Online testing of real-time systems

Brian Nielsen
bnielsen@cs.aau.dk

With

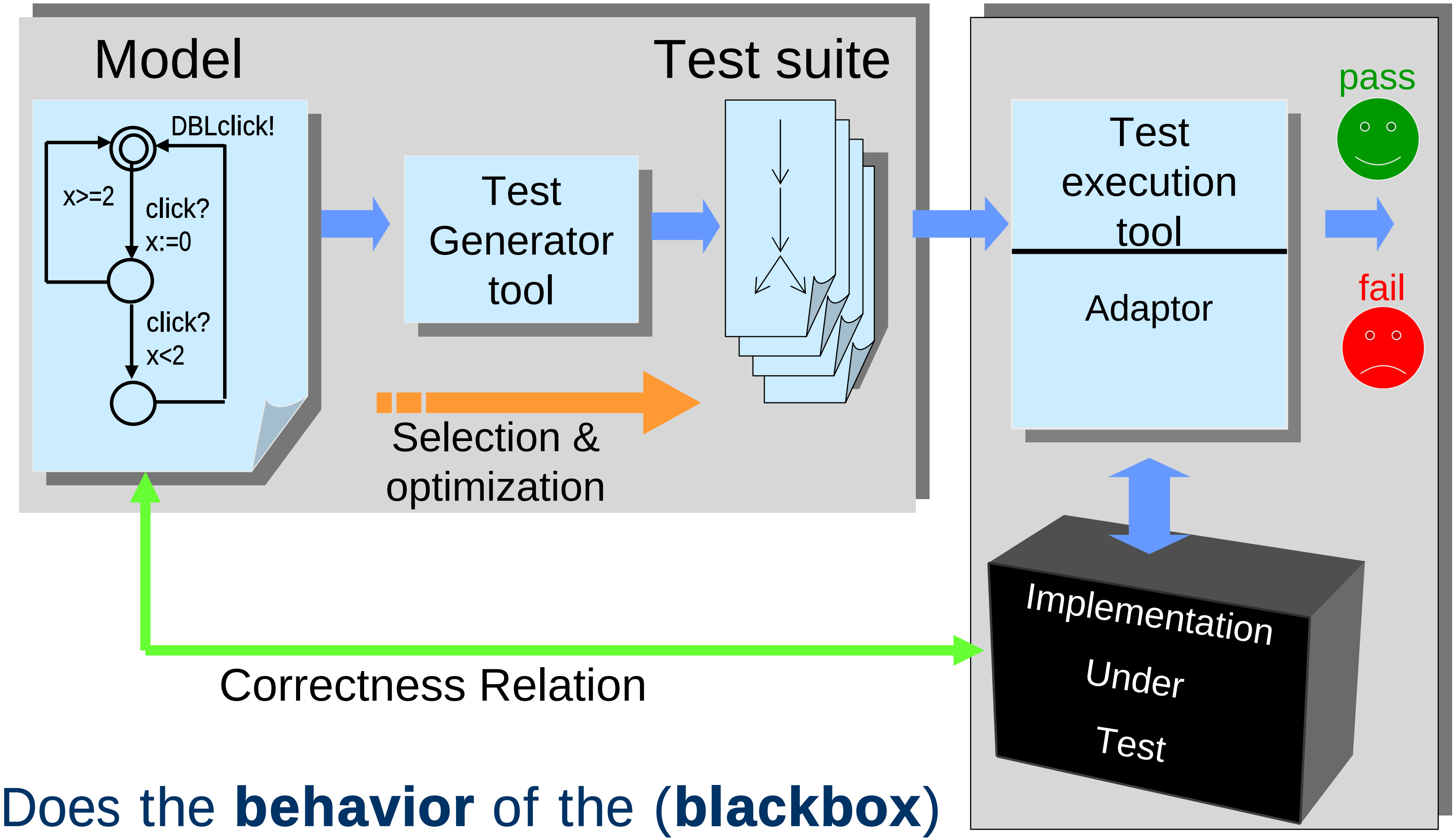
Kim Larsen, Marius Mikucionis, Arne Skou

AALBORG UNIVERSITY DENMARK

BRICS
Basic Research in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Automated Model Based Conformance Testing



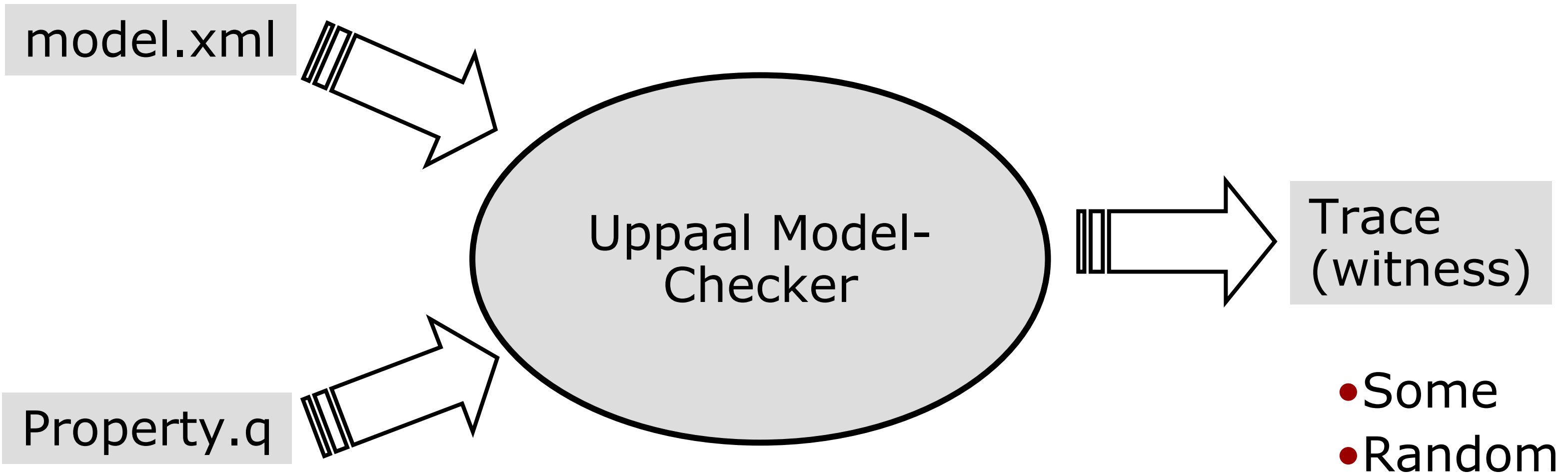
Does the **behavior** of the (**blackbox**) implementation ***comply*** to that of the specification?

Offline Test Generation Using Uppaal

Basic Research in Computer Science

CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

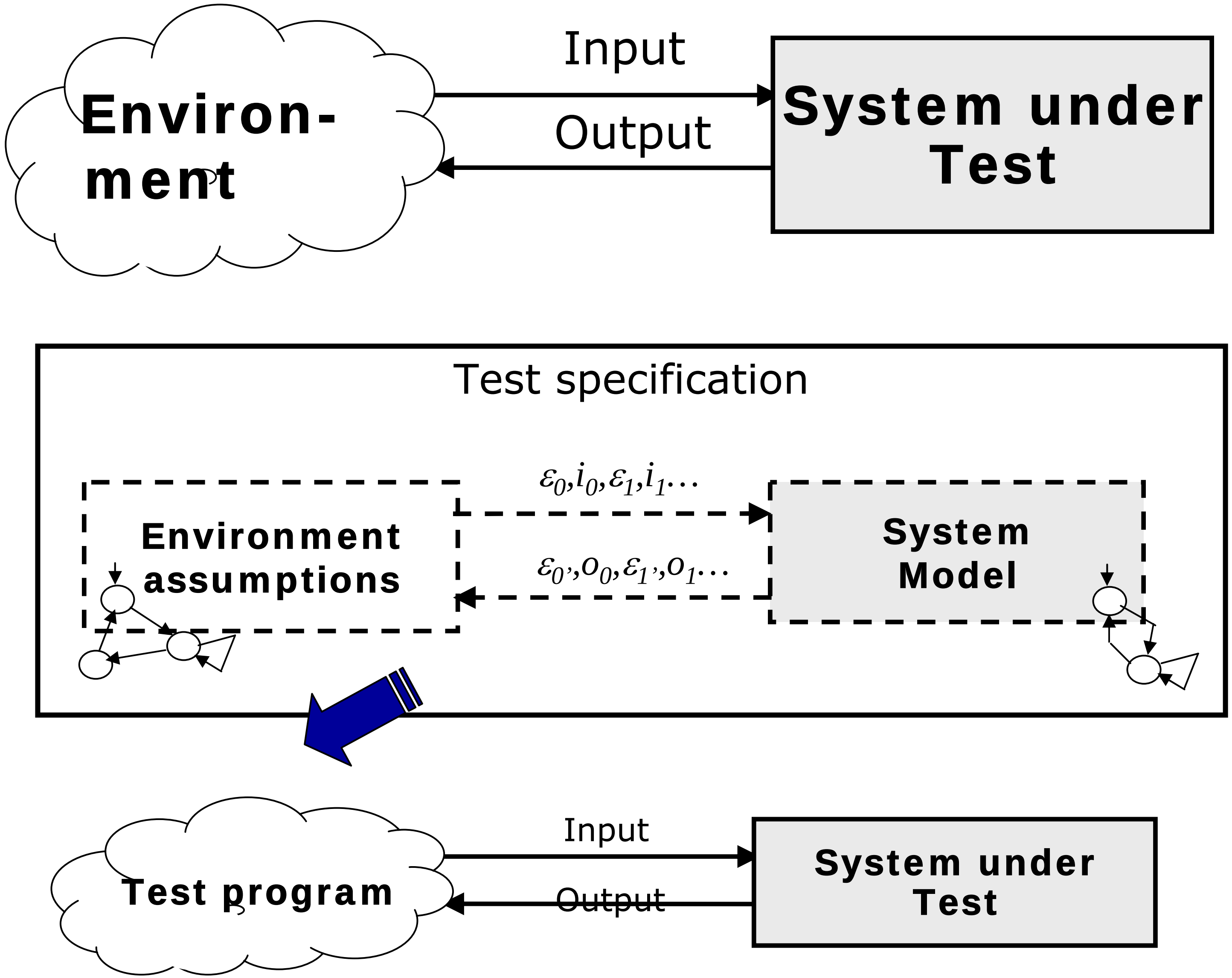
Test generation using model-checking



- Some
- Random
- Shortest
- Cheapest

- Use trace scenario as test case??!!

