A Theory of Model-Based Testing with Labelled Transition Systems
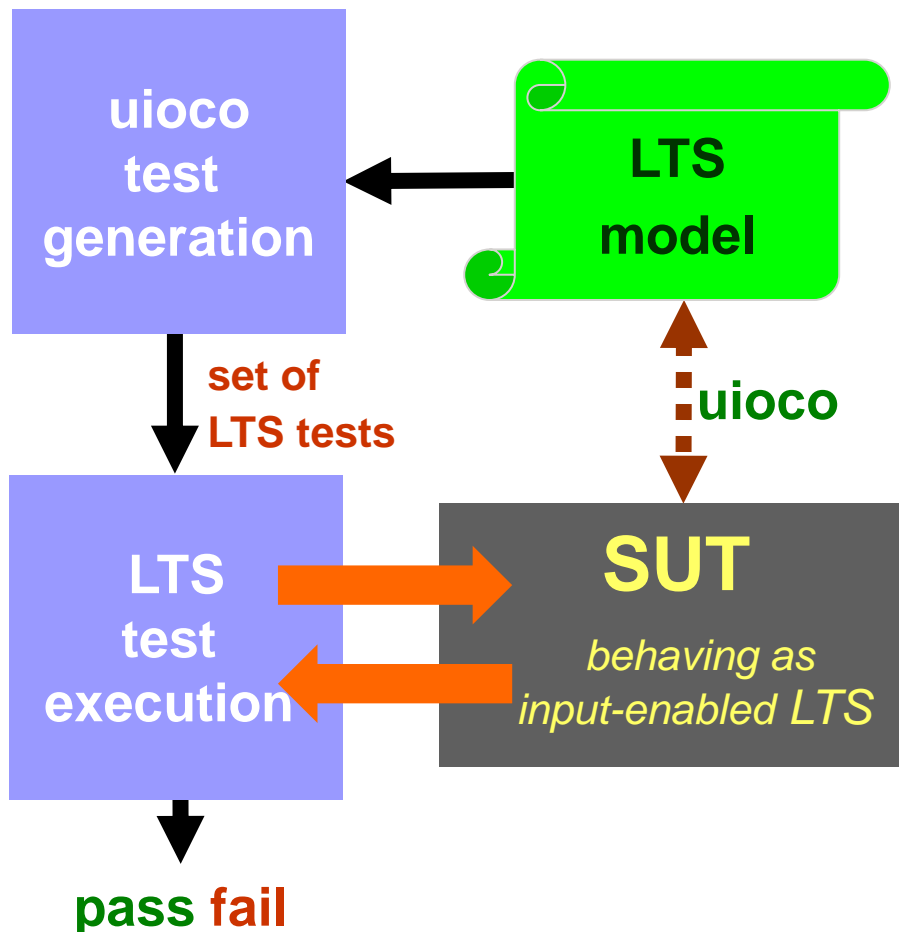
*Test Generation for **uioco***
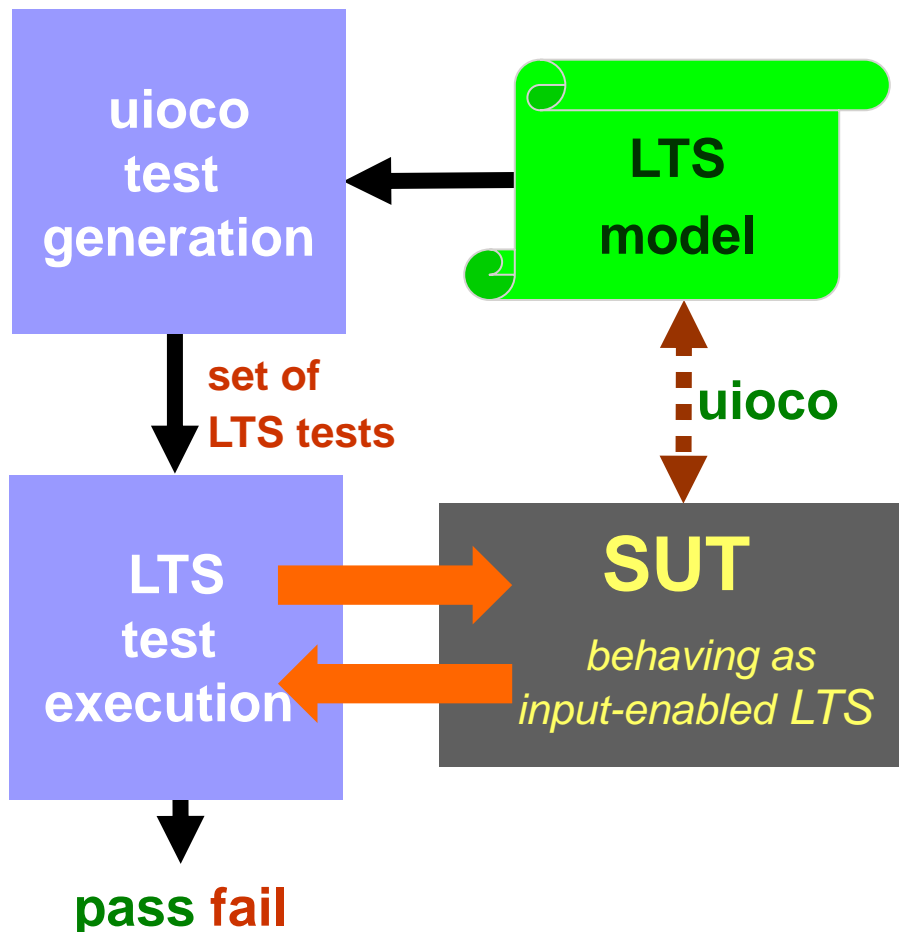
MBT : Labelled Transitions Systems

uioco test generation

LTS model

set of LTS tests

uioco

LTS test execution

SUT

behaving as input-enabled LTS

pass **fail**

MBT with LTS topics:

☞ specification model

☞ system under test (SUT)

☞ SUT model

☞ conformance **uioco**

☞ test cases

☞ test generation

☞ test execution

☞ test result analysis

☞ sound & exhaustive

MBT : Labelled Transitions Systems

uioco test generation

LTS model

set of LTS tests

uioco

LTS test execution

SUT

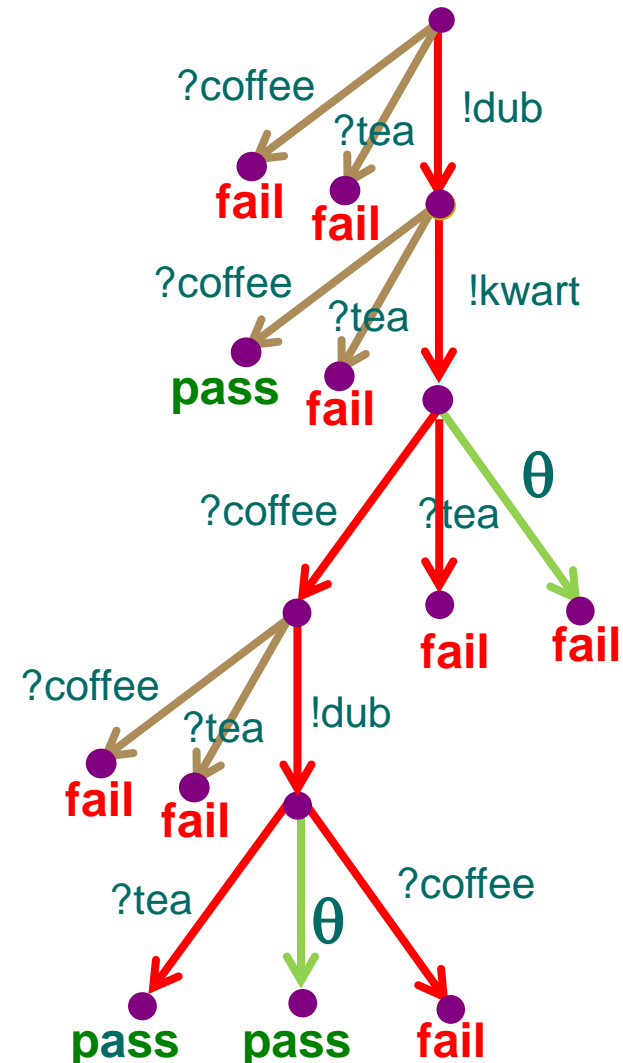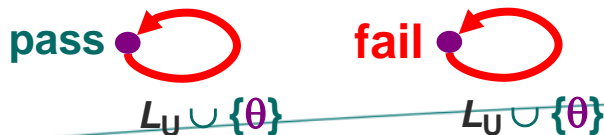*behaving as input-enabled LTS*

pass fail

MBT with LTS topics:

