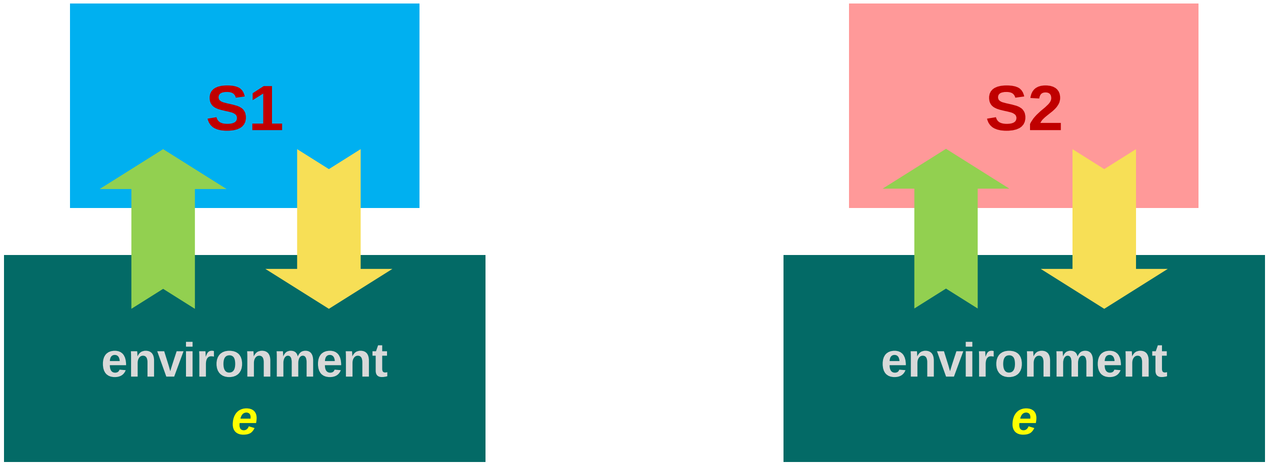
A Theory of Model-Based Testing with Labelled Transition Systems

Various Topics

Testability Assumption

Comparing Transition Systems



$$S1 \approx S2 \quad \Leftrightarrow \quad \forall\, e \in E.\ \ obs\,(e, S1) = obs\,(e, S2)$$

↓ ↓

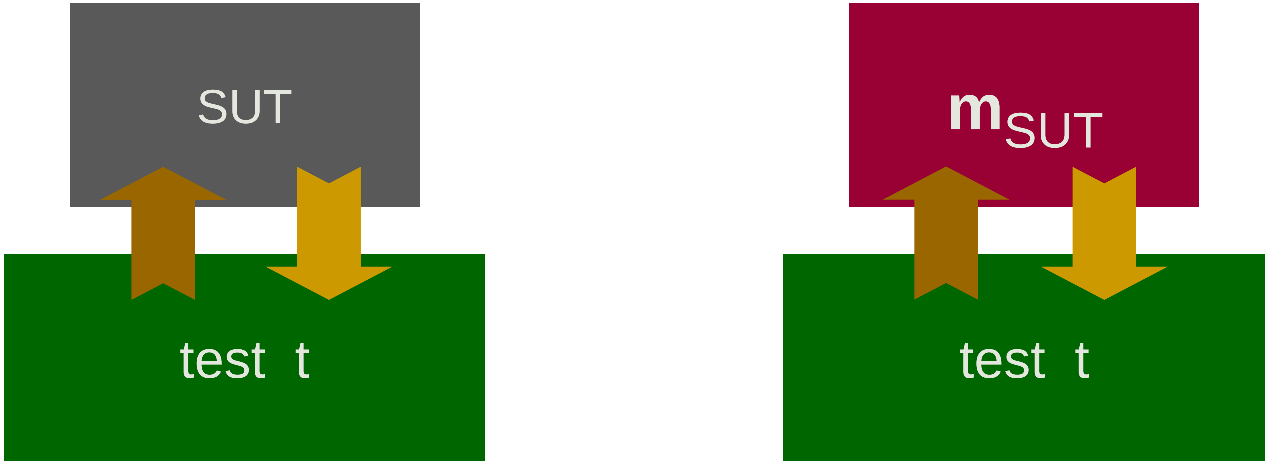
? ?

MBT: Testability Assumption

Testability assumption :

$\forall$ **SUT** . $\exists$ $\mathbf{m}_{\mathbf{SUT}}$ $\in$ **IOTS** .

$\forall$ **t** $\in$ **TEST** . **SUT passes t** $\Leftrightarrow$ $\mathbf{m}_{\mathbf{SUT}}$ **passes t**



MBT : Completeness

$$\text{SUT } \textbf{passes } T_s \overset{?}{\Leftrightarrow} \text{SUT } \textbf{conforms to } s$$

SUT **passes** T_s

$\Leftrightarrow$ SUT **passes** T_s $\Leftrightarrow_{def}$ $\forall t \in T_s$. SUT **passes** t

$\forall t \in T_s$. SUT **passes** t

$\Leftrightarrow$ *testability assumption*: $\forall t \in TEST$. SUT **passes** t $\Leftrightarrow$ m_{SUT} **passes** t

$\forall t \in T_s$. m_{SUT} **passes** t

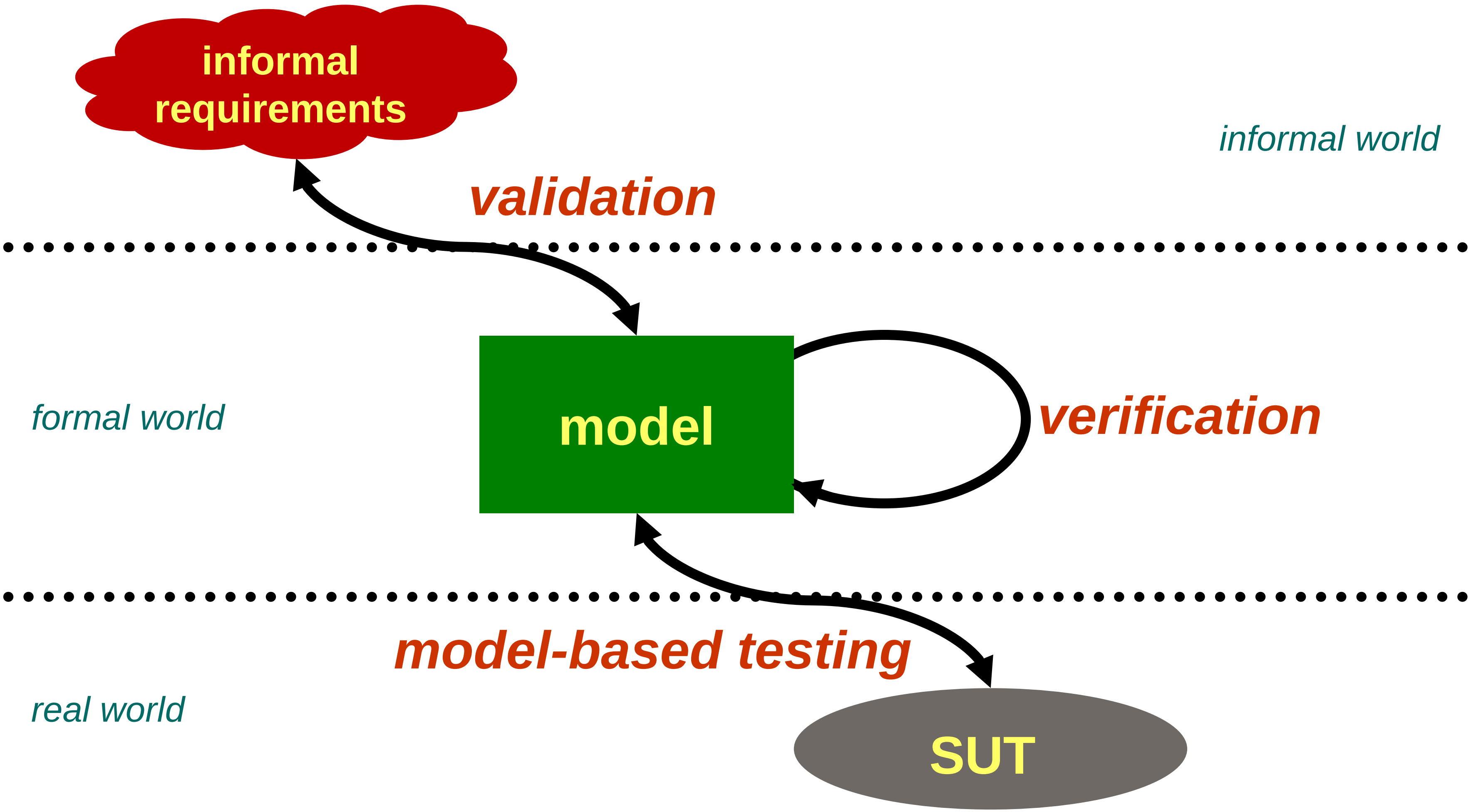
$\Leftrightarrow$ *prove*: $\forall m \in MOD$. ($\forall t \in T_s$. m **passes** t) $\Leftrightarrow$ m **uioco** s