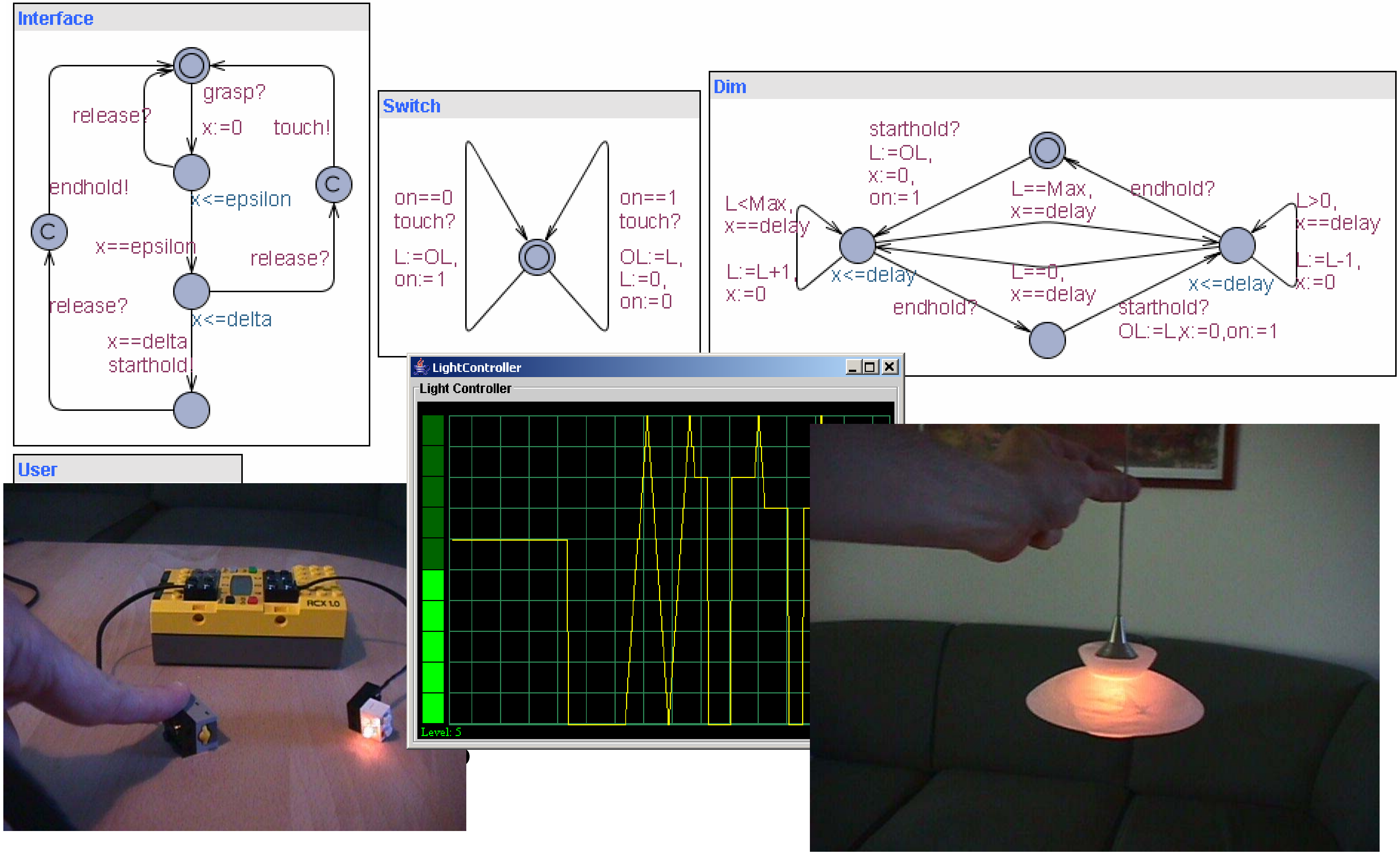
Model based Testing



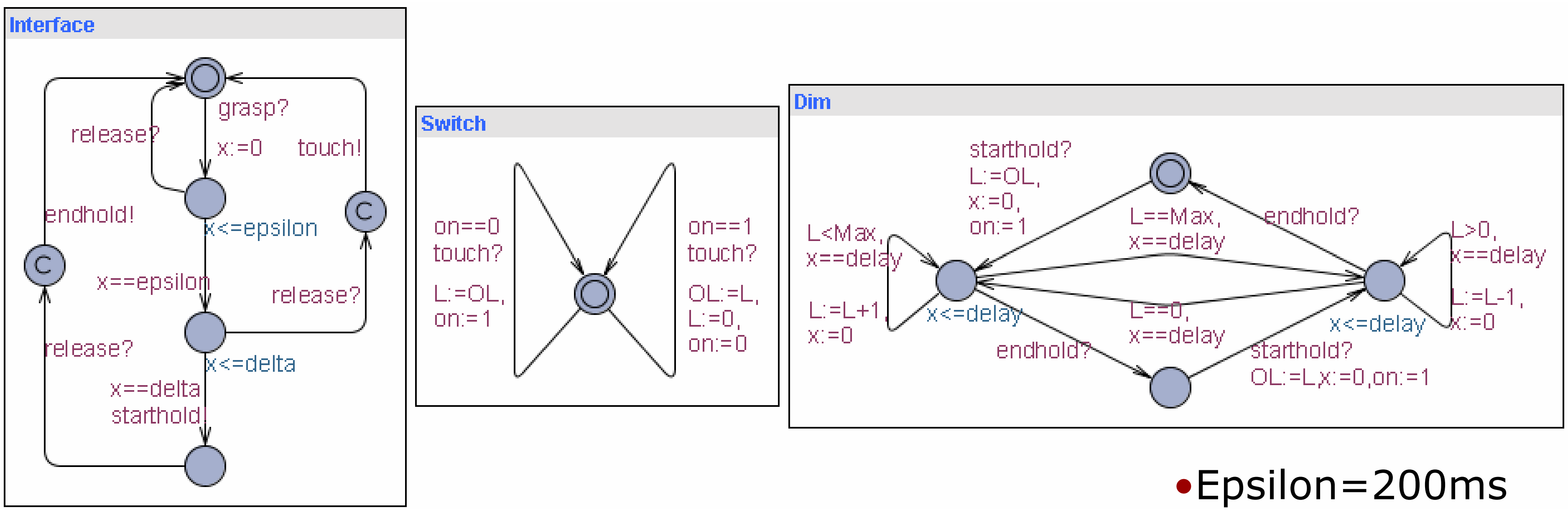
Controllable Timed Automata

- **Input Enabled**: all inputs can always be accepted
- **Output Urgent**: enabled outputs will occur immediately
- **Determinism**: two transitions with same input/output leads to the same state
- **Isolated Outputs**: if an output is enabled, no other output is enabled

