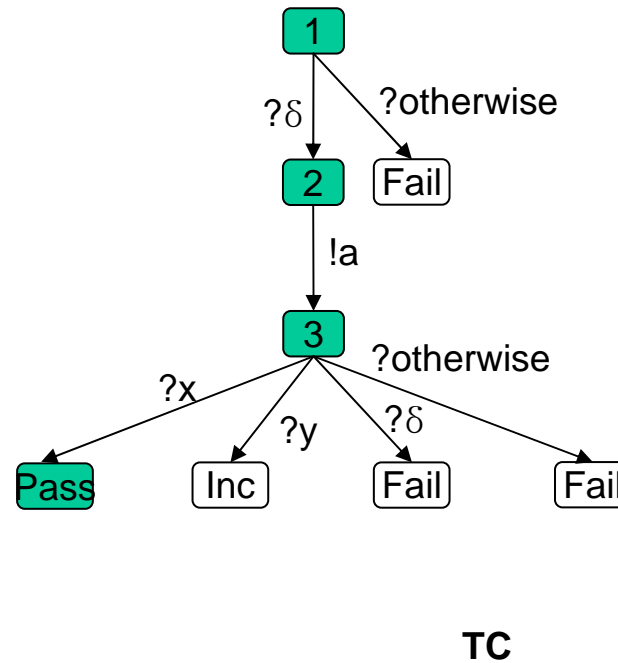
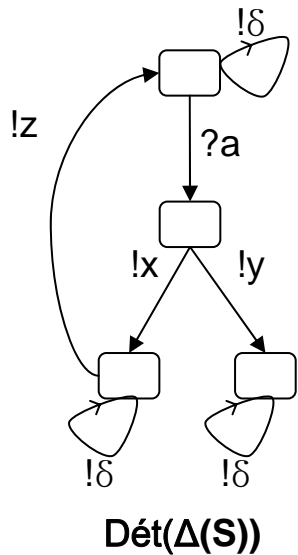
Extension à d'autres type de test (robustesse et ioco_robuste, etc.)

Extension à des systèmes temporisés (automates temporisés et ioco_temporisé)

Quelques exercices...

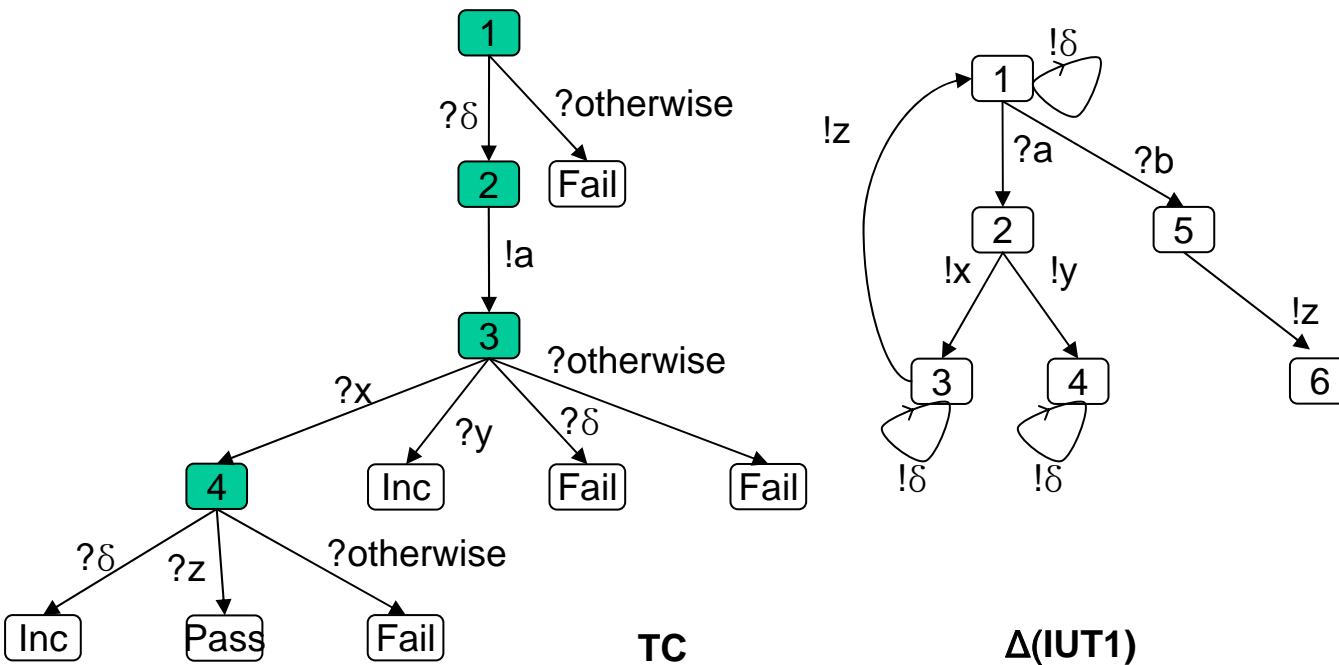
Exercices

Ex1. En considérant la relation de conformité *ioco*, donner un cas de test *TC* pour la spécification *S* vérifiant la propriété « *Après la réception de a, j'envoie un x* ».



Exercices

Ex2. Donner les exécutions possibles du cas de test *TC* avec *IUT1*



Exercices

Ex3. Donner les exécutions possibles du cas de test *TC* avec *IUT2*.