☞ specification model

☞ system under test (SUT)

☞ SUT model

☞ conformance uioco

☞ **test cases**

☞ test generation

☞ test execution

☞ test result analysis

☞ sound & exhaustive

Test Cases

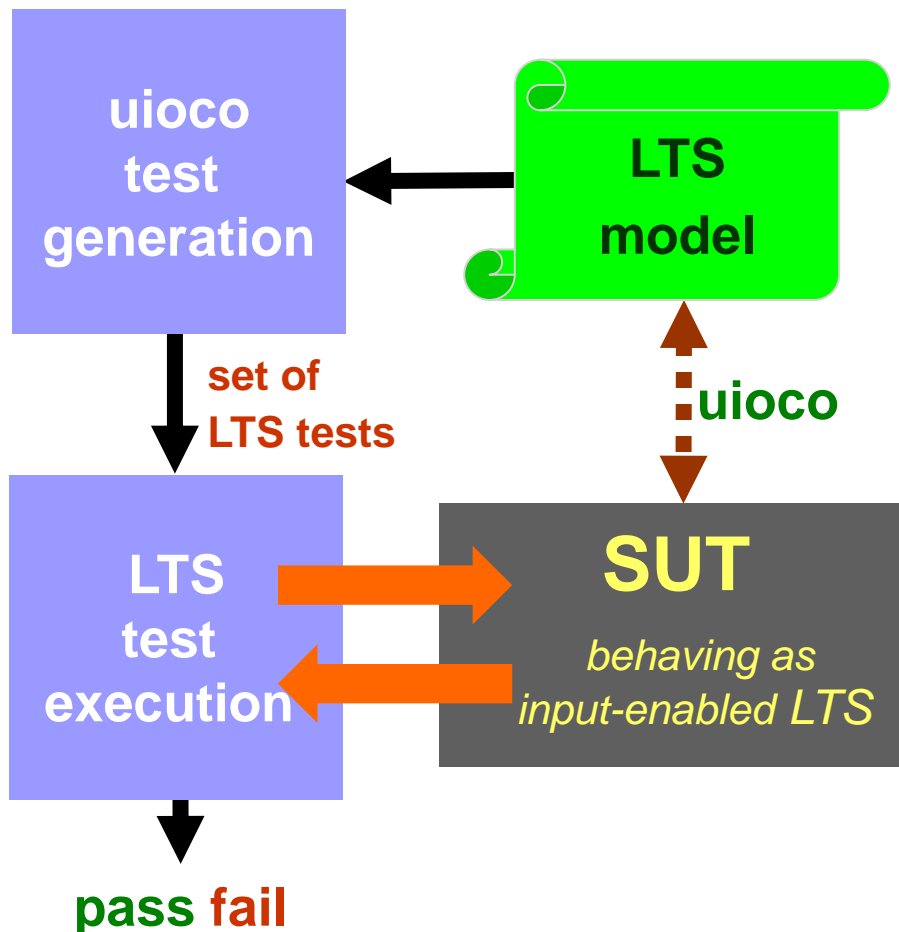*model of a test case*

    = *labelled transition system*

– labels in $\mathbf{L_I} \cup \mathbf{L_U} \cup \{ \theta \}$

– 'quiescence' / 'time-out' label θ

– tree-structured

– finite, deterministic

– sink states **pass** and **fail**

– from each state $\neq$ **pass**, **fail** :

    • either one input **!a** and all outputs **?x**

    • or all outputs **?x** and θ

pass   **fail**

$L_U \cup \{\theta\}$ $L_U \cup \{\theta\}$

?coffee   ?tea   !dub

fail   **fail**

?coffee   ?tea   !kwart

pass   **fail**

?coffee   ?tea θ

fail   **fail**

?coffee   ?tea   !dub

fail   **fail**

?tea θ ?coffee

pass   **pass**   **fail**

MBT : Labelled Transitions Systems

uioco test generation

LTS model

set of LTS tests

uioco

LTS test execution

SUT

behaving as input-enabled LTS

pass fail

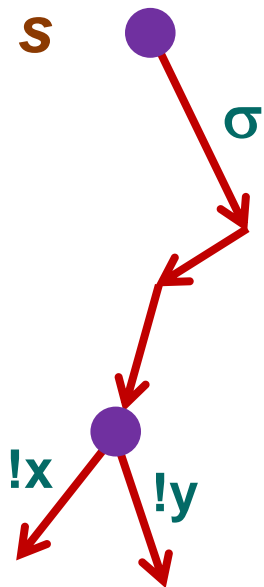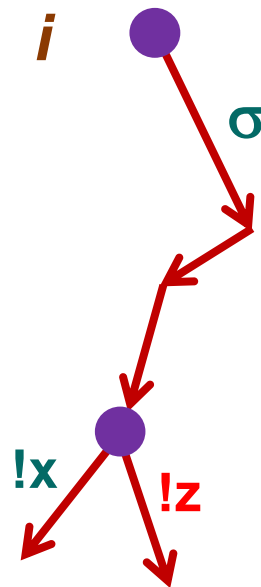**MBT with LTS topics:**

☞ specification model

☞ system under test (SUT)

☞ SUT model

☞ conformance uioco

☞ test cases

☞ **test generation**

☞ test execution

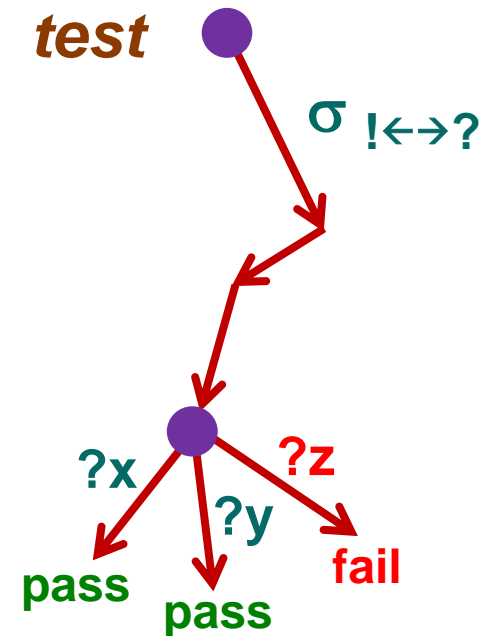☞ test result analysis

☞ sound & exhaustive

Test Generation

$$i \ \text{uioco} \ s \ =_{def} \ \forall \ \sigma \in \textit{Utraces} \ (s) : \ \textit{out} \ (i \ \text{after} \ \sigma) \subseteq \textit{out} \ (s \ \text{after} \ \sigma)$$

s

σ

!x !y

out (**s** after σ)

= { !x, !y }

i

σ

!x !z

out (**i** after σ)

= { !x, !z }

test

σ !↔?