Touch-sensitive Light-Controller



Timed Tests



- Epsilon=200ms
- Delta=500ms

EXAMPLE test cases for **Interface**

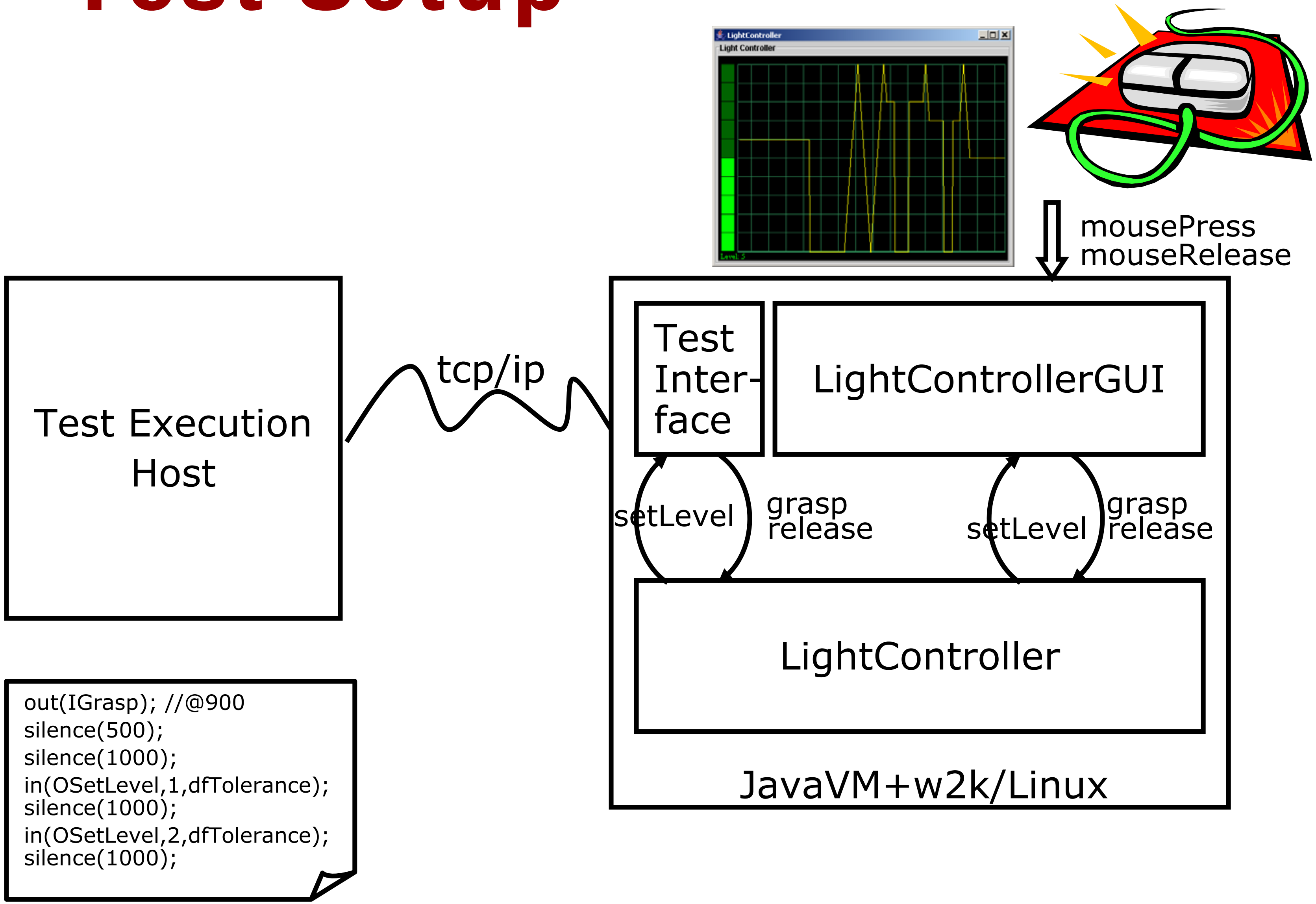
0·grasp!·210·release!·touch?.PASS

0·grasp!·317·release!·touch?·2½·grasp!·220·release!·touch?·PASS

1000·grasp!·517·starthold?·100·release!·endhold?·PASS

INFINITELY MANY SEQUENCES!!!!!!

Test Setup


mousePress
mouseRelease
Test Execution Host
tcp/ip
Test Inter-face
LightControllerGUI
setLevel
grasp
release
setLevel
grasp
release
LightController
JavaVM+w2k/Linux
out(IGrasp); //@900
silence(500);
silence(1000);
in(OSetLevel,1,dfTolerance);
silence(1000);
in(OSetLevel,2,dfTolerance);
silence(1000);

“Scripts” for LightControl

Events: const IGrasp=0; const int IRelease=1; const int OSetLevel=0;

- void **out**(int eventNo);

  send eventNo to IUT at now();
- void **silence**(int msDelay);

  expect no outputs for msDelay: otherwise fail
- void **in** (int eventNo,int par, int msTolerance);

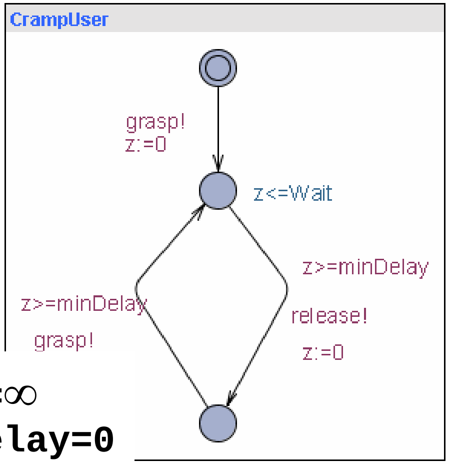
  expect input event(par) before now()+msTolerance otherwise fail
- void **at**(int eventNo, int par, int msTime, int msTolerance);

  expect input eventNo(par) at time msTime from start of test+/- msTolerance

Test Purposes 1

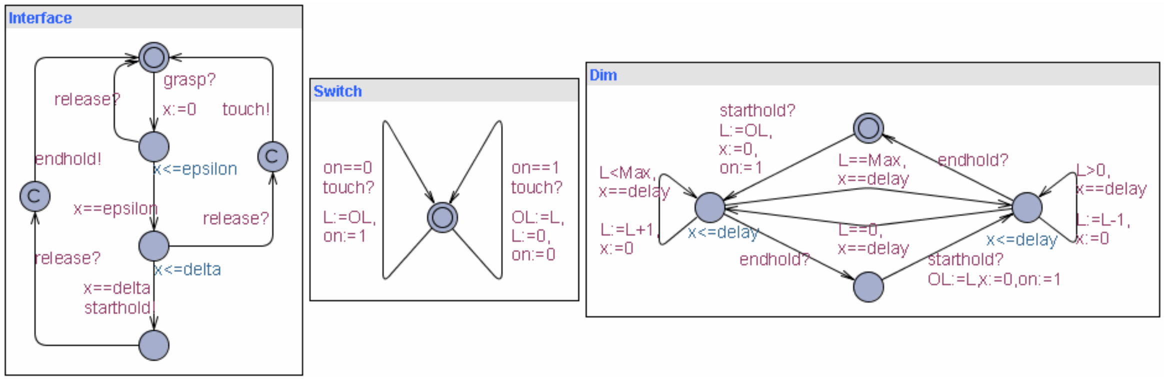
A specific test objective (or observation) the tester wants to make on SUT

Environment model



Wait=∞
minDelay=0

System model



TP1: Check that the light can become bright:

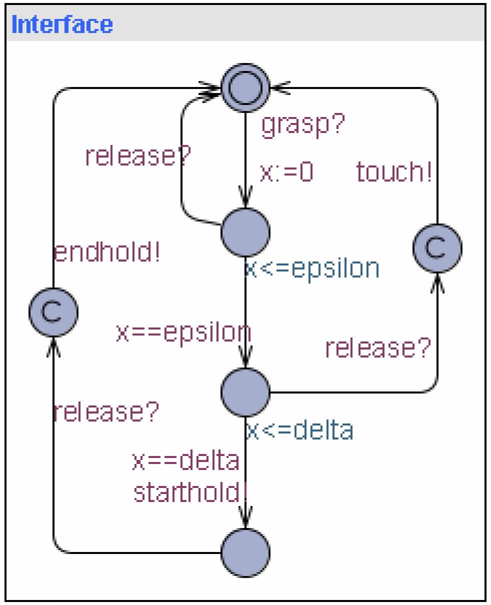
E<> L==10

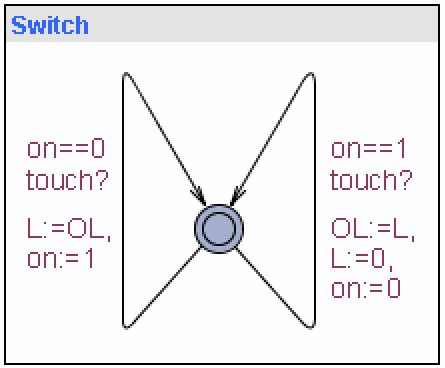
- *Shortest* (and *fastest*) Test:

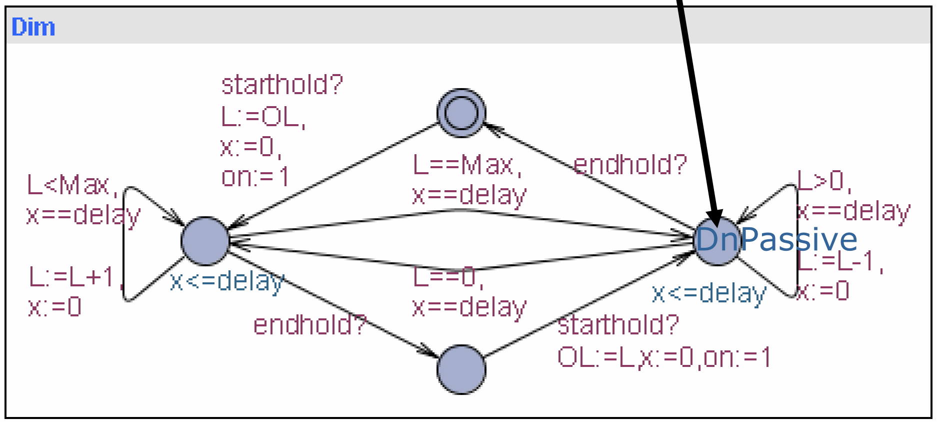
```
out(IGrasp);silence(500);in(OSetLevel,0);silence(1000);
in(OSetLevel,1);silence(1000);in(OSetLevel,2); silence(1000);
in(OSetLevel,3);silence(1000);in(OSetLevel,4);silence(1000);
in(OSetLevel,5);silence(1000);in(OSetLevel,6);silence(1000);
in(OSetLevel,7);silence(1000);in(OSetLevel,8);silence(1000);
in(OSetLevel,9);silence(1000);in(OSetLevel,10);
out(IRelease);
```