m_{SUT} **uioco** s

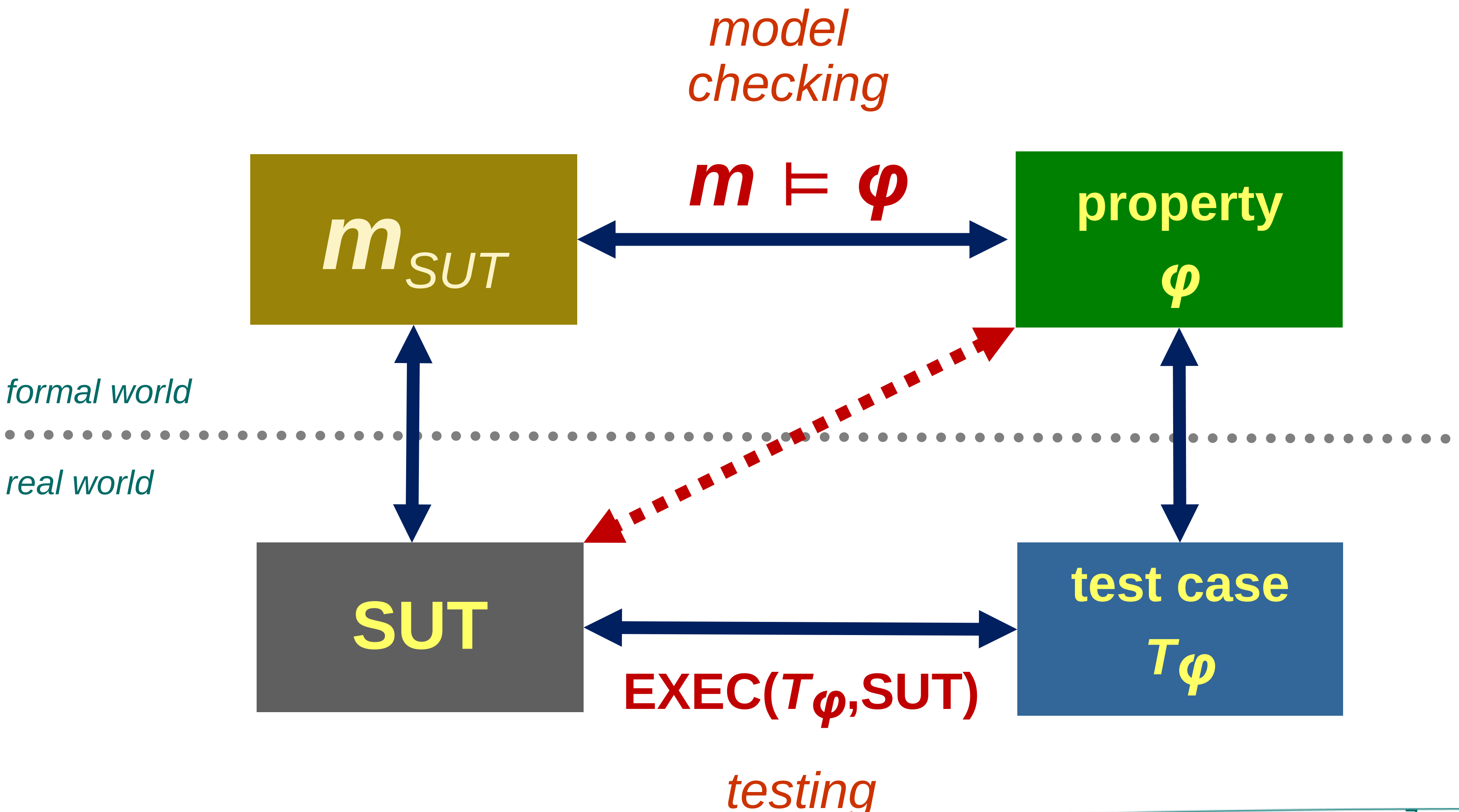
$\Leftrightarrow$ *define*: SUT **conforms to** s iff m_{SUT} **uioco** s

SUT **conforms to** s

Validation, Verification, Testing



Verification and Testing



Verification and Testing

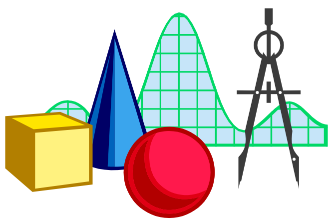
Model-based verification :

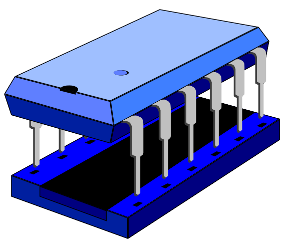
- formal manipulation
- prove properties
- performed on model

Model-based testing :

- experimentation
- show error
- concrete system

formal world



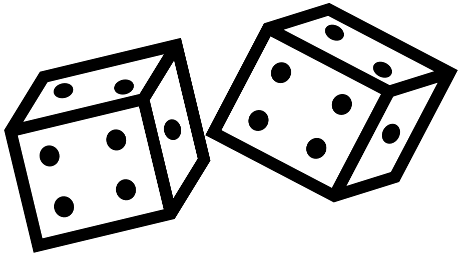


concrete world

Verification is only as good as the validity of the model on which it is based

Testing can only show the presence of errors, not their absence

Testability Assumption : Adder



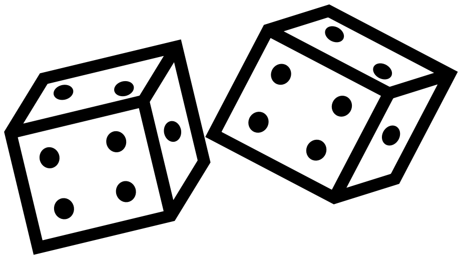
Test a function adding numbers of two dice:

int add (int x, y) for $x, y \in [1\ldots6]$

Is the following a complete test suite?

(1,1) (1,2) . . . . . (1,6)
(2,1) (2,2) . . . . . (2,6)
. . .
. . .
. . .
(6,1) (6,2) . . . . . (6,6)

Testability Assumption : Adder



Test a function adding numbers of two dice:

int add (int x, y) for $x, y \in [1\ldots6]$

The test suite

(1,1) (1,2) . . . . . . (1,6)
(2,1) (2,2) . . . . . . (2,6)
. . .
. . .
(6,1) (6,2) . . . . . . (6,6)

is sound & exhaustive if

- the testability assumption is that implementation

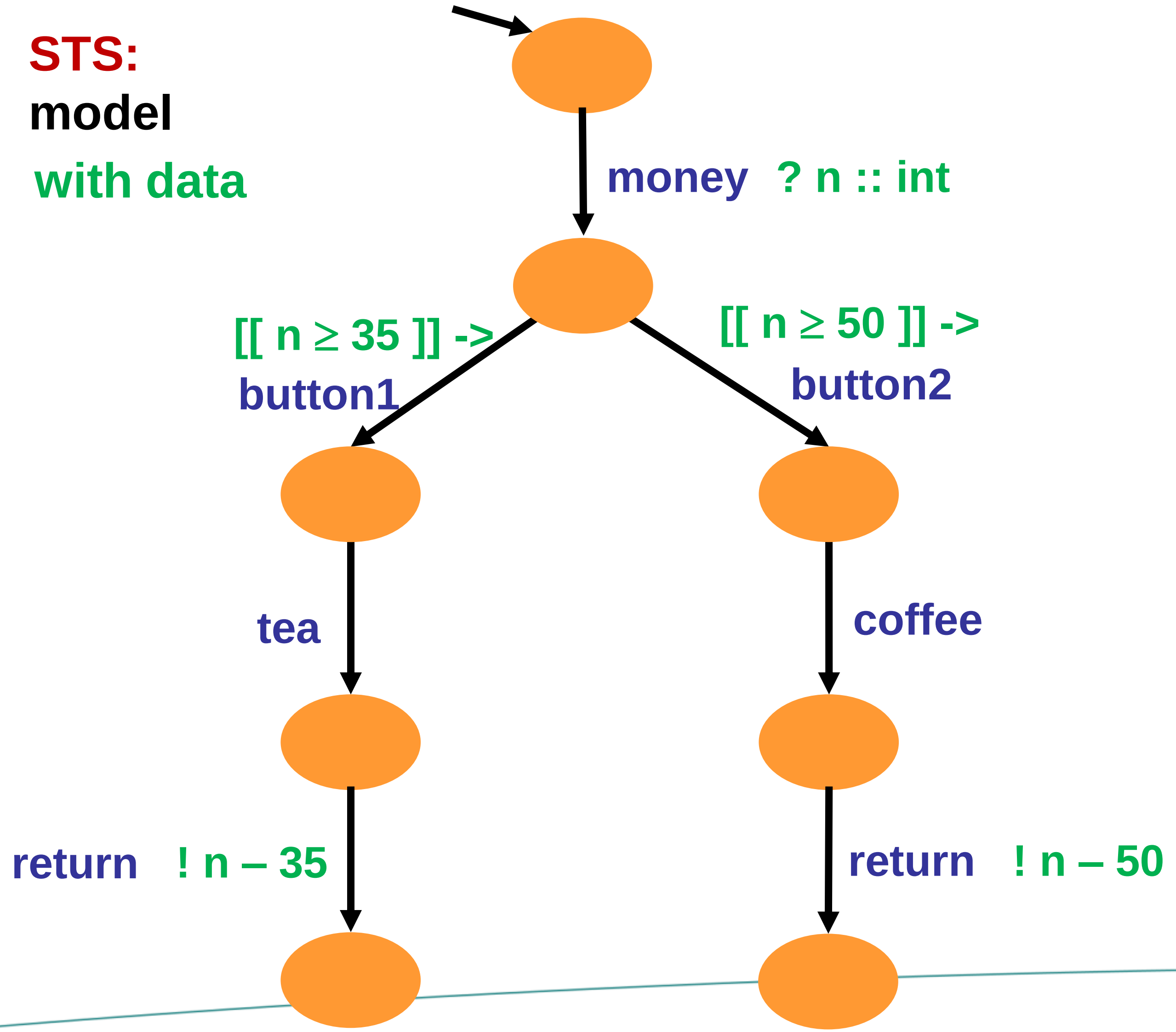
  can be modelled as functions : $\boldsymbol{i} :: [1..6] \times [1..6] \rightarrow \text{Int}$

