The ioco Theory for Model-Based Testing with Labelled Transition Systems

Jan Tretmans

jan.tretmans@tno.nl

The ioco Theory

for Model-Based Testing

Overview

- Models LTS
- Comparing LTS
  - equivalences
- Correctness
  - implementation relations
  - **ioco**
- Testing LTS
  - test generation
  - test execution
- Correctness & Testing
  - soundness
  - exhaustiveness
- SUT: Black-Box & Formal
  - test assumption

Labelled Transition Systems

Labelled Transition Systems

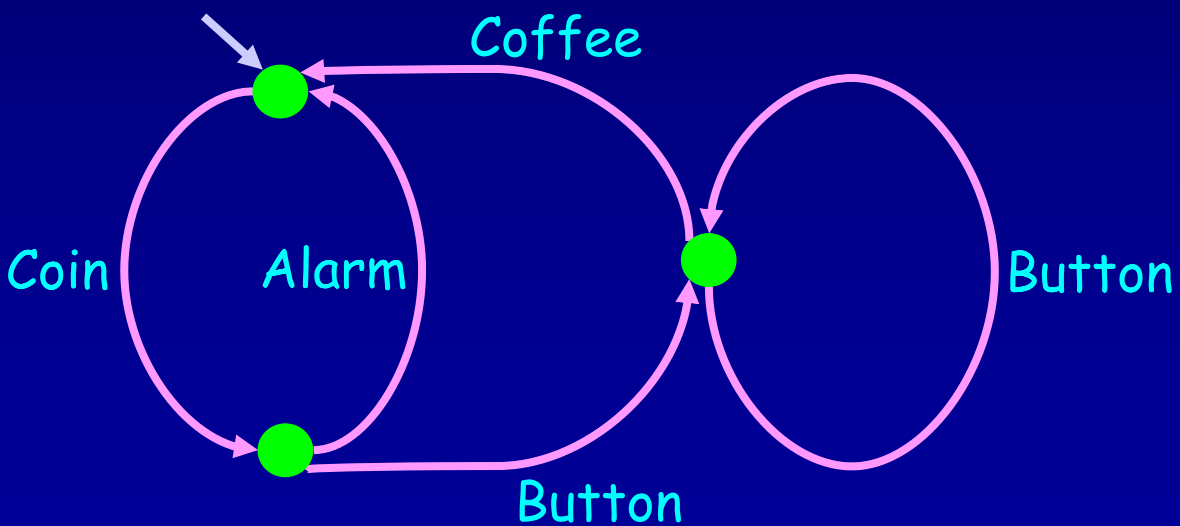
Labelled Transition System $\langle\, \mathbf{S},\ \mathbf{L},\ \mathbf{T},\ s_0 \,\rangle$

states

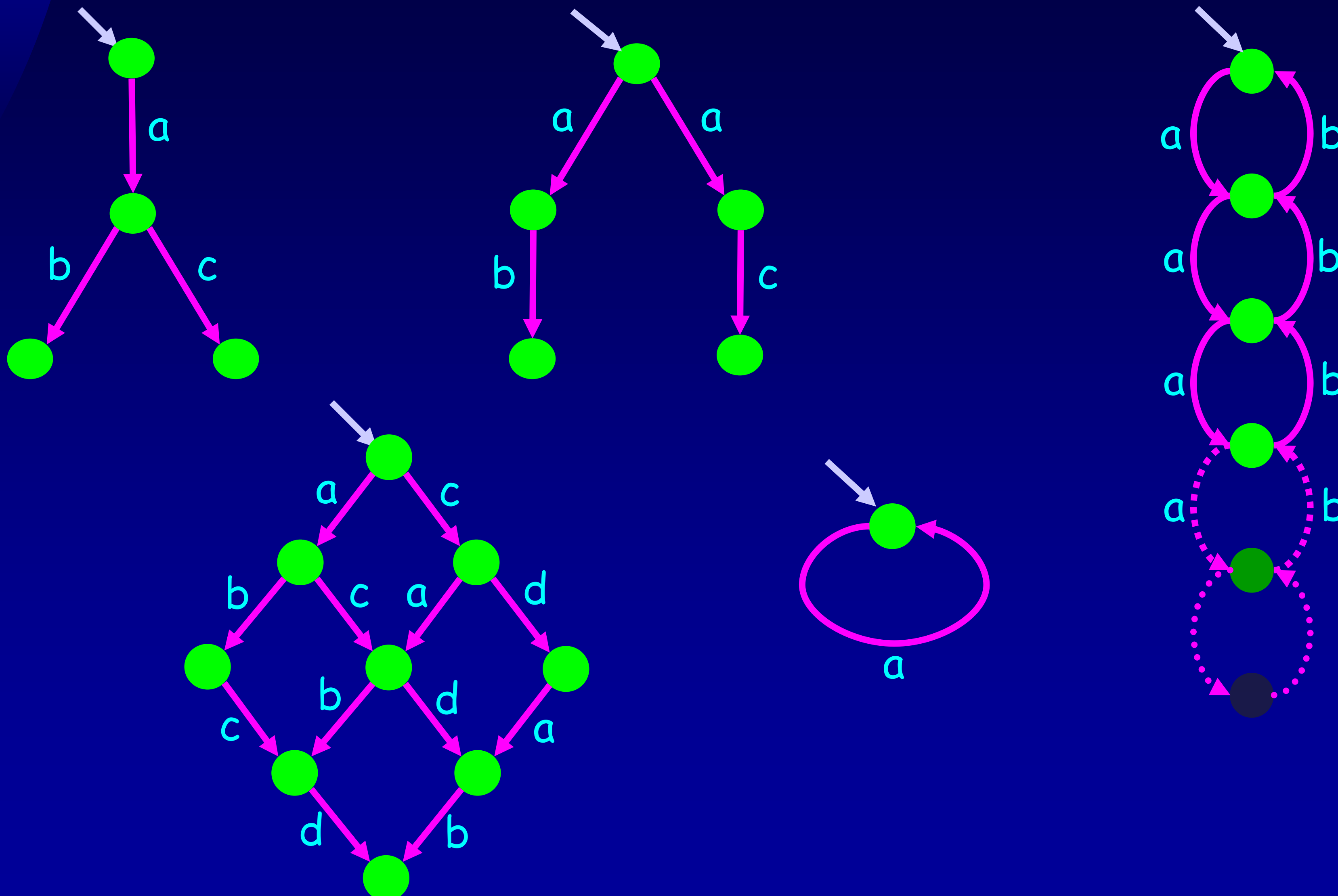
actions

transitions
$T \subseteq S \times (L \cup \{\tau\}) \times S$

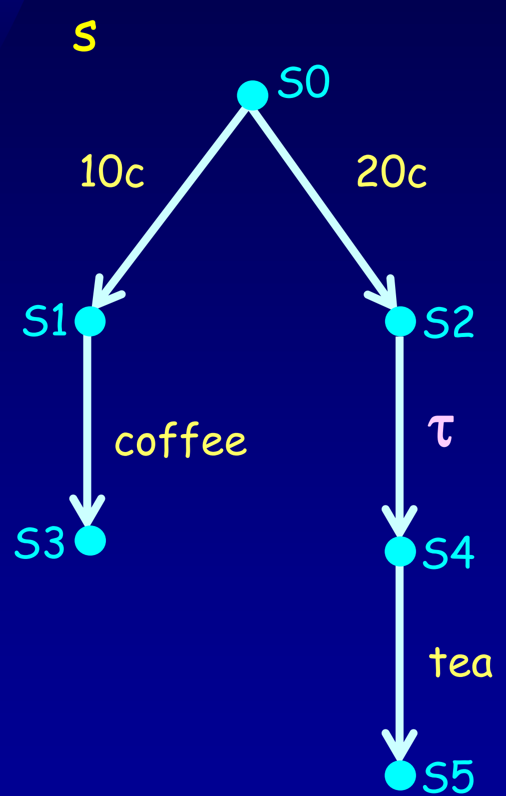
initial state
$s_0 \in S$



Labelled Transition Systems



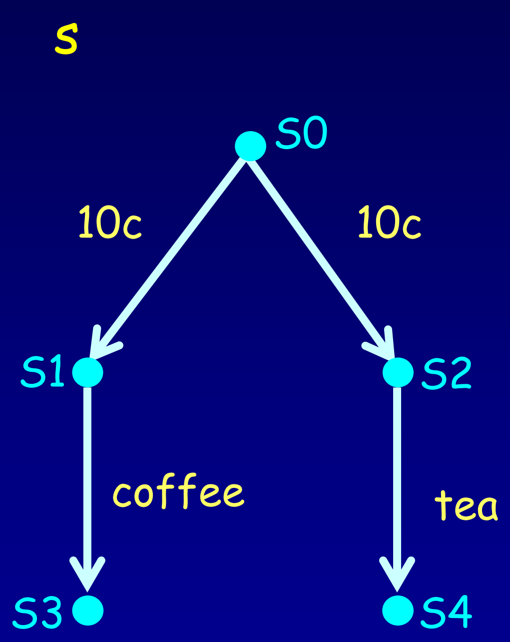
Labelled Transition Systems



$L = \{ 10c, 20c, coffee, tea, soup \}$

$S0 \xrightarrow{10c} S1$	transition
$S0 \xRightarrow{10c\ \ coffee} S3$	transition composition
$S0 \xRightarrow{20c\ \ tea}$	executable sequence
$S0 \not\xRightarrow{20c\ \ soup}$	non-executable sequence
$LTS(L)$	all transition systems over L

Labelled Transition Systems



Sequences of observable actions:

traces (s) = $\{ \sigma \in L^* \mid s \overset{\sigma}{\Longrightarrow} \}$

traces(s) = $\{ \varepsilon$, 10c, 10c coffee, 10c tea $\}$

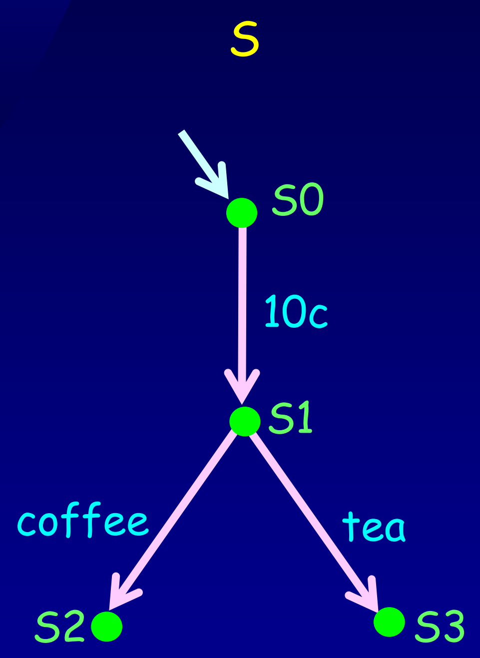
Reachable states:

s **after** σ = $\{ s' \mid s \overset{\sigma}{\Longrightarrow} s' \}$

s **after** 10c = { S1, S2 }

s **after** 10c tea = { S4 }

Representation of LTS



- Explicit :

  ⟨ {S0,S1,S2,S3},

  {10c,coffee,tea},

  { (S0,10c,S1), (S1,coffee,S2), (S1,tea,S3) },

  S0 ⟩

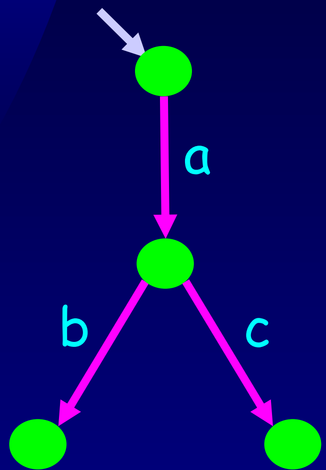
- Transition tree / graph

- Language / behaviour expression :

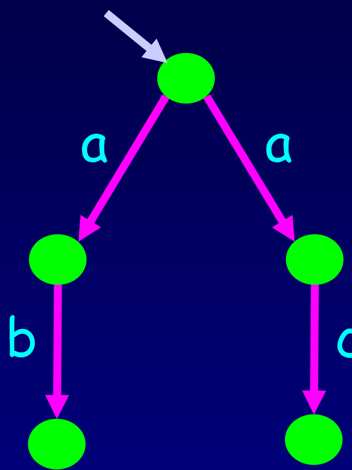
  S ::=

  10c >-> (coffee >-> STOP ## tea >-> STOP)

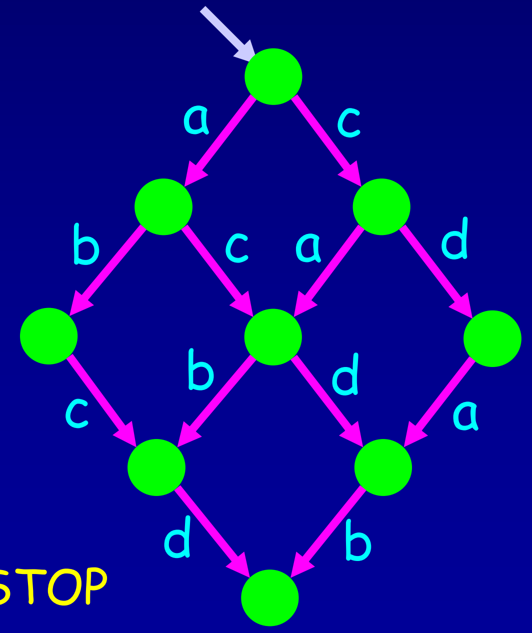
Representation of LTS




```
a >->
  ( b >-> STOP
   ##
    c >-> STOP
  )
```

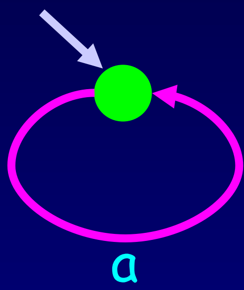



```
   a >-> b >-> STOP
## a >-> c >-> STOP
```




```
a >-> b >-> STOP   |||   c >-> d >-> STOP
```