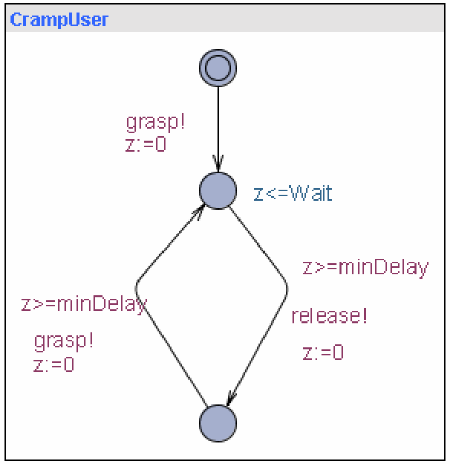
Test Purposes 2

TP2: Check that controller can enter location 'DnPassive':
`E<> Dim.DnPassive`







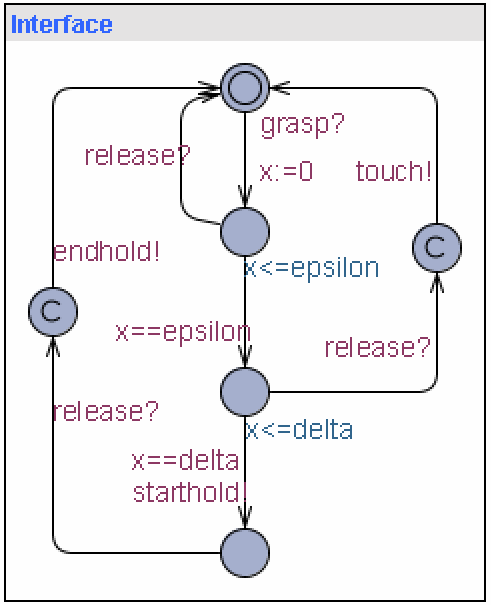


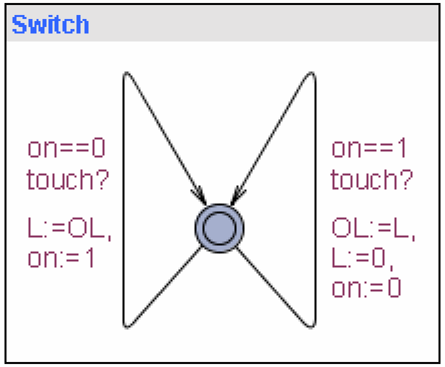
- If delay=1000
- *Shortest* (and *fastest*) Test:

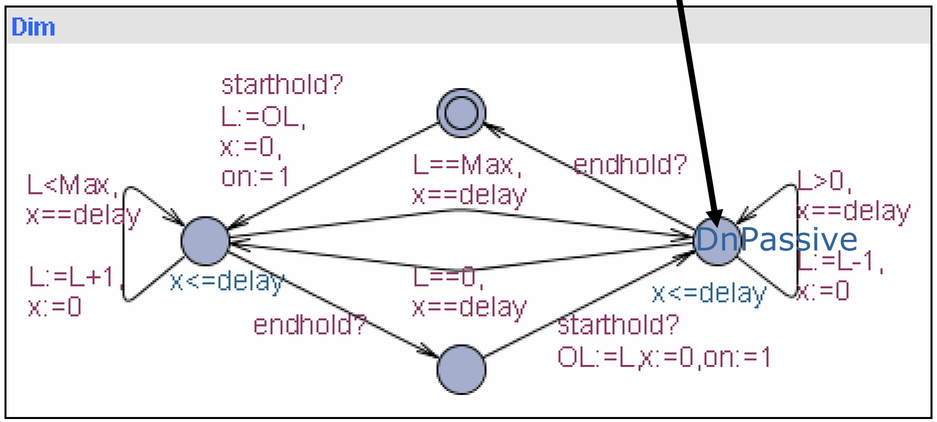
```
out(IGrasp);
silence(500);
in(OSetLevel,0);
out(IRelease);
out(IGrasp);
silence(500);
```

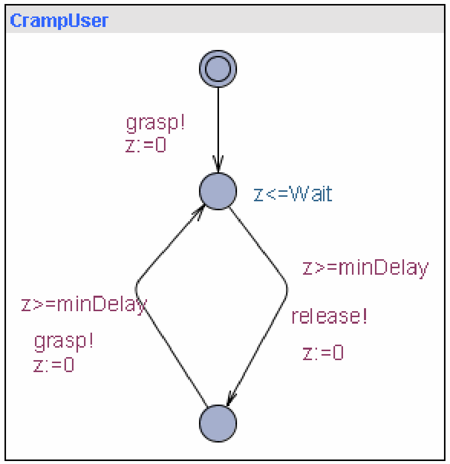
Test Purposes 2

TP2: Check that controller can enter location ‘DnPassive’:
E<> Dim.DnPassive









- If delay=40?
- *Shortest* Test:

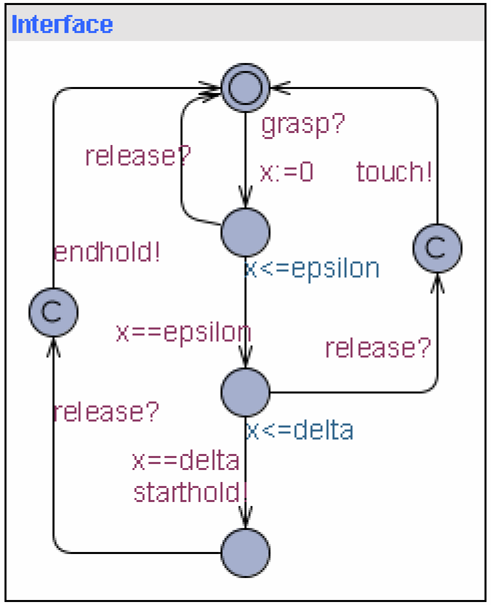
```
out(IGrasp);
silence(500);
in(OSetLevel,0);
out(IRelease);
out(IGrasp);
silence(500);
```

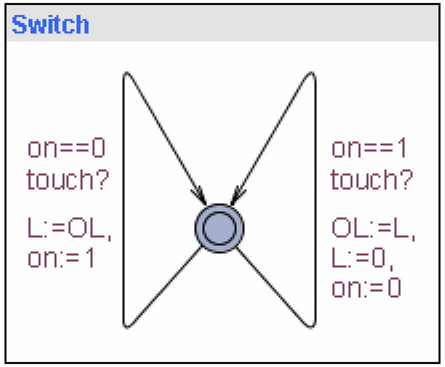
- *Fastest* Test:

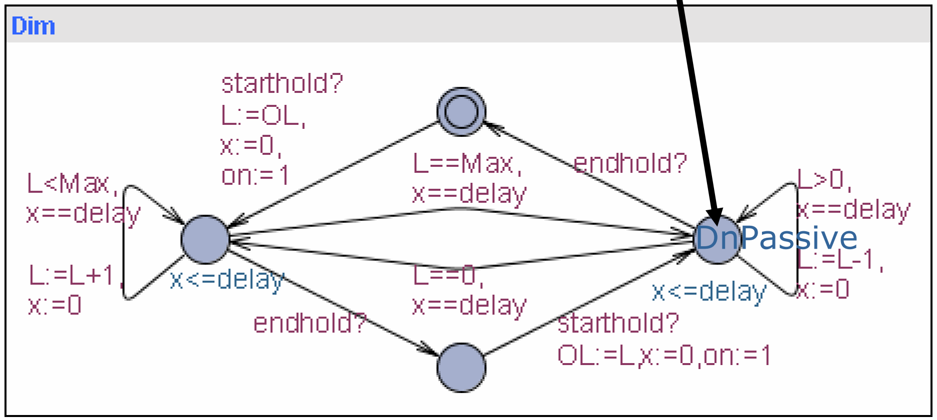
```
out(IGrasp);silence(500);in(OSetLevel,0);silence(40);
in(OSetLevel,1);silence(40);in(OSetLevel,2); silence(40);
in(OSetLevel,3);silence(40);in(OSetLevel,4); silence(40);
in(OSetLevel,5);silence(40);in(OSetLevel,6); silence(40);
in(OSetLevel,7);silence(40);in(OSetLevel,8); silence(40);
in(OSetLevel,9);silence(40);in(OSetLevel,10);silence(40);
```

Test Purposes 2

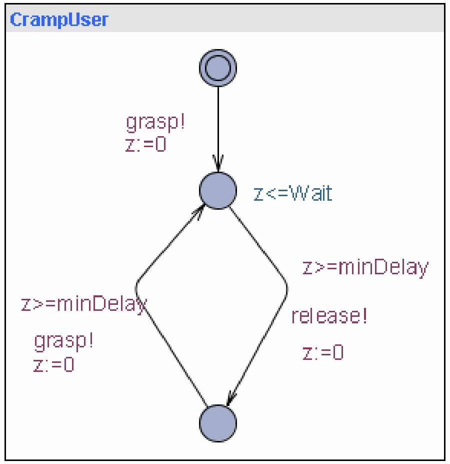
TP2: Check that controller can enter location 'DnPassive':
E<> Dim.DnPassive







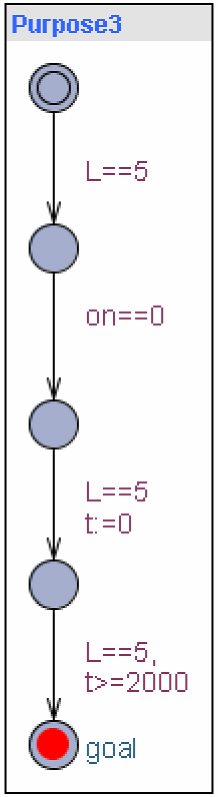
- If Wait=1500 and minDelay=400?