?x ?y ?z

pass pass fail

out (**test** after σ)

= { !x, !y, !z }

Test Generation

$$i \;\; \textbf{uioco} \;\; s \;\;\; =_{\textbf{def}} \;\; \forall \, \sigma \in \textit{Utraces}\,(s) : \;\; \textit{out}\,(i \text{ after } \sigma) \subseteq \textit{out}\,(s \text{ after } \sigma)$$

s

σ

!x !y

δ

i

σ

!x !z

δ

test

$\sigma_{\, ! \leftrightarrow ?}$

?x ?y ?z θ

pass

pass pass fail pass

out (*s* after σ)

= { !x, !y, δ }

out (*i* after σ)

= { !x, !z, δ }

out (*test* after σ)

= { !x, !y, !z, θ }
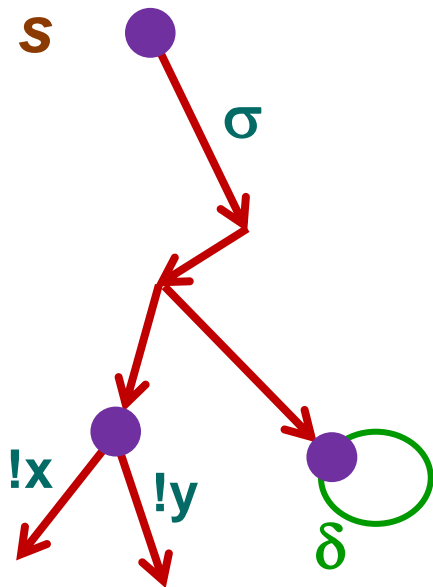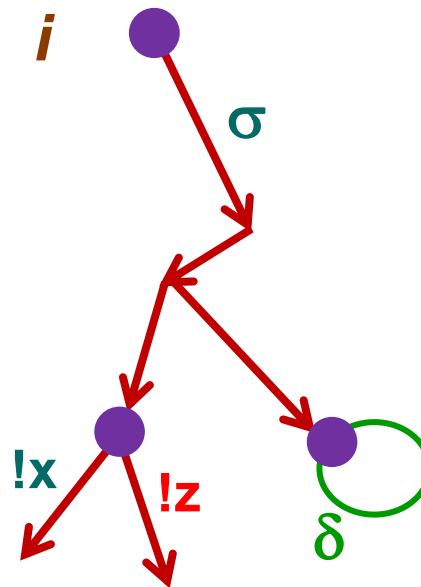
Test Generation Algorithm : *uioco*
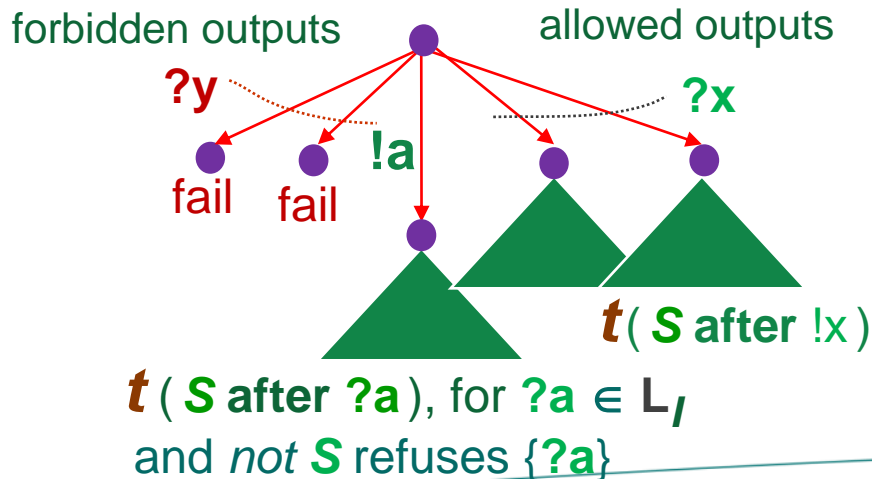
Algorithm to generate a test case $t(S)$

from a transition system state set S, with $S \neq \varnothing$, and initially $S = s_0$ **after** ε

Apply the following steps recursively, non-deterministically :

1	end test case

- **pass**

2	supply input **!a**

forbidden outputs   allowed outputs

?y ...   **!a** ... **?x**

fail   fail

$t(S$ **after** !x$)$

$t(S$ **after** ?a$)$, for **?a** $\in$ L$_I$
and *not* S refuses {**?a**}

3	observe all outputs

forbidden outputs   allowed outputs

?y θ **?x**

fail   fail

$t(S$ **after** !x$)$
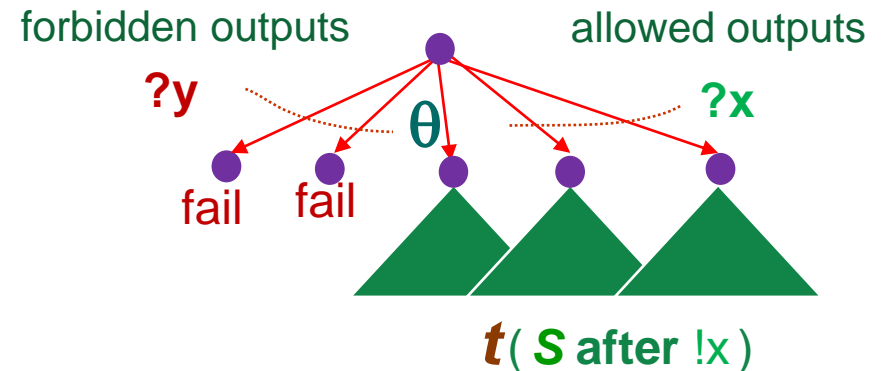
allowed outputs (or δ):   **!x** $\in out(S)$

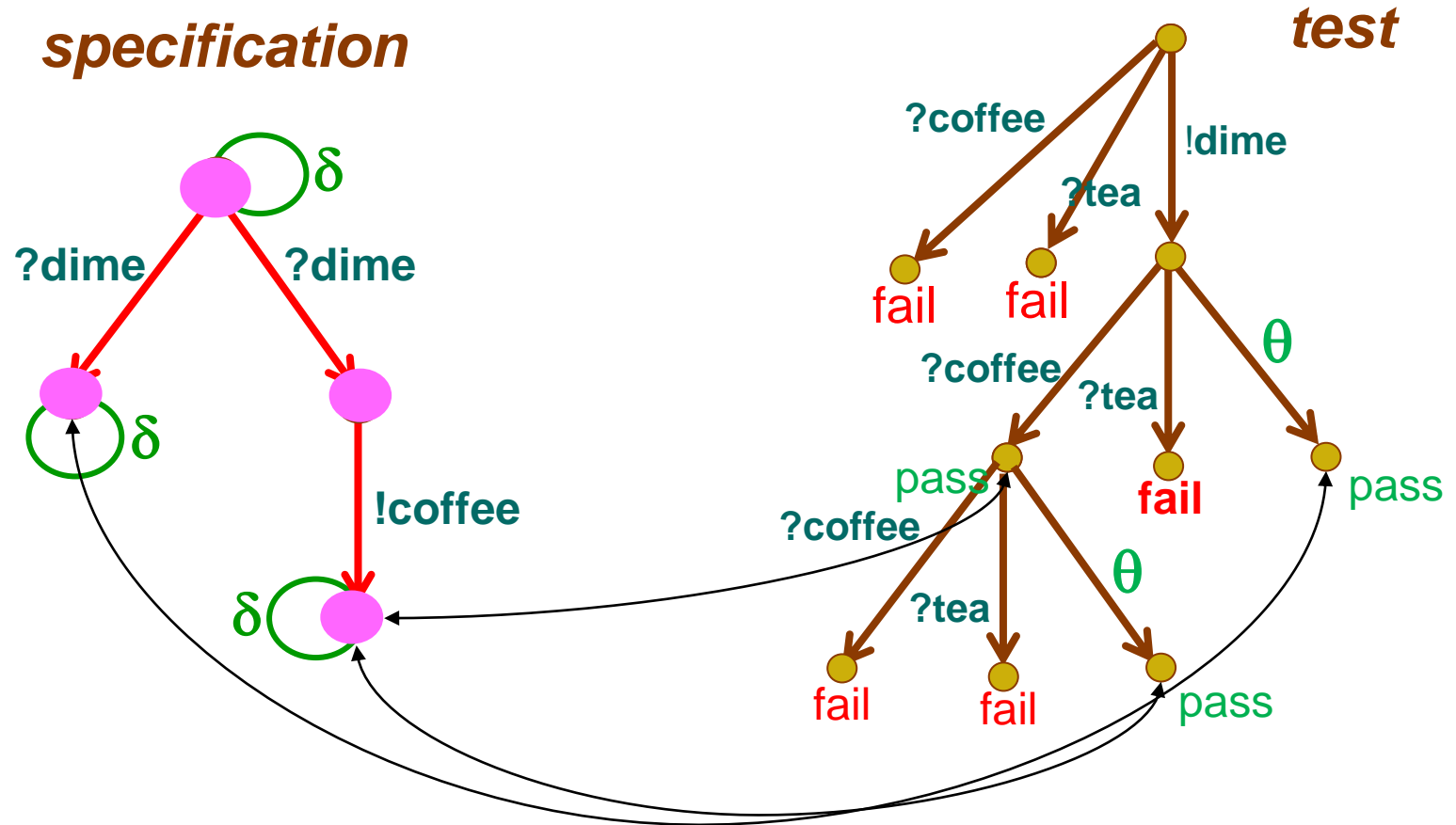forbidden outputs (or δ):   **!y** $\notin out(S)$

11

Example : *uioco* Test Generation

s
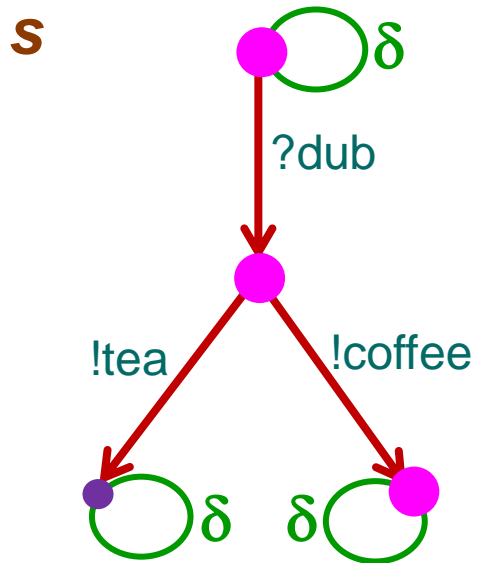
test

L = { ?dub, !coffee, !tea, !choc }
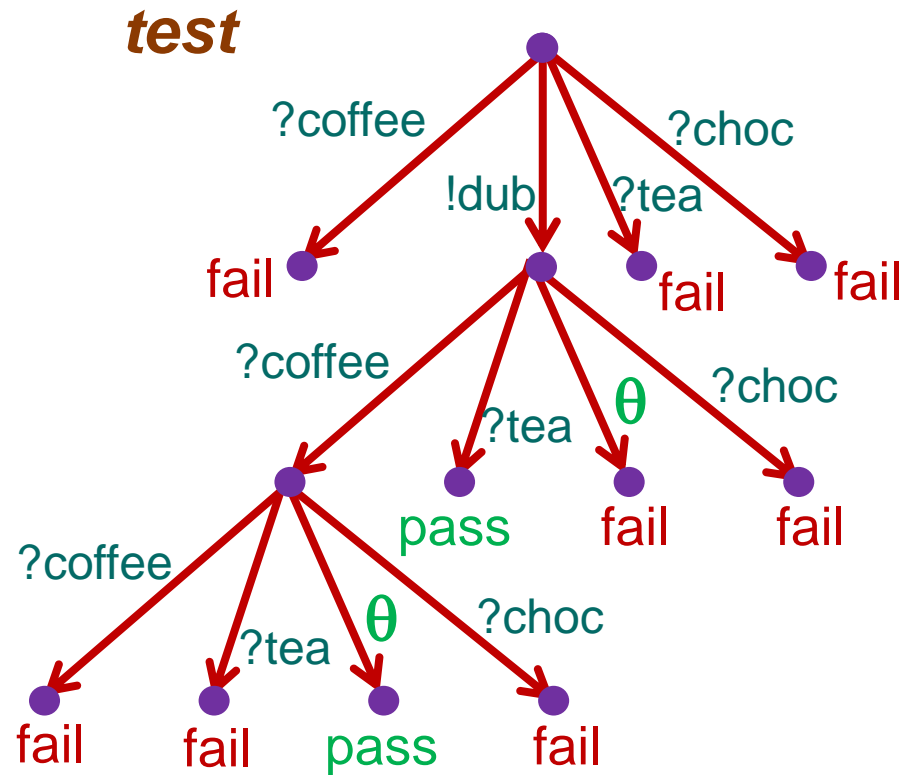
Example : *uioco* Test Generation

Example : *uioco* Test Generation

Equation solver for $y^2 = x$

test

specification **s**

! y
$(y^2 = x)$

? x (x >= 0)

?otherwise

fail

!9

?otherwise

fail

?3

pass

?-3

?otherwise

fail

!4

?otherwise

fail

?2

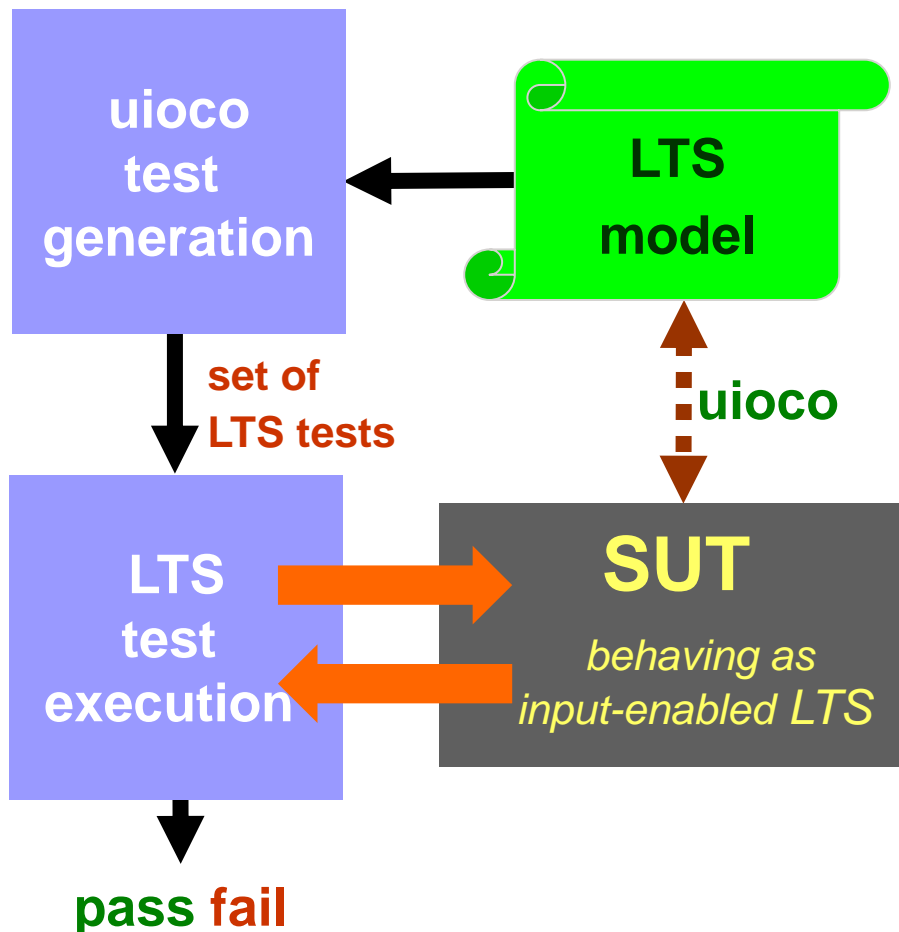pass

?-2

pass
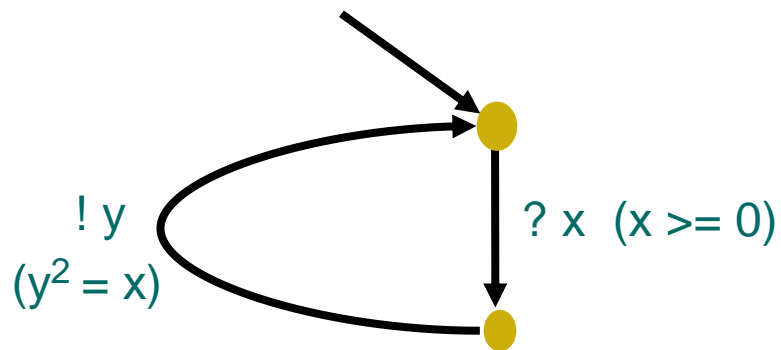
To cope with non-deterministic behaviour, tests are not linear traces, but trees
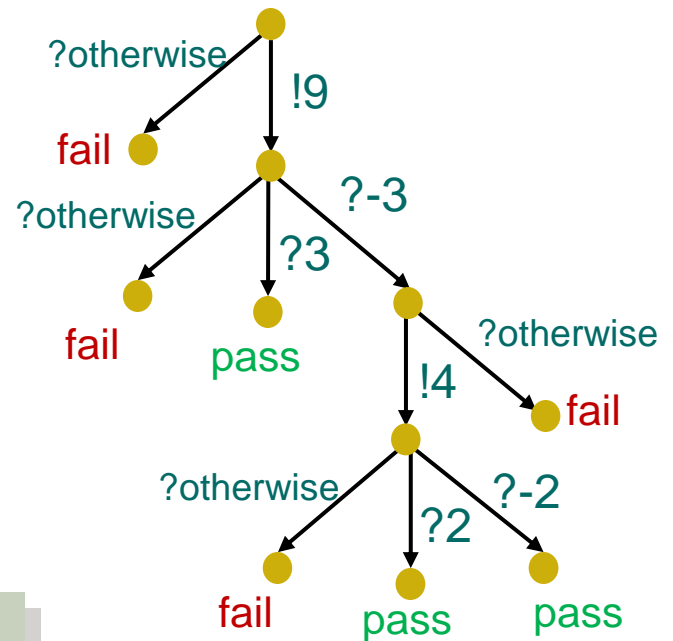
15

MBT : Labelled Transitions Systems

uioco test generation

LTS model

set of LTS tests

uioco

LTS test execution

SUT

behaving as input-enabled LTS

pass fail

MBT with LTS topics:

☞ specification model

☞ system under test (SUT)

☞ SUT model

☞ conformance uioco

☞ test cases

☞ test generation

☞ **test execution**

☞ test result analysis

☞ sound & exhaustive

Test Execution

Test execution = all possible parallel test runs of test **t**
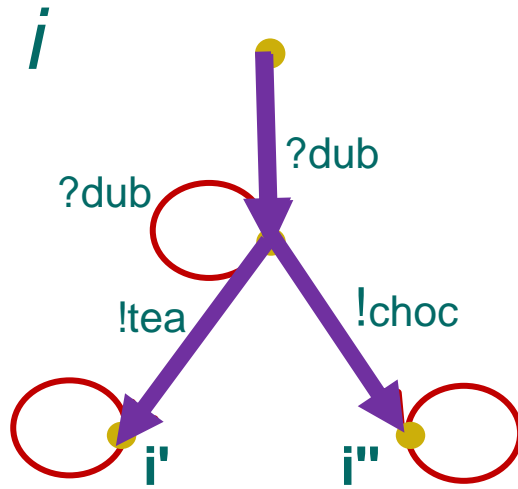
with implementation **i** going to state pass or fail

Test run : $\quad$ **t** ⌉| **i** $\xrightarrow{\sigma}$ pass ⌉| **i'** $\quad$ or $\quad$ **t** ⌉| **i** $\xrightarrow{\sigma}$ fail ⌉| **i'**

$$\frac{t \xrightarrow{a} t', \quad i \xrightarrow{a} i'}{t \,⌉|\, i \xrightarrow{a} t' \,⌉|\, i'} \qquad \frac{i \xrightarrow{\tau} i'}{t \,⌉|\, i \xrightarrow{\tau} t \,⌉|\, i'}$$

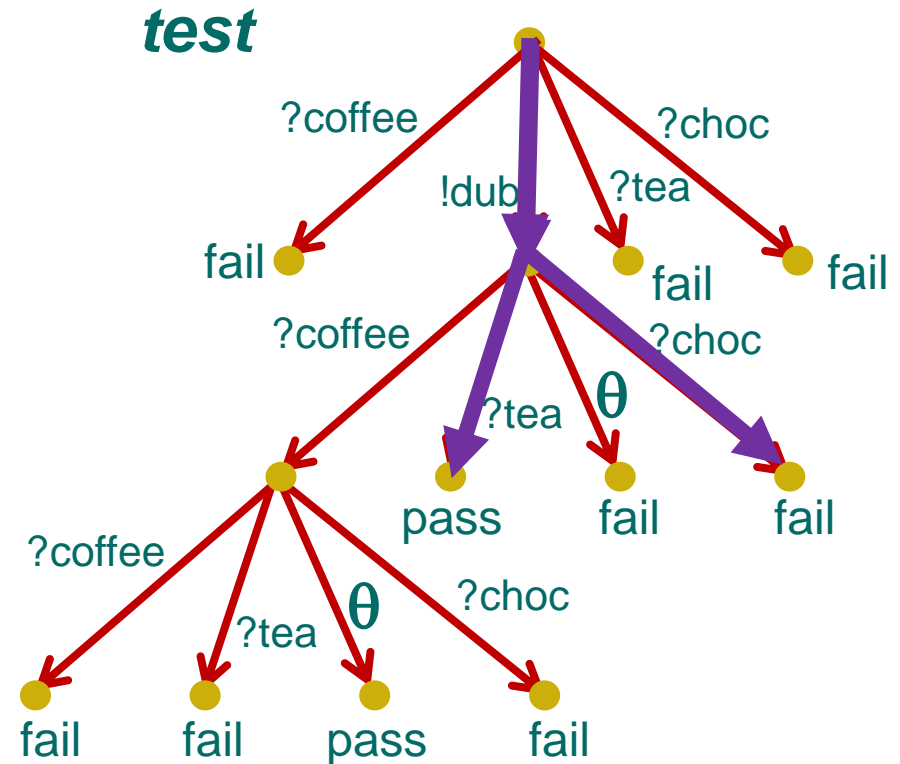$$\frac{t \xrightarrow{\theta} t', \quad i \xrightarrow{\delta} i'}{t \,⌉|\, i \xrightarrow{\theta} t' \,⌉|\, i'}$$
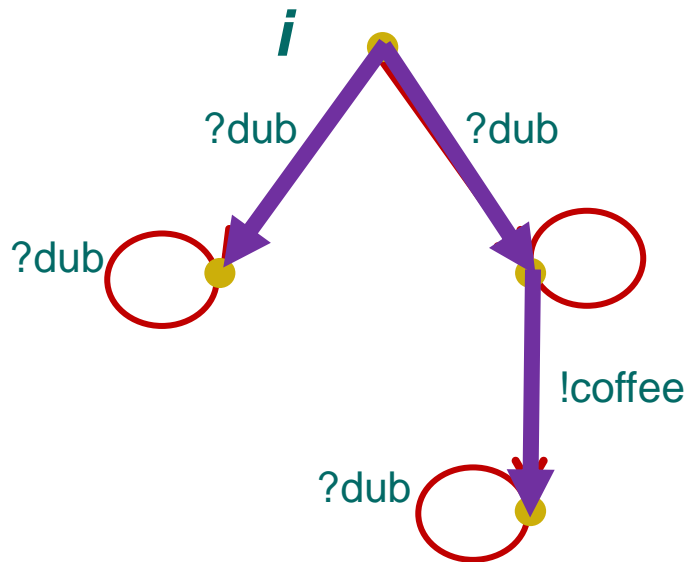
Example : Test Execution

i

test

Two test runs :

$t \rceil\mid i \xrightarrow{\text{dub tea}} pass \rceil\mid i'$

$t \rceil\mid i \xrightarrow{\text{dub choc}} fail \rceil\mid i''$

i fails **t**

18

Example : Test Execution

i

test

?dub   ?dub

?dub

!coffee

?dub

?coffee
?tea   !dub

fail
fail

?coffee
?tea   θ

?coffee   fail   pass
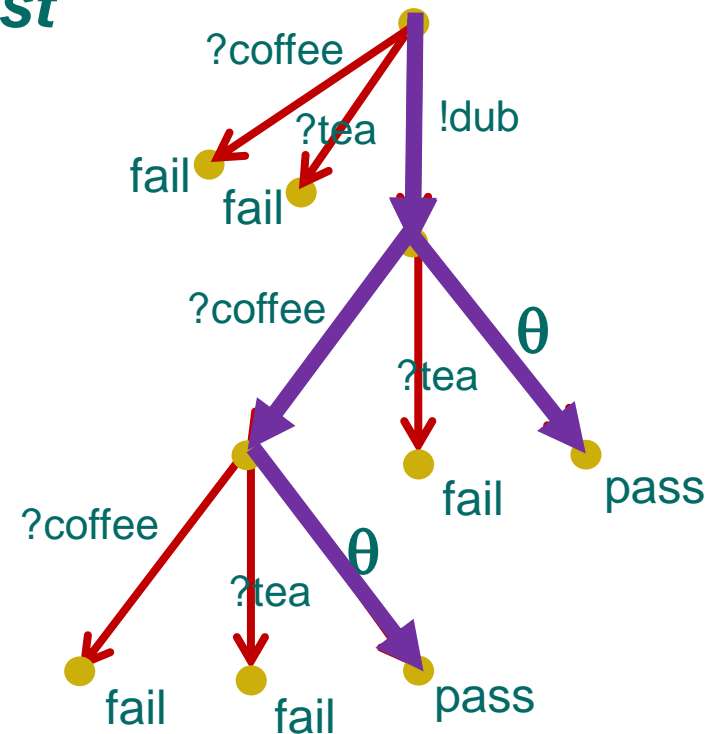
?tea   θ

fail   fail   pass

Two test runs :

$$t \rceil| \ i \xrightarrow{\text{dub} \ \theta} \text{pass} \rceil| \ i'$$

$$t \rceil| \ i \xrightarrow{\text{dub coffee} \ \theta} \text{pass} \rceil| \ \textbf{i'}$$

i passes **t**

Example : Test Execution
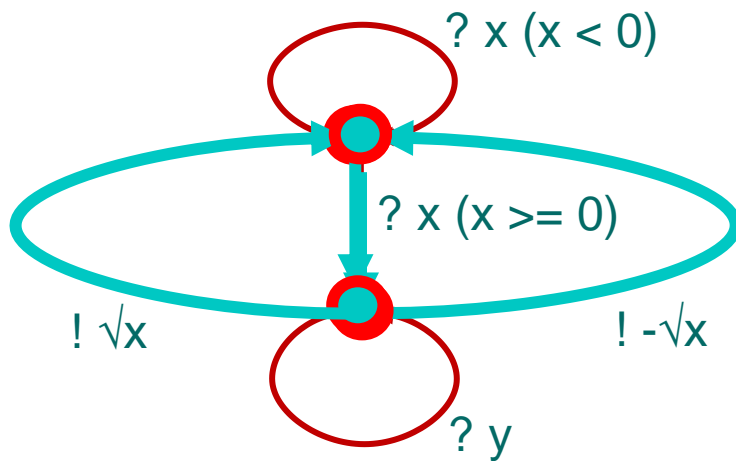
implementation ⌐| test
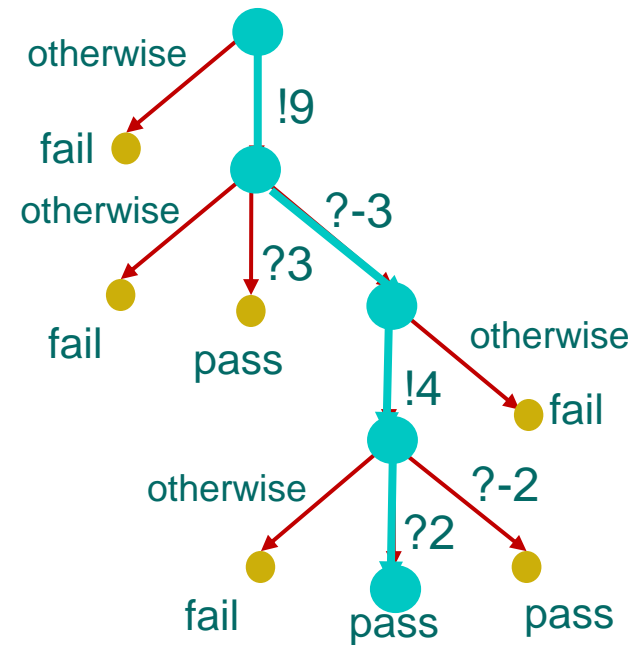
i passes **t**

MBT : Labelled Transitions Systems
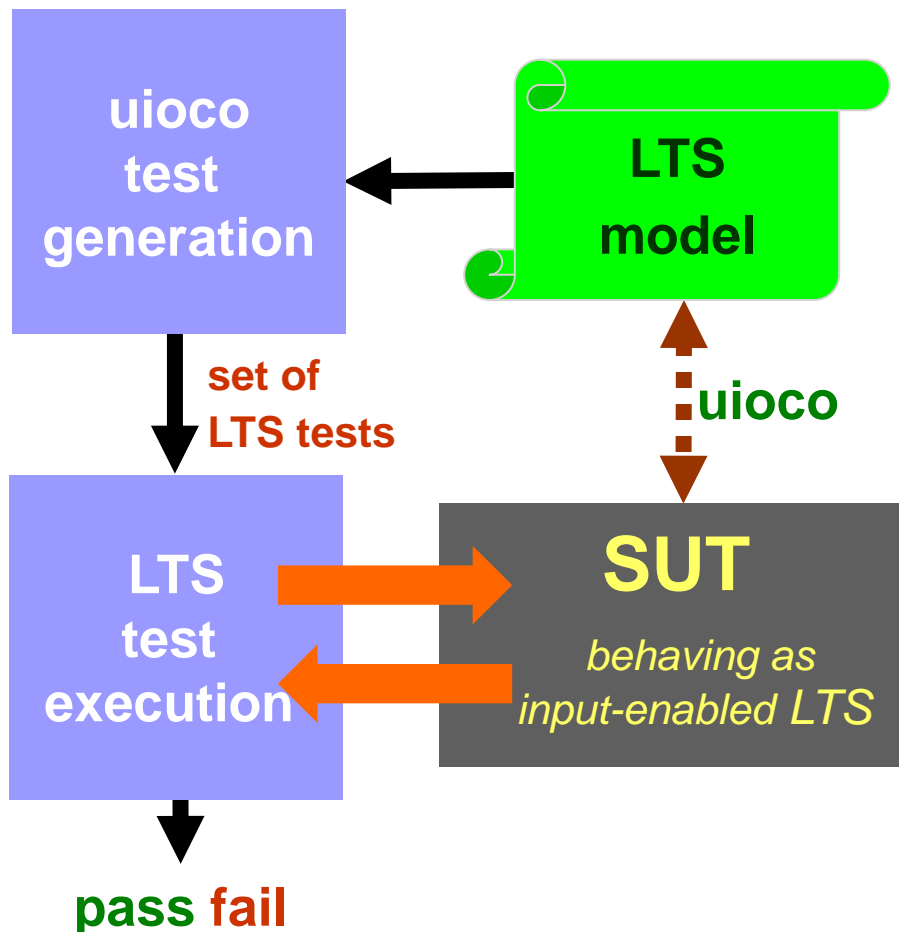
uioco test generation

set of LTS tests

LTS model

uioco

SUT

behaving as input-enabled LTS

LTS test execution

pass fail

MBT with LTS topics:

☞ specification model

☞ system under test (SUT)

☞ SUT model

☞ conformance uioco

☞ test cases

☞ test generation

☞ test execution

☞ **test result analysis**

☞ **sound & exhaustive**

21

Validity of Test Generation

For every test **t** generated with algorithm we have:

- Soundness :
  t will never fail with correct implementation

$$\mathbf{i} \ \mathbf{uioco} \ \mathbf{s} \quad \text{implies} \quad \mathbf{i} \ \mathbf{passes} \ \mathbf{t}$$

- Exhaustiveness :
  each incorrect implementation can be detected
  with a generated test t

$$\mathbf{i} \ \cancel{\mathbf{uioco}} \ s \quad \text{implies} \quad \exists \ \mathbf{t} : \ \mathbf{i} \ \mathbf{fails} \ \mathbf{t}$$

Soundness and Exhaustiveness

s

test

?coffee

!dime

?tea

fail

fail

?coffee

?tea

θ

fail

pass
fail

?coffee

?tea

θ

fail
pass

fail

pass

still
sound

not sound
anymore

?dime

?dime

!coffee

δ

δ

δ
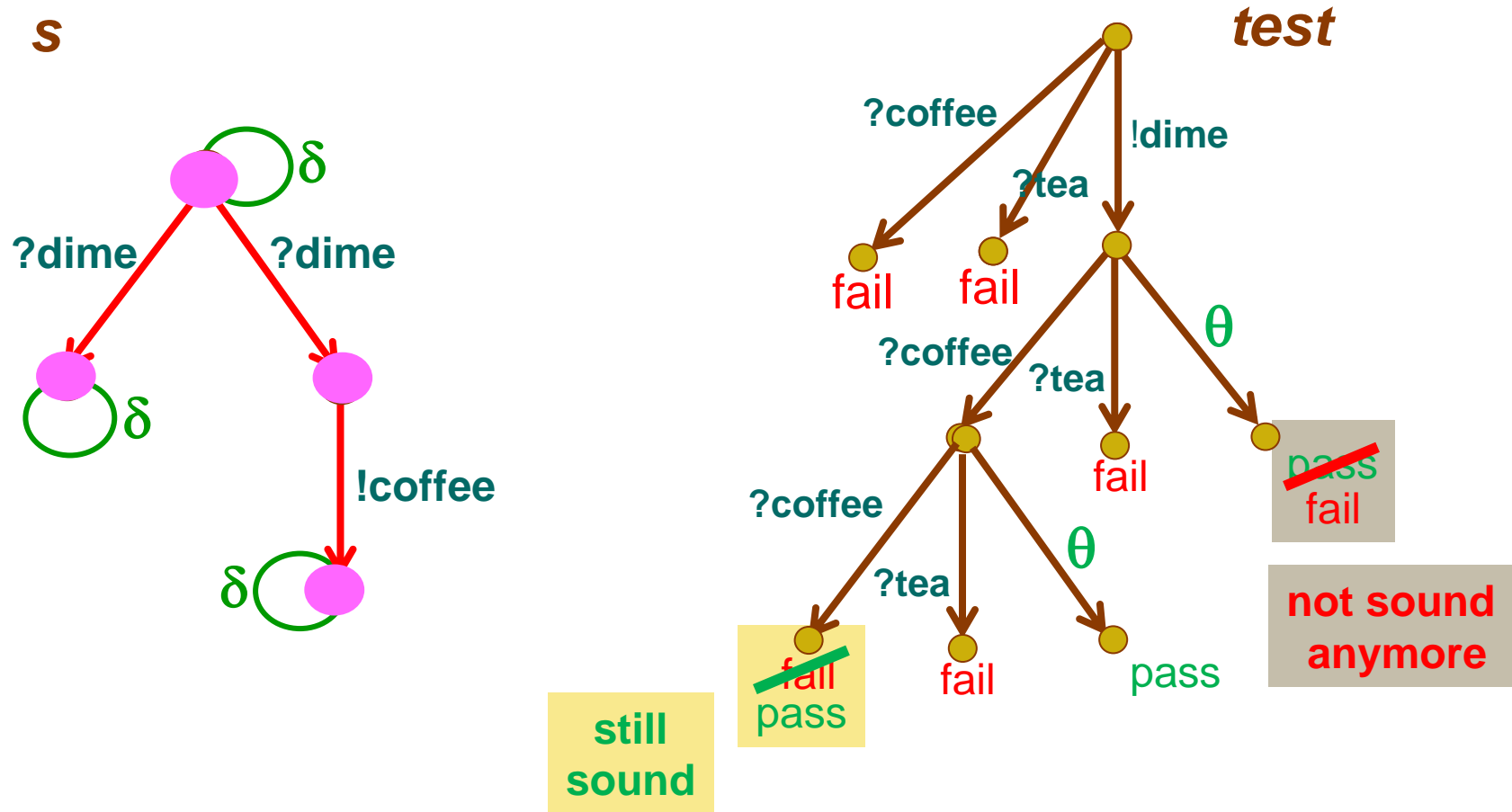
MBT with *uioco* is Sound and Exhaustive

gen : LTS
$\rightarrow \wp$ **(TTS)**

S $\in$ LTS

i uioco s

t ⫮ SUT

SUT
behaving as input-enabled LTS

pass fail
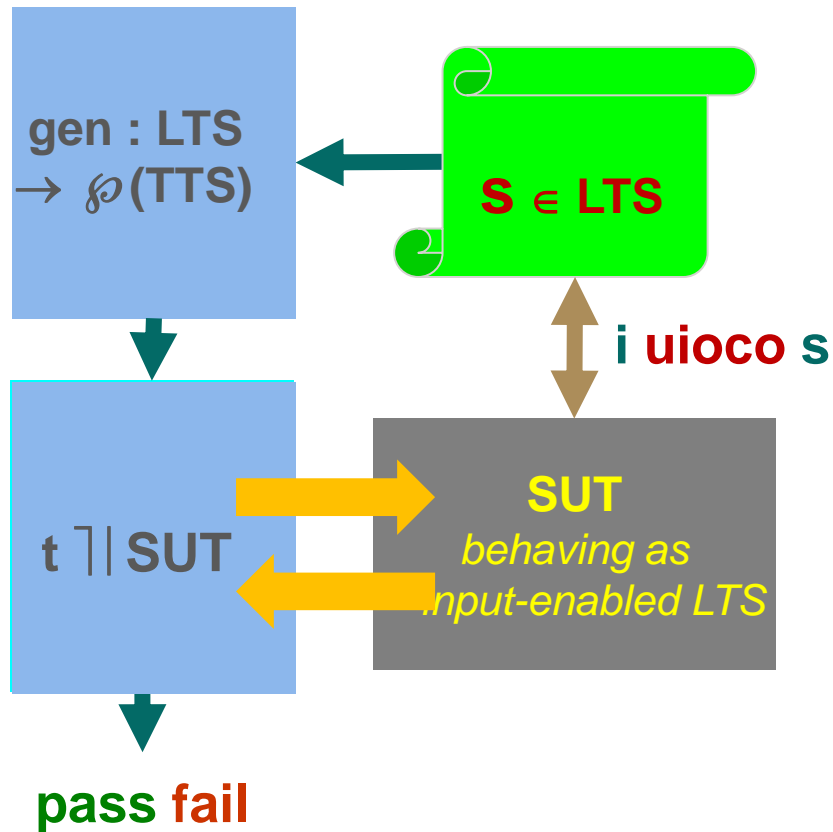
Testability assumption :

$\forall$**SUT**$\in$IMP . $\exists$**m**$_{SUT}$ $\in$IOTS .
   $\forall$t$\in$TESTS .
      SUT passes t $\Leftrightarrow$ **m**$_{SUT}$ **passes** t

Prove soundness and exhaustiveness:

$\forall$**m**$\in$IOTS .
   ($\forall$t$\in$gen(**s**) . **m passes** t)
      $\Leftrightarrow$ **m uioco s**

SUT comforms to s

exhaustive ⇑ ⇓ *sound*

SUT passes gen(s)

MBT : Model-Based Testing
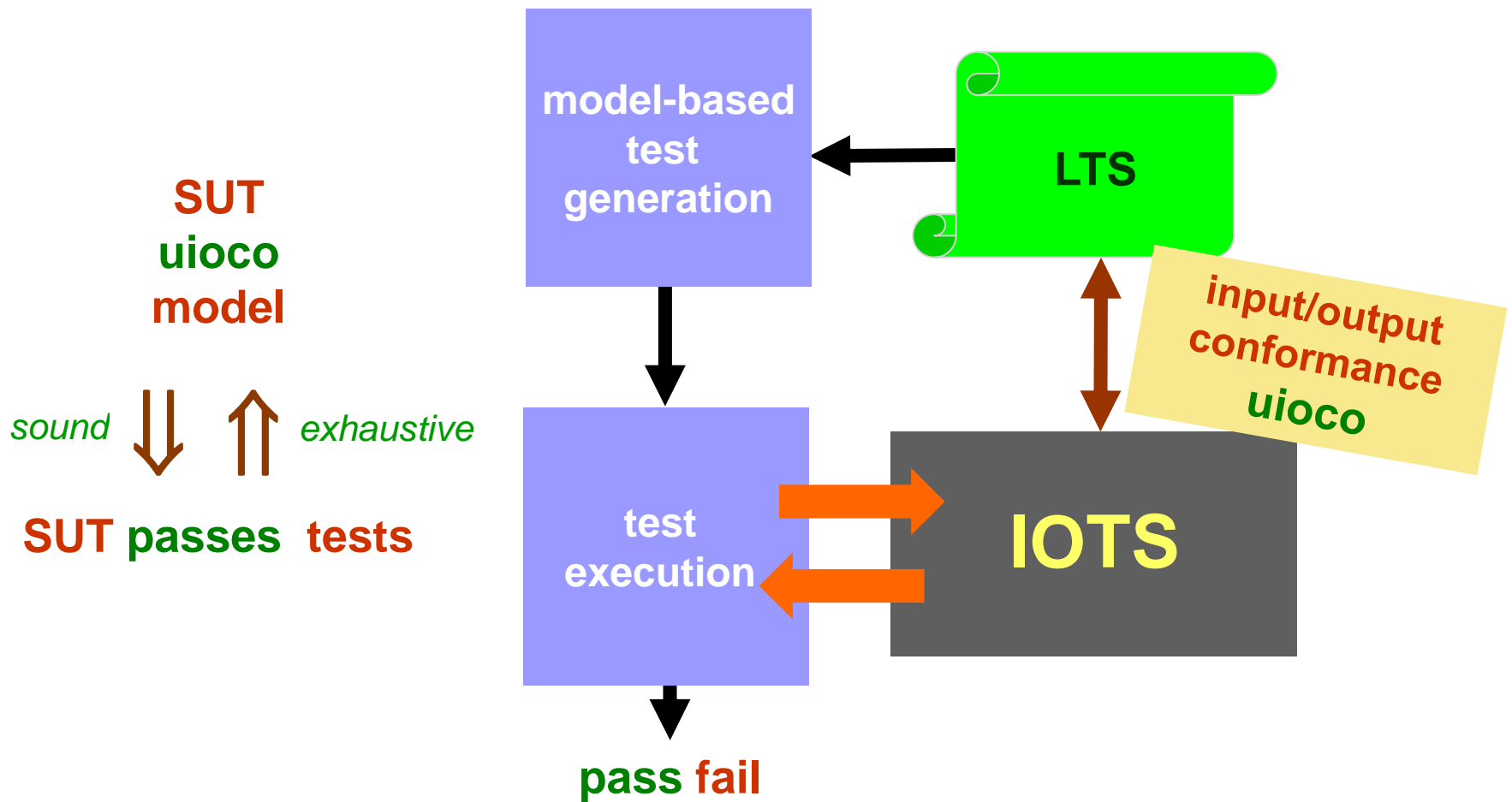
SUT
uioco
model

sound ⇓ ⇑ exhaustive

SUT **passes** tests

model-based test generation

LTS

input/output conformance uioco

test execution

IOTS

pass fail

MBT : Labelled Transitions Systems

uioco test generation

LTS model

LTS test execution

set of LTS tests

uioco

SUT

behaving as input-enabled LTS

pass fail
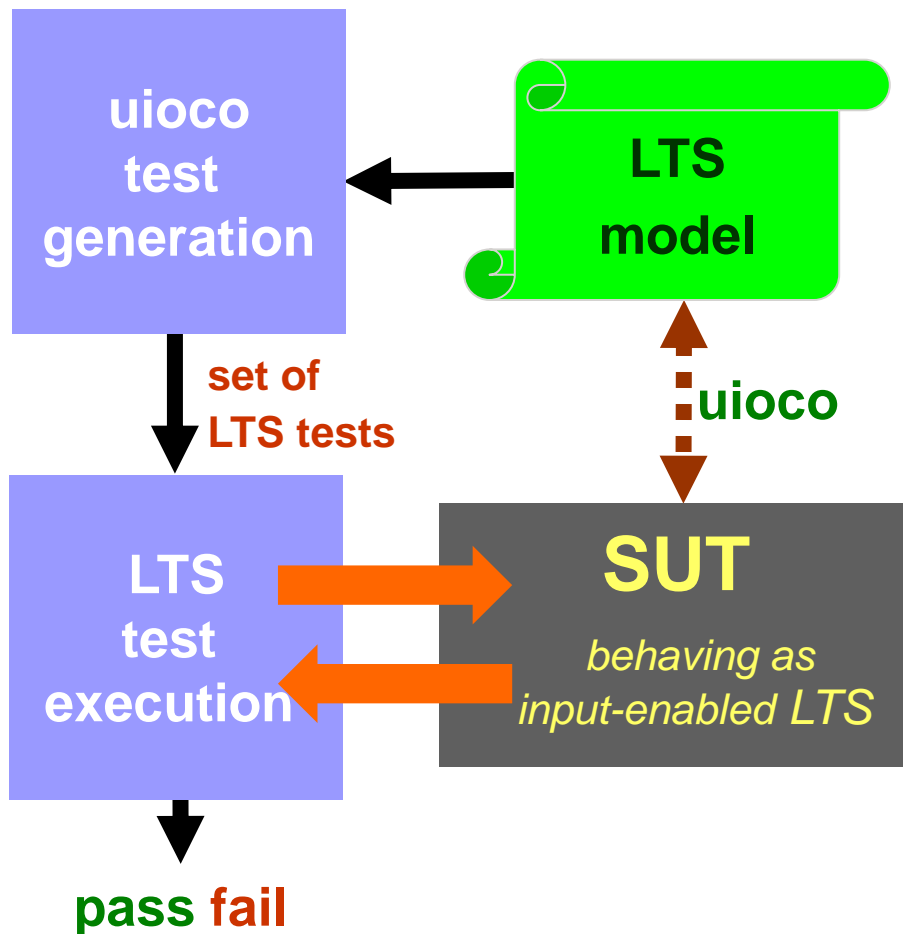
MBT with LTS topics:

☞ specification model

☞ system under test (SUT)

☞ SUT ...

☞ conformance uioco

☞ test cas...

☞ test generation

☞ test execution

☞ test result analysis

☞ sound & exhaustive