Model-Based Testing with Data:

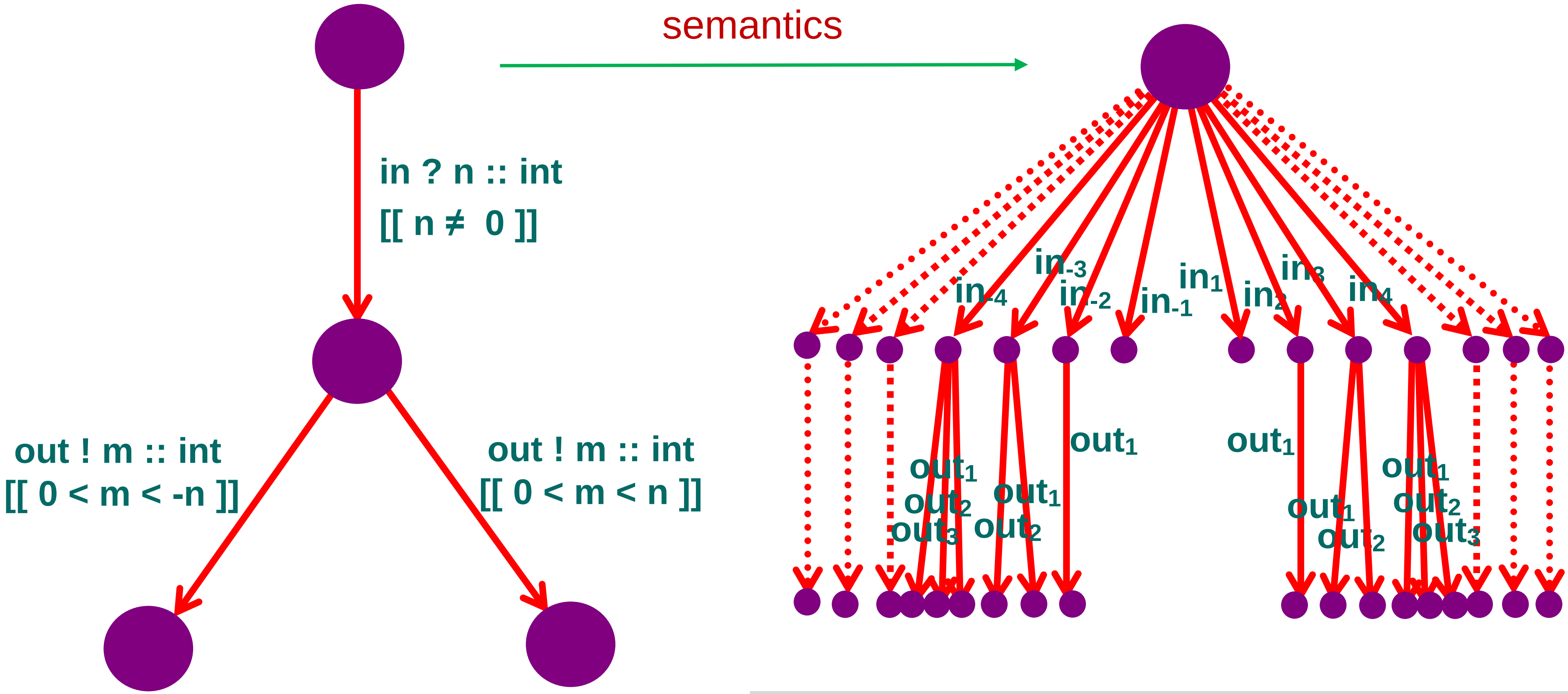
Symbolic Transition Systems

STS : Symbolic Transition Systems

STS:
model
with data



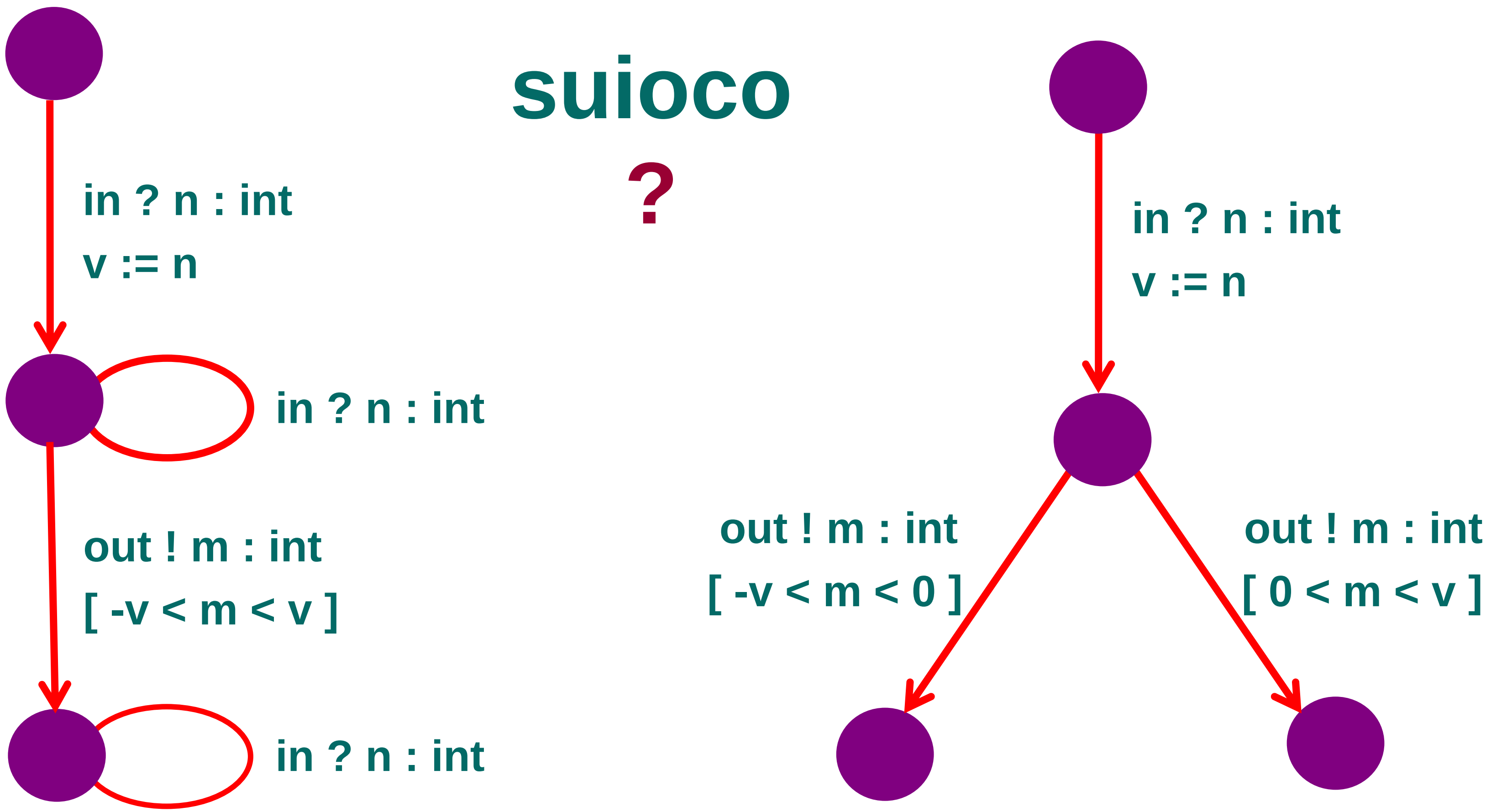
STS : Symbolic Transition Systems



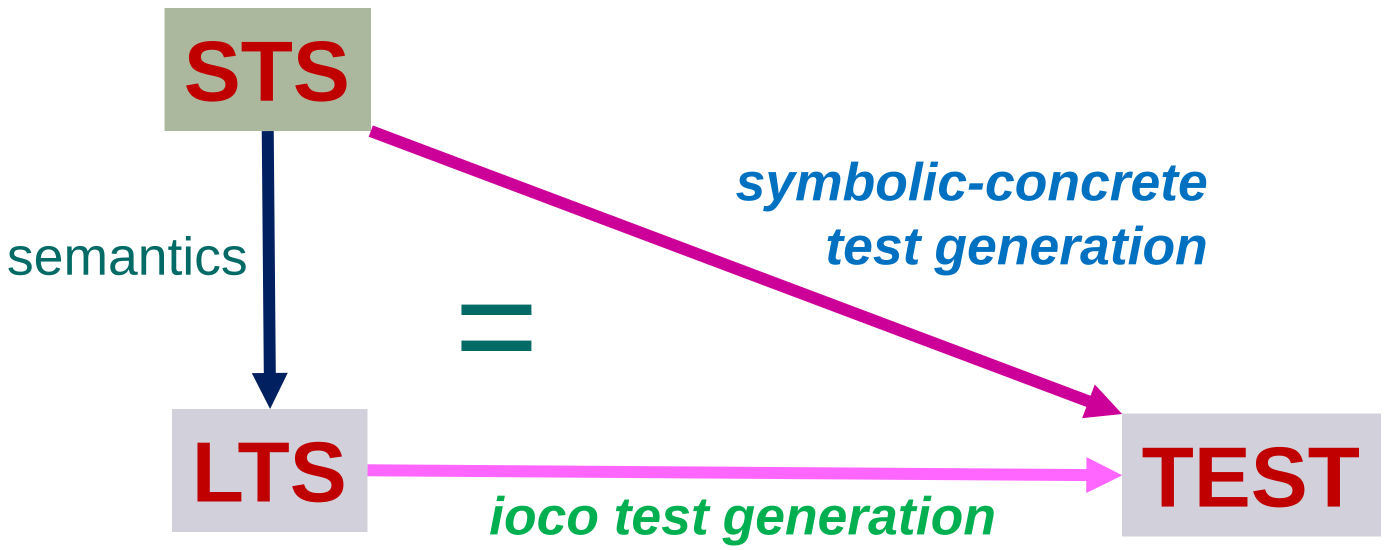
Disadvantages unfolded representation:

- infinity
- loss of information (e.g. for test selection)

suioco : symbolic uioco



TorXakis : Lift Test Generation



sioco : Symbolic ioco

Specification: IOSTS $\mathcal{S}(\iota_S) = \langle L_S, l_S, \mathcal{V}_S, \mathcal{I}, \Lambda, \rightarrow_S\rangle$
Implementation: IOSTS $\mathcal{P}(\iota_P) = \langle L_P, l_P, \mathcal{V}_P, \mathcal{I}, \Lambda, \rightarrow_P\rangle$
both initialised, implementation input-enabled, $\mathcal{V}_S \cap \mathcal{V}_P = \emptyset$
$\mathcal{F}_s$: a set of symbolic extended traces satisfying $[\![\mathcal{F}_s]\!]_{\iota_S} \subseteq \mathit{Straces}((l_0, \iota))$;

$\mathcal{P}(\iota_P)$ $\mathbf{sioco}_{\mathcal{F}_s}$ $\mathcal{S}(\iota_S)$ iff

$$\forall(\sigma,\chi)\in\mathcal{F}_s\ \forall\lambda_\delta\in\Lambda_U\cup\{\delta\} : \iota_P\cup\iota_S \models \overline{\forall}_{\widehat{\mathcal{I}}\cup\mathcal{I}}\big(\Phi(l_P,\lambda_\delta,\sigma)\wedge\chi\rightarrow\Phi(l_S,\lambda_\delta,\sigma)\big)$$

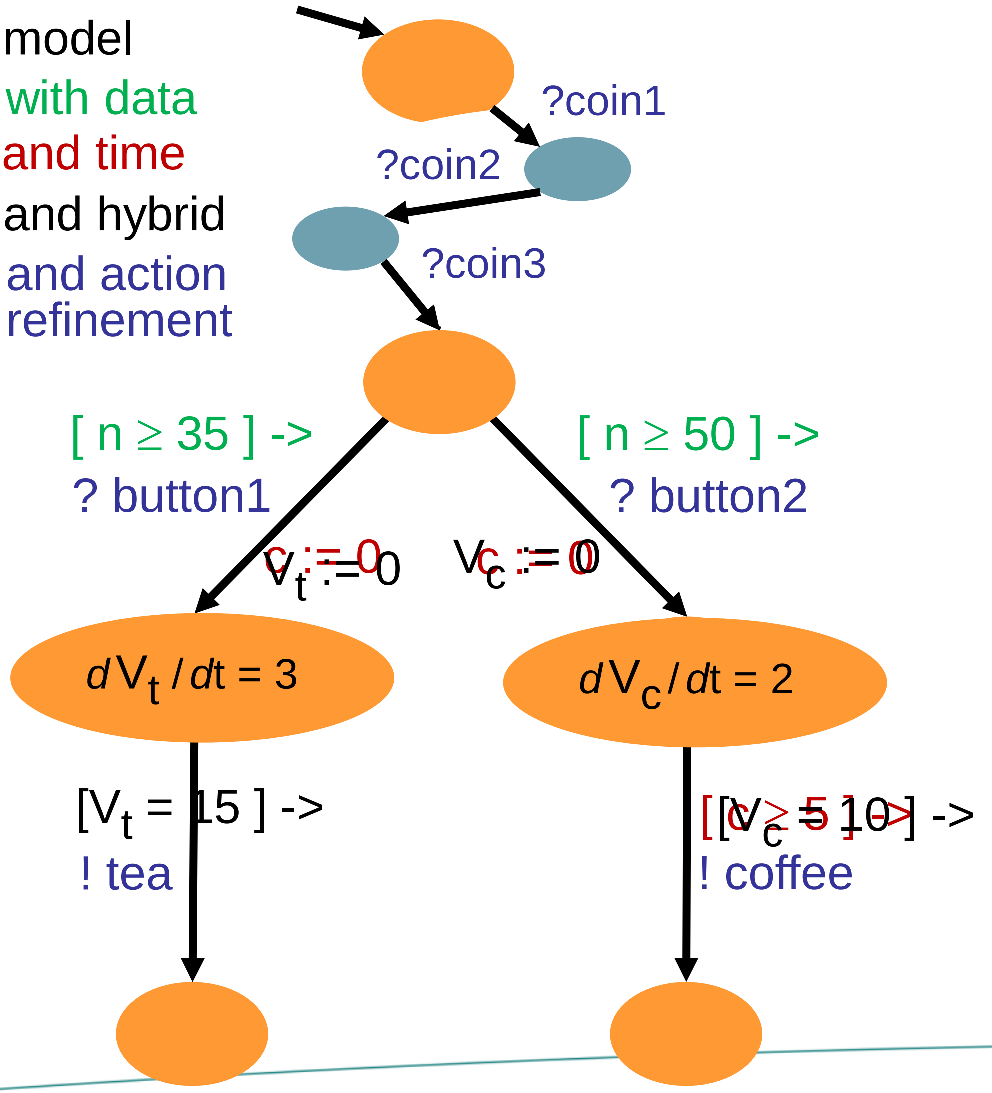
$$\text{where } \Phi(\xi,\lambda_\delta,\sigma) = \bigvee\{\varphi\wedge\psi \mid (\lambda_\delta,\ \varphi,\ \psi)\in\mathbf{out}_s((\xi,\top,\mathsf{id})_0\,\mathbf{after}_s(\sigma,\top))\}$$

Theorem 1.

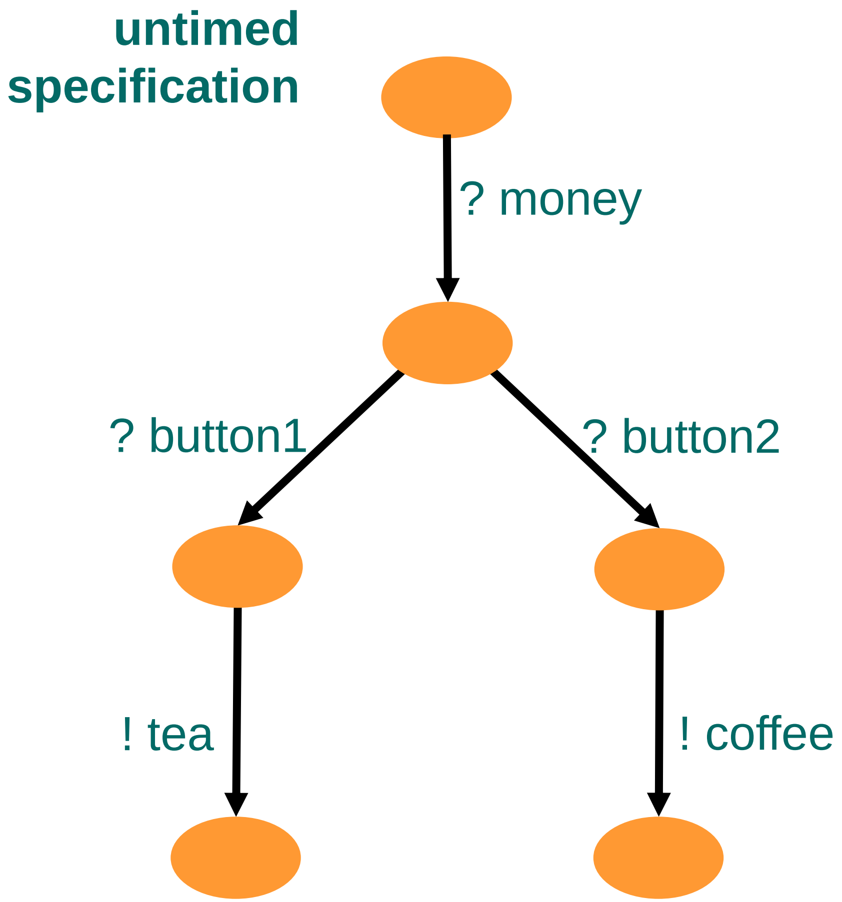
$$\mathcal{P}(\iota_P)\ \mathbf{sioco}_{\mathcal{F}_s}\ \mathcal{S}(\iota_S) \quad \mathit{iff} \quad [\![\mathcal{P}]\!]_{\iota_P}\ \mathbf{ioco}_{[\![\mathcal{F}_s]\!]_{\iota_S}}\ [\![\mathcal{S}]\!]_{\iota_S}$$

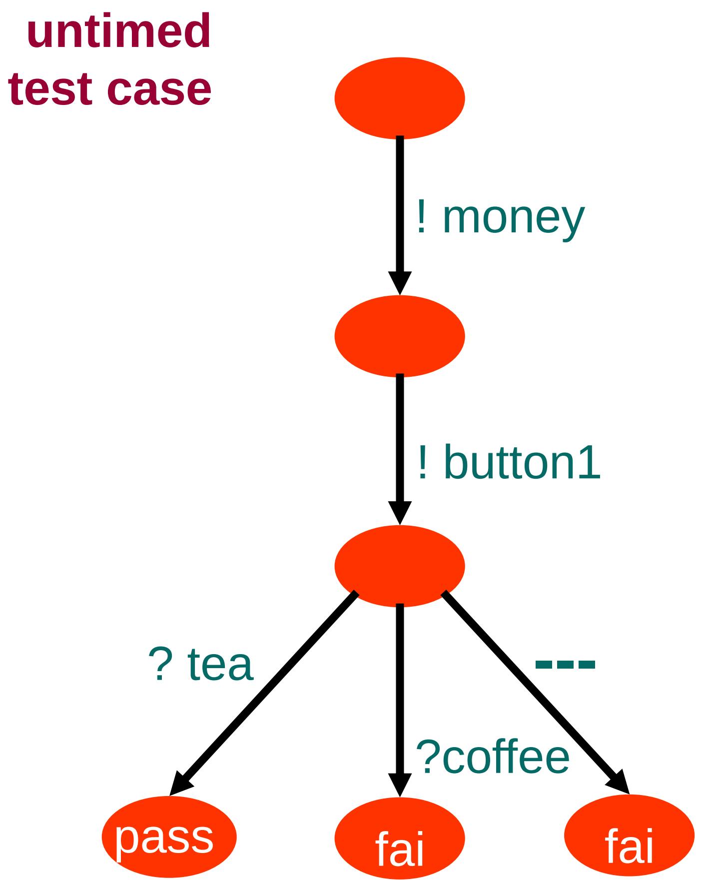
Transition Systems : Other Extensions

model
with data
and time
and hybrid
and action
refinement

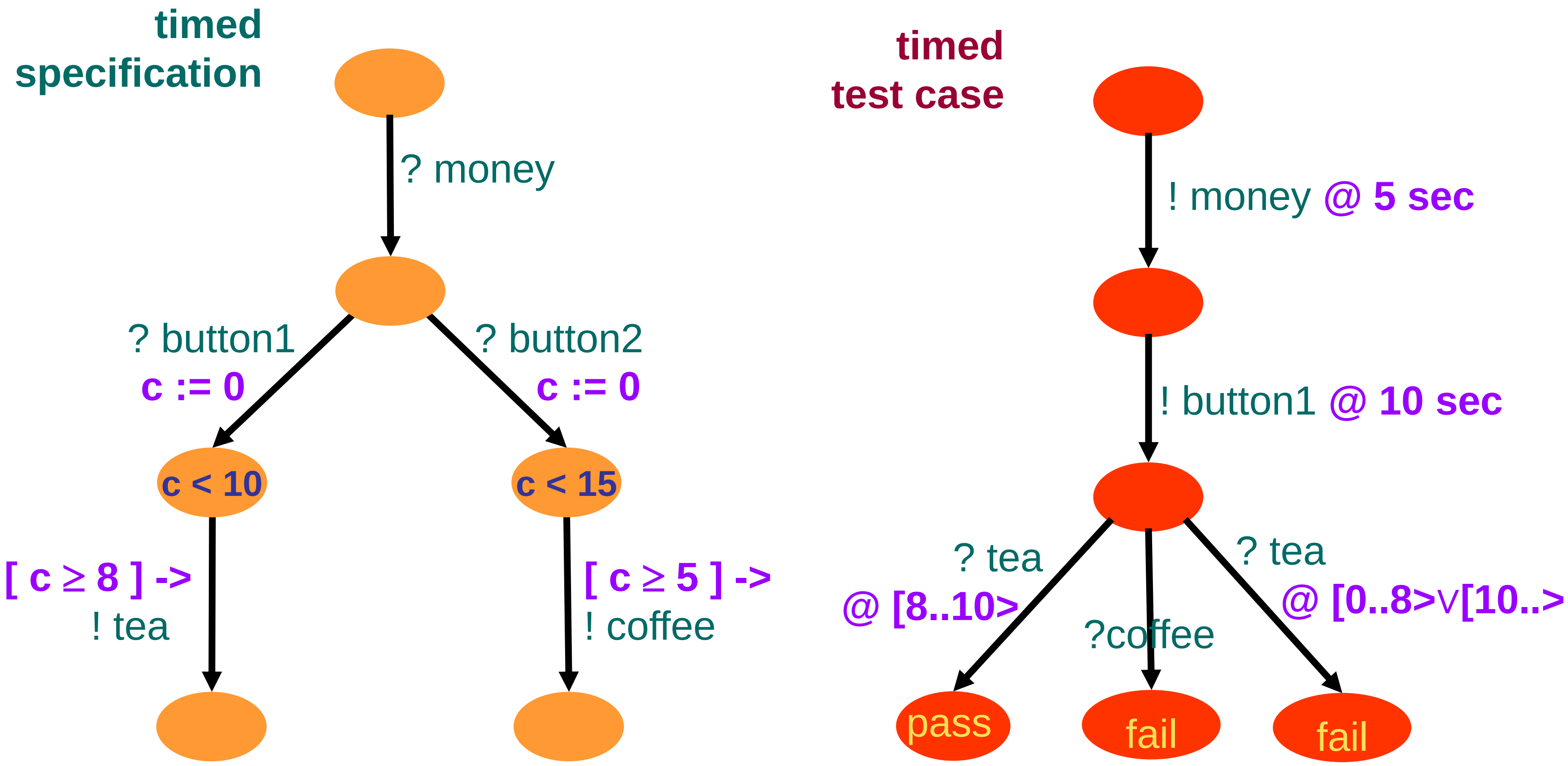


Real-Time MBT





Real-Time MBT



uioco *variations*

Variations on a Theme

- i **ioco** $s \;\Leftrightarrow\; \forall \sigma \in Straces(s) : out\,(\,i \text{ after } \sigma) \subseteq out\,(\,s \text{ after } \sigma)$
- $i \leq_{ior} s \;\Leftrightarrow\; \forall \sigma \in (\,L \cup \{\delta\}\,)^* : out\,(\,i \text{ after } \sigma) \subseteq out\,(\,s \text{ after } \sigma)$
- i **ioconf** $s \;\Leftrightarrow\; \forall \sigma \in traces(s) : out\,(\,i \text{ after } \sigma) \subseteq out\,(\,s \text{ after } \sigma)$
- $i\ ioco_F\ s \;\Leftrightarrow\; \forall \sigma \in F : out\,(\,i \text{ after } \sigma) \subseteq out\,(\,s \text{ after } \sigma)$
- i **uioco** $s \;\Leftrightarrow\; \forall \sigma \in Utraces(s) : out\,(\,i \text{ after } \sigma) \subseteq out\,(\,s \text{ after } \sigma)$
- **i** mioco **s** multi-channel ioco
- **i** wioco **s** non-input-enabled ioco
- **i** eco **e** environmental conformance
- **i** sioco **s** symbolic ioco
- **i** suioco **s** symbolic uioco
- **i** (r)tioco **s** (real) timed tioco (Aalborg, Twente, Grenoble, Bordeaux,.....)
- **i** $ioco_r$ **s** refinement ioco
- **i** dioco **s** distributed ioco
- **i** hioco **s** hybrid ioco
- **i** qioco **s** quantified ioco
- **i** poco **s** partially observable game ioco
- **i** $stioco_D$ **s** real time and symbolic data . . . . . .

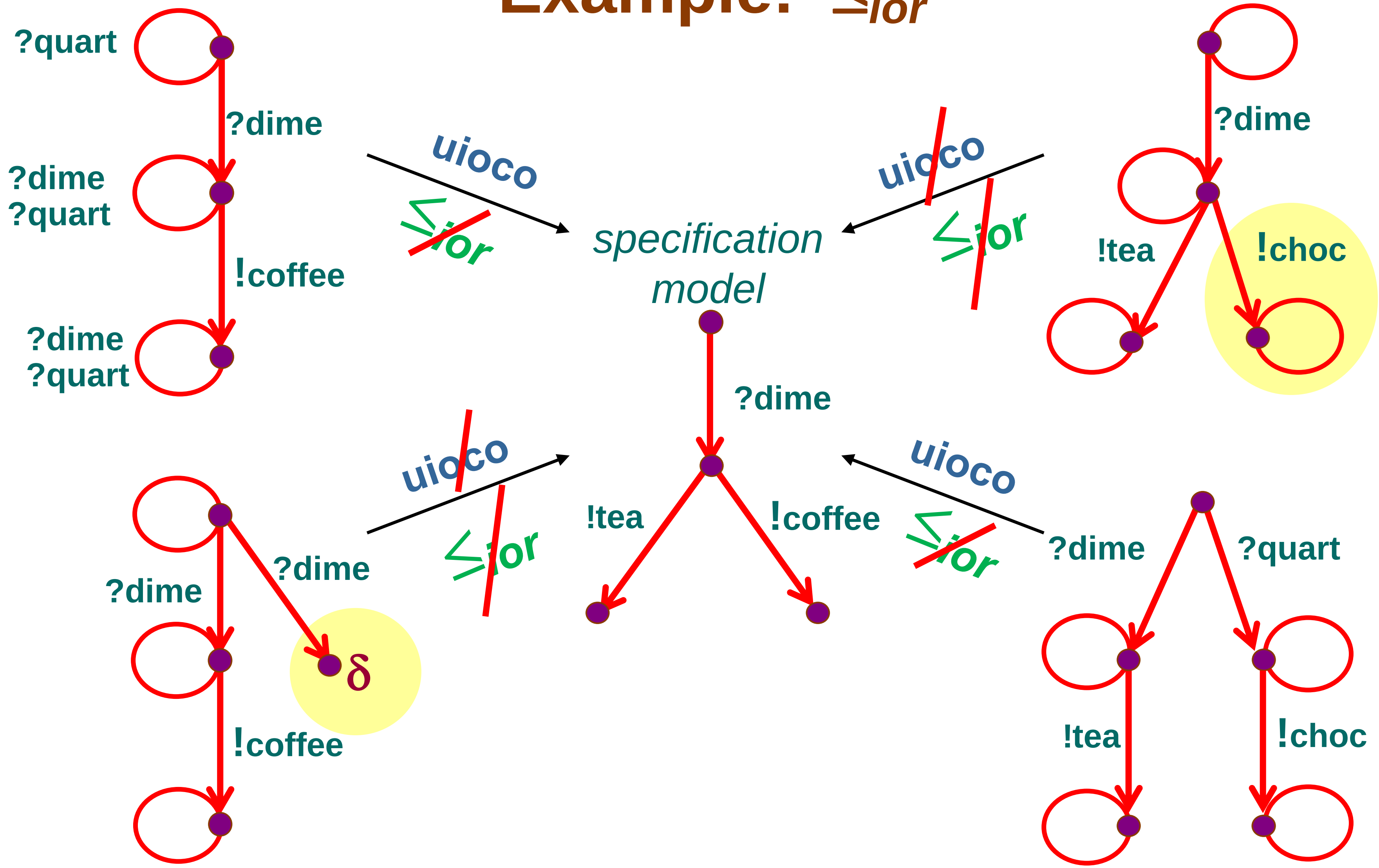
uioco *variations*

ioco$_F$ *: Varying Trace Sets*

Variations on a Theme

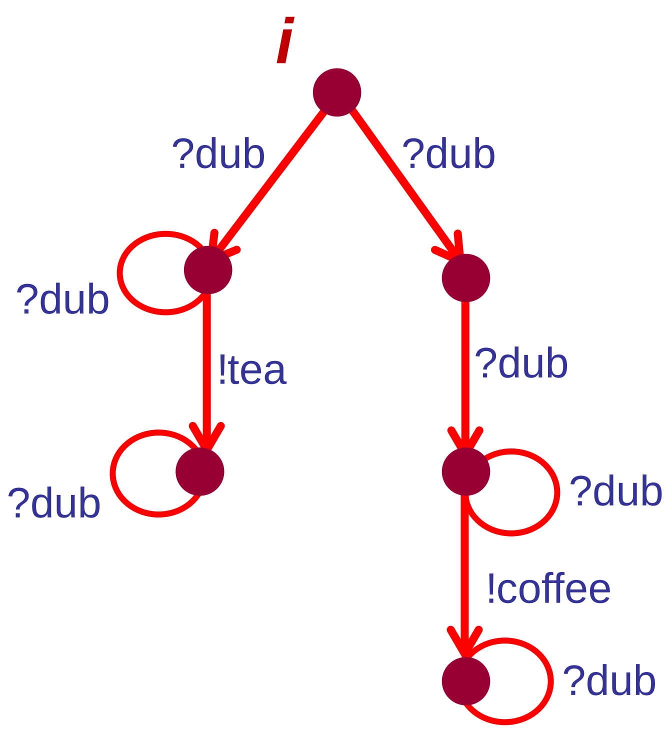
- **i** ioco **s** $\Leftrightarrow$ $\forall \sigma \in \mathrm{Straces}(s)$: out (i after σ) $\subseteq$ out (s after σ)
- **i** $\leq_{ior}$ **s** $\Leftrightarrow$ $\forall \sigma \in (L \cup \{\delta\})^*$: out (i after σ) $\subseteq$ out (s after σ)
- **i** ioconf **s** $\Leftrightarrow$ $\forall \sigma \in \mathrm{traces}(s)$: out (i after σ) $\subseteq$ out (s after σ)
- **i** ioco_F **s** $\Leftrightarrow$ $\forall \sigma \in F$: out (i after σ) $\subseteq$ out (s after σ)
- **i** uioco **s** $\Leftrightarrow$ $\forall \sigma \in \mathrm{Utraces}(s)$: out (i after σ) $\subseteq$ out (s after σ)
- **i** mioco **s** multi-channel ioco
- **i** wioco **s** non-input-enabled ioco
- **i** eco **e** environmental conformance
- **i** sioco **s** symbolic ioco
- **i** suioco **s** symbolic uioco
- **i** (r)tioco **s** (real) timed tioco (Aalborg, Twente, Grenoble, Bordeaux,.....)
- **i** ioco_r **s** refinement ioco
- **i** dioco **s** distributed ioco
- **i** hioco **s** hybrid ioco
- **i** qioco **s** quantified ioco
- **i** poco **s** partially observable game ioco
- **i** stioco_D **s** real time and symbolic data . . . . . .

Example: $\leq_{ior}$



Example: (*u*)*ioco*

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

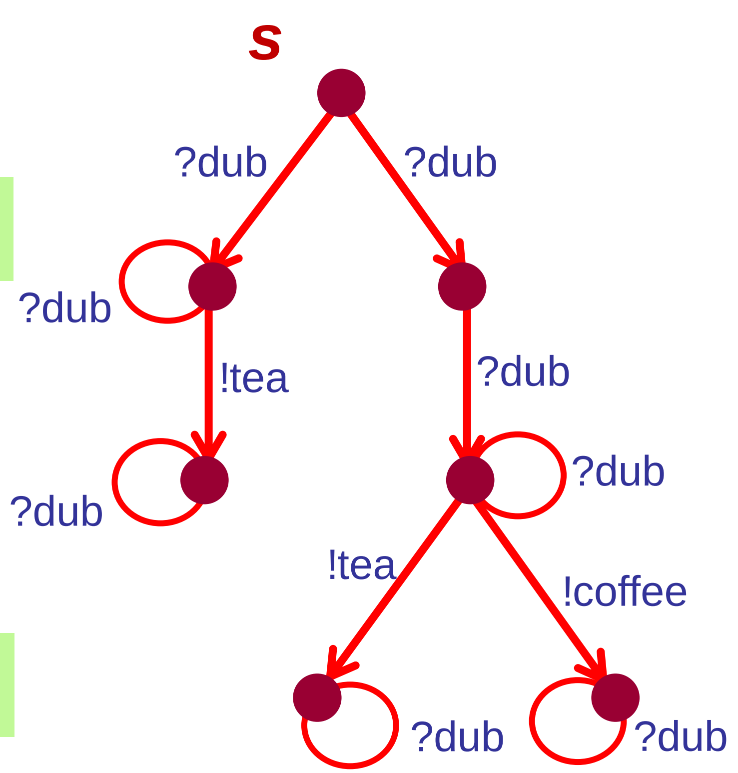


i **uioco** *s*

s **~~uioco~~** *i*

i **ioconf** *s*

s **ioconf** *i*

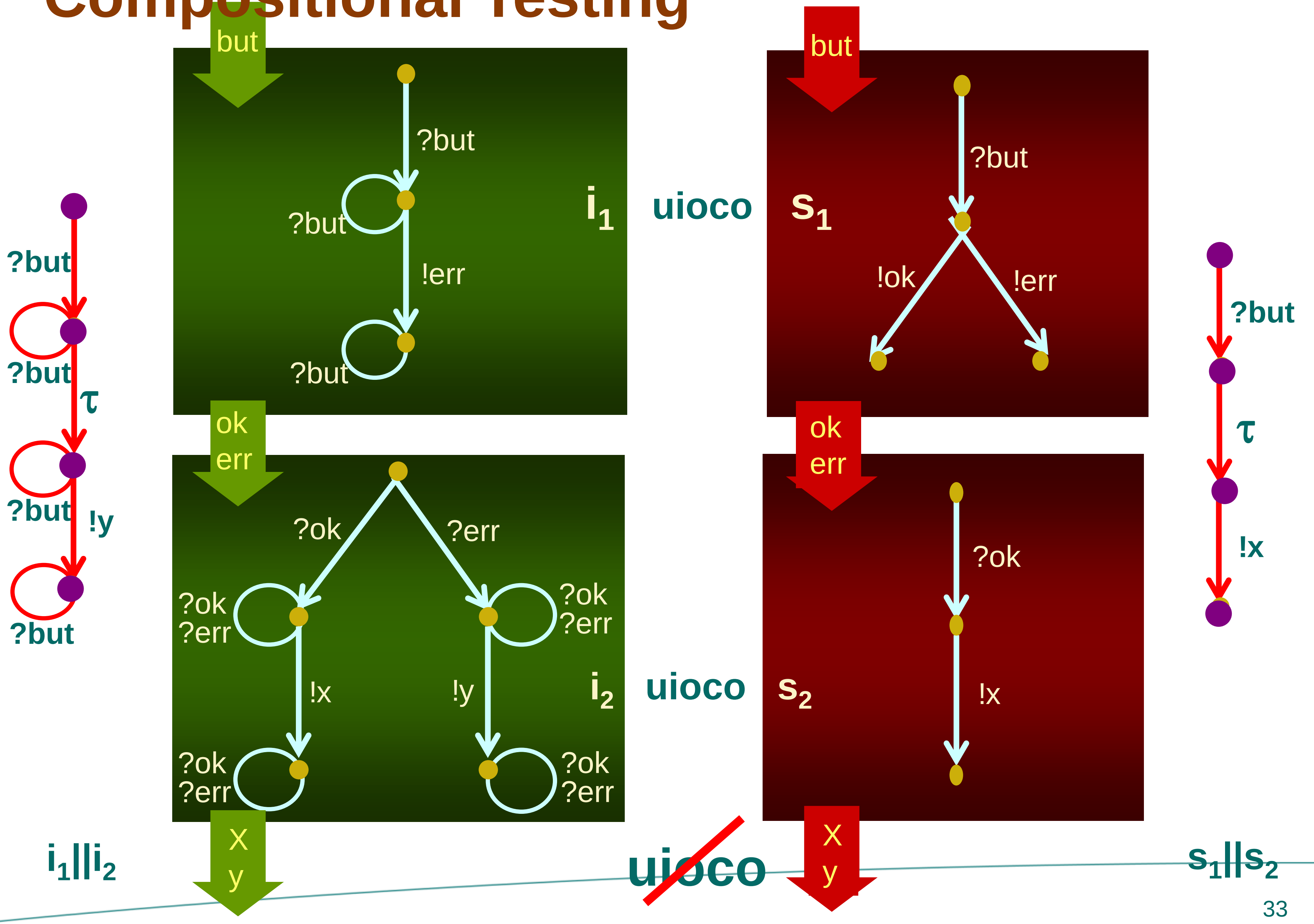


out (***i*** **after** ?dub.?dub) = *out* (***s*** **after** ?dub.?dub) = { !tea, !coffee }

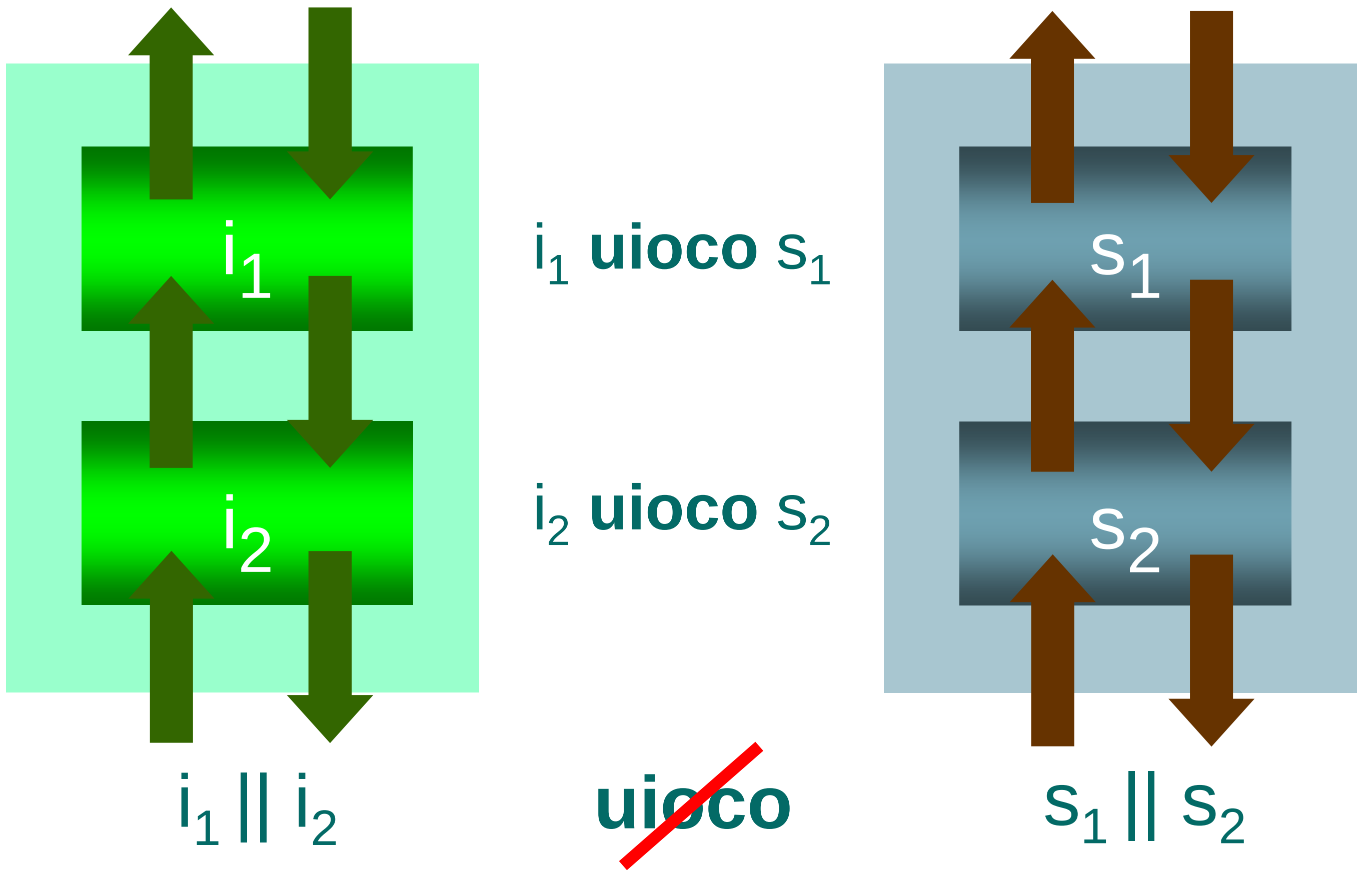
out (***i*** **after** ?dub.δ.?dub) = { !coffee } $\neq$ *out* (***s*** **after** ?dub.δ.?dub) = { !tea, !coffee }

Compositionality

Compositional Testing

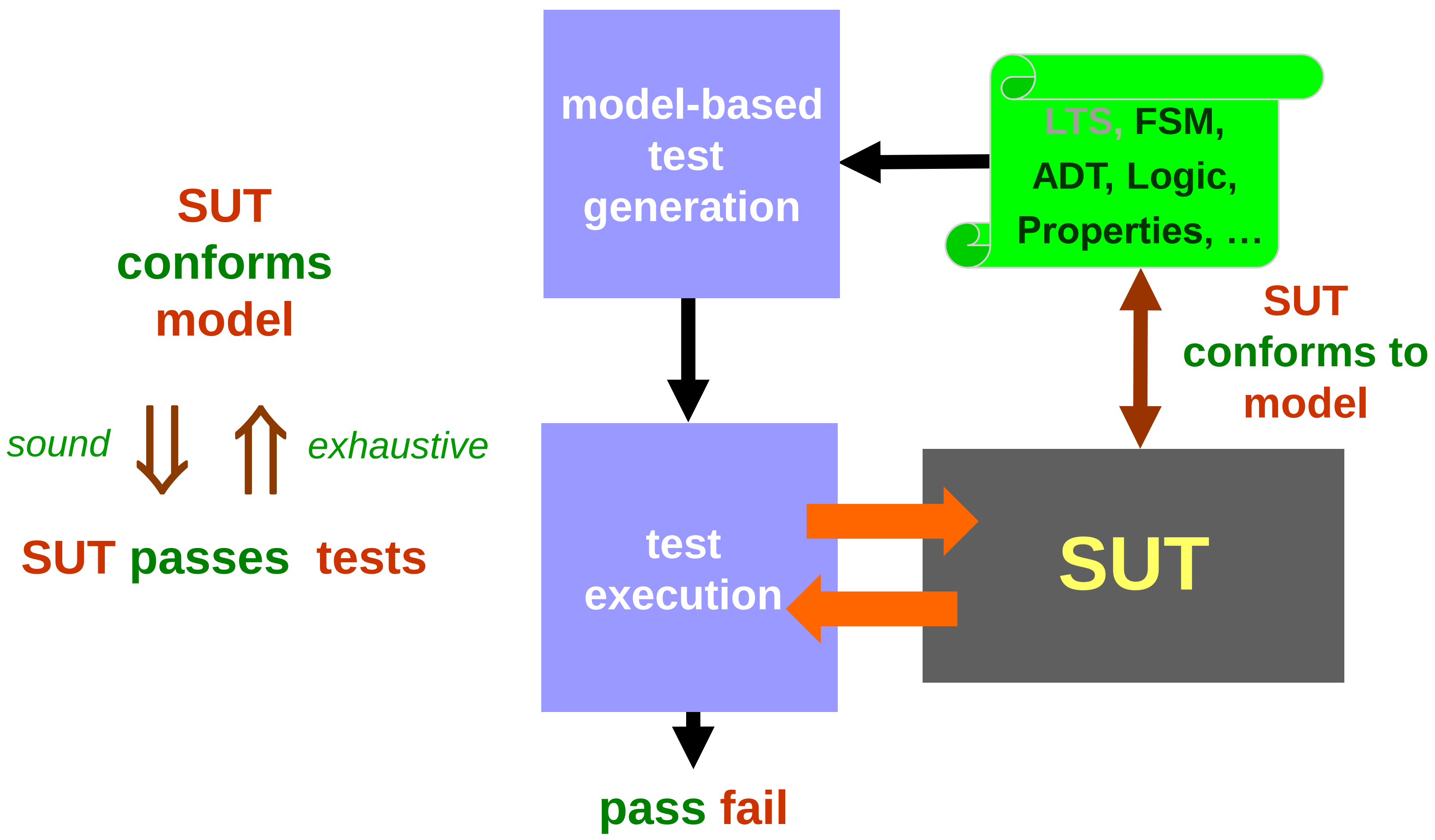


Compositional Testing

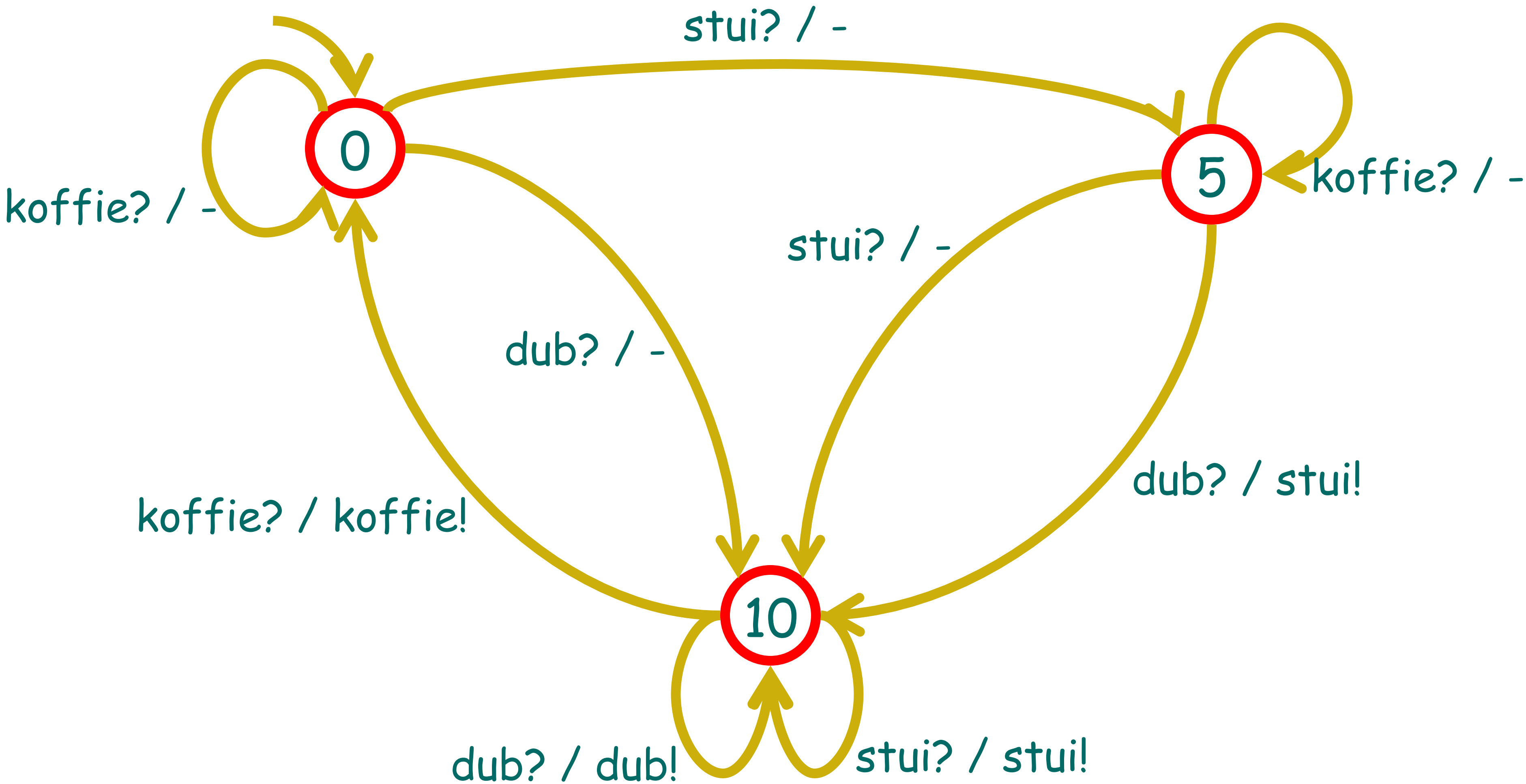


If s_1, s_2 input enabled - $s_1, s_2 \in$ IOTS - then **ioco** is preserved !

MBT : Model-Based Testing

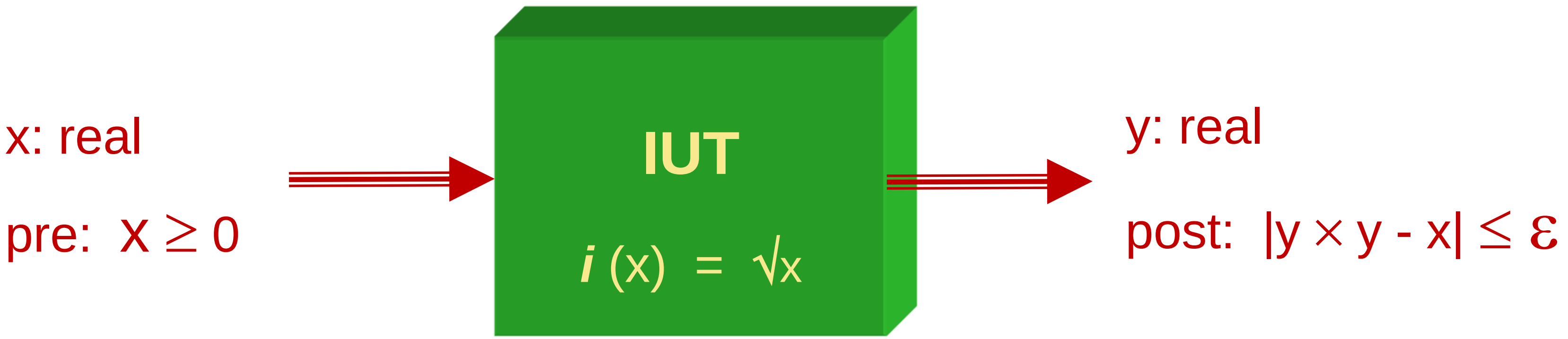


MBT : Finite State Machines (FSM)



36

MBT : Property-Based Testing



- Specification: property over x and y
  - property(x,y) = $x \geq 0 \implies |y \times y - x| \leq \varepsilon$
- Implementation is function $\boldsymbol{i} :: X \to Y$
- Test set $T \subseteq X$
  - Tools like G∀ST and QuickCheck generate thousands of tests by systematic traversal of all values of type X
  - But still: what is a "good" set ?