Ask a tool

Test Purposes 3

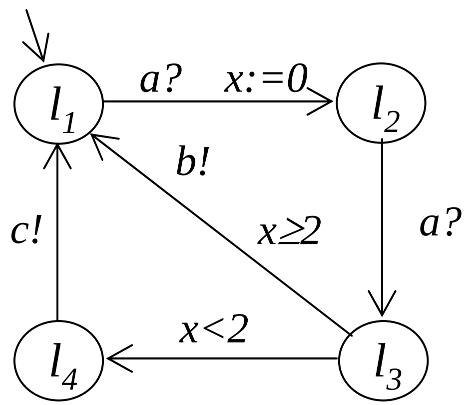
TP3: Check that controller re-sets light level to previous value after switch-on. **`E<> Purpose1.goal`**



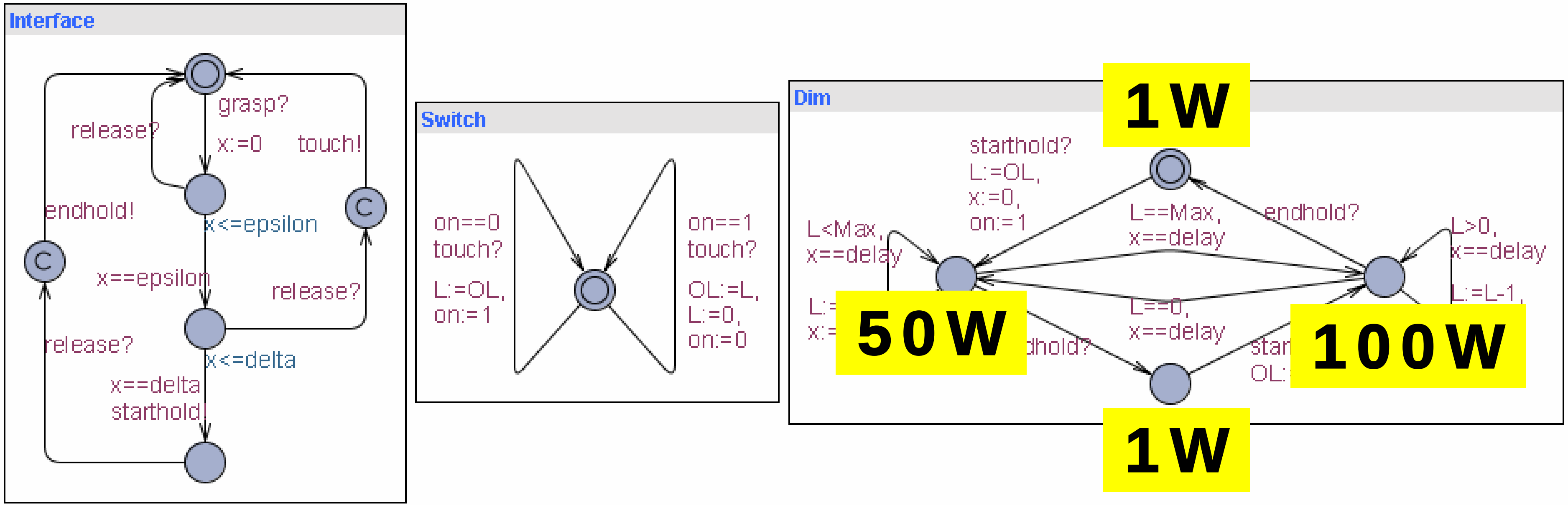

```
out(IGrasp);   //set level to 5
silence(500);
in(OSetLevel,0);
silence(1000);
in(OSetLevel,1);
silence(1000);
in(OSetLevel,2);
silence(1000);
in(OSetLevel,3);
silence(1000);
in(OSetLevel,4);
silence(1000);
in(OSetLevel,5);
out(IRelease);

out(IGrasp);   //touch To Off
silence(200);
out(IRelease);
in(OSetLevel,0);

out(IGrasp);   //touch To On
silence(200);
out(IRelease);
in(OSetLevel,5);

silence(2000);
```

Coverage Based Test Generation

- Multi purpose testing
- Cover measurement
- Examples:
  - Location coverage,
  - Edge coverage,
  - Definition/use pair coverage



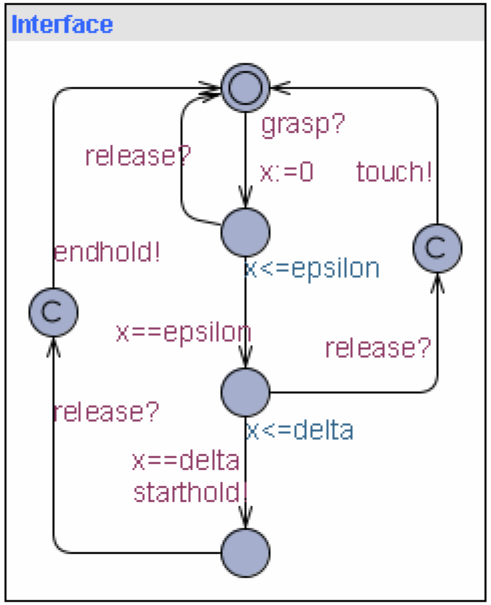
Optimal Tests

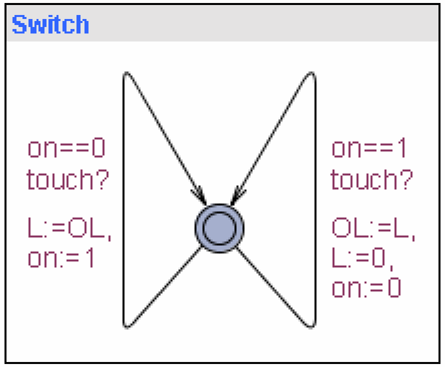


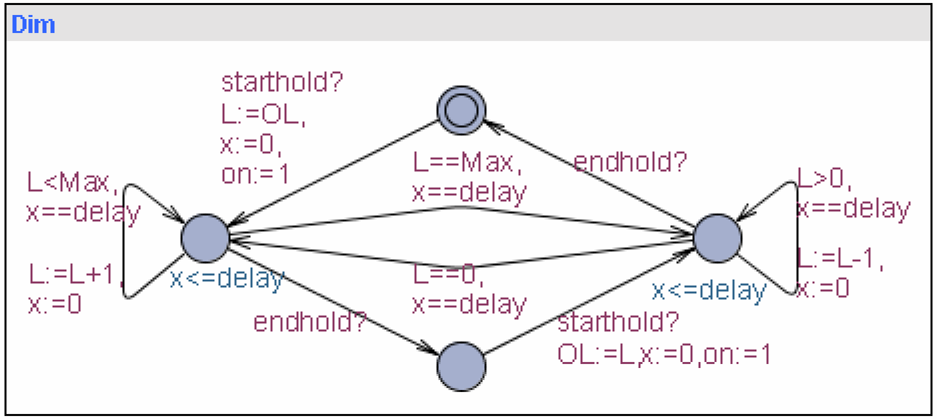
- ***Shortest*** test for max light??
- ***Fastest*** test for max light??
- ***Fastest*** edge-covering test suite??
- Least ***power*** consuming test??

Fastest Edge Coverage

Cost=12600 ms








```
out(IGrasp);              //touch:switch light on
silence(200);
out(IRelease);
in(OSetLevel,0);

out(IGrasp); //@200         // touch: switch light off
silence(200);
out(IRelease);//touch
in(OSetLevel,0);

//9
out(IGrasp); //@400         //Bring dimmer from ActiveUp
silence(500); //hold        //To Passive DN (level=0)
in(OSetLevel,0);
out(IRelease);
```

Page 1

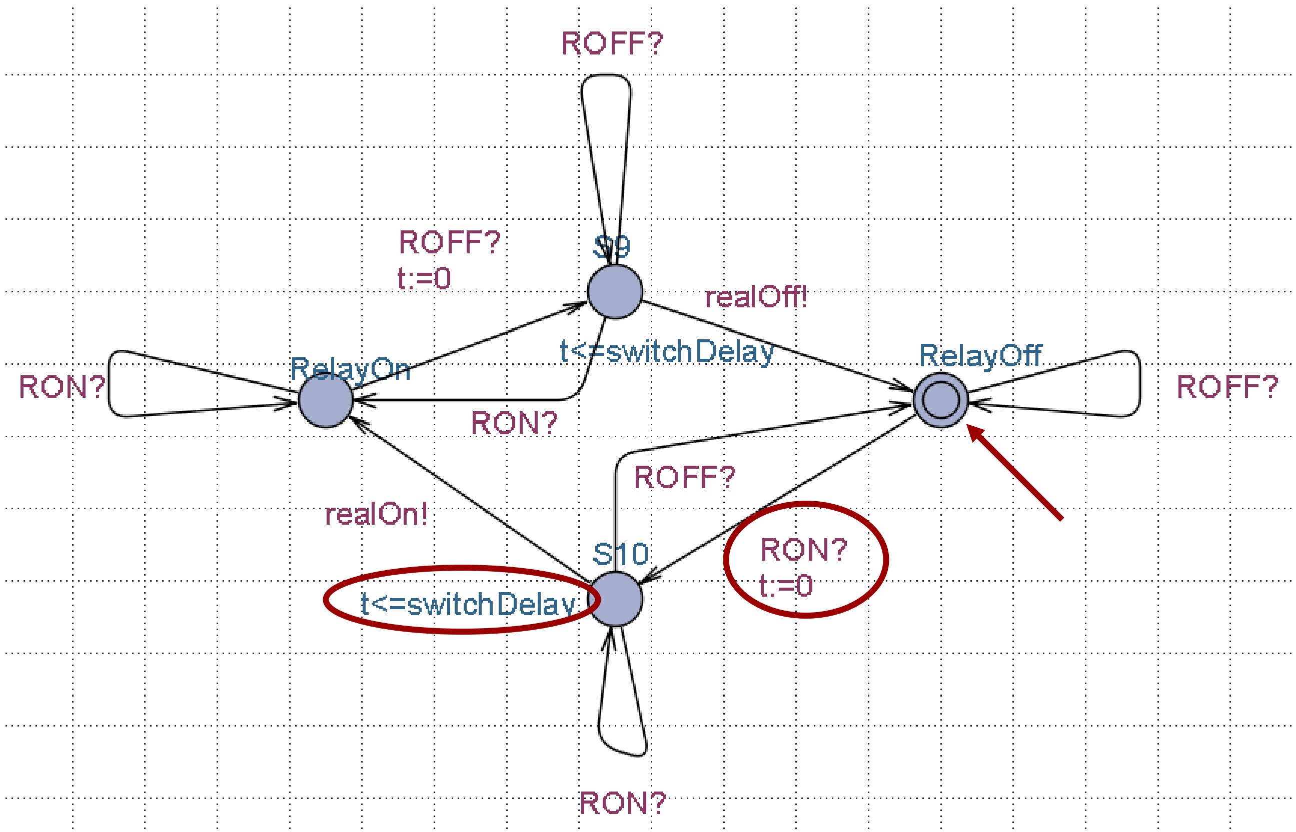
```
//13
out(IGrasp); //@900         // Bring dimmer  PassiveDn->ActiveDN->
silence(500);//hold         // ActiveUP+increase to level 10
silence(1000); in(OSetLevel,1);
silence(1000); in(OSetLevel,2);
silence(1000); in(OSetLevel,3);
silence(1000); in(OSetLevel,4);
silence(1000); in(OSetLevel,5);
silence(1000); in(OSetLevel,6);
silence(1000); in(OSetLevel,7);
silence(1000); in(OSetLevel,8);
silence(1000); in(OSetLevel,9);
silence(1000); in(OSetLevel,10
silence(1000);  in(OSetLevel,9); //bring dimm State to ActiveDN

out(IRelease);              //check release->grasp is ignored
out(IGrasp); //@12400
out(IRelease);
silence(dfTolerance);
```

Page 2

Non-Determinism

Modeling Timing Tolerances

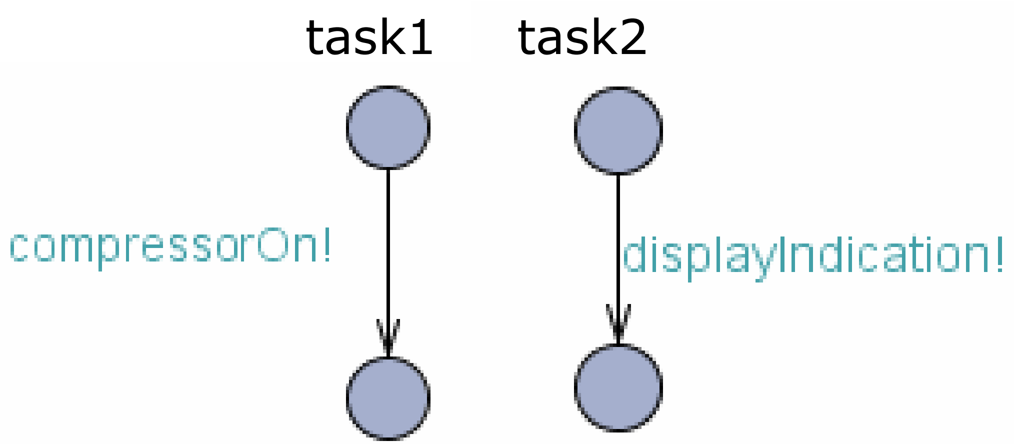
A relay is required to switch on (event realOn!) ***within*** *switchDelay* time units after a request (event RON?)



Non-Determinism

Modeling Action uncertainty

- Event output ordering of two concurrent tasks in the IUT may be unknown



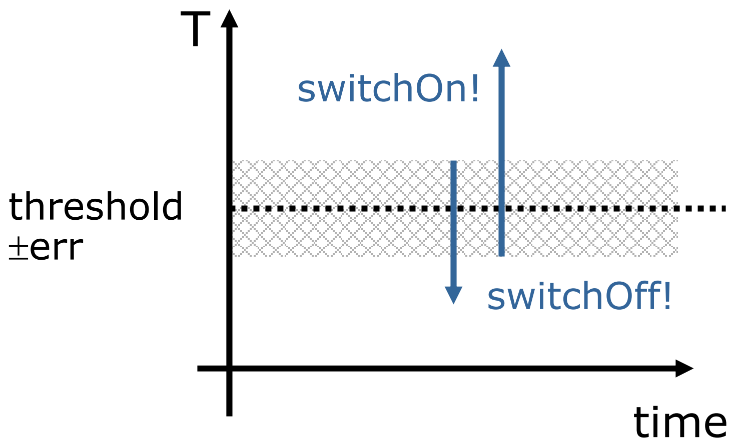
- CompressorOn then displayIndication, or
- displayIndication then compressorOn???

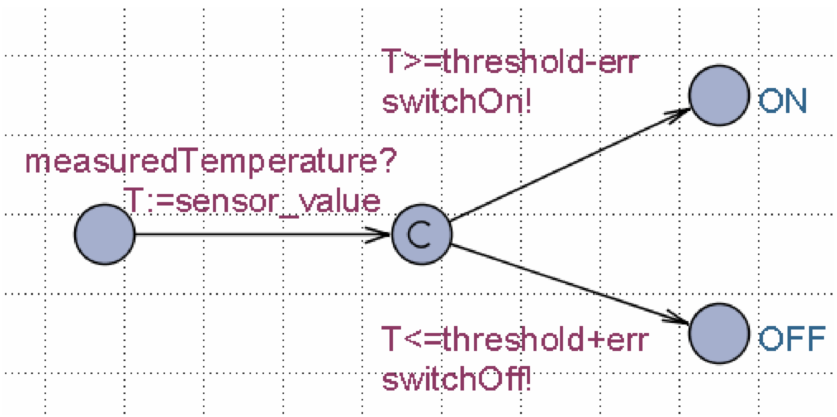
Non-Determinism

Modeling Action uncertainty

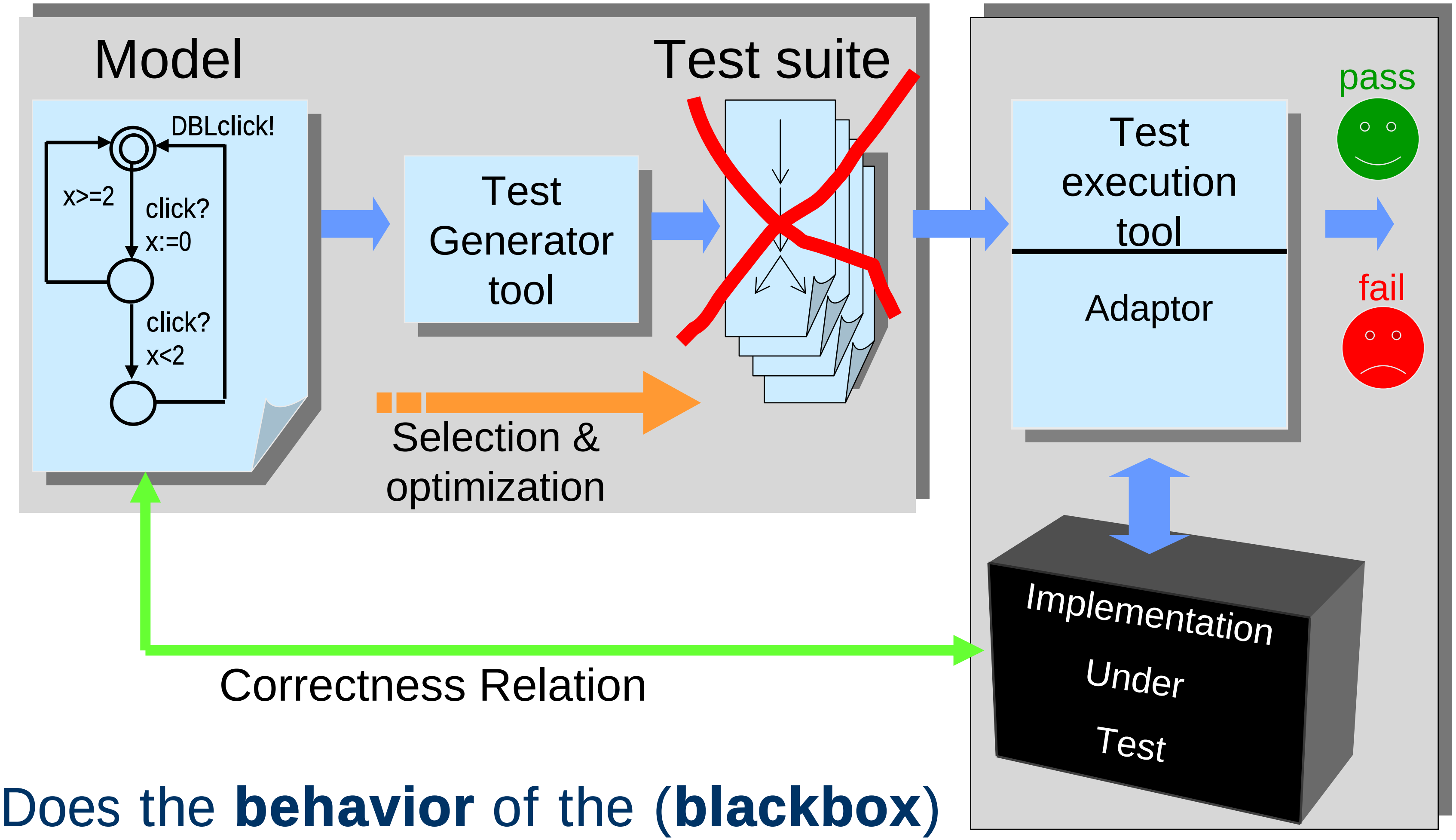
- A controller switches a relay when a control variable crosses a threshold value
- The value of the control variable is not known ***precisely***

⇒allow error margin (model by non-deterministic choice)



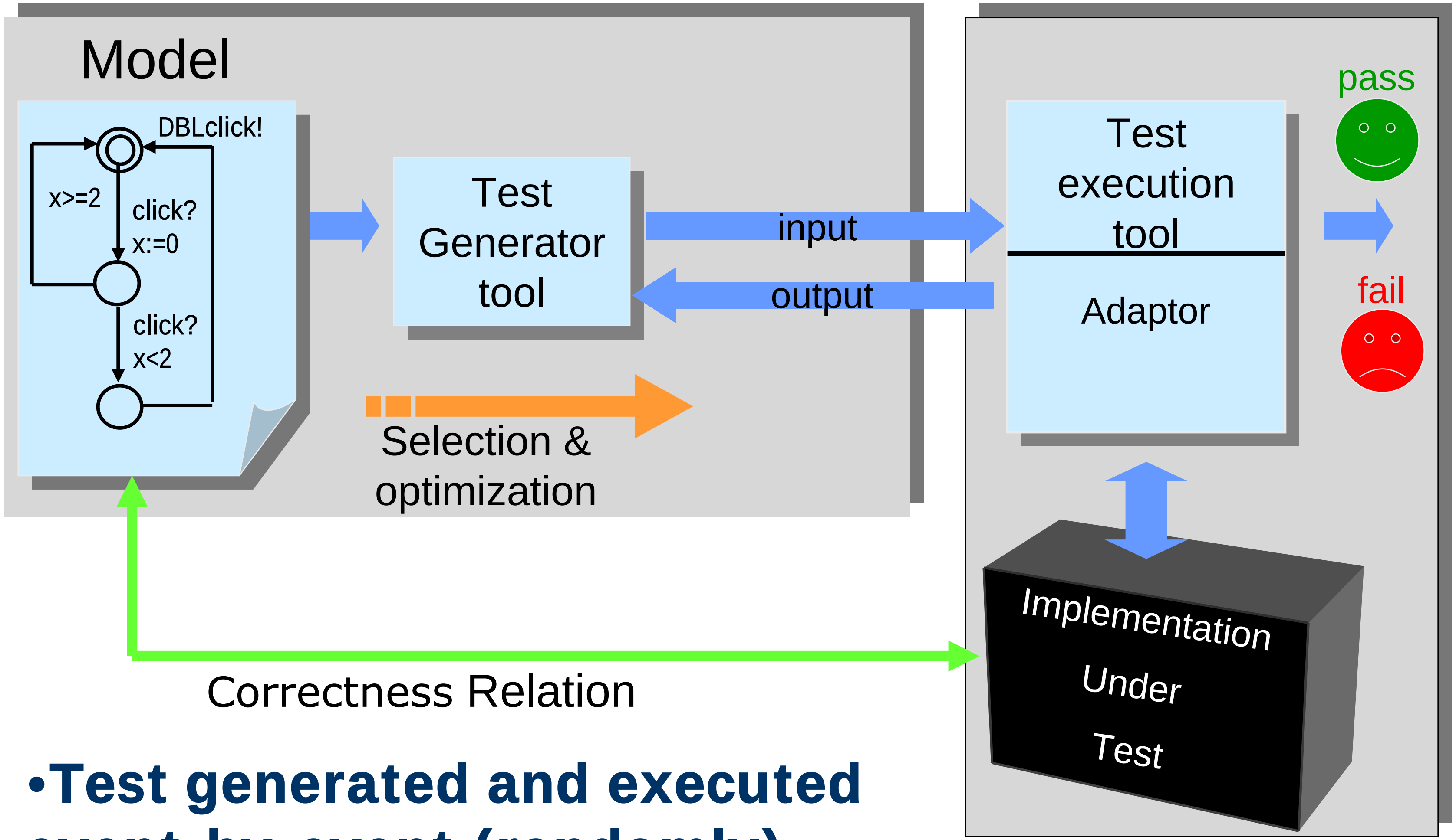


Automated Model Based Conformance Testing



Does the **behavior** of the (**blackbox**) implementation ***comply*** to that of the specification?

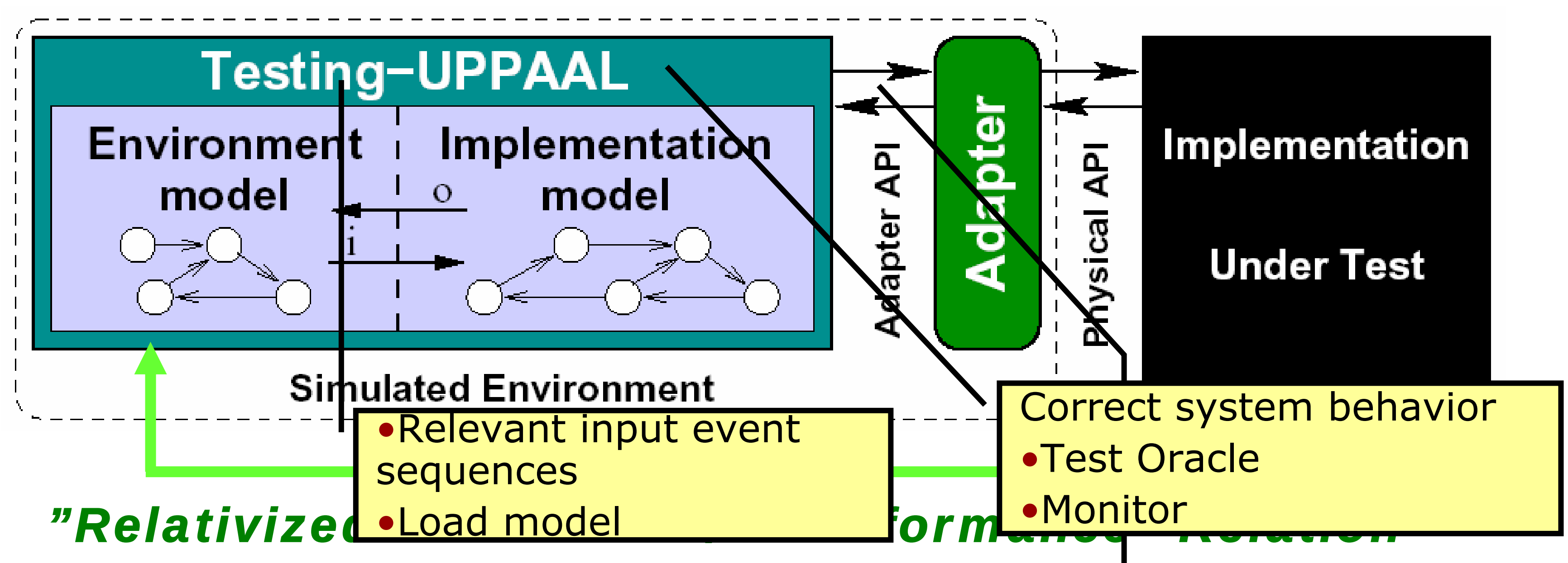
Online Testing



- **Test generated and executed event-by-event (randomly)**
- **A.K.A on-the-fly testing**

Tron Framework

- **UppAal-TRON:** **T**esting **R**eal-Time Systems **On**line
- Spec = UppAal Timed Automata *Network: Env || IUT*



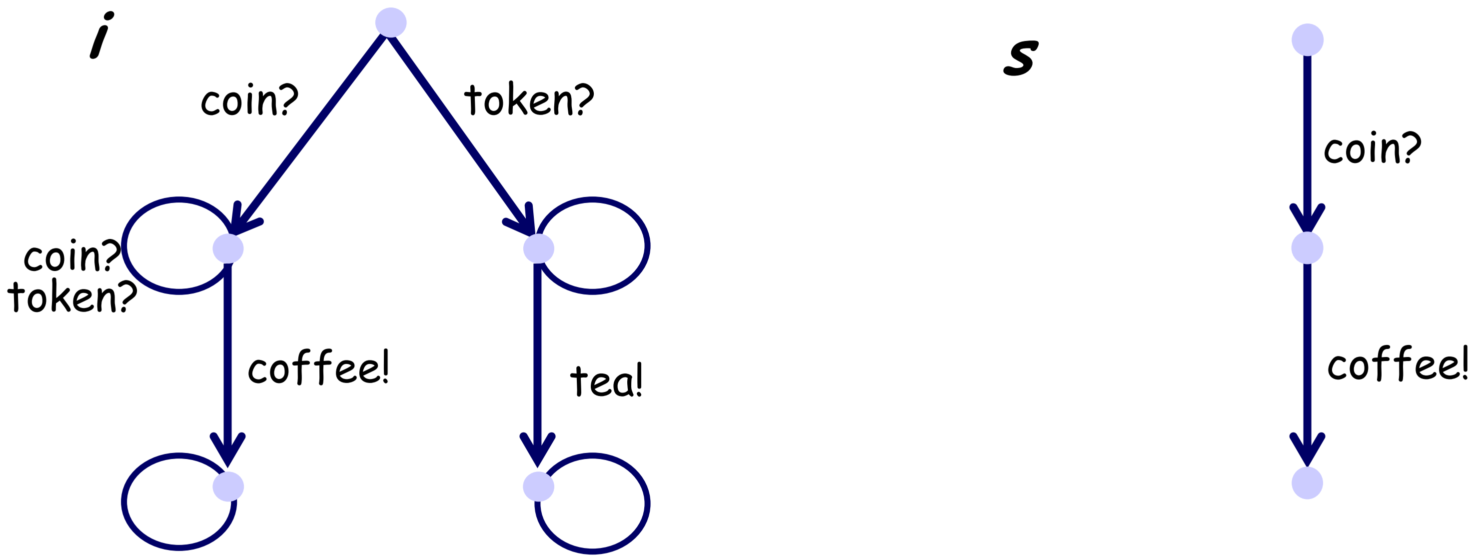
Timed Trace: $i_1.2½.o_1.3.o_2.19.i_2.5.i_3$

Correctness Relation

BRICS
Basic Research
in Computer Science

CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

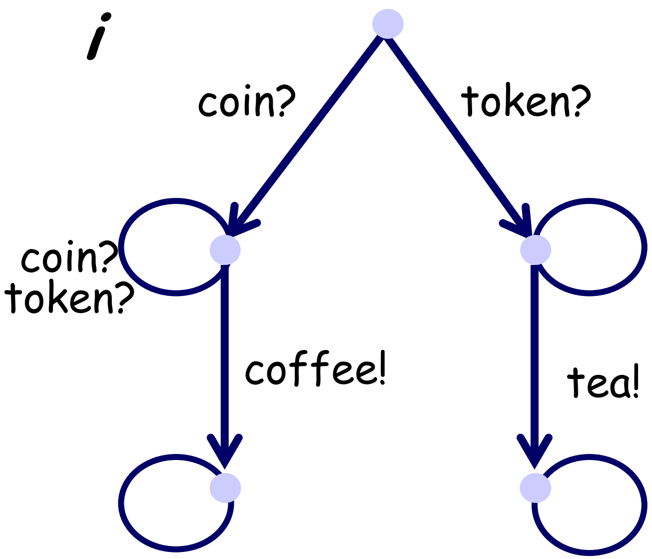
Tretman's IOCO



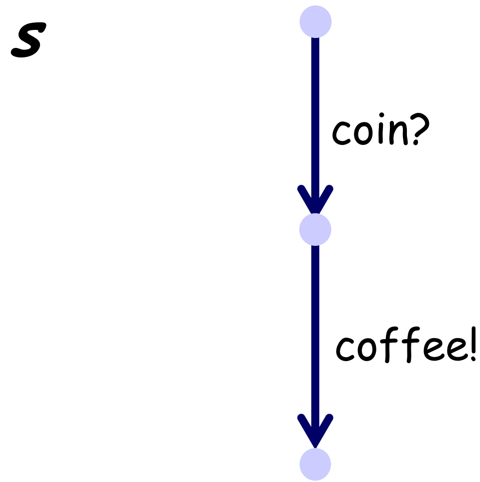
- Labeled Transition Systems (LTS) as semantic model
- **Input** actions (?) are controlled by the environment
- **Output** actions (!) are controlled by the implementation
- Implementations are *input enabled*
- *Testing hypothesis:* IUT can be modeled by some (unknown) LTS

[Jan Tretmans].

I conforms-to S ??

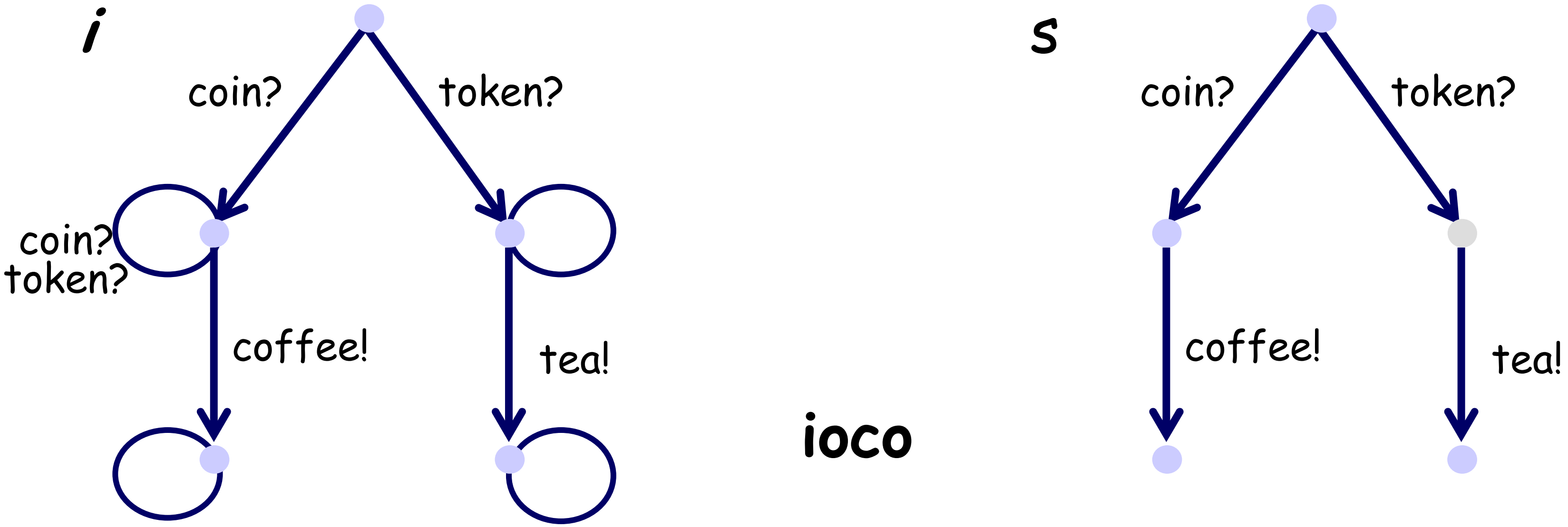


ioco



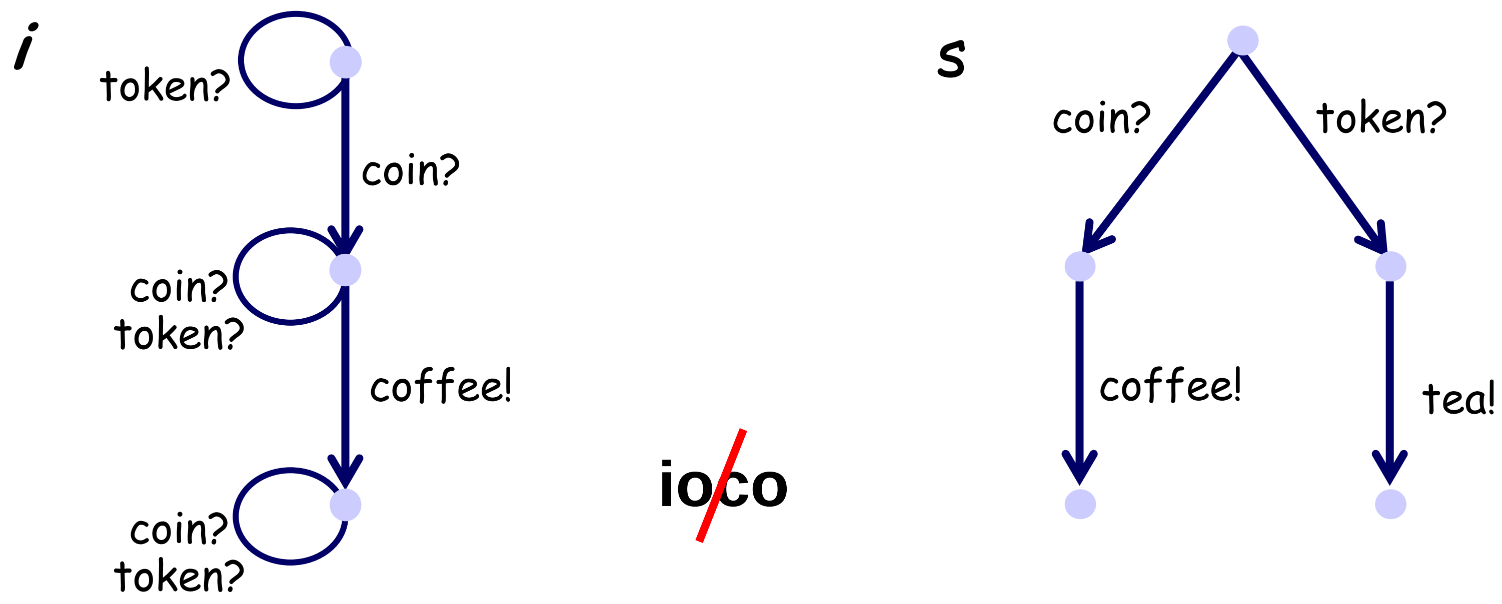
[Jan Tretmans].

I conforms-to S ??



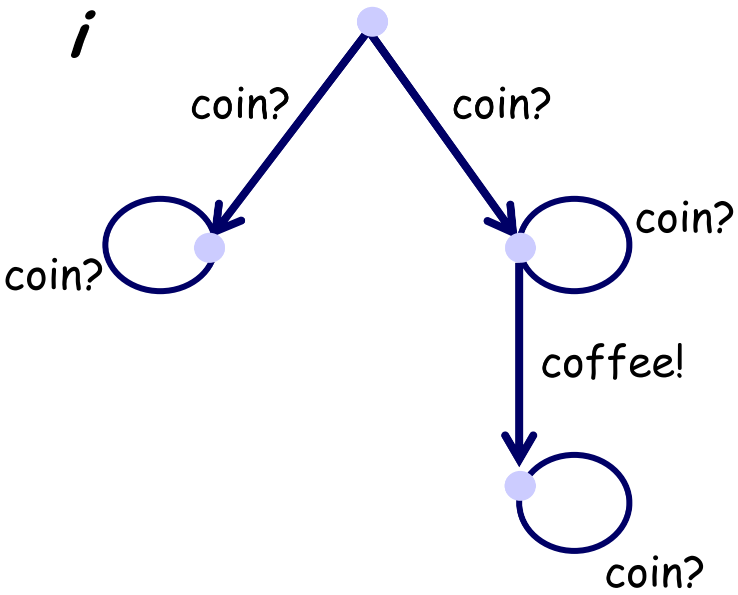
[Jan Tretmans].

I conforms-to S ??

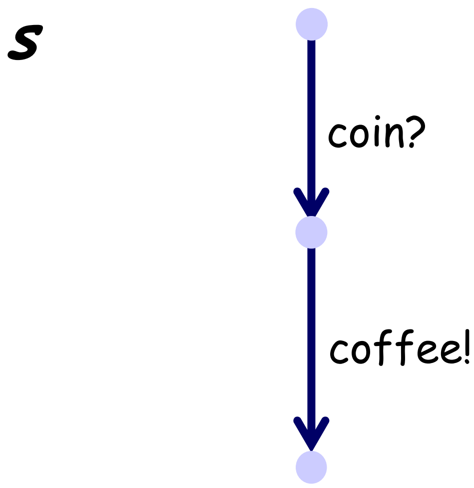


[Jan Tretmans].

I conforms-to S ??



ioco



[Jan Tretmans].

Tretman's IOCO

"*The*" conformance relation used for blackbox testing of (untimed) reactive systems

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

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

$p \xrightarrow{\delta} p$ iff $\forall\, o! \in L_U \cup \{\tau\} : p \overset{o!}{\not\longrightarrow}$