Representation of LTS

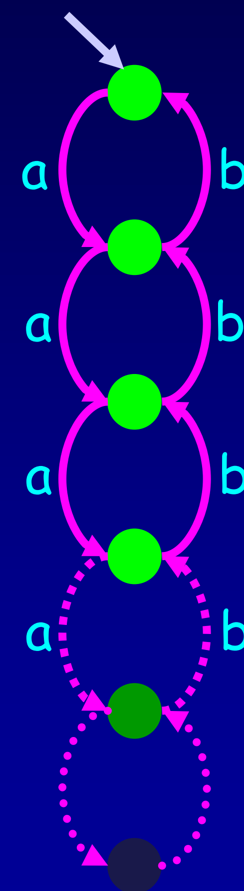


P, where

P ::= a >-> P

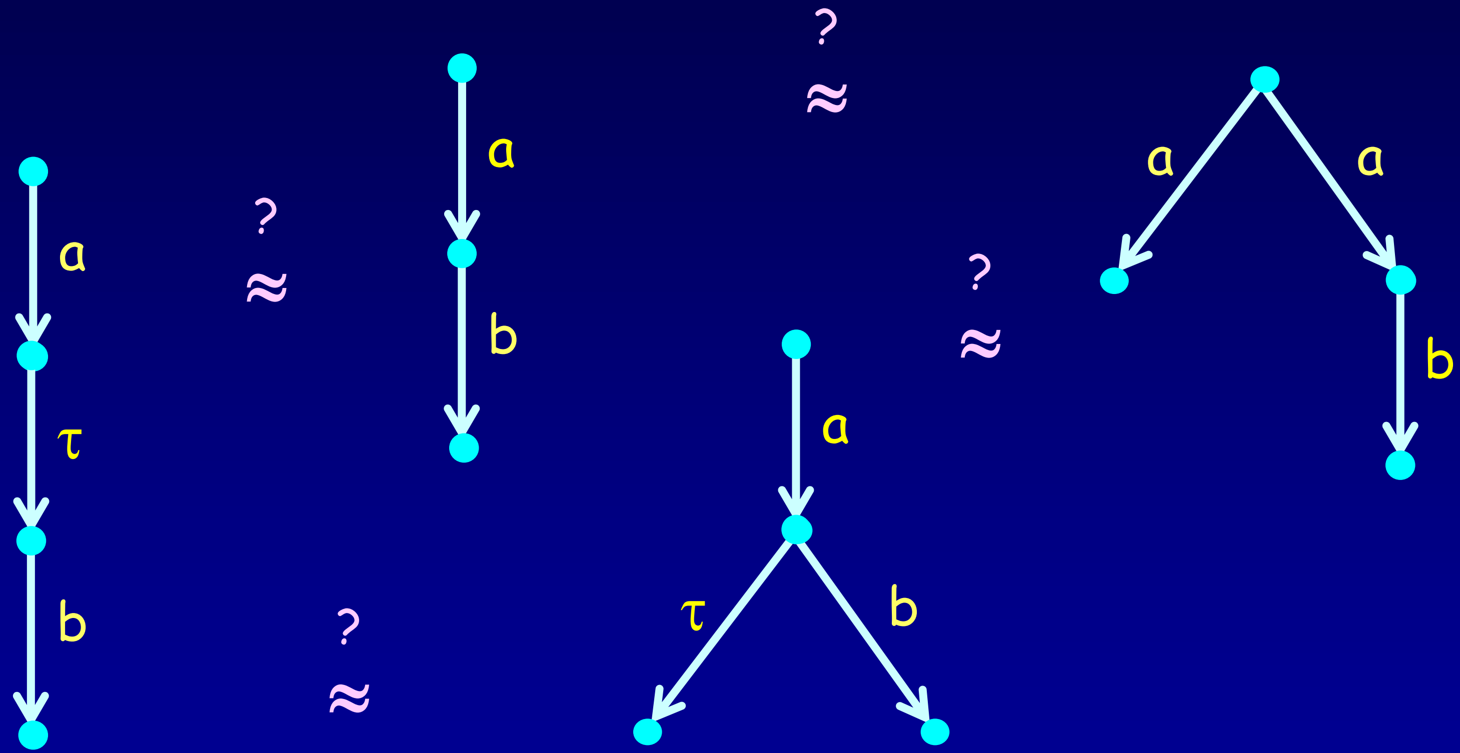
Q, where

Q ::=
a >-> (b >-> STOP ||| Q)



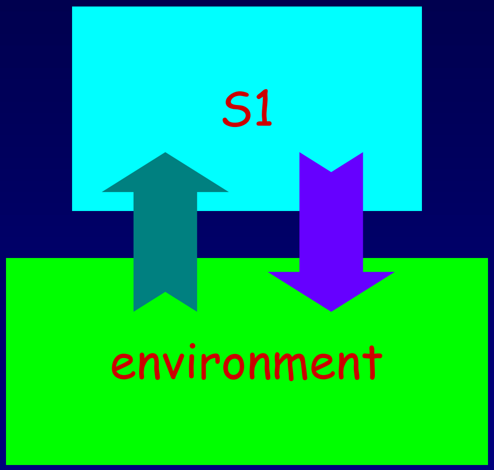
Equivalences on Labelled Transition Systems

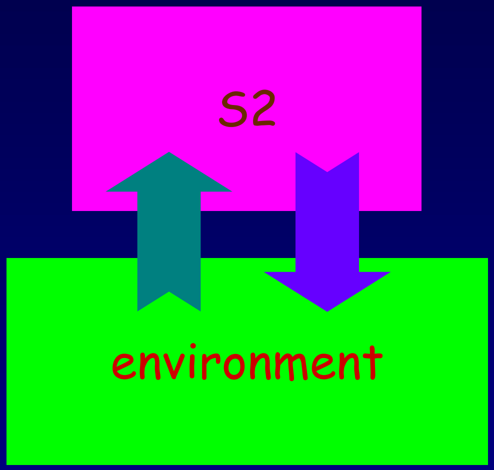
Observable Behaviour



" Some transition systems are more equal than others "

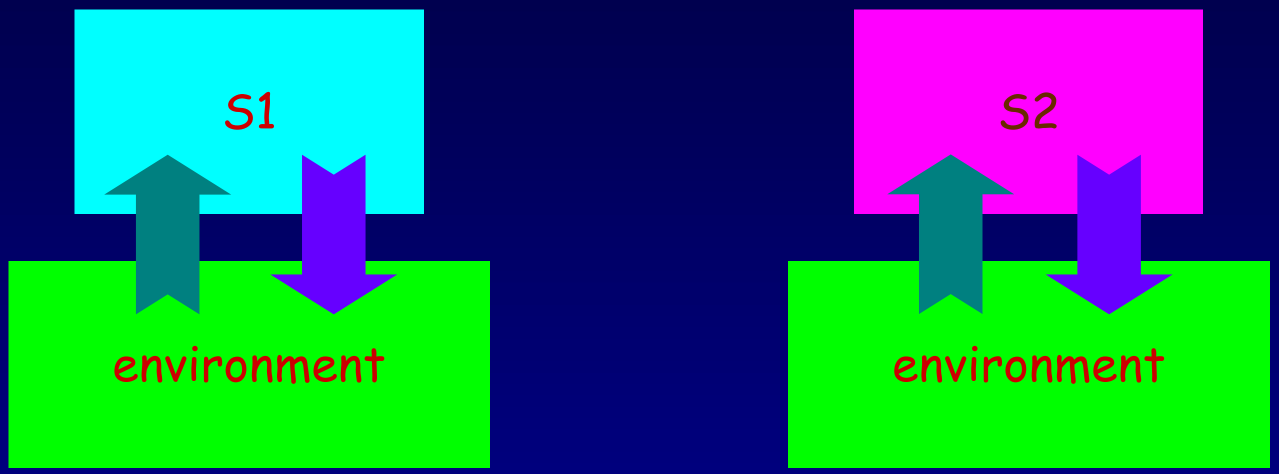
Comparing Transition Systems





- Suppose an environment interacts with the systems:
  - the environment tests the system as black box by observing and actively controlling it;
  - the environment acts as a tester;
- Two systems are equivalent if they pass the same tests.

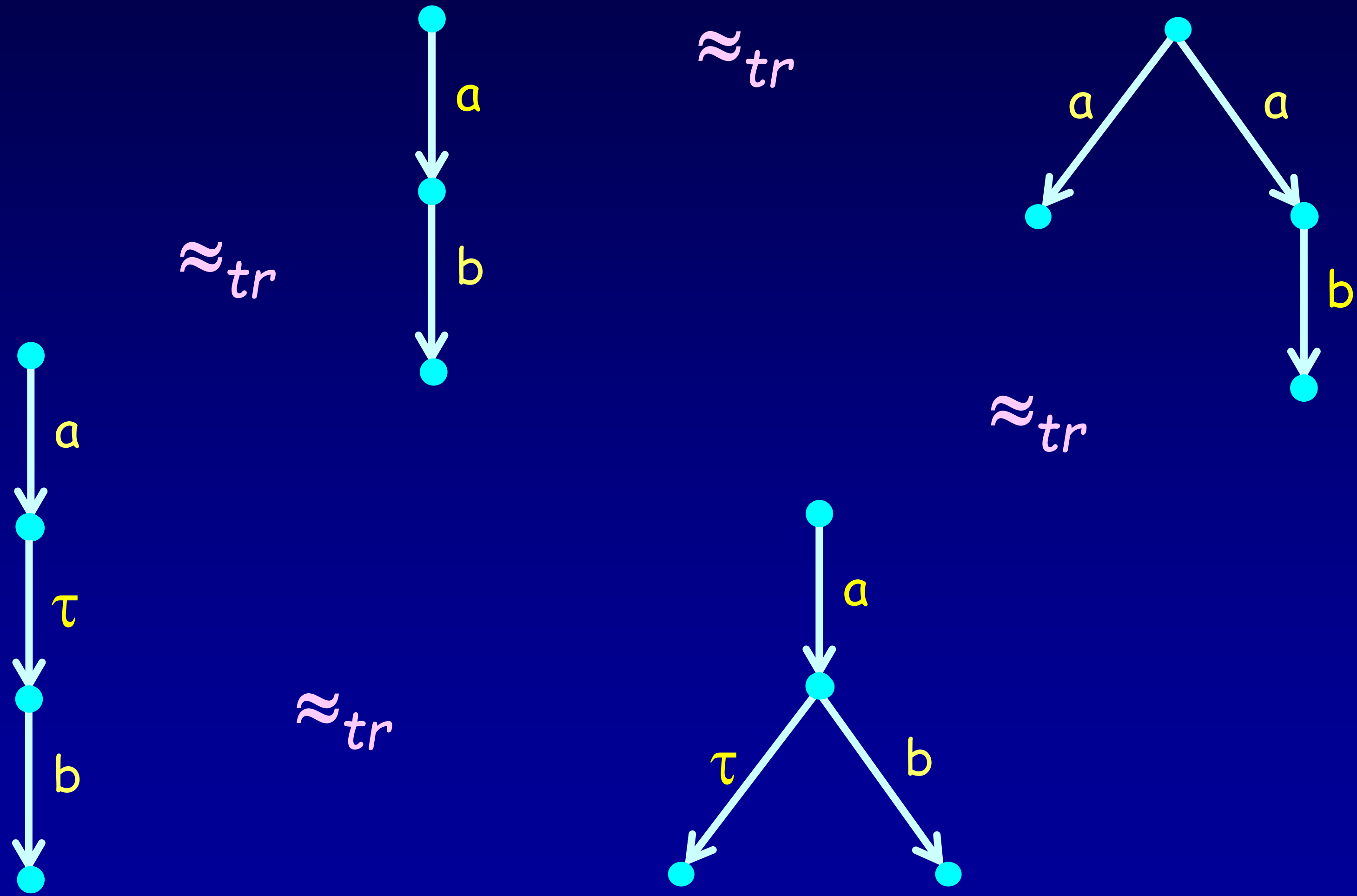
Trace Equivalence



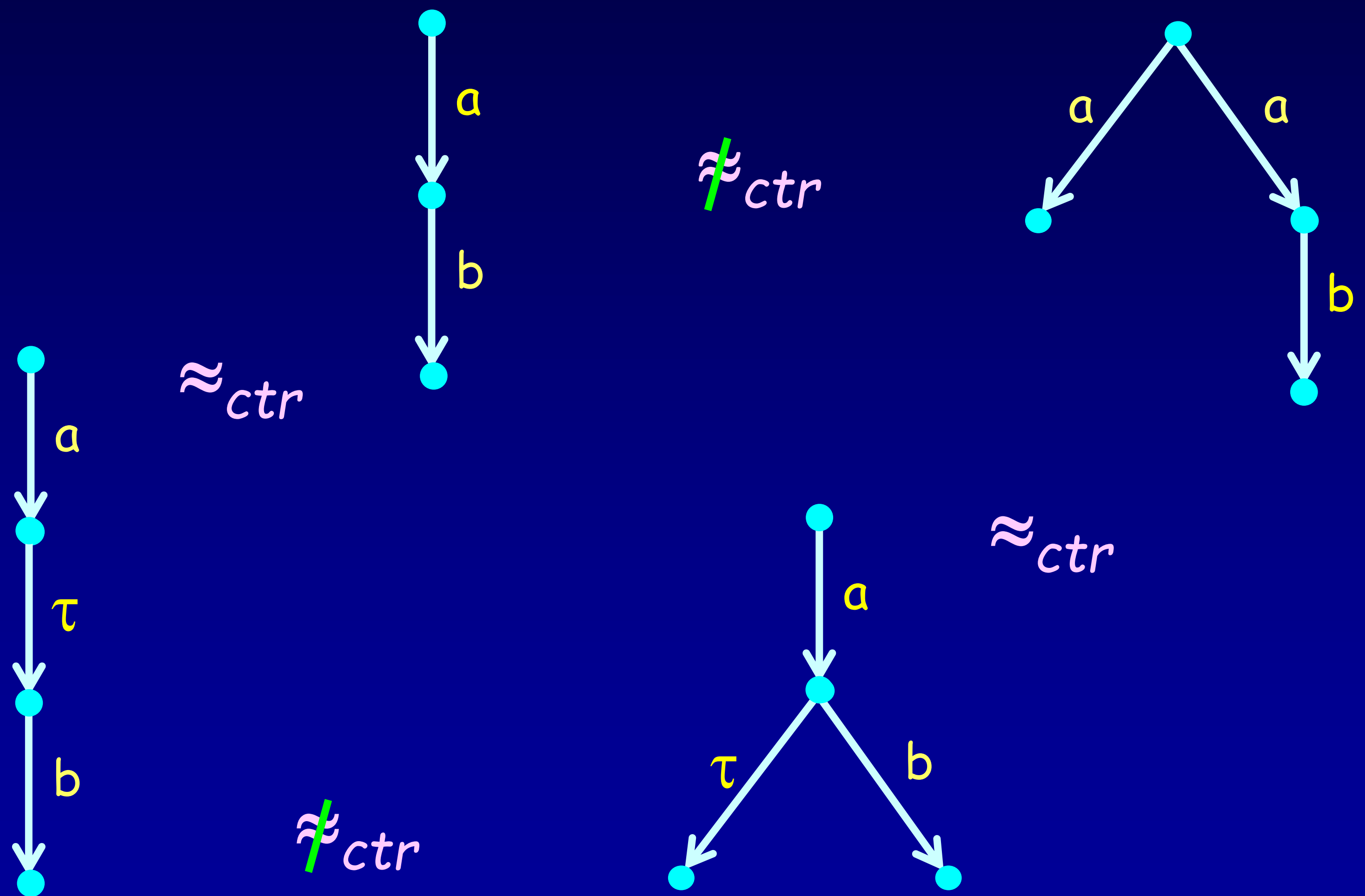
$S1 \approx_{tr} S2 \quad \Leftrightarrow \quad traces(S1) = traces(S2)$

Traces: $traces(s) = \{ \sigma \in L^* \mid s \stackrel{\sigma}{\Longrightarrow} \}$

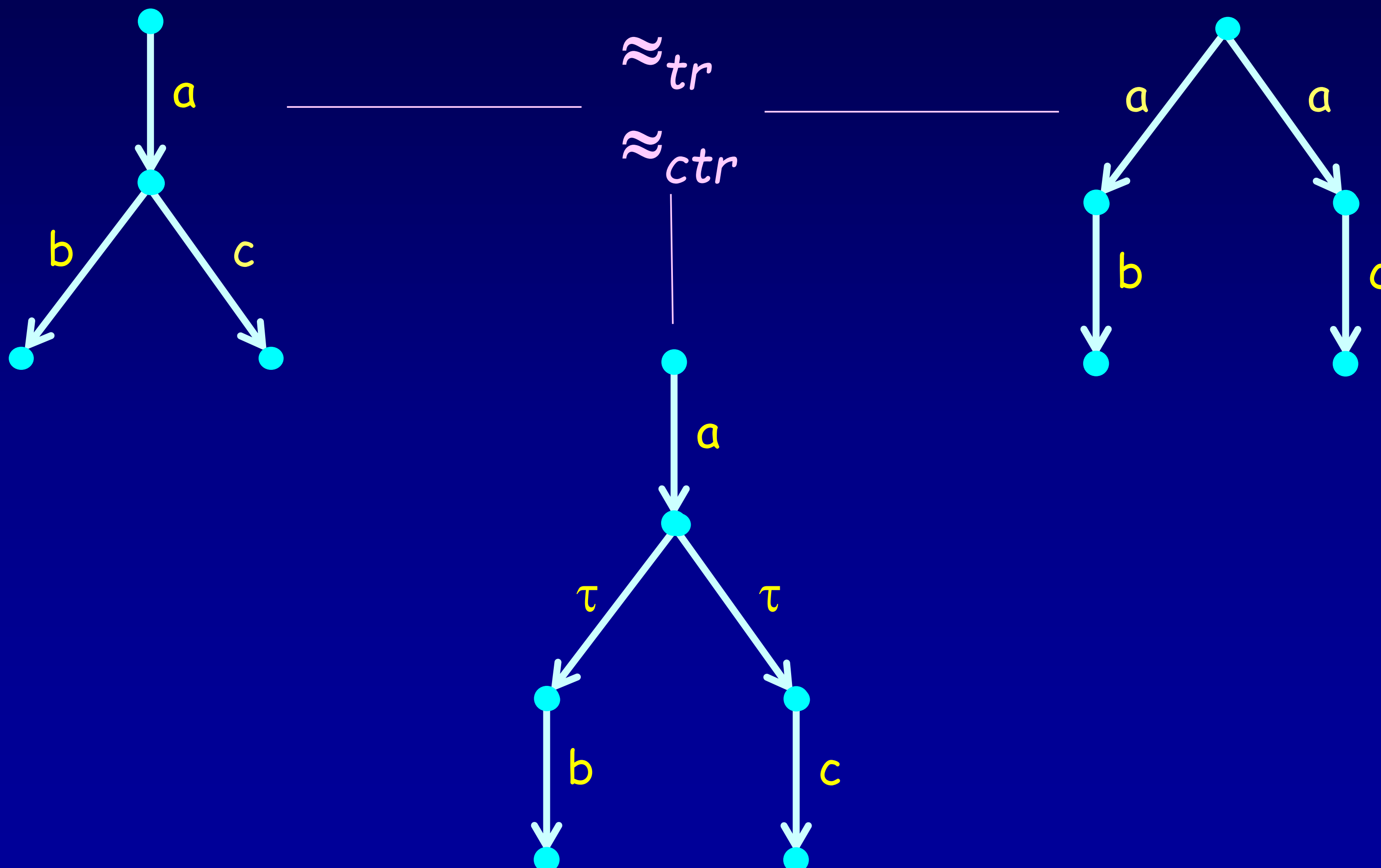
Trace Equivalence



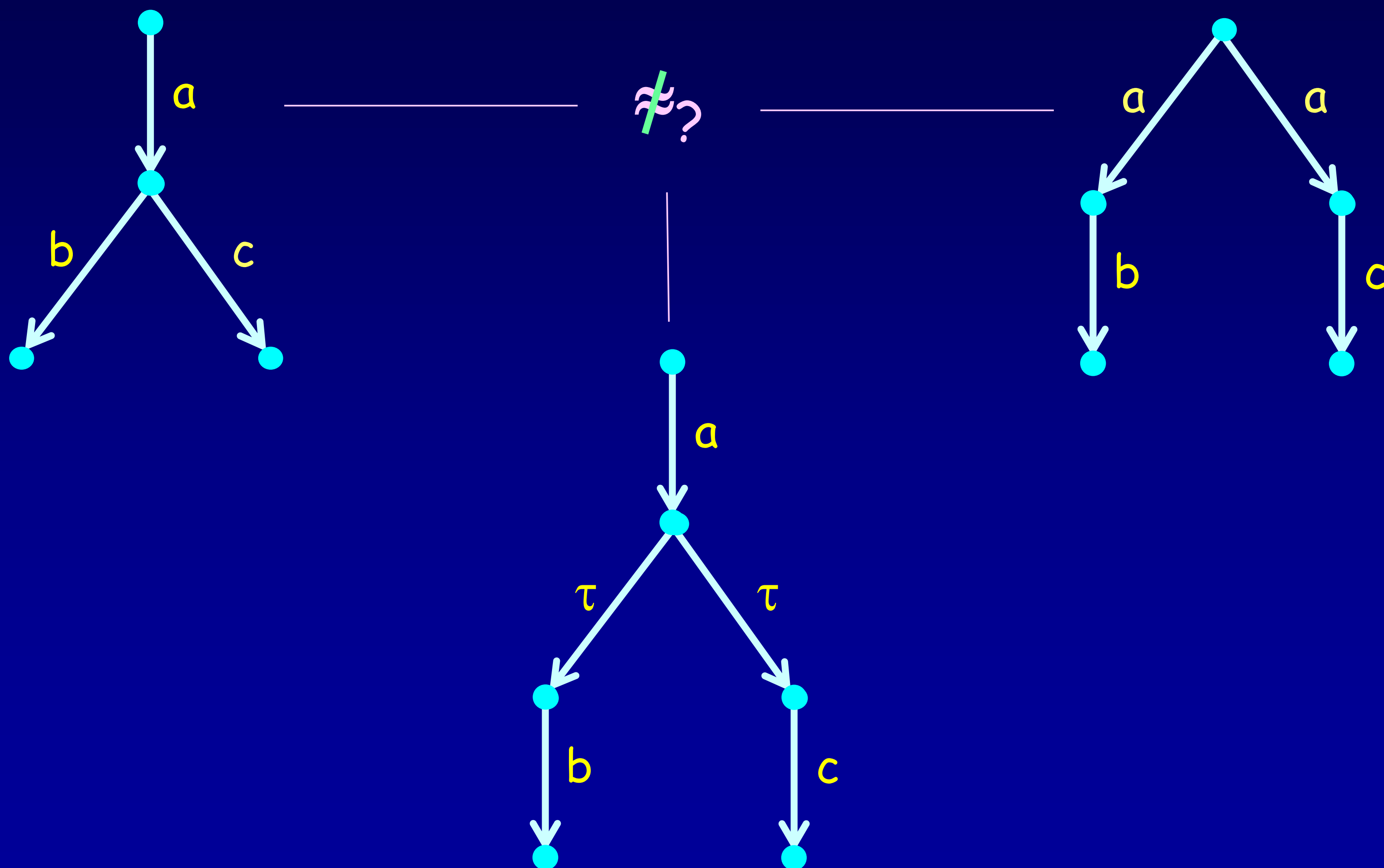
Completed Trace Equivalence



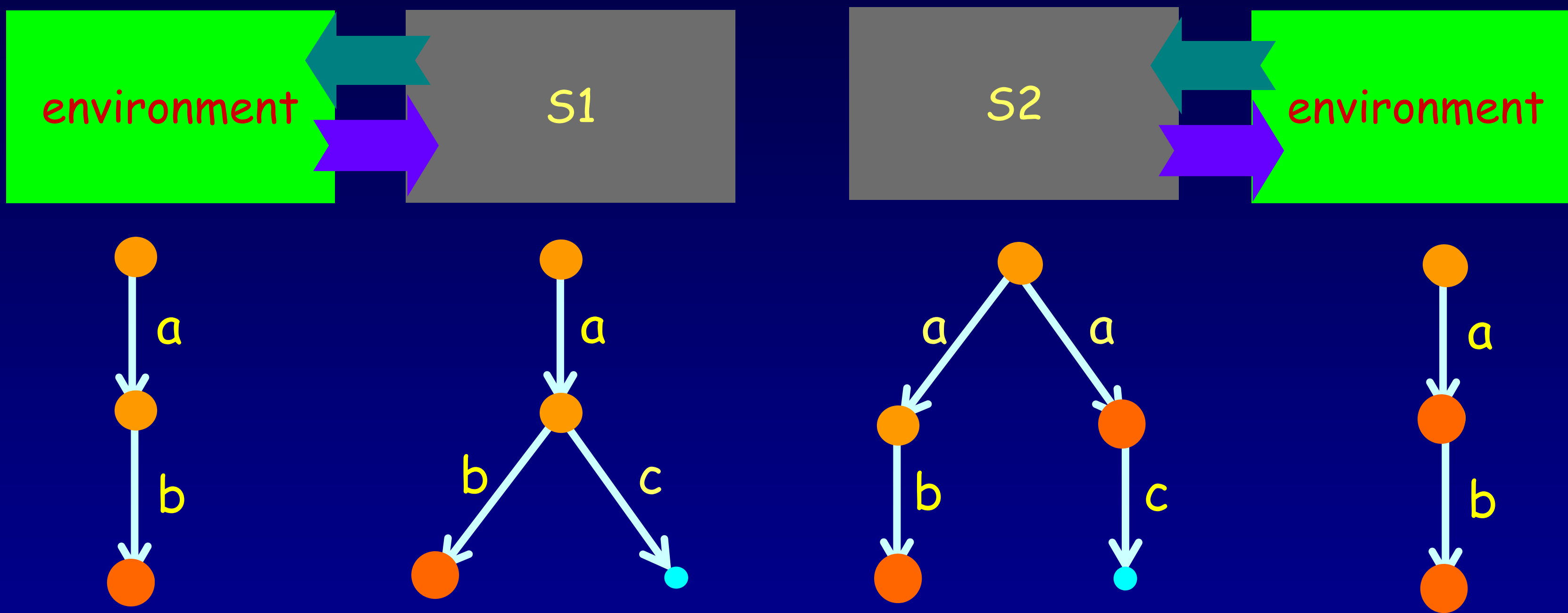
(Completed) Trace Equivalence



(Completed) Trace Equivalence : Others ?



Comparing Systems : Testing Equivalence



a b √

$\not\approx_{te}$

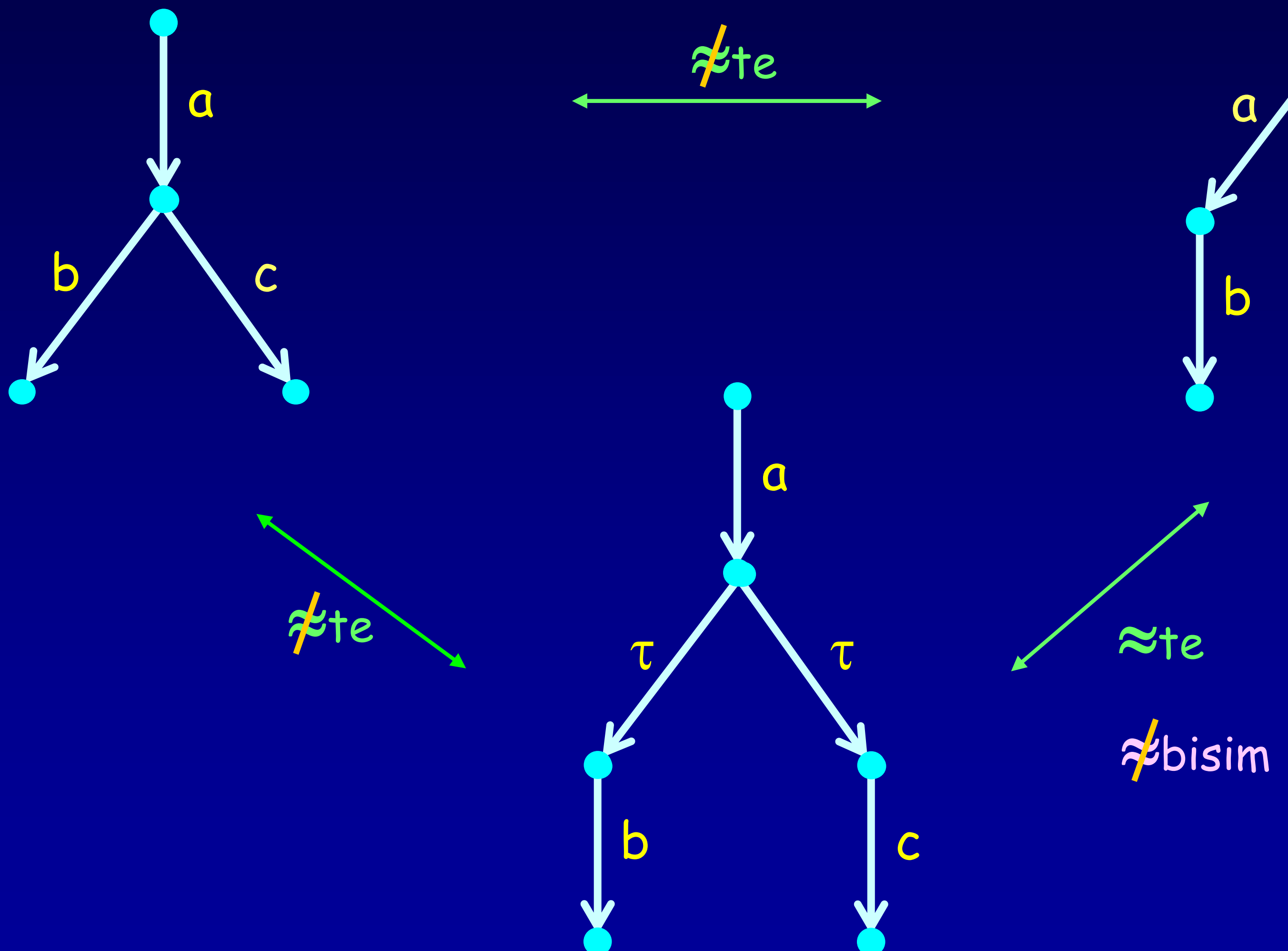
a b √

a √

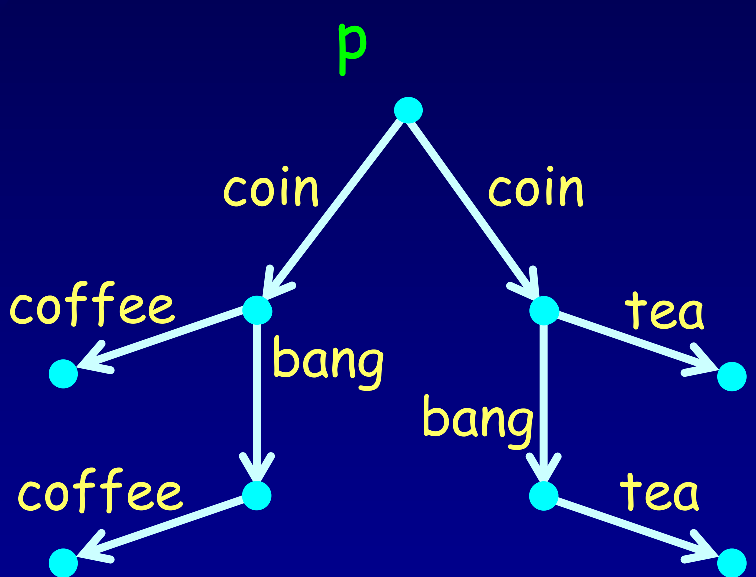
S1 after ~~a refuses~~ {b}

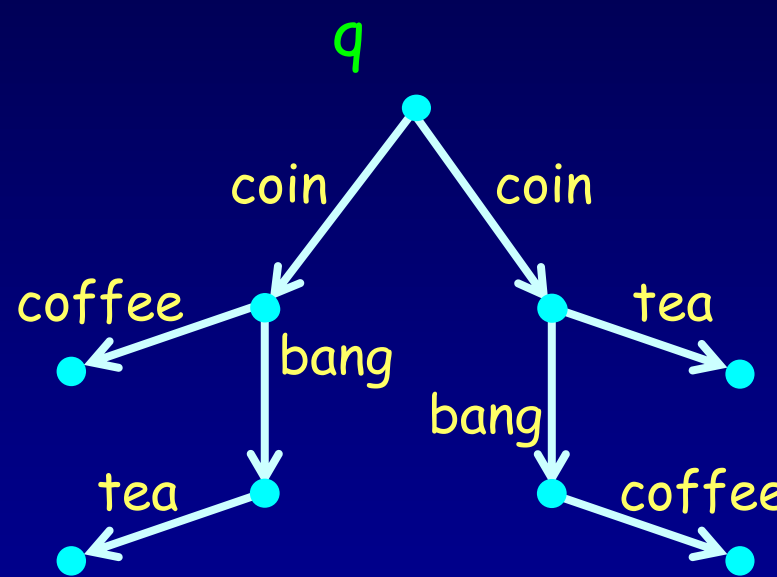
S2 after a refuses {b}

Testing Equivalence



Testing Equivalence



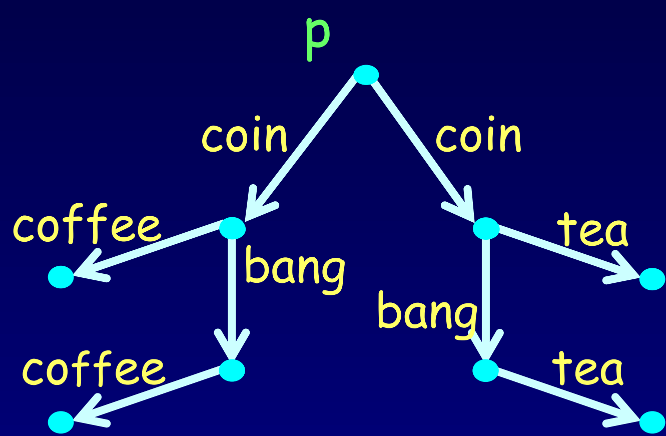


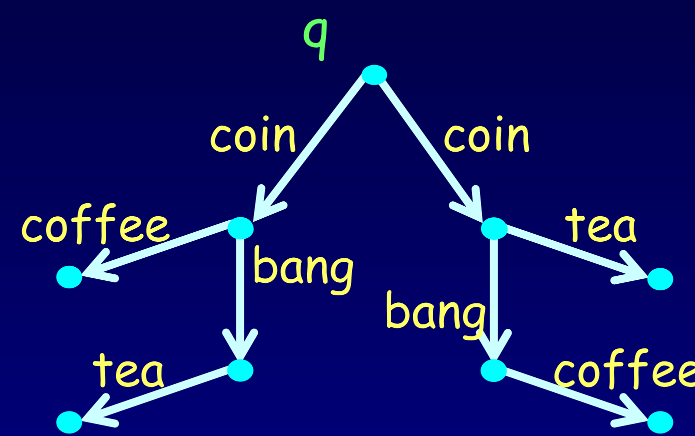
p $\approx_{te}$ q

But:

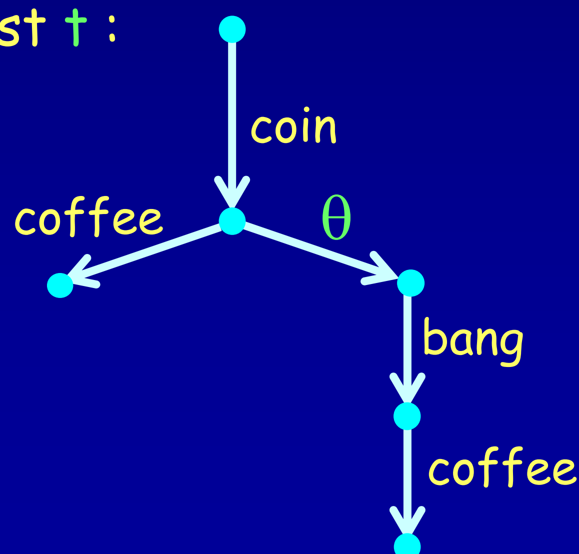
if you want coffee you will eventually always succeed in q but not p !?

Refusal Equivalence





Test t :



θ only possible
if nothing else is possible

coin θ bang coffee $\surd$ $\notin$ obs (p || t)

coin θ bang coffee $\surd$ $\in$ obs (q || t)

$p \not\approx_{rf} q$

Equivalences on Transition Systems

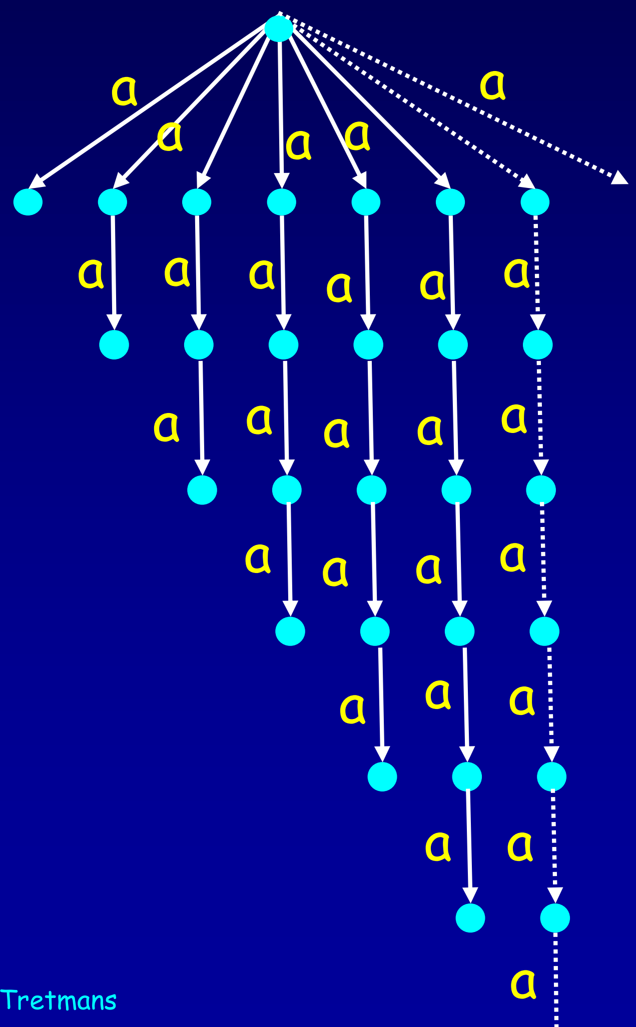
strong

weak

isomorphism	now you need to observe τ's ......
bisimulation (weak)	test an LTS with another LTS, and undo, copy, repeat as often as you like
failure trace = refusal	test an LTS with another LTS, and try again (continue) after failure
failures = testing	test an LTS with another LTS
completed trace	observing sequences of actions and their end
trace	observing sequences of actions

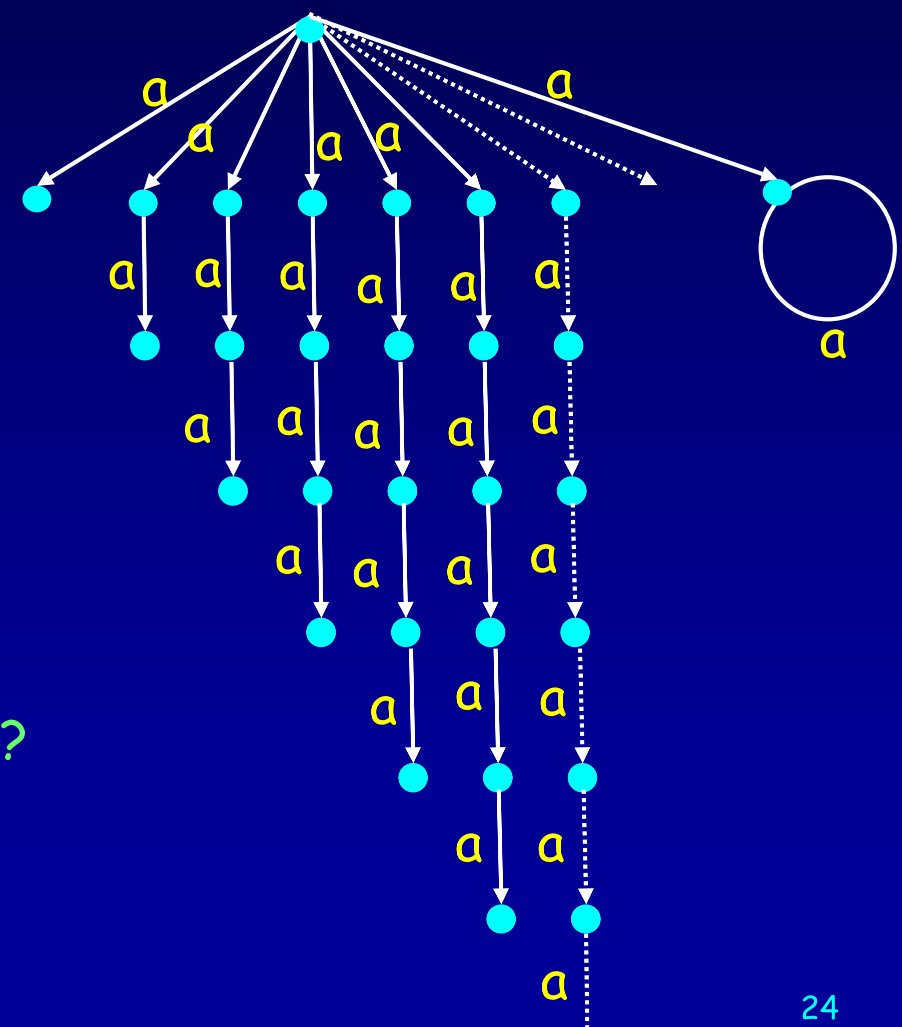
Equivalences : Examples

p



$\approx$?

q



Non-Equivalence Relations on Labelled Transition Systems

Implementation Relations

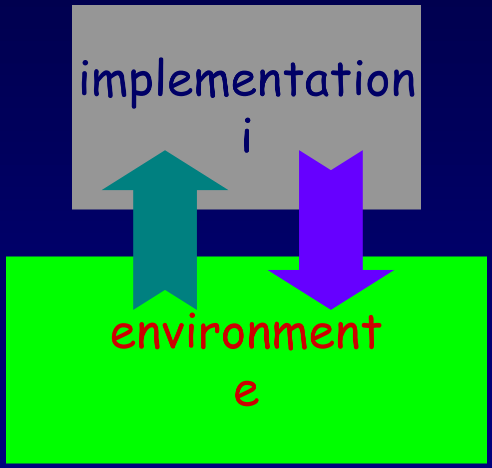
Conformance Relations

Refinement Relations

Pre-Orders

Preorders on Transition Systems

$i \in$ LTS



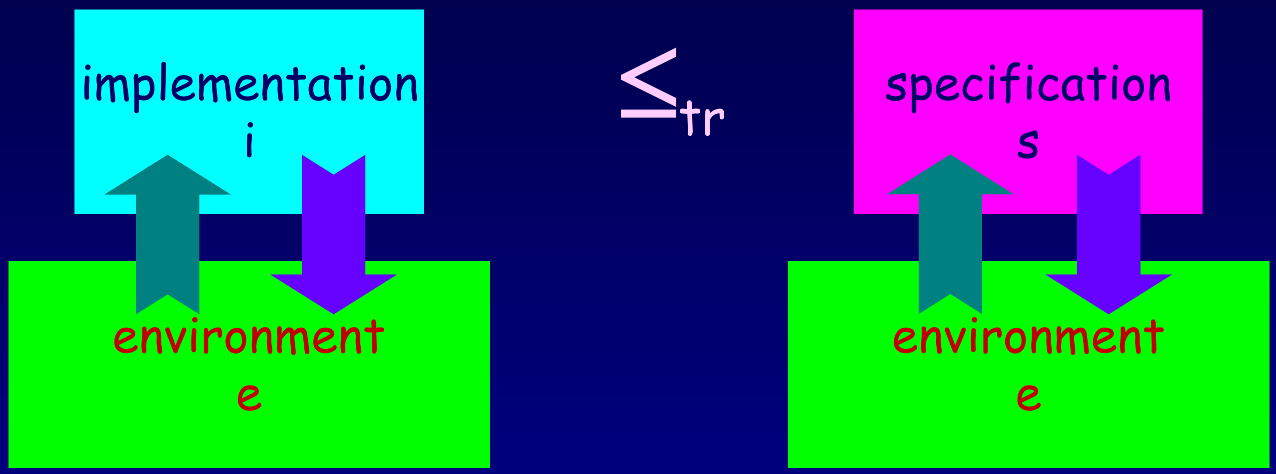
$\leq$

$s \in$ LTS

☞ Suppose an environment interacts with the black box implementation i and with the specification s :

- i correctly implements s
  if all observation of i can be related to observations of s

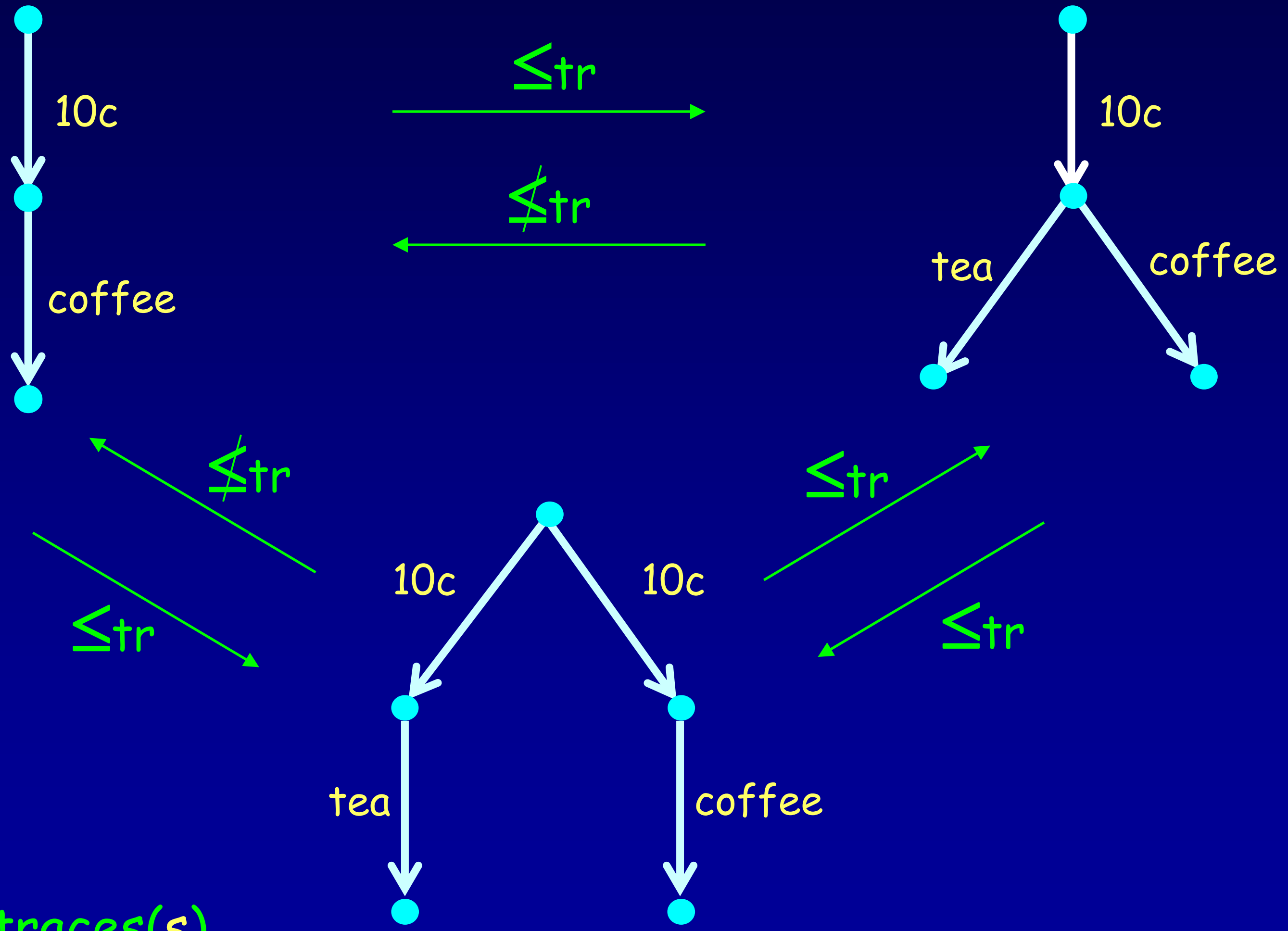
Trace Preorder



$$i \ \leq_{tr} \ s \quad \Leftrightarrow \quad traces(i) \subseteq traces(s)$$

Traces: $\quad traces(s) = \{ \sigma \in L^* \mid s \stackrel{\sigma}{\Longrightarrow} \}$

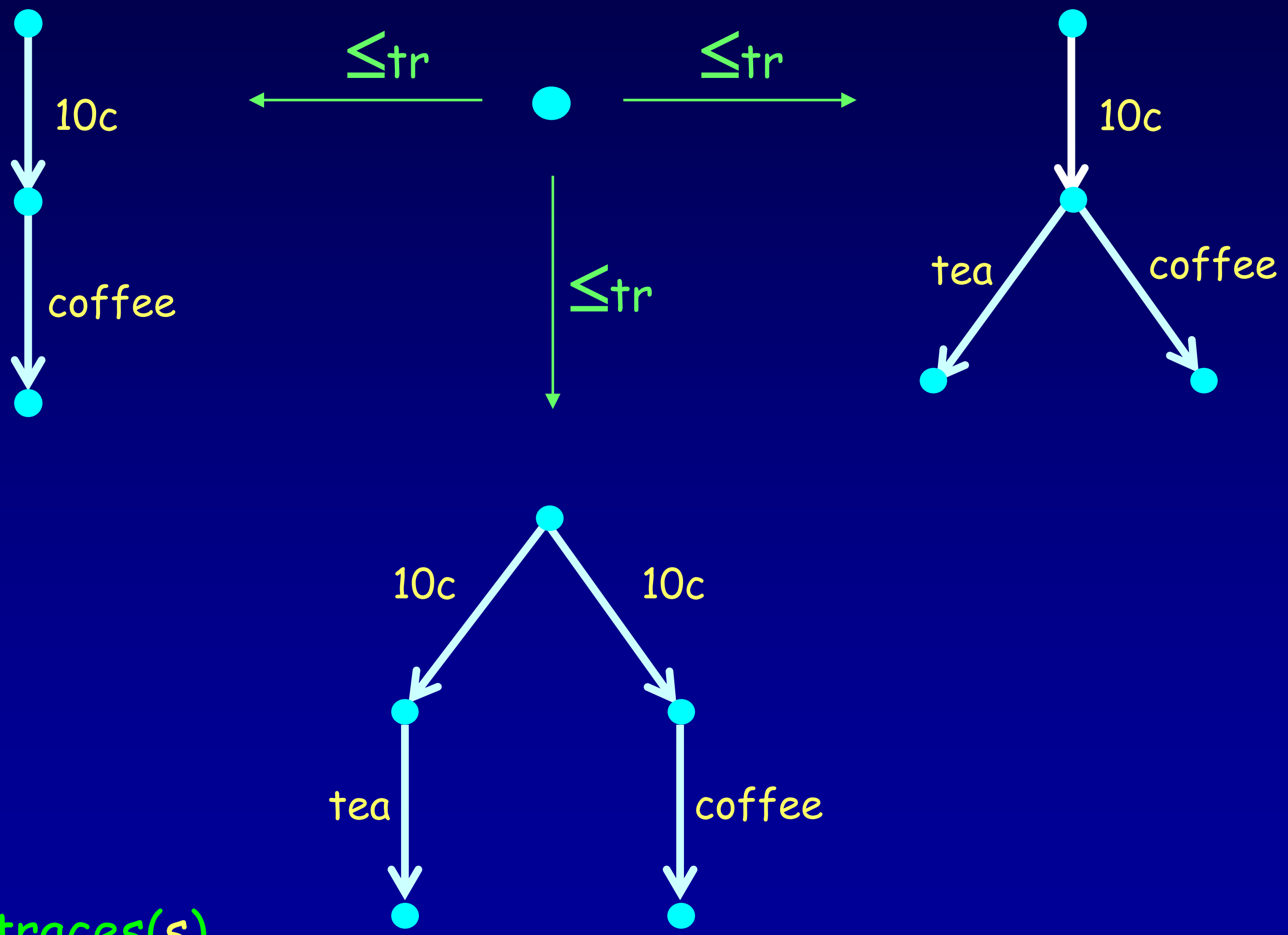
Trace Preorder



$i \leq_{tr} s$ =

$traces(i) \subseteq traces(s)$

Trace Preorder

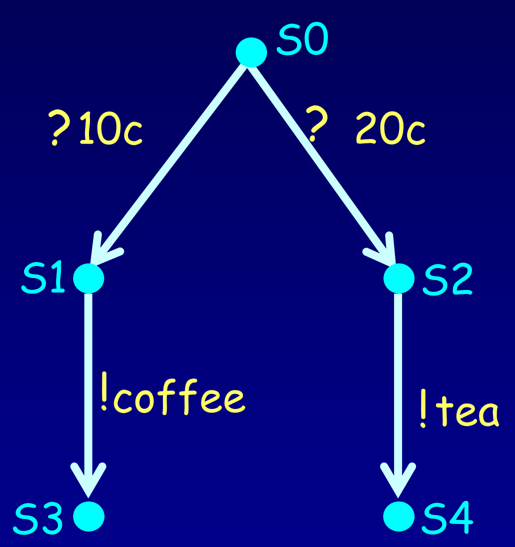


$i \leq_{tr} s$ =

$traces(i) \subseteq traces(s)$

Implementation Relation **ioco** for Labelled Transition Systems with Inputs and Outputs

Input-Output Transition Systems

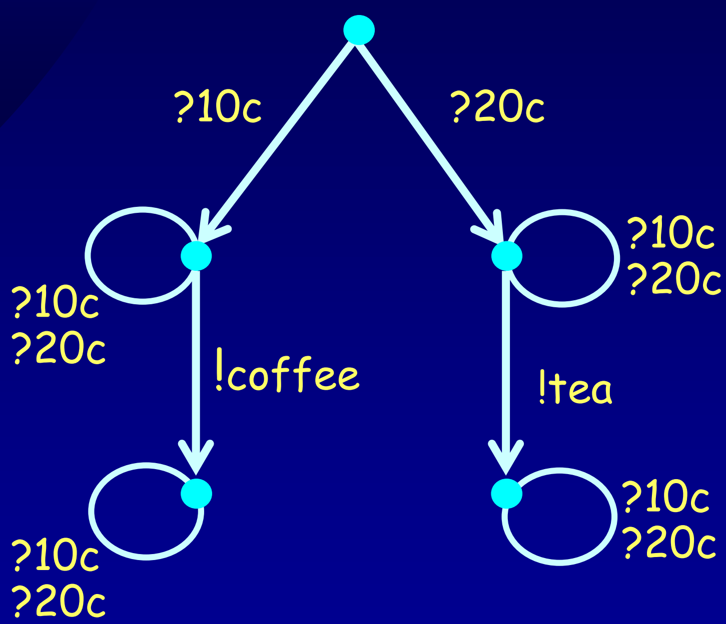


L_I = { ?10c, ?20c }

L_U = { !coffee, !tea }

10c, 20c	coffee, tea
from user to machine initiative with user machine cannot refuse	from machine to user initiative with machine user cannot refuse
input L_I	output L_U
$L_I \cap L_U = \emptyset$	$L_I \cup L_U = L$

Input-Output Transition Systems



L_I = { ?10c, ?20c }

L_U = { !coffee, !tea }

Input-Output Transition Systems

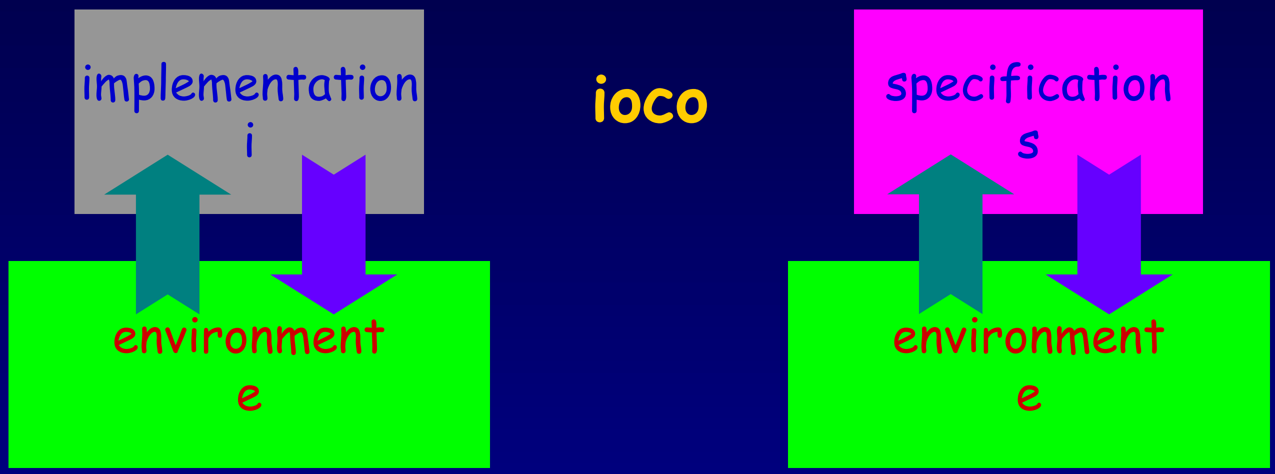
$IOTS(L_I, L_U) \subseteq LTS(L_I \cup L_U)$

IOTS is LTS with Input-Output and always enabled inputs:

for all states s,

for all inputs $?a \in L_I$: $s \stackrel{?a}{\Longrightarrow}$

Input-Output Transition Systems with ioco



$i \in IOTS(L_I, L_U)$ $\quad$ $s \in LTS(L_I, L_U)$

ioco $\subseteq IOTS(L_I, L_U) \times LTS(L_I, L_U)$

Observing IOTS where system inputs interact with environment outputs, and v.v.

Correctness

Implementation Relation **ioco**

$$i \ \textbf{ioco} \ s \ =_{def} \ \forall \sigma \in \mathit{Straces}(s) : \ \mathit{out}(i \ \textbf{after} \ \sigma) \subseteq \mathit{out}(s \ \textbf{after} \ \sigma)$$

$$p \xrightarrow{\delta} p \quad = \quad \forall \ !x \in L_U \cup \{\tau\} \ . \ p \overset{!x}{\not\longrightarrow}$$

$$\mathit{Straces}(s) \quad = \quad \{ \ \sigma \in (L \cup \{\delta\})^* \ | \ s \overset{\sigma}{\Longrightarrow} \ \}$$

$$p \ \textbf{after} \ \sigma \quad = \quad \{ \ p' \ | \ p \overset{\sigma}{\Longrightarrow} p' \ \}$$

$$\mathit{out}(P) \quad = \{ \ !x \in L_U \ | \ p \xrightarrow{!x}, \ p \in P \ \} \cup \{ \ \delta \ | \ p \xrightarrow{\delta} p, \ p \in P \ \}$$

Correctness

Implementation Relation **ioco**

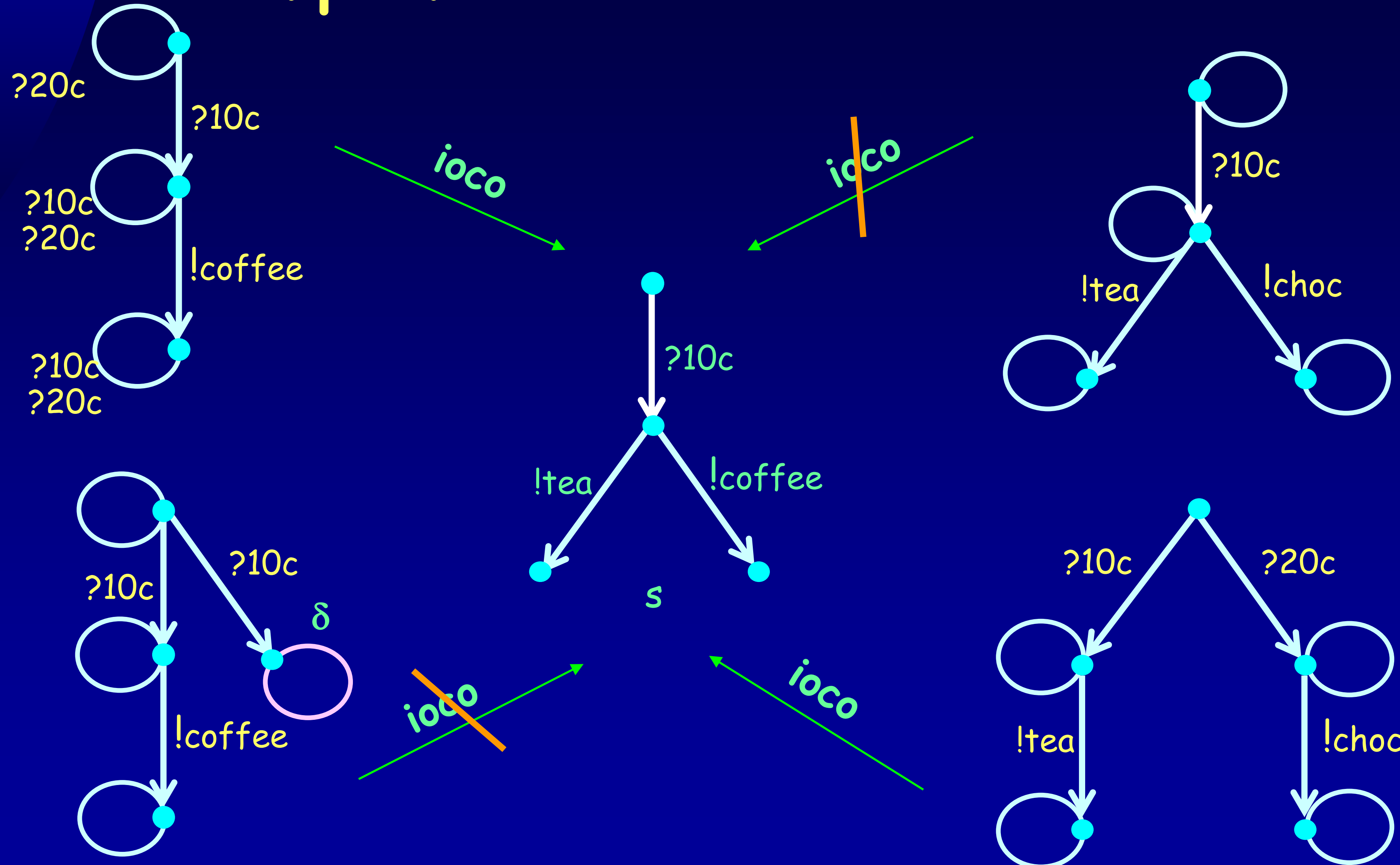
$$i \ \mathbf{ioco}\ s \ =_{def} \ \forall \sigma \in \mathit{Straces}(s):\ \mathit{out}(i\ \mathbf{after}\ \sigma) \subseteq \mathit{out}(s\ \mathbf{after}\ \sigma)$$

Intuition:

i **ioco**-conforms to s, iff

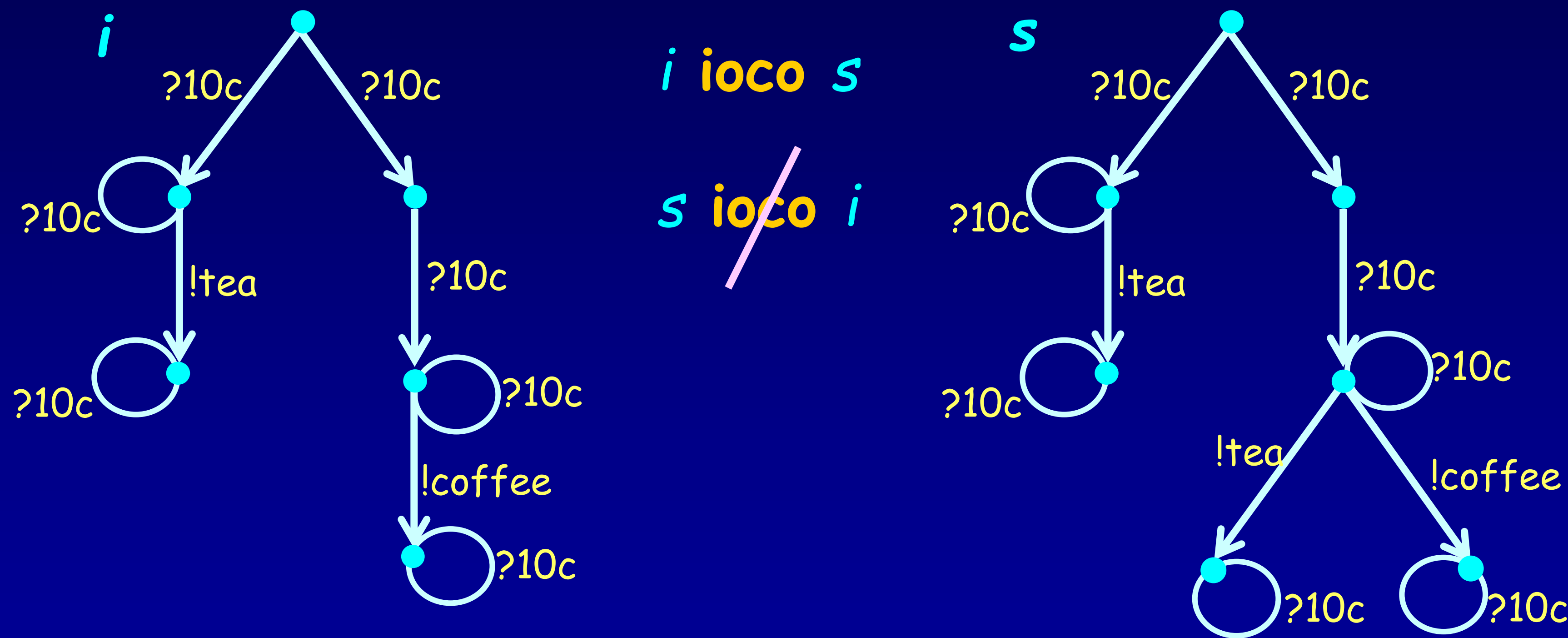
- if i produces output x after trace σ,
  then s can produce x after σ
- if i cannot produce any output after trace σ,
  then s cannot produce any output after σ (*quiescence* δ)

Implementation Relation ioco



Implementation Relation ioco

i **ioco** $s \;=_{def}\; \forall \sigma \in Straces\,(s) : \; out\,(i \textbf{ after } \sigma) \subseteq out\,(s \textbf{ after } \sigma)$

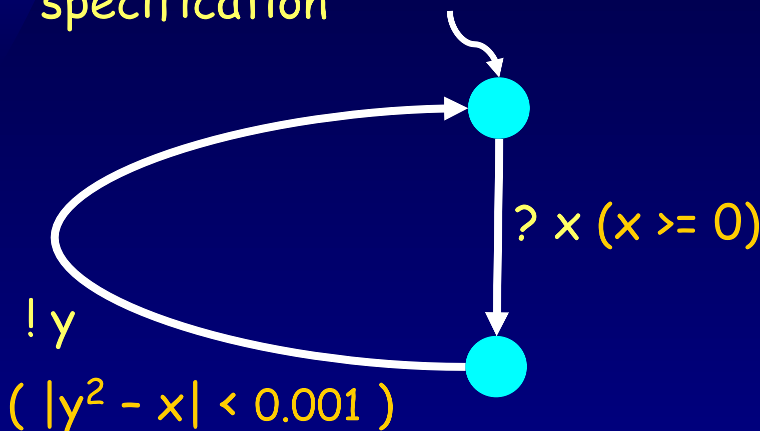


$out\,(i \textbf{ after } ?10c.?10c) = out\,(s \textbf{ after } ?10c.?10c) = \{\, !tea, !coffee \,\}$

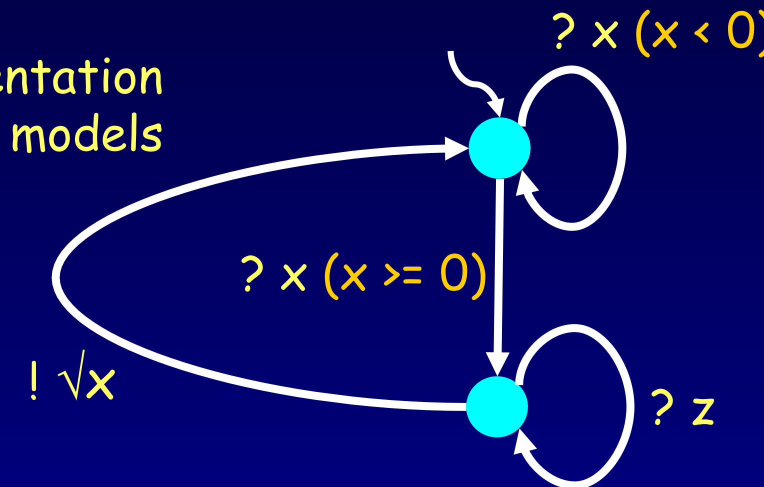
$out\,(i \textbf{ after } ?10c.\delta.?10c) = \{\, !coffee \,\} \neq out\,(s \textbf{ after } ?10c.\delta.?10c) = \{\, !tea, !coffee \,\}$

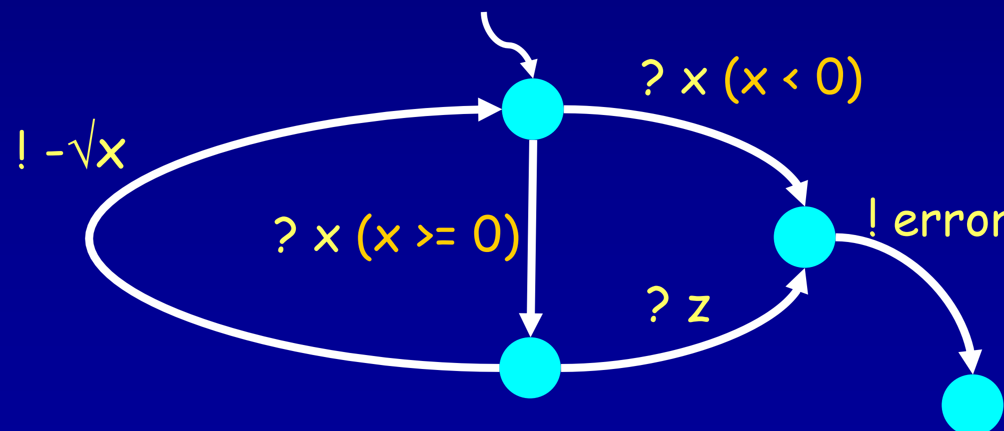
Implementation Relation ioco

specification



implementation
models

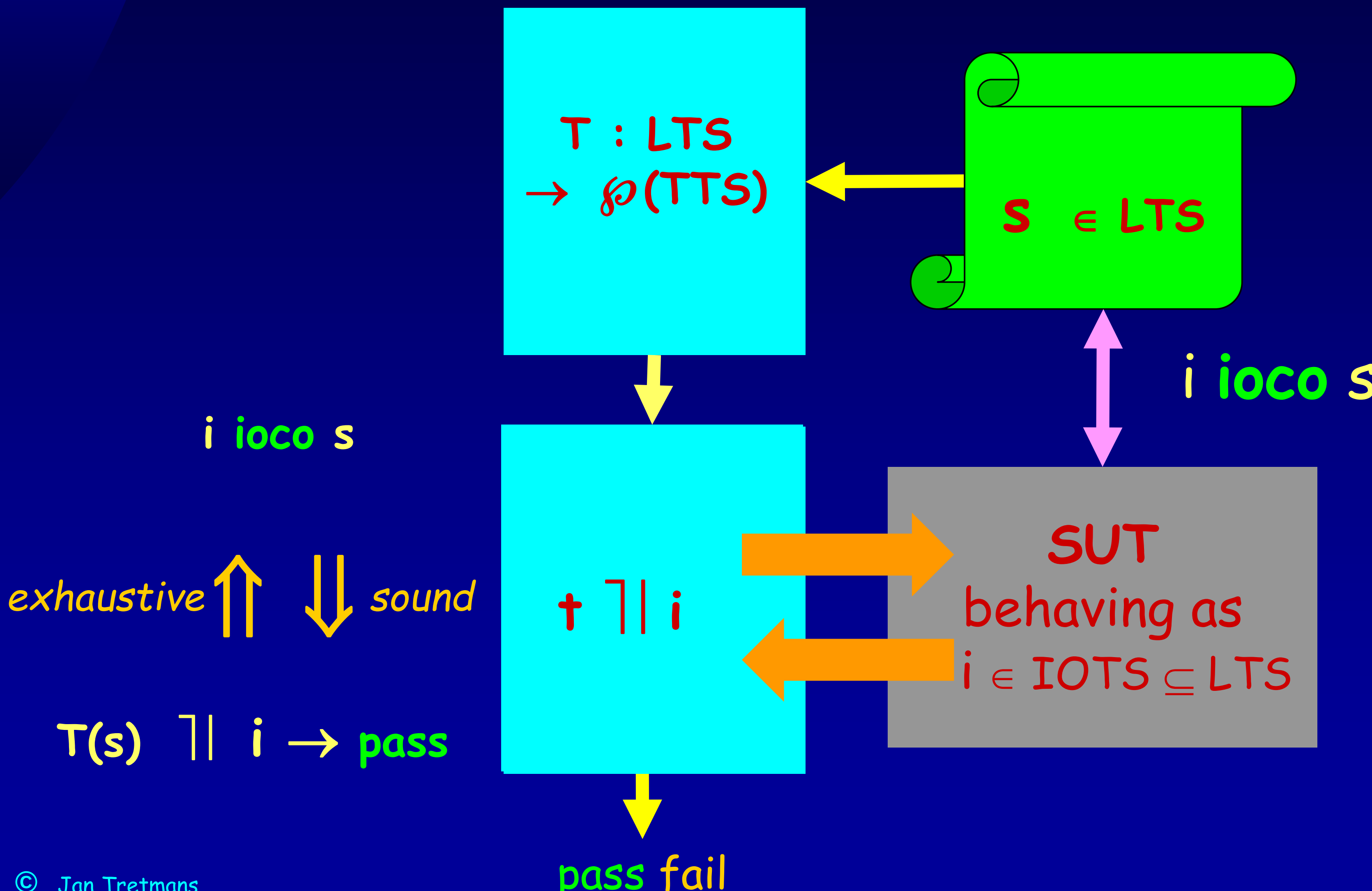




LTS and ioco allow:

- non-determinism
- under-specification
- the specification of properties rather than construction

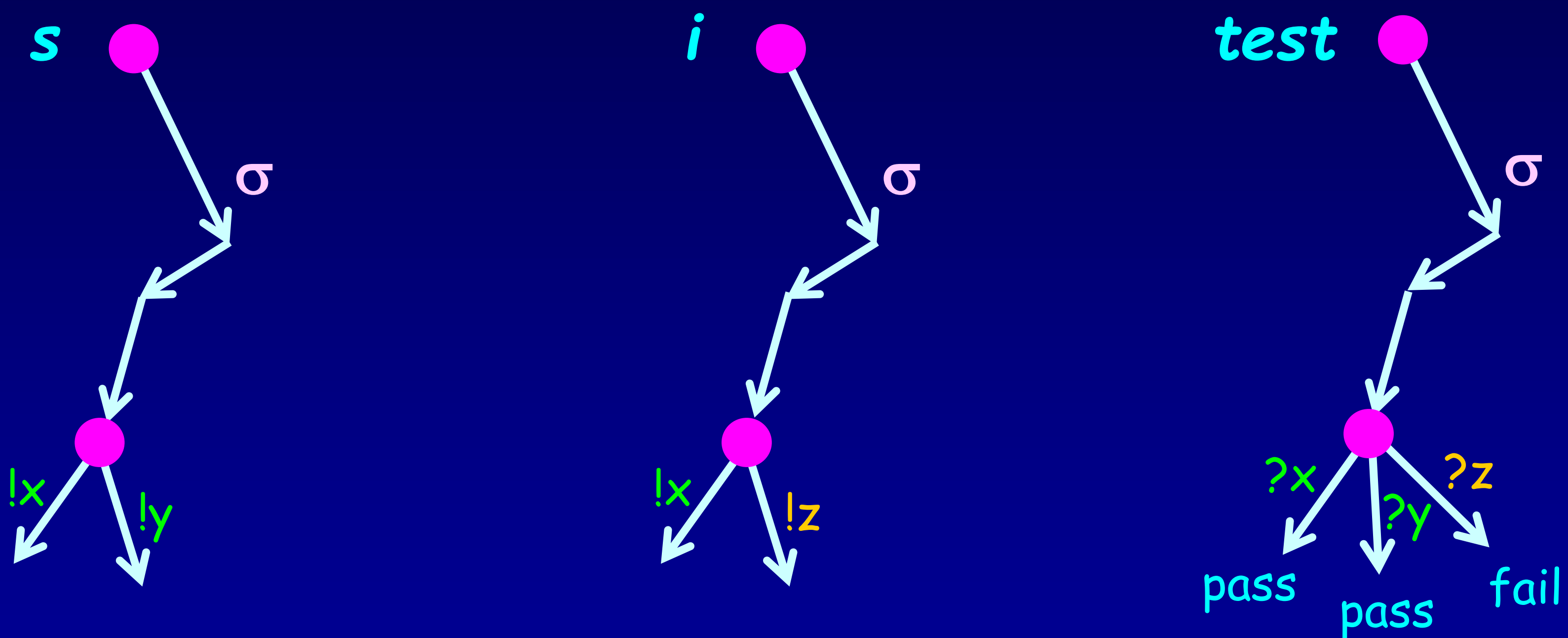
Model Based Testing with Transition Systems



Test Cases, Test Generation, and Test Execution for Labelled Transition Systems

Test Generation

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



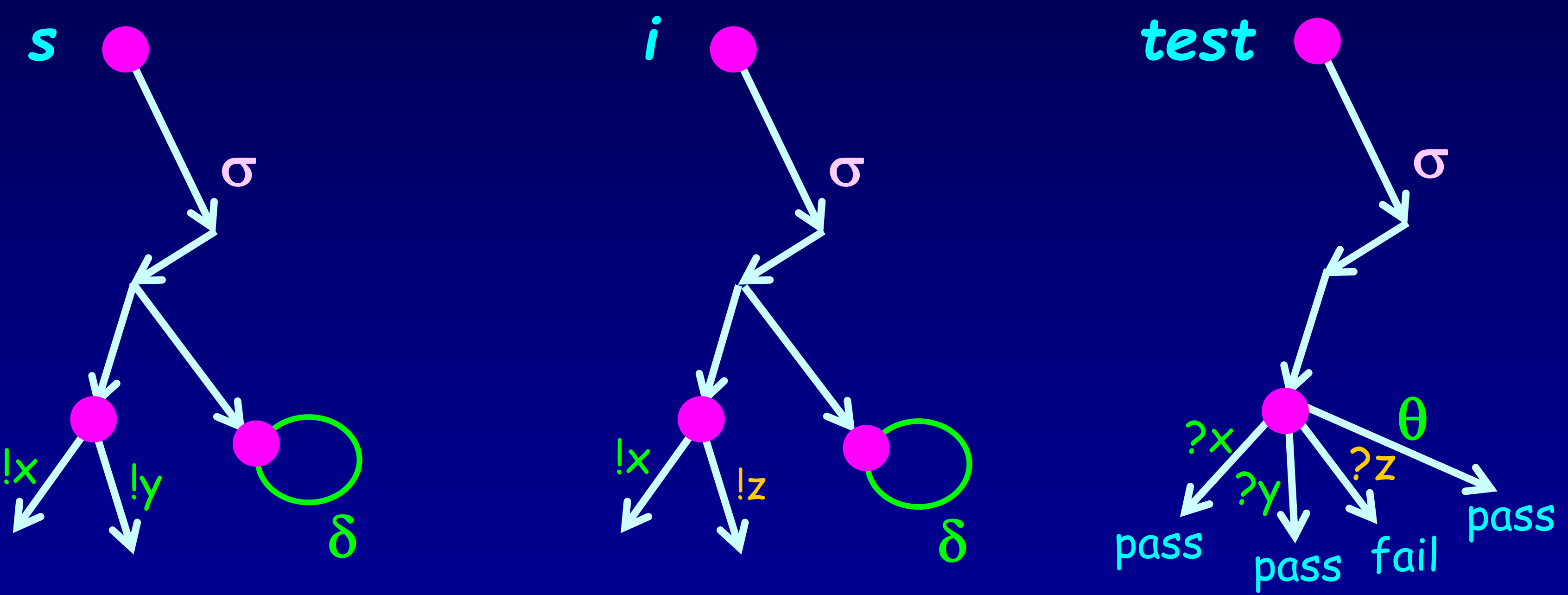
out (s **after** σ) = { !x, !y }

out (i **after** σ) = { !x, !z }

out (*test* **after** σ) = L_U

Test Generation

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



out (s **after** σ)

= { !x, !y, δ }

out (i **after** σ)

= { !x, !z, δ }

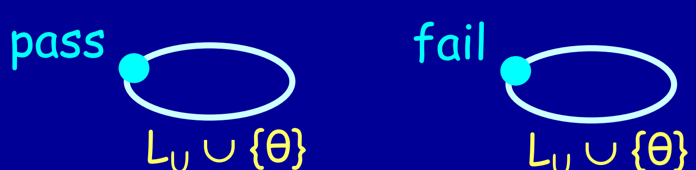
out (*test* **after** σ)

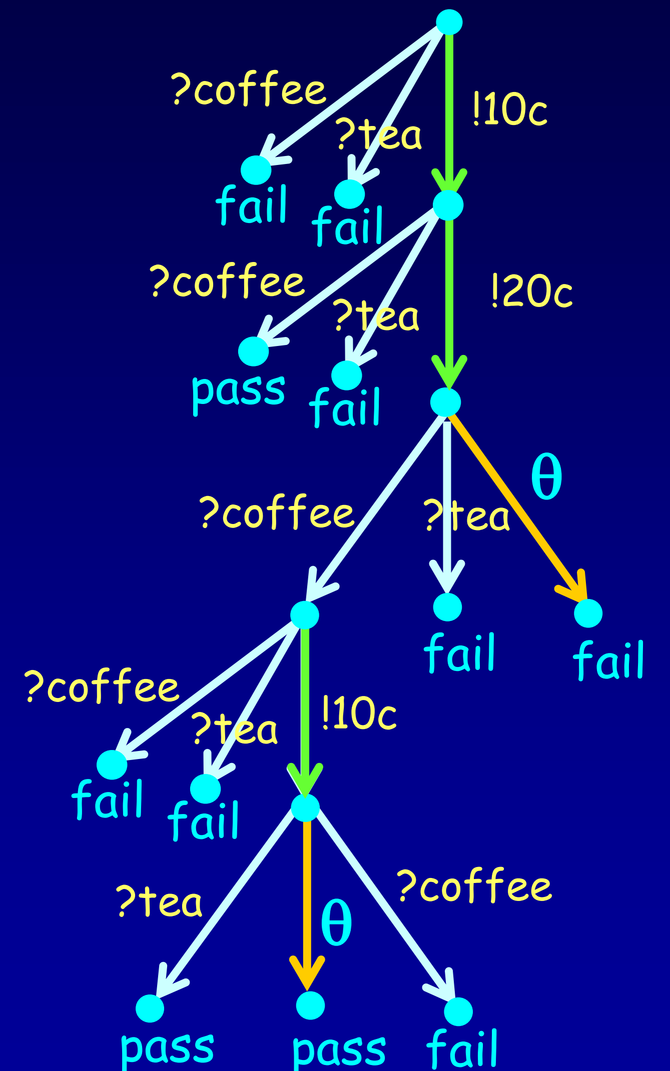
$= L_U \cup \{ \theta \}$

Test Cases

Model of a test case
= transition system :

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





Test Generation Algorithm

Algorithm

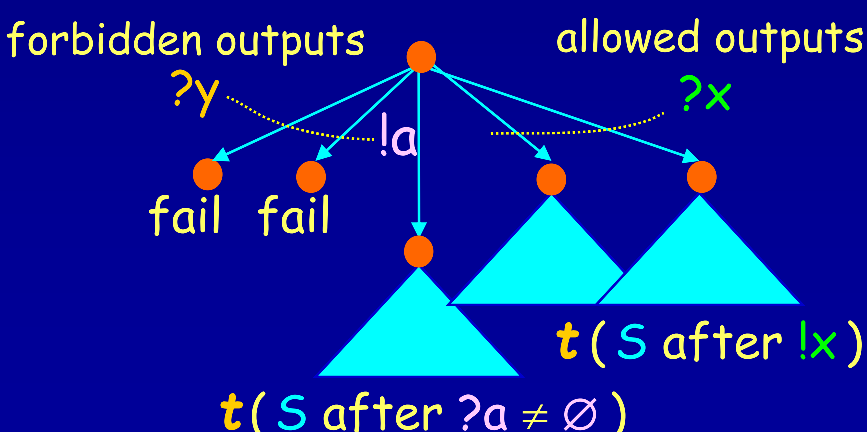
To generate a test case $\boldsymbol{t}(S)$ from a transition system specification S, with $S \neq \varnothing$: set of states (initially $S = s_0$ after ε)

Apply the following steps recursively, non-deterministically:

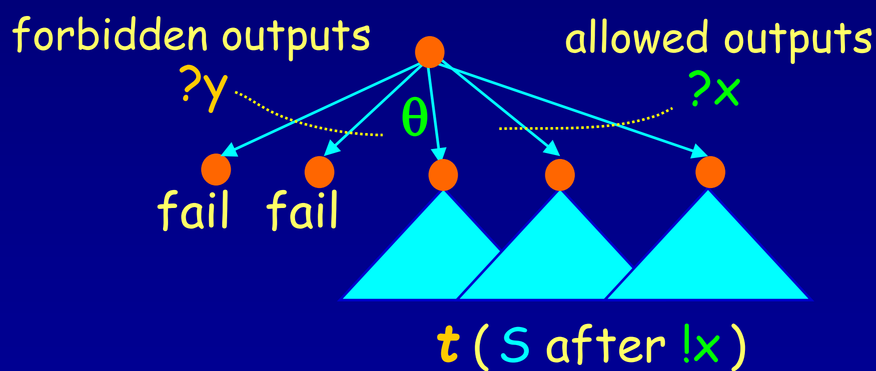
1 end test case

● pass

2 supply input !a



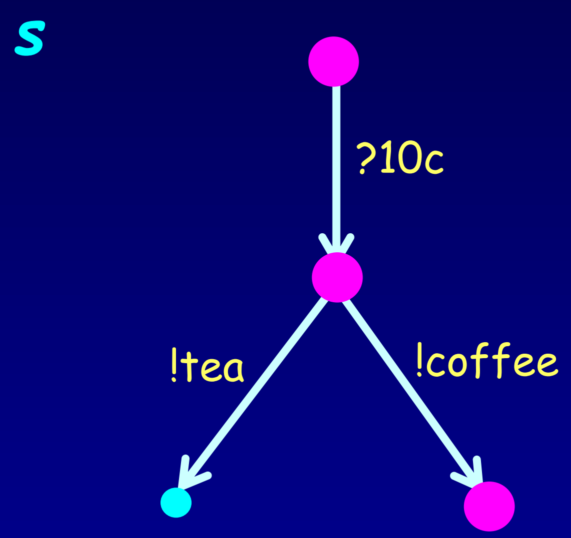
3 observe all outputs

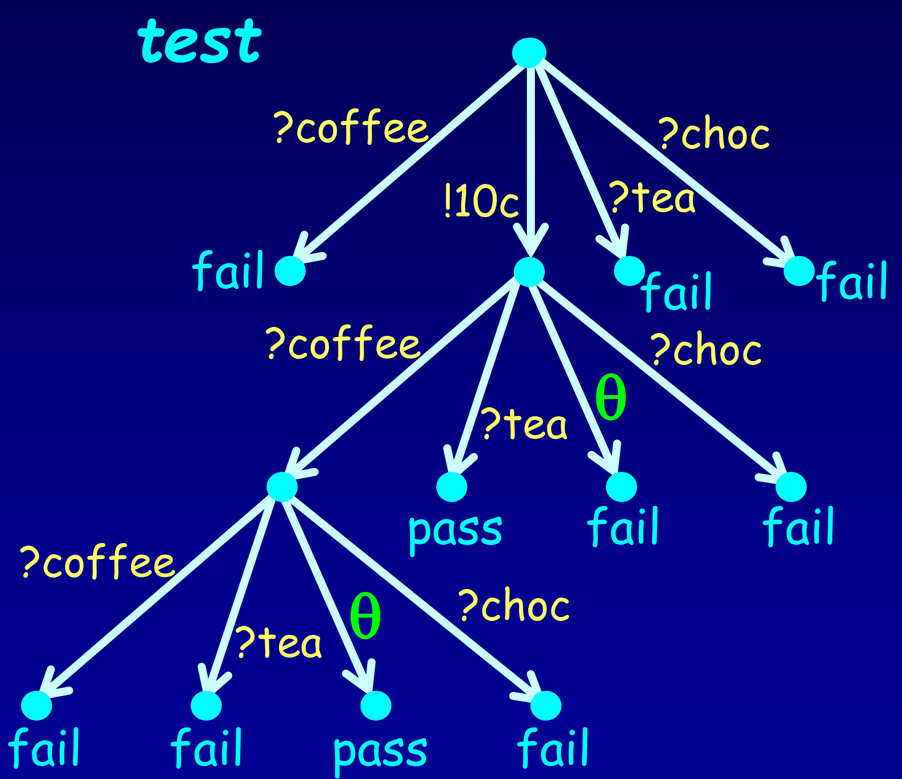


allowed outputs (or δ): !x $\in$ *out* (S)

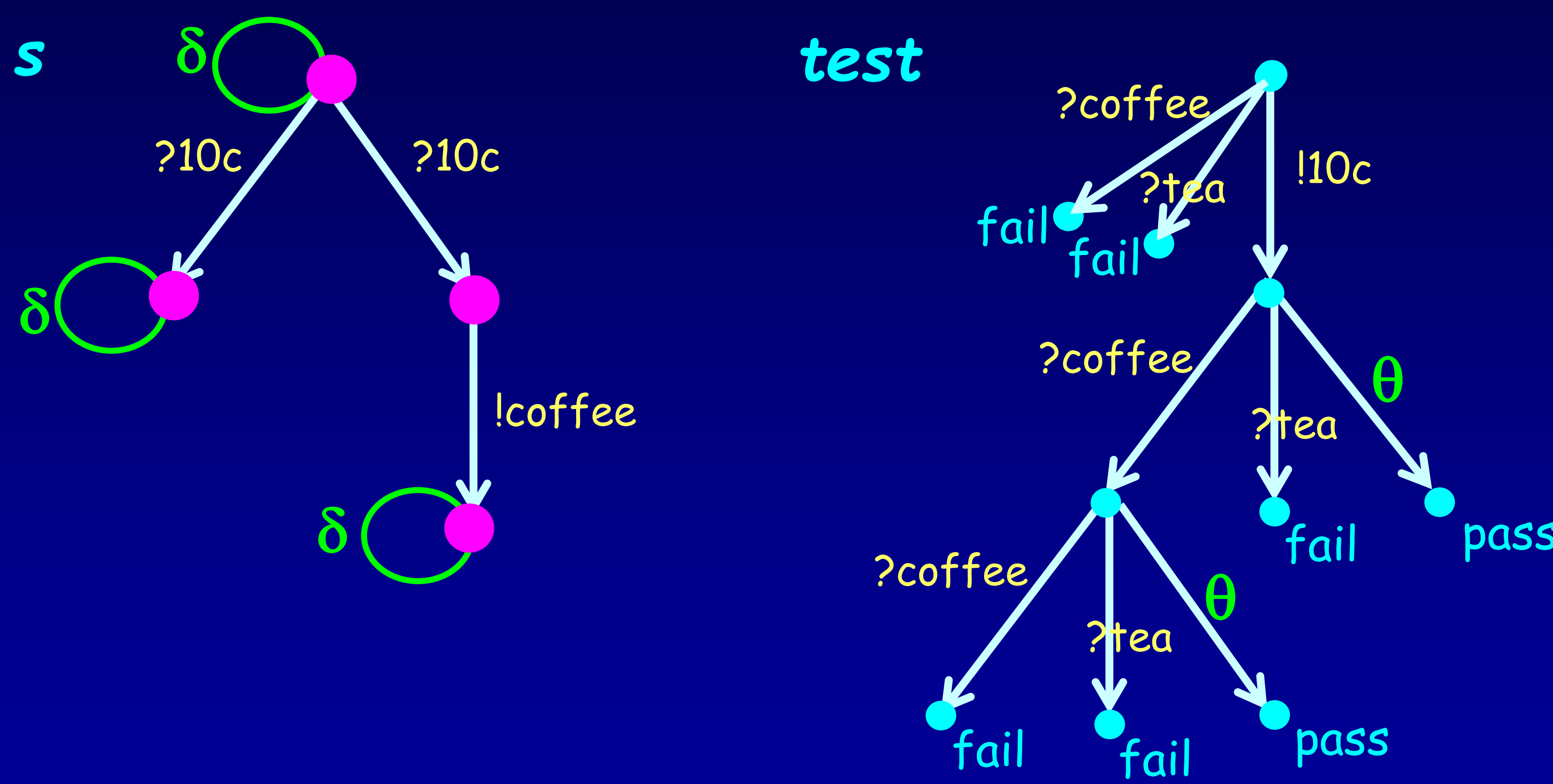
forbidden outputs (or δ): !y $\notin$ *out* (S)

Test Generation Example



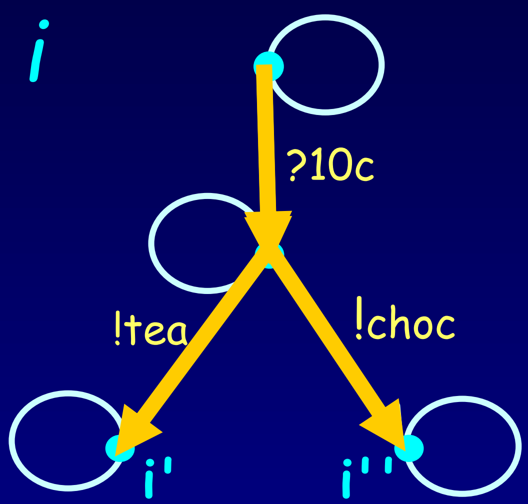


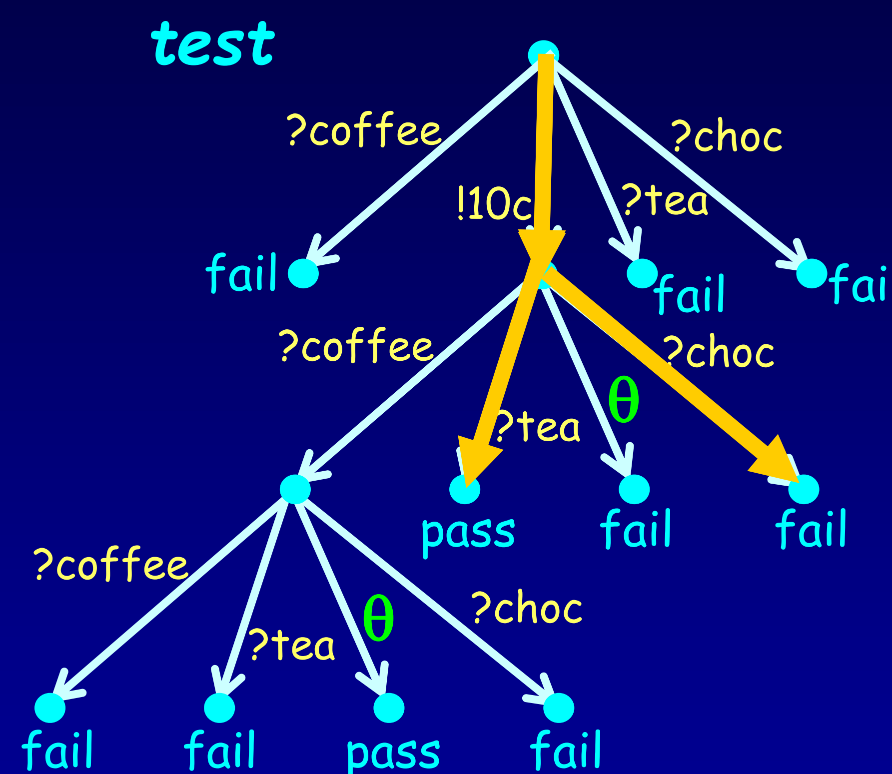
Test Generation Example



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

Test Execution Example





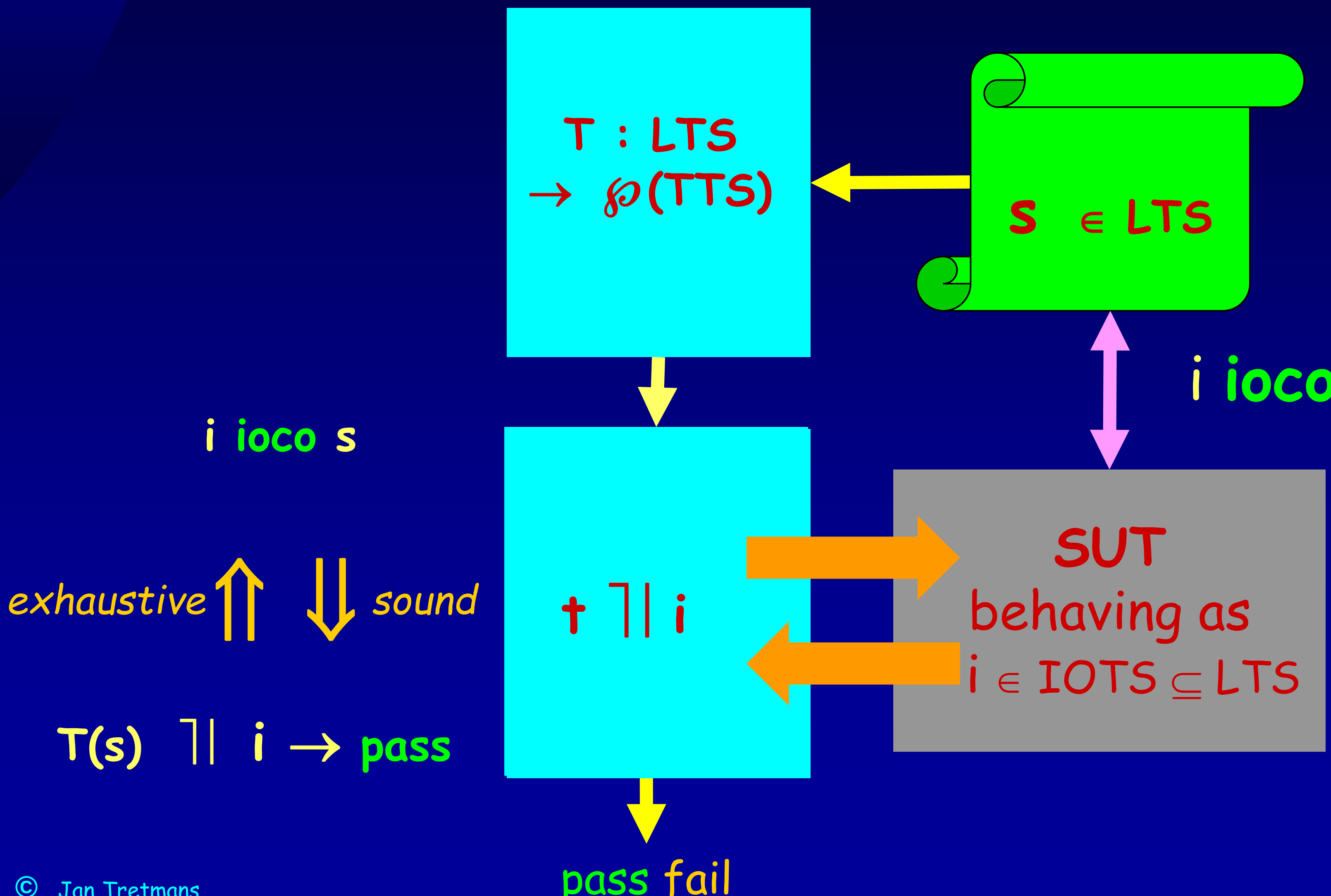
Two test runs :

$t \rceil\!|\, i \xRightarrow{\text{10c tea}} \text{pass} \rceil\!|\, i'$

$t \rceil\!|\, i \xRightarrow{\text{10c choc}} \text{fail} \rceil\!|\, i''$

i fails t

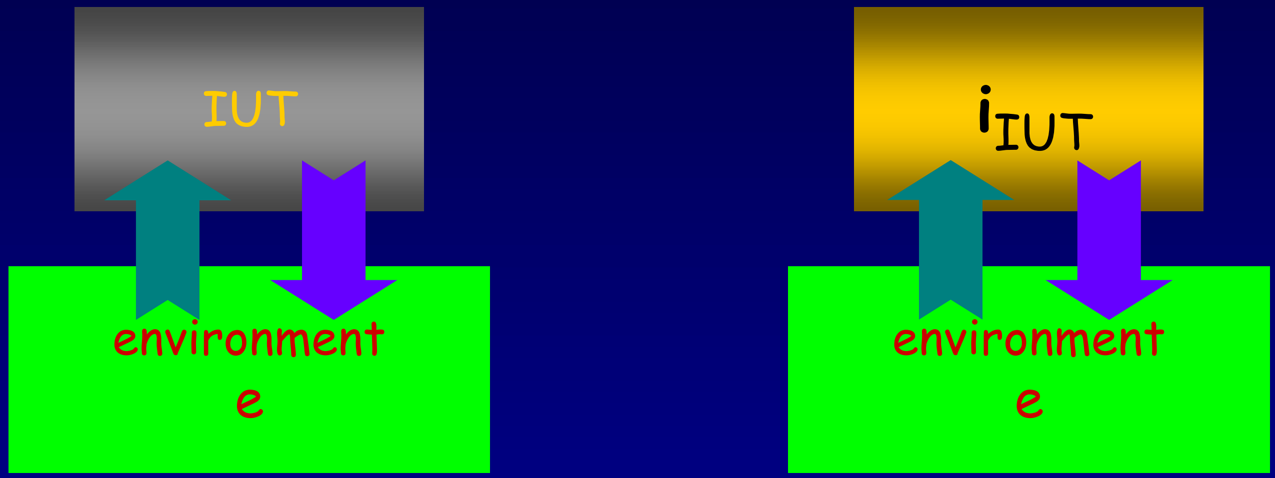
Model Based Testing with Transition Systems



Testability Assumption

(Test Hypothesis)

Comparing Transition Systems: An Implementation and a Model



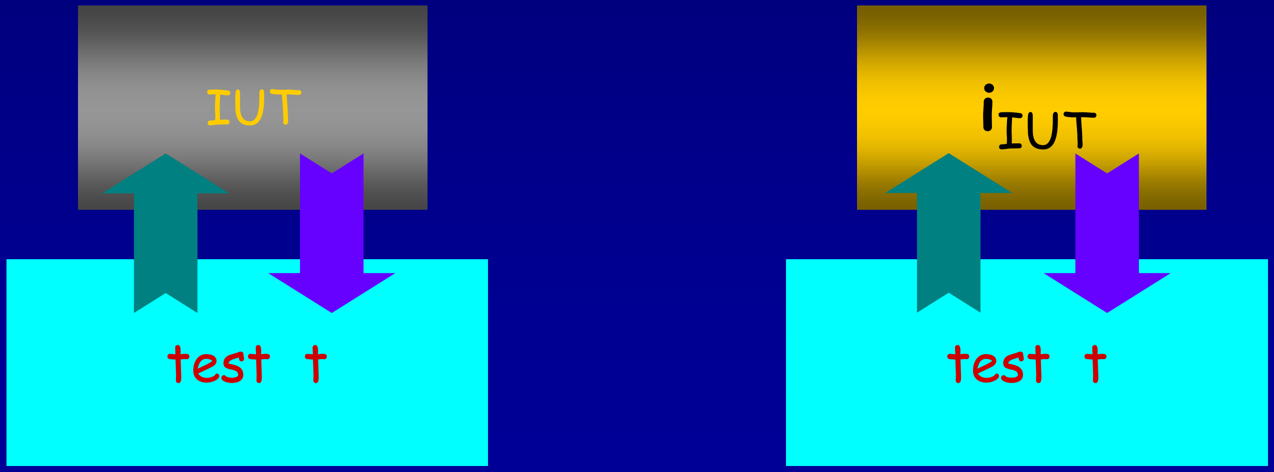
$$IUT \approx i_{IUT} \Leftrightarrow \forall\ e \in E\ .\ \ obs\,(e, IUT) = obs\,(e, i_{IUT})$$

Formal Testing : Test Assumption

Test assumption :

$\forall$ IUT . $\exists$ $i_{IUT} \in$ MOD.

$\forall$ t $\in$ TEST . IUT passes t $\Leftrightarrow$ i_{IUT} passes t



Soundness and Exhaustiveness

Validity of Test Generation

For every test t generated with algorithm we have:

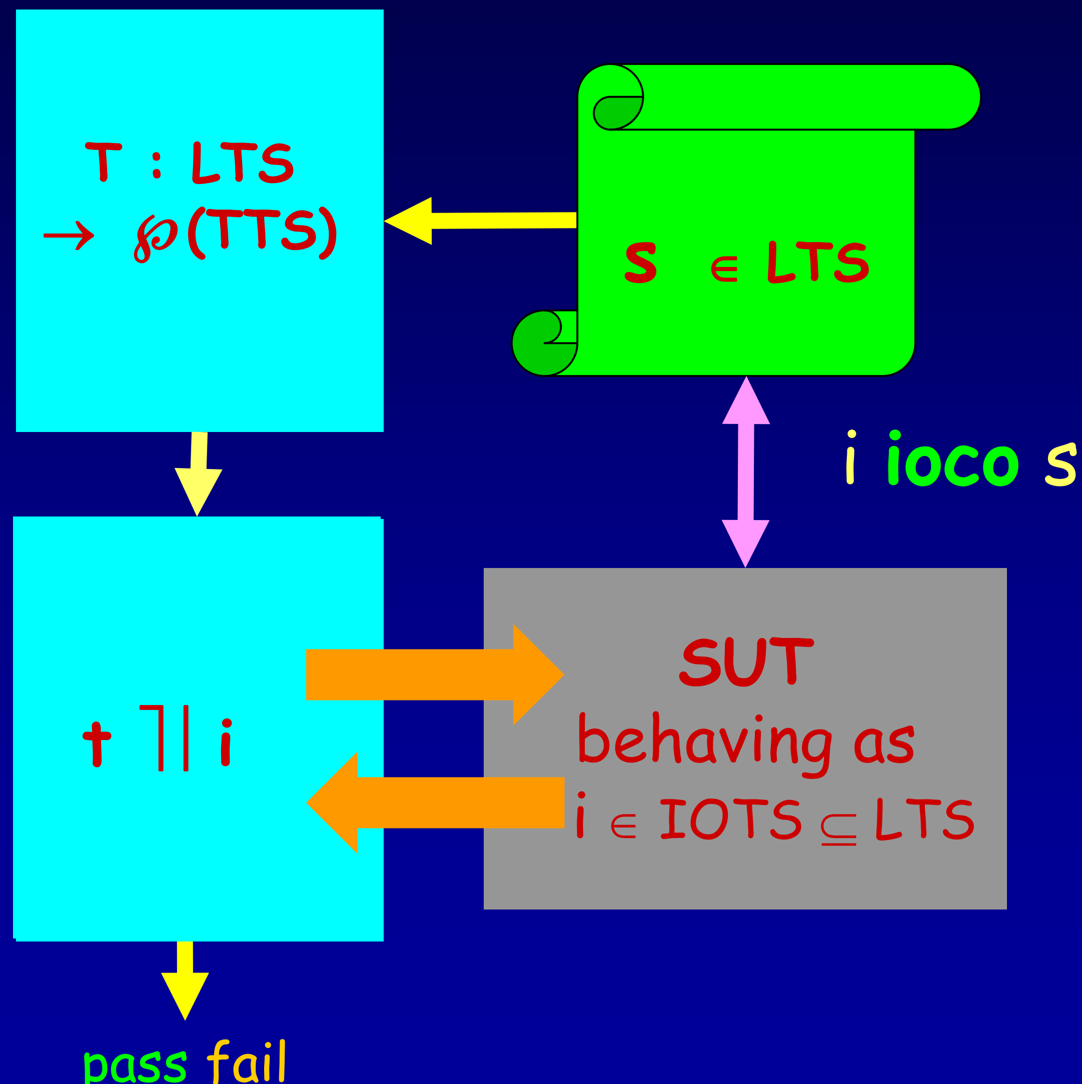
☞ Soundness :
t will never fail with correct implementation

i **ioco** s   implies   i passes t

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

i $\cancel{\textbf{ioco}}$ s   implies $\exists$ t : i fails t

Model Based Testing with Transition Systems

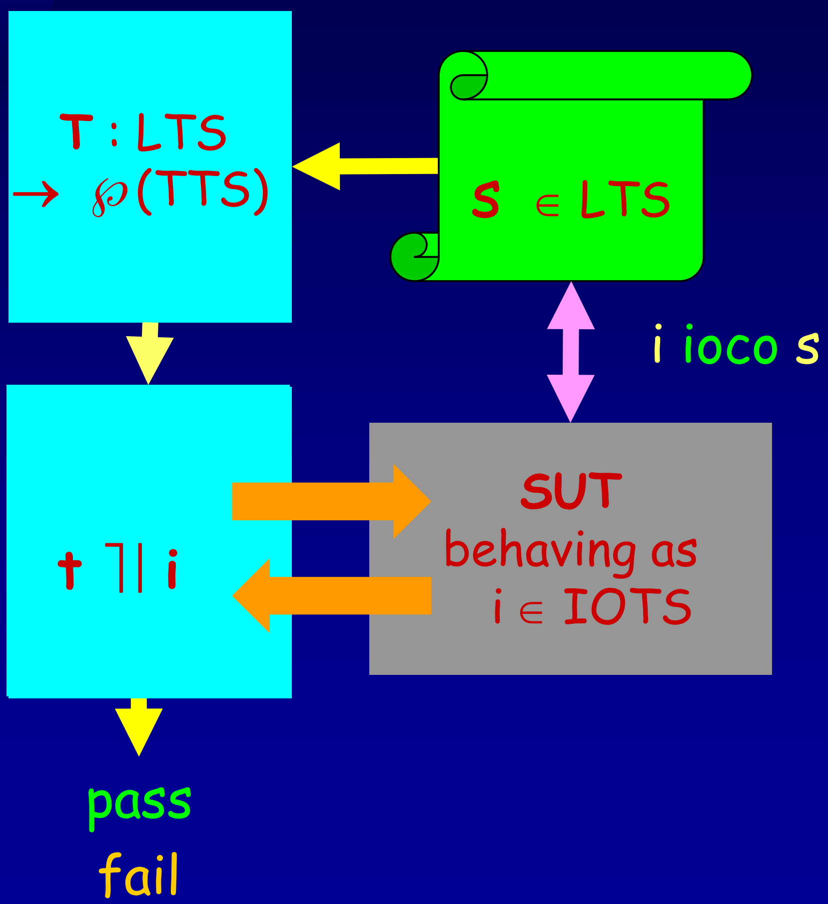


i ioco s

exhaustive ⇑ ⇓ *sound*

T(s) ⌉| i → pass

The ioco Theory for Model-Based Testing



Test assumption :

$\forall IUT \in IMP \,.\, \exists i_{IUT} \in IOTS \,.$

$\forall t \in TEST \,.\, IUT$ **passes** t

$\Leftrightarrow\ i_{IUT}$ passes t

Proof soundness and exhaustiveness:

$\forall i \in IOTS \,.$

$(\ \forall t \in T(s) \,.\ i$ **passes** $t\ )$

$\Leftrightarrow\quad i$ **ioco** s

SUT ioco s

exhaustive $\Uparrow\quad\Downarrow$ *sound*

SUT ⌉| T(s) → pass