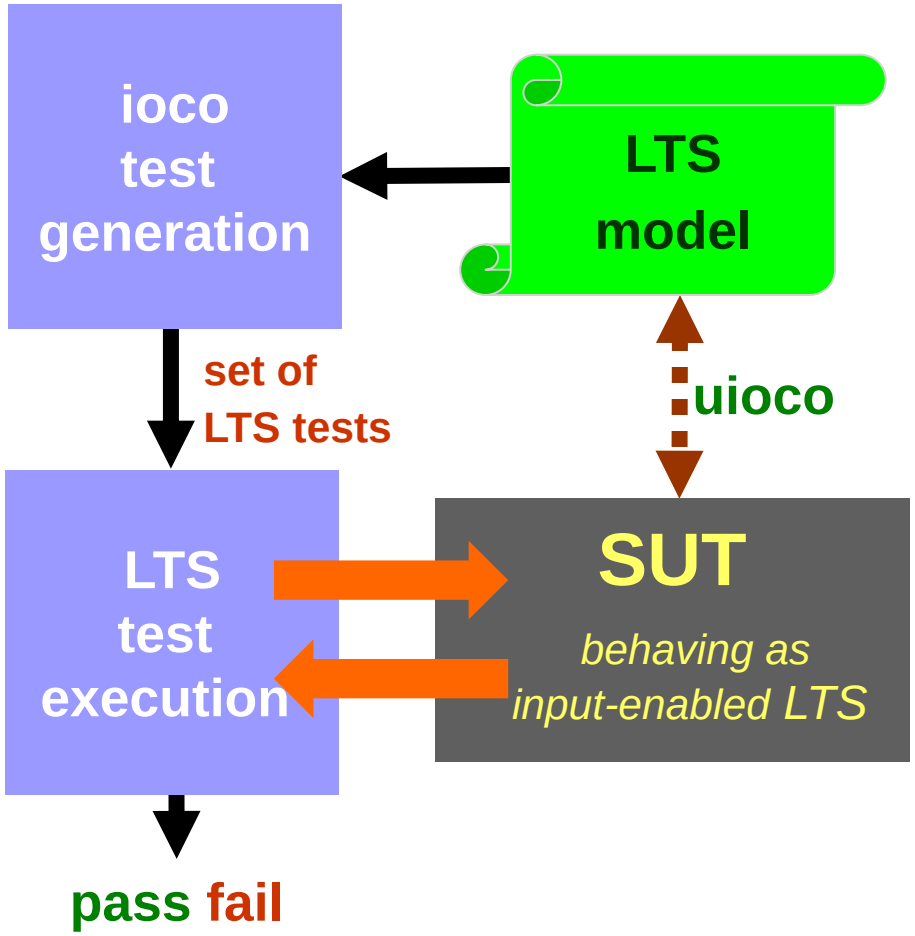
A Theory of Model-Based Testing with Labelled Transition Systems

*Test Generation for **uioco***

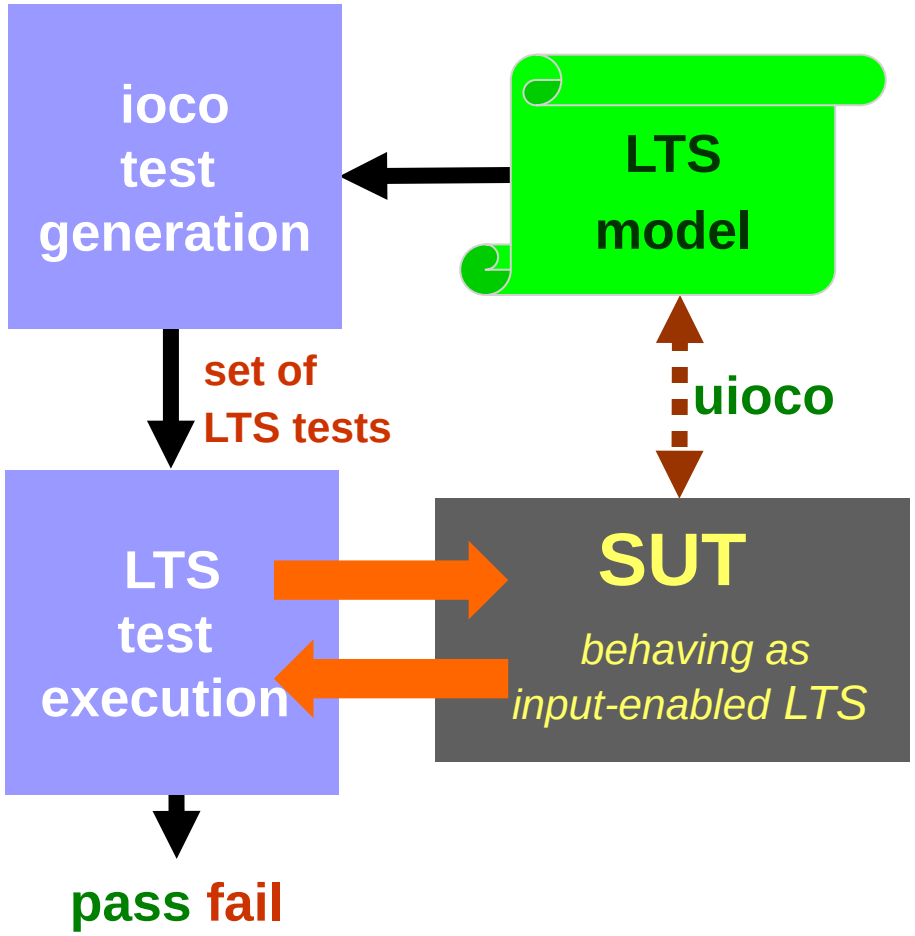
MBT : Labelled Transitions Systems



MBT with LTS topics:

- specification model
- implementation (SUT)
- implementation model
- conformance **uioco**
- test cases
- test generation
- test execution
- test result analysis
- sound & exhaustive

MBT : Labelled Transitions Systems



MBT with LTS topics:

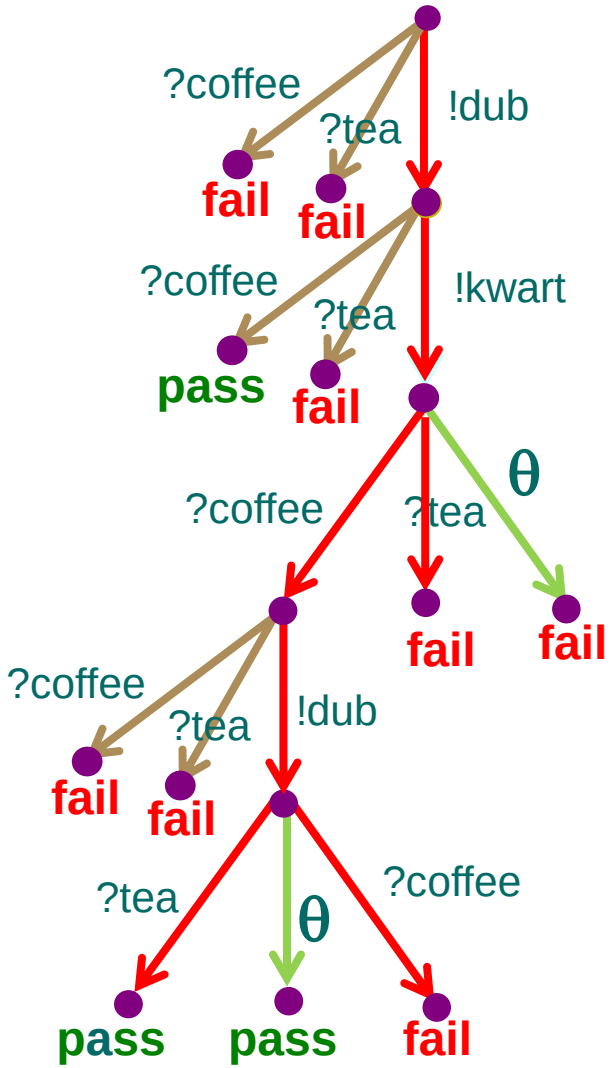
- specification model
- implementation (SUT)
- implementation model
- conformance **uioco**
- test cases
- test generation
- test execution
- test result analysis
- sound & exhaustive

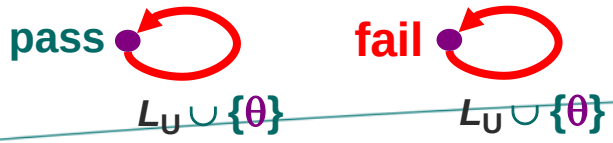
Test Cases

model of a test case

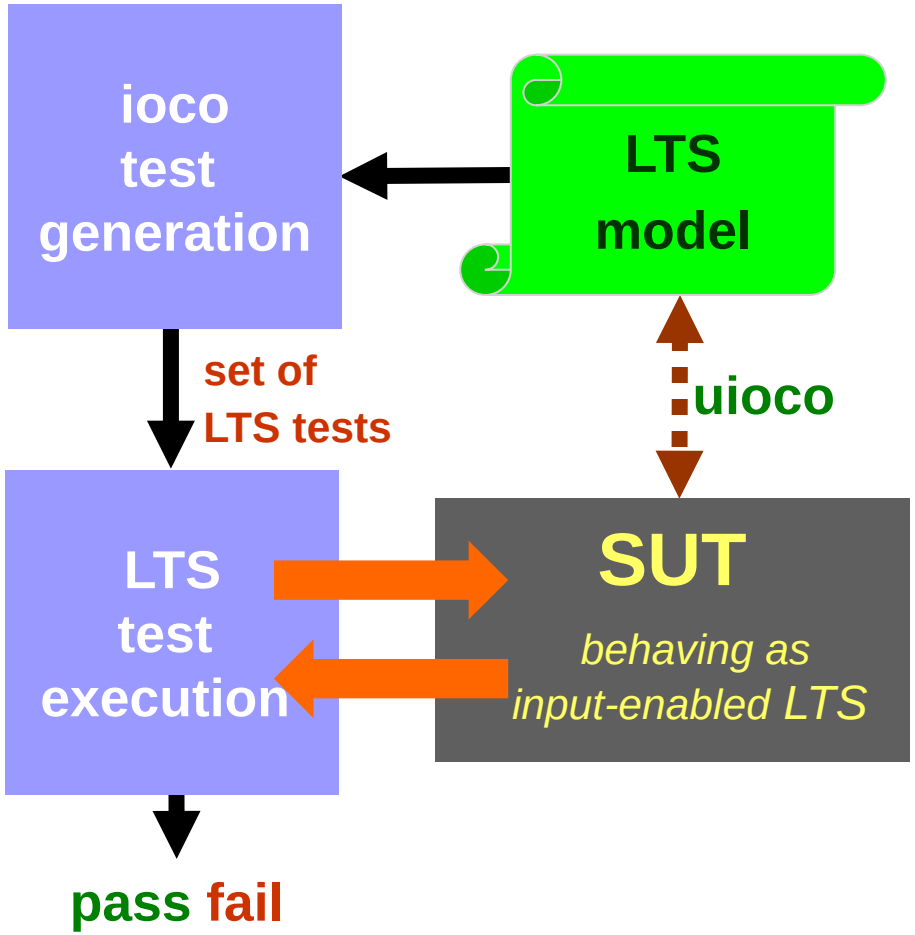
= labelled transition system

- labels in $L_I \cup L_U \cup \{ \theta \}$
- ‘quiescence’ / ‘time-out’ label θ
- tree-structured
- finite, deterministic
- sink states **pass** and **fail**
- from each state ≠ **pass**, **fail** :
  - either one input **!a** and all outputs **?x**
  - or all outputs **?x** and θ





MBT : Labelled Transitions Systems

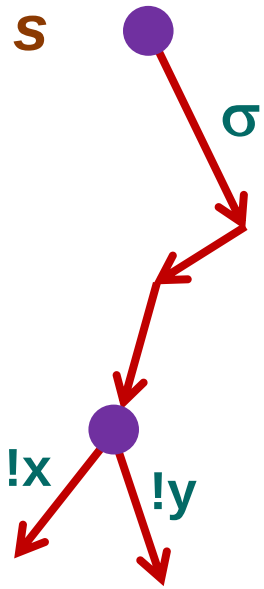


MBT with LTS topics:

- specification model
- implementation (SUT)
- implementation model
- conformance **uioco**
- test cases
- test generation
- test execution
- test result analysis
- sound & exhaustive

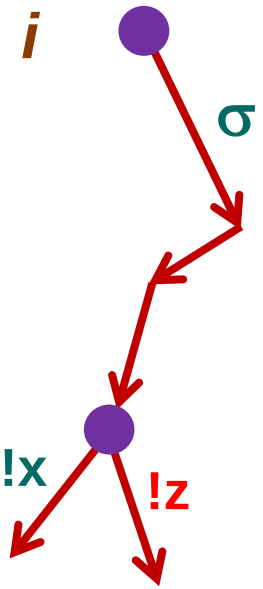
Test Generation

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



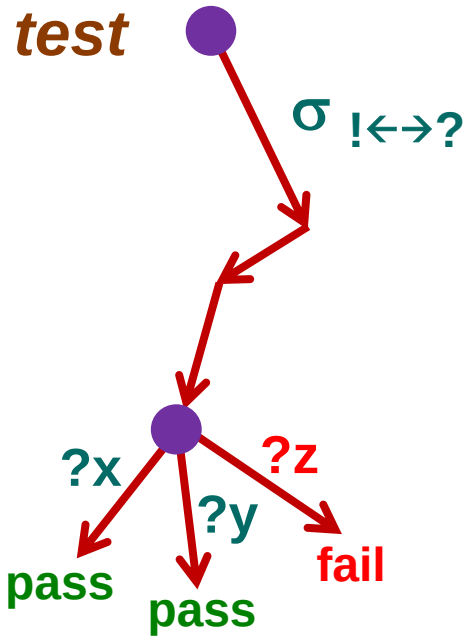
out **(** ***s*** **after** σ **)**

= { **!x**, **!y** }



out **(** ***i*** **after** σ **)**

= { **!x**, **!z** }

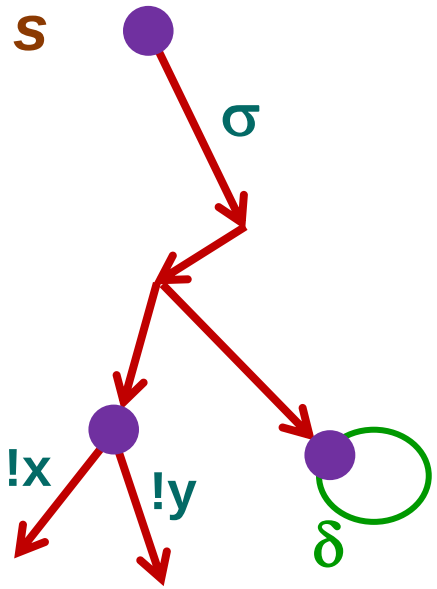


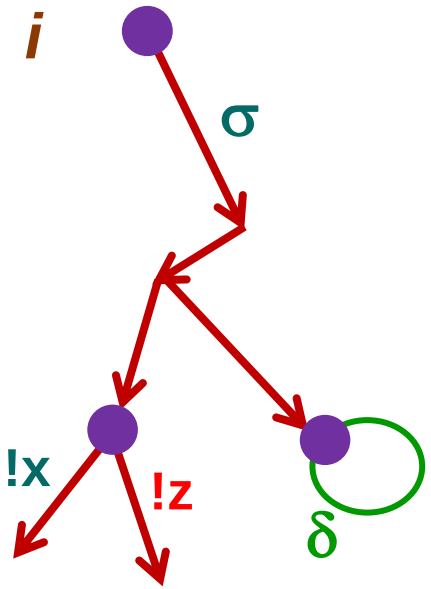
out **(** ***test*** **after** σ **)**

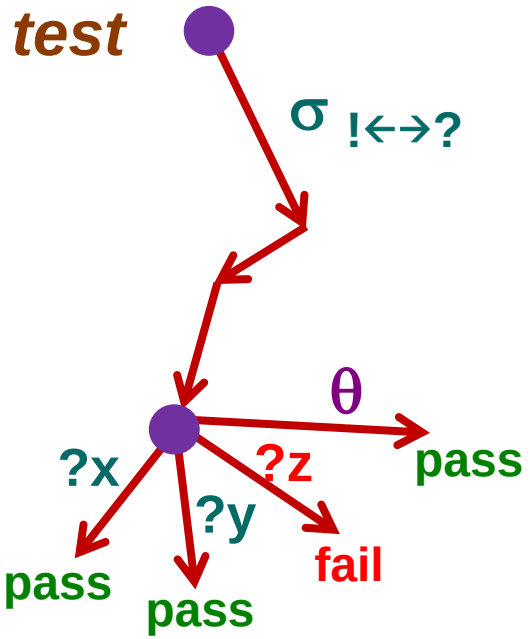
= { **!x**, **!y**, **!z** }

Test Generation

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







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***)

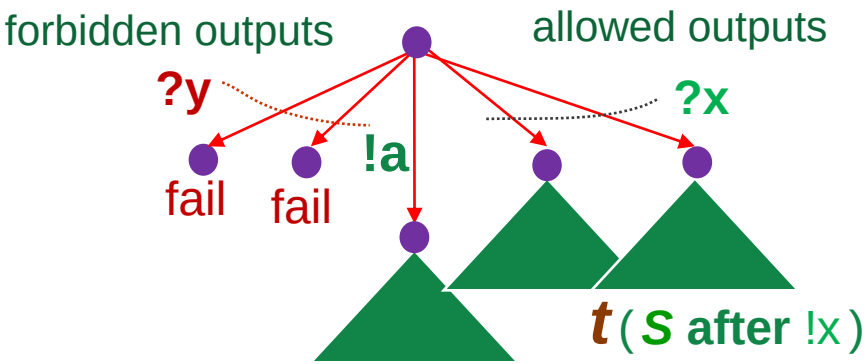
from a transition system state set ***S***, with ***S*** ≠ ∅, and initially` ***S*** = s_0 **after** ε

Apply the following steps recursively, non-deterministically :

1 end test case

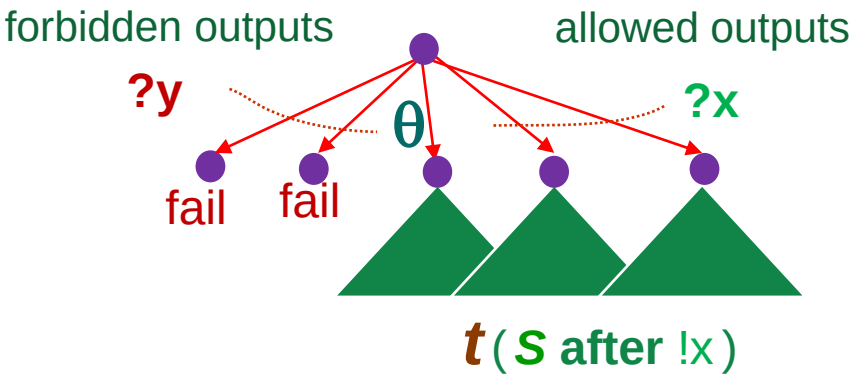
● **pass**

2 supply input !a



t (***S*** **after ?a**), for **?a** ∈ L_I
and *not* ***S*** refuses {**?a**}

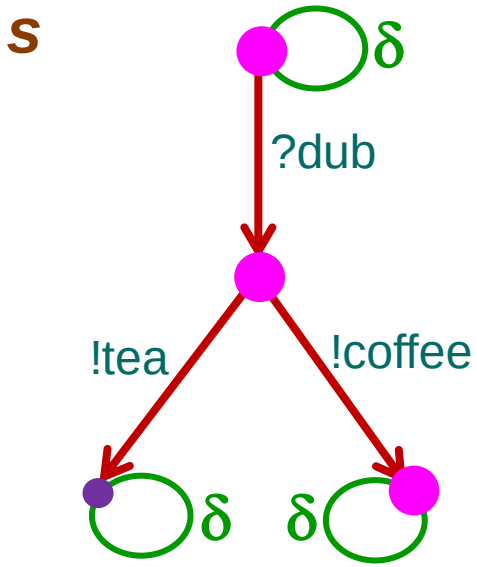
3 observe all outputs



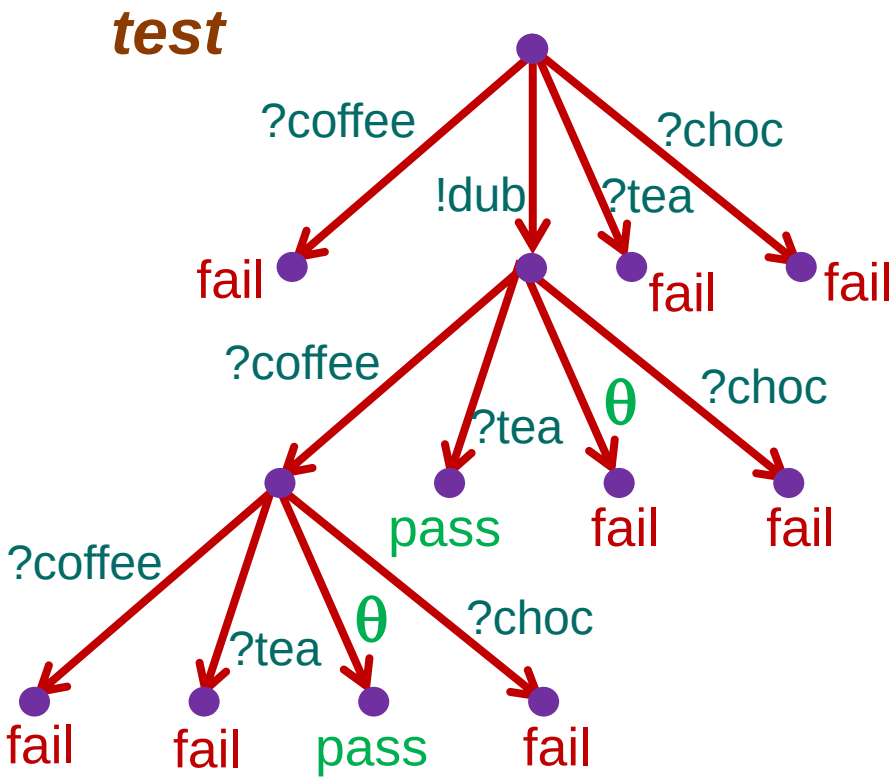
allowed outputs (or δ): **!x** ∈ *out* (***S***)

forbidden outputs (or δ): **!y** ∉ *out* (***S***)

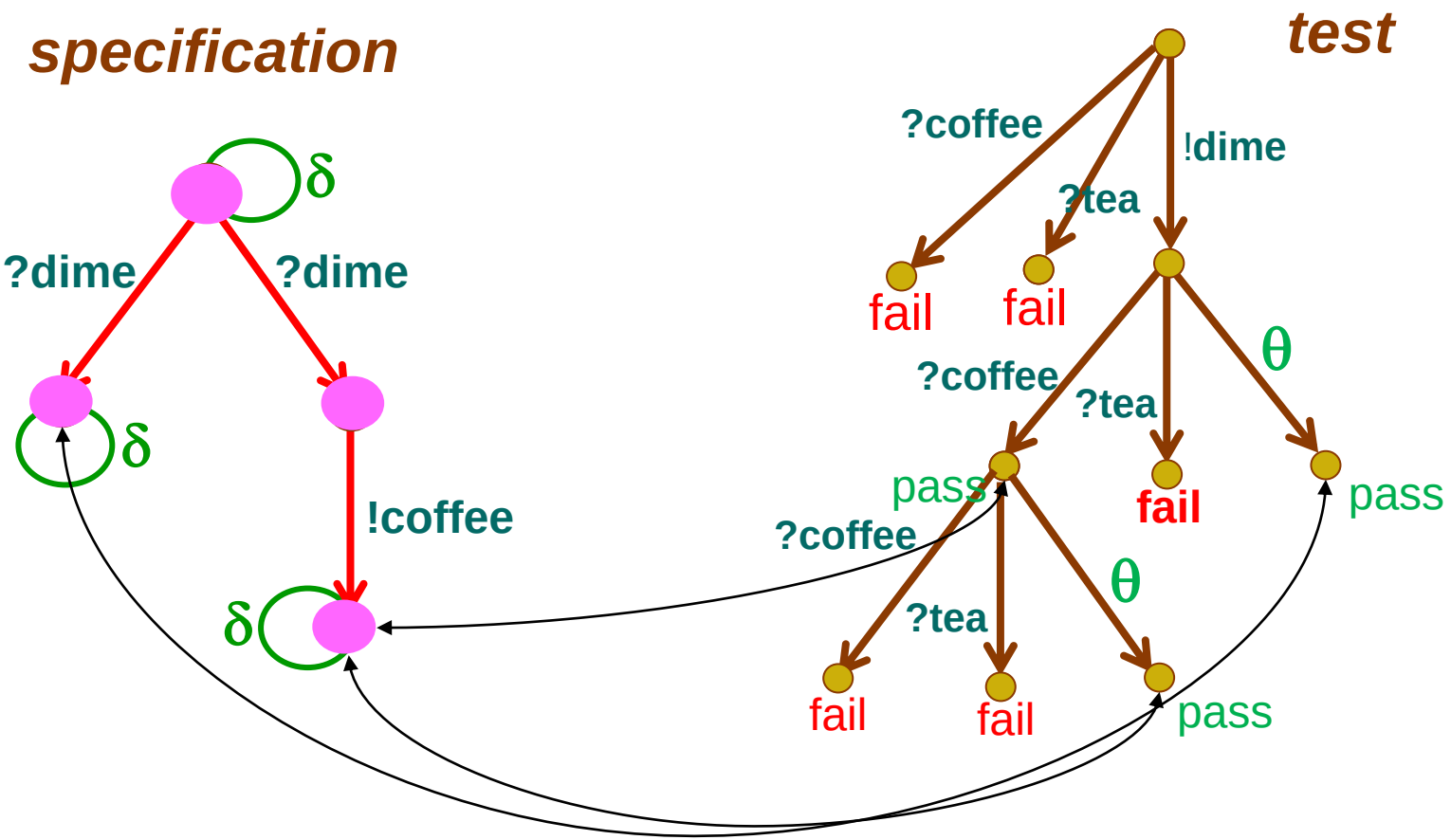
Example : *uioco* Test Generation



L = { ?dub, !coffee,
!tea, !choc }



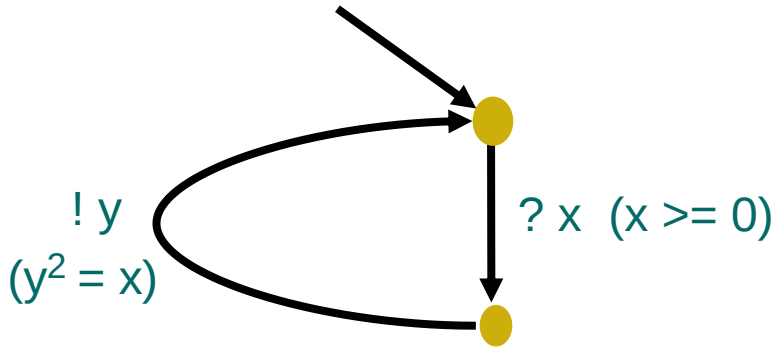
Example : *uioco* Test Generation



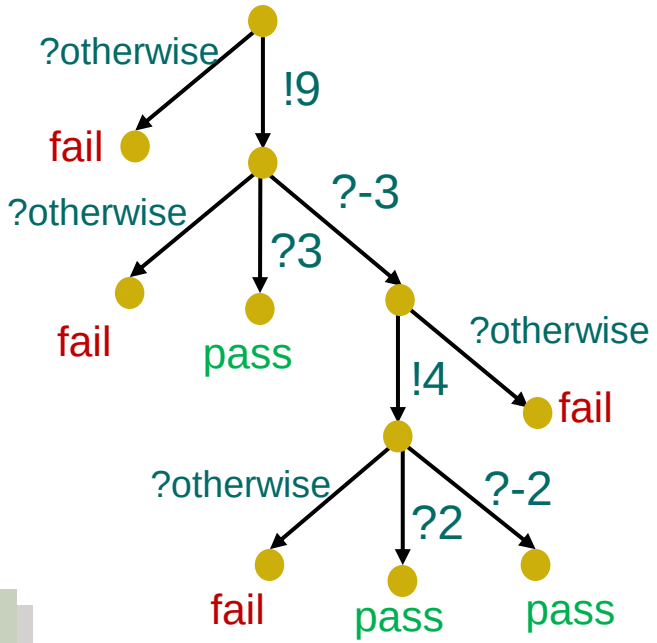
Example : *uioco* Test Generation

Equation solver for $y^2=x$

specification ***s***

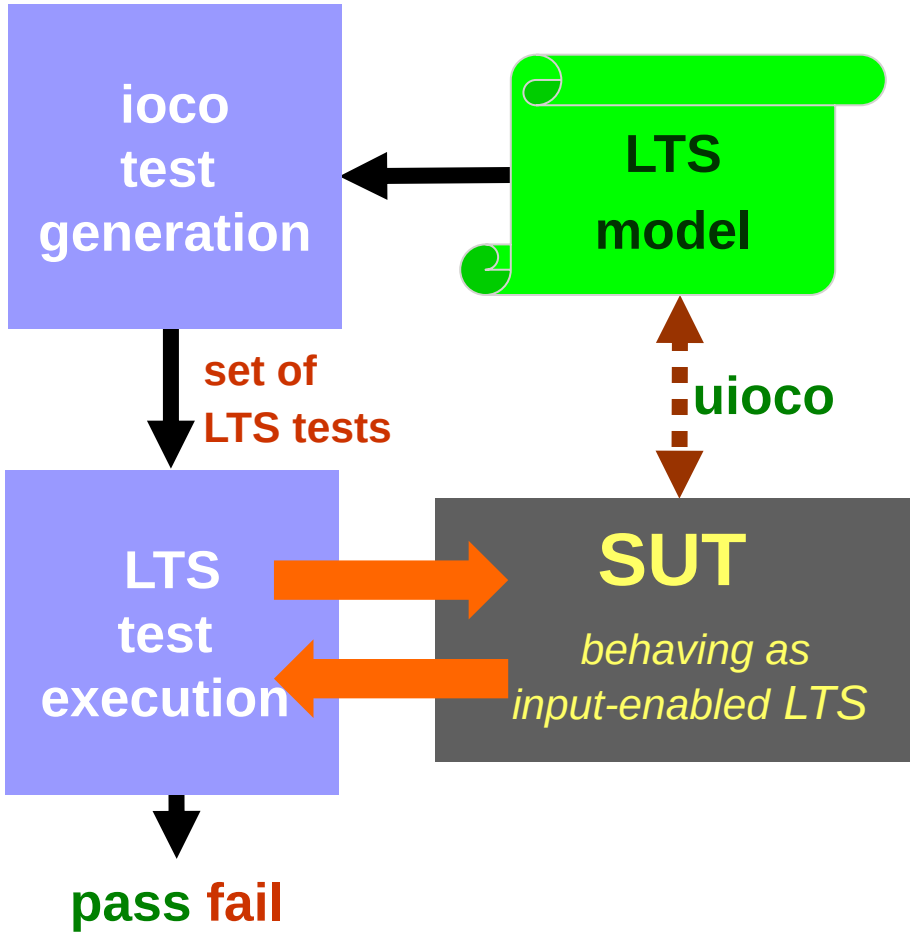


test



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

MBT : Labelled Transitions Systems



MBT with LTS topics:

- specification model
- implementation (SUT)
- implementation model
- conformance **uioco**
- test cases
- test generation
- test execution
- test result analysis
- sound & exhaustive

Test Execution

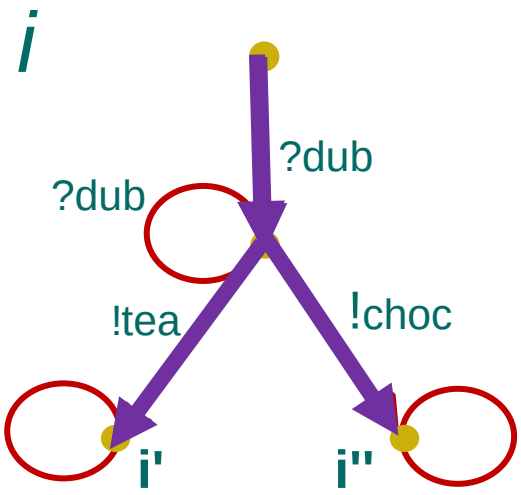
Test execution = all possible parallel test runs of test **t**
with implementation **i** going to state pass or fail

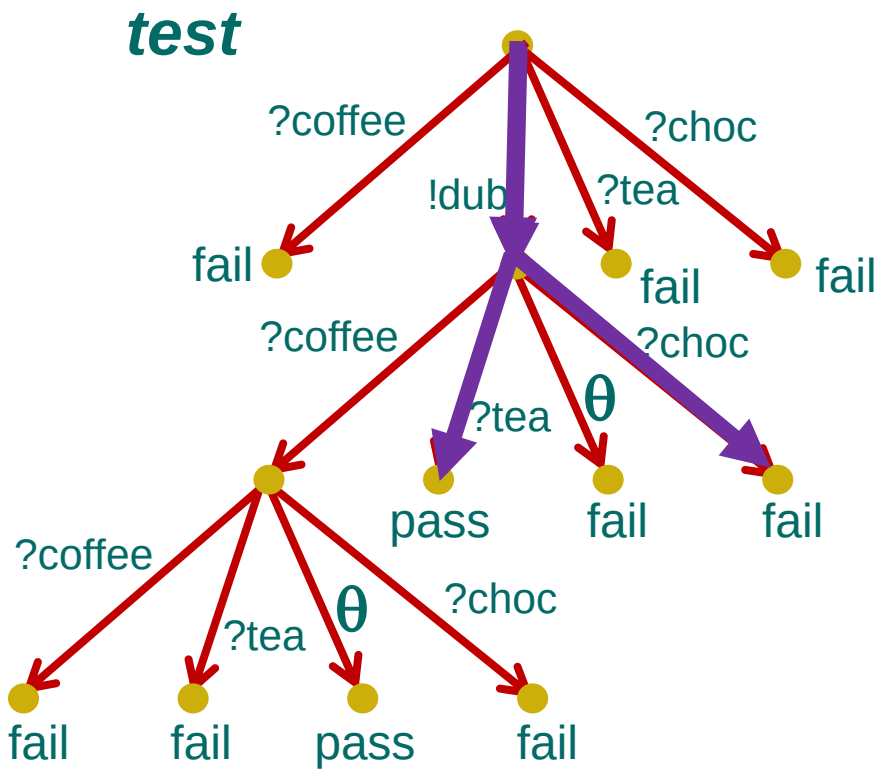
Test run : $t \lceil\!\lceil i \xRightarrow{\sigma} pass \lceil\!\lceil i'$ or $t \lceil\!\lceil i \xRightarrow{\sigma} fail \lceil\!\lceil i'$

$$\frac{t \xrightarrow{a} t', \quad i \xrightarrow{a} i'}{t \lceil\!\lceil i \xrightarrow{a} t' \lceil\!\lceil i'} \qquad \frac{i \xrightarrow{\tau} i'}{t \lceil\!\lceil i \xrightarrow{\tau} t \lceil\!\lceil i'}$$

$$\frac{t \xrightarrow{\theta} t', \quad i \xrightarrow{\delta} i'}{t \lceil\!\lceil i \xrightarrow{\theta} t' \lceil\!\lceil i'}$$

Example : Test Execution





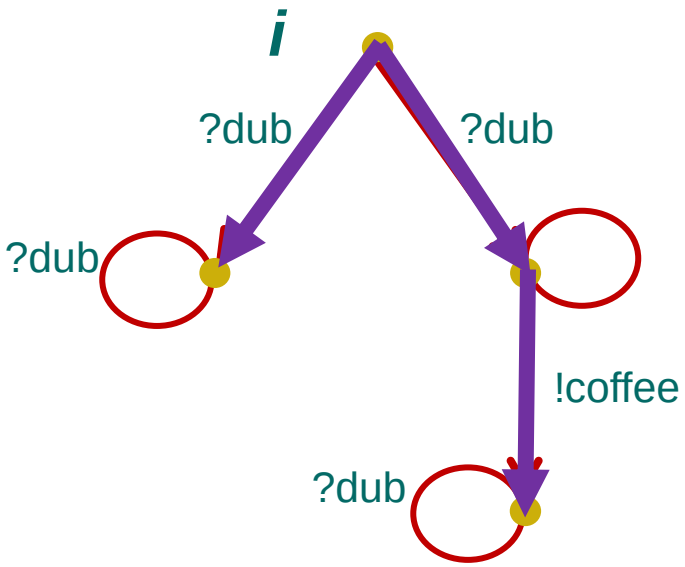
Two test runs :

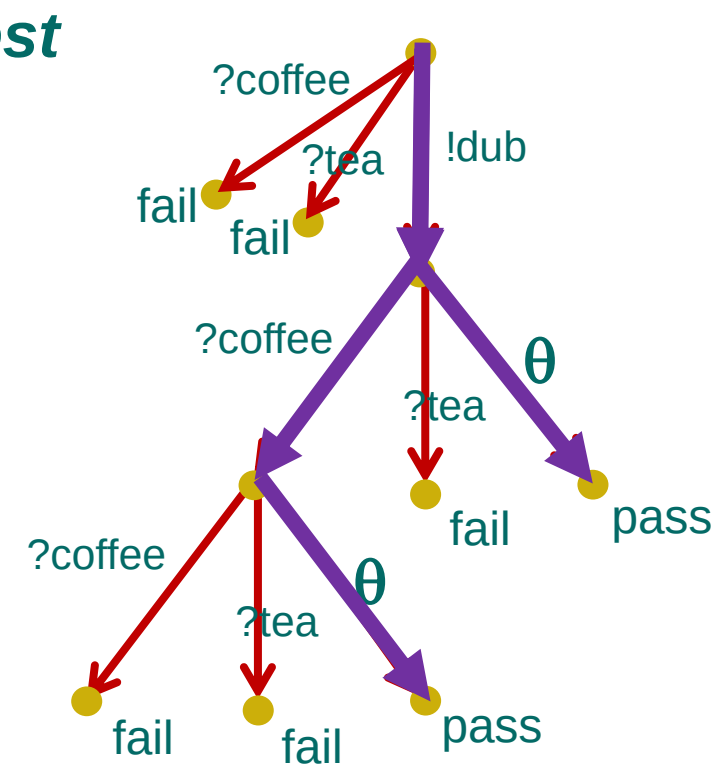
t ⫐ i ⟹ (dub tea) pass ⫐ i'

t ⫐ i ⟹ (dub choc) fail ⫐ i''

i fails t

Example : Test Execution





Two test runs :

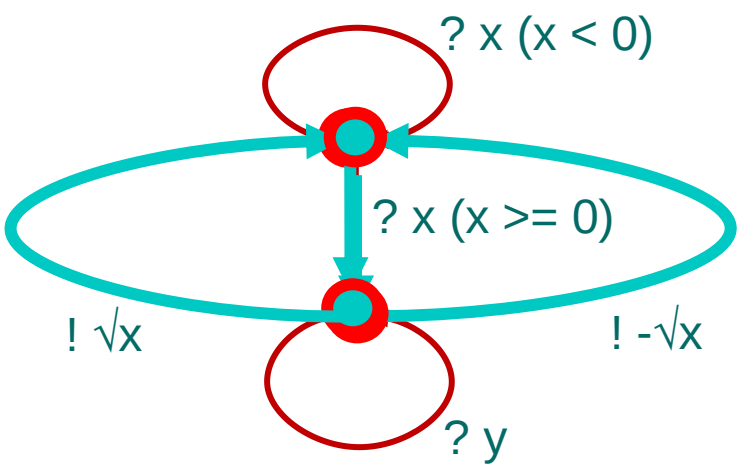
$t \lceil\!\lceil\ i \xRightarrow{dub\ \theta} pass \lceil\!\lceil\ i'$

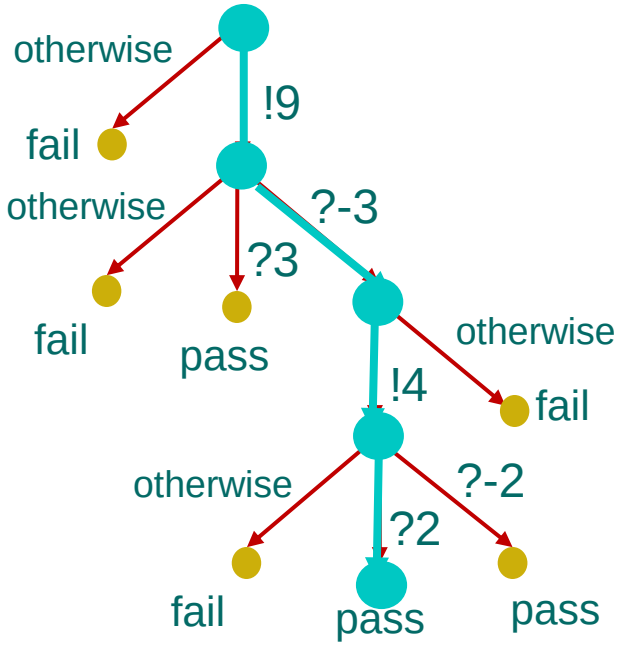
$t \lceil\!\lceil\ i \xRightarrow{dub\ coffee\ \theta} pass \lceil\!\lceil\ i'$

i passes **t**

Example : Test Execution

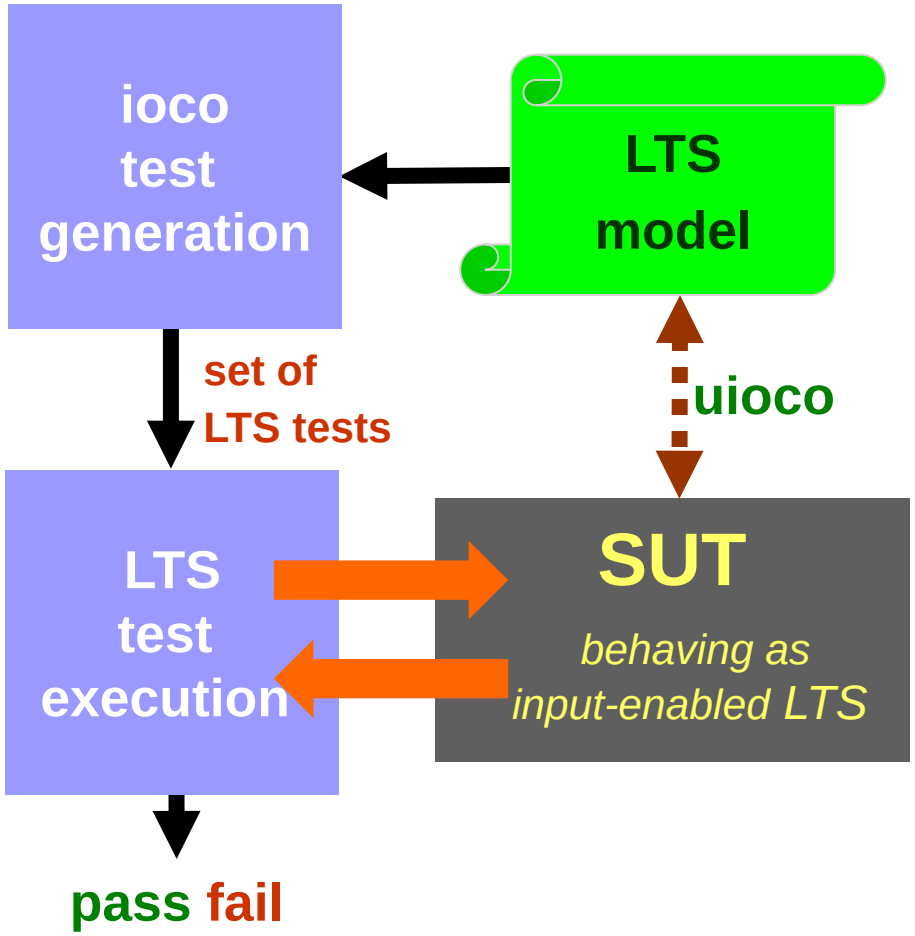
implementation ⌉| test





i passes **t**

MBT : Labelled Transitions Systems



MBT with LTS topics:

- specification model
- implementation (SUT)
- implementation model
- conformance **uioco**
- test cases
- test generation
- test execution
- test result analysis
- sound & exhaustive

Validity of Test Generation

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

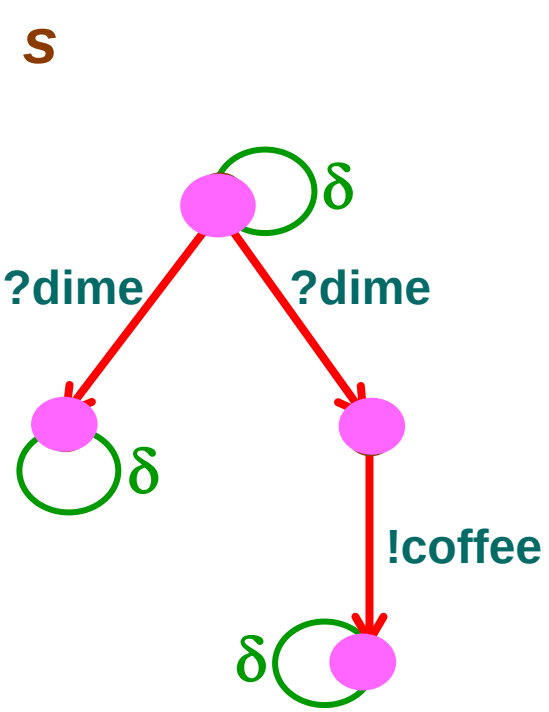
- Soundness :
  t will never fail with correct implementation

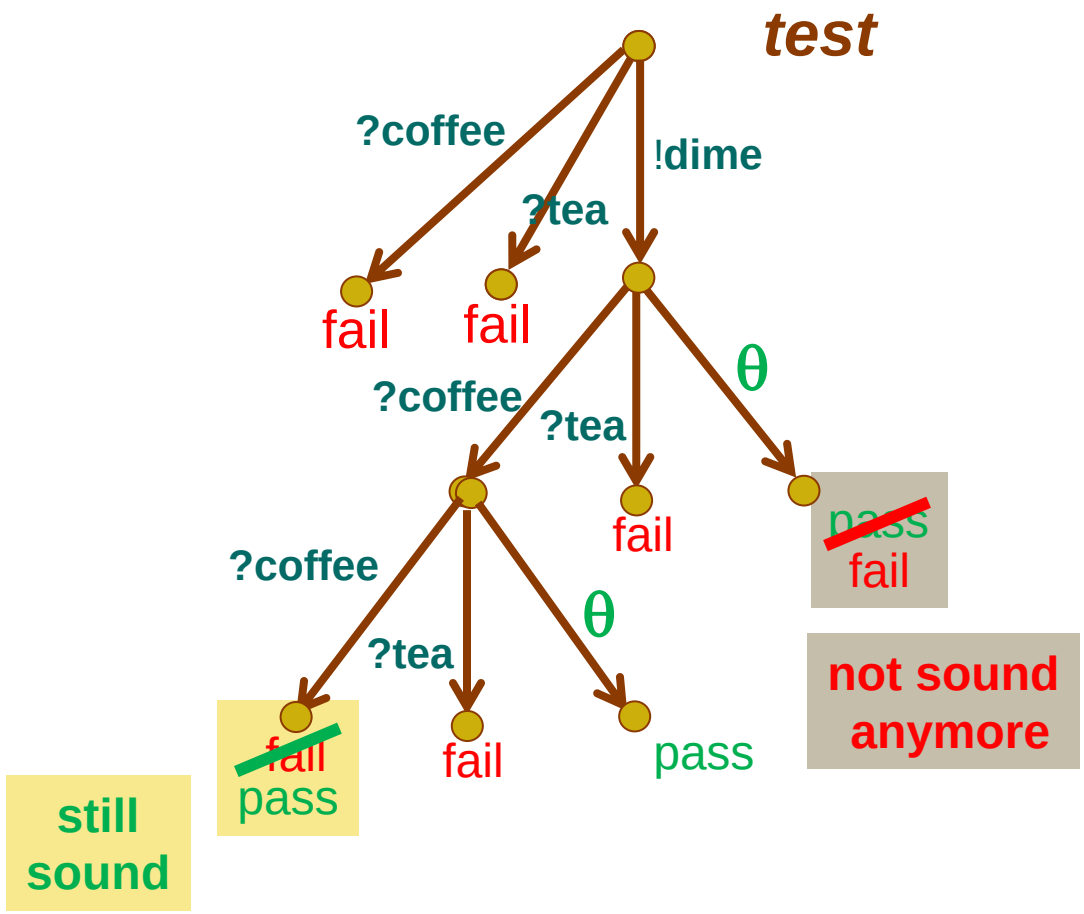
  i uioco s implies **i passes t**

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

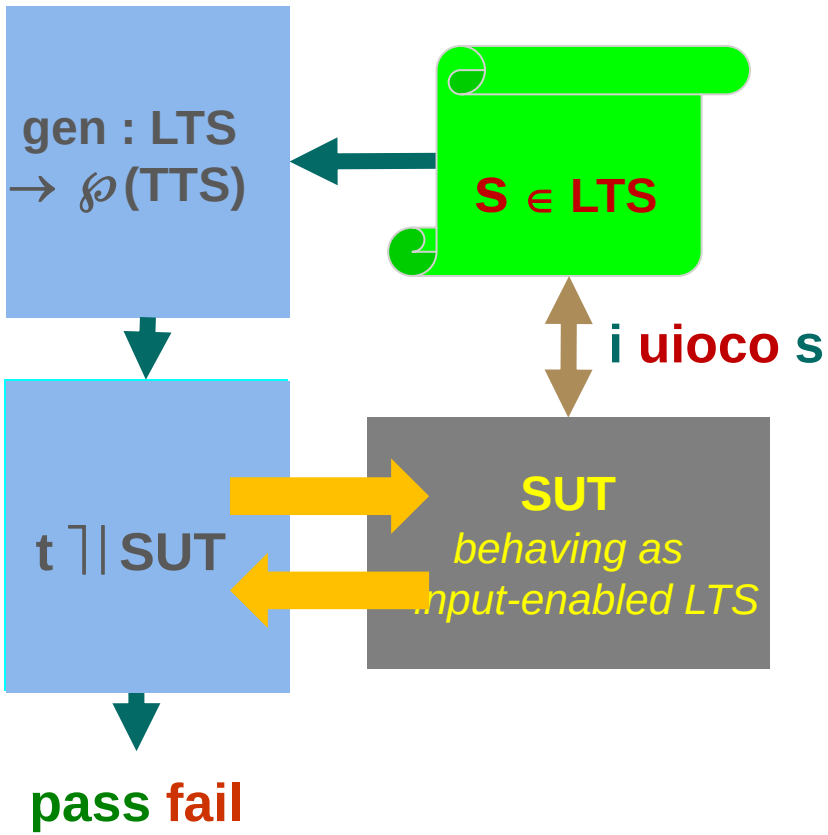
  i uioco s implies ∃ **t : i fails t**

Soundness and Exhaustiveness





MBT with *uioco* is Sound and Exhaustive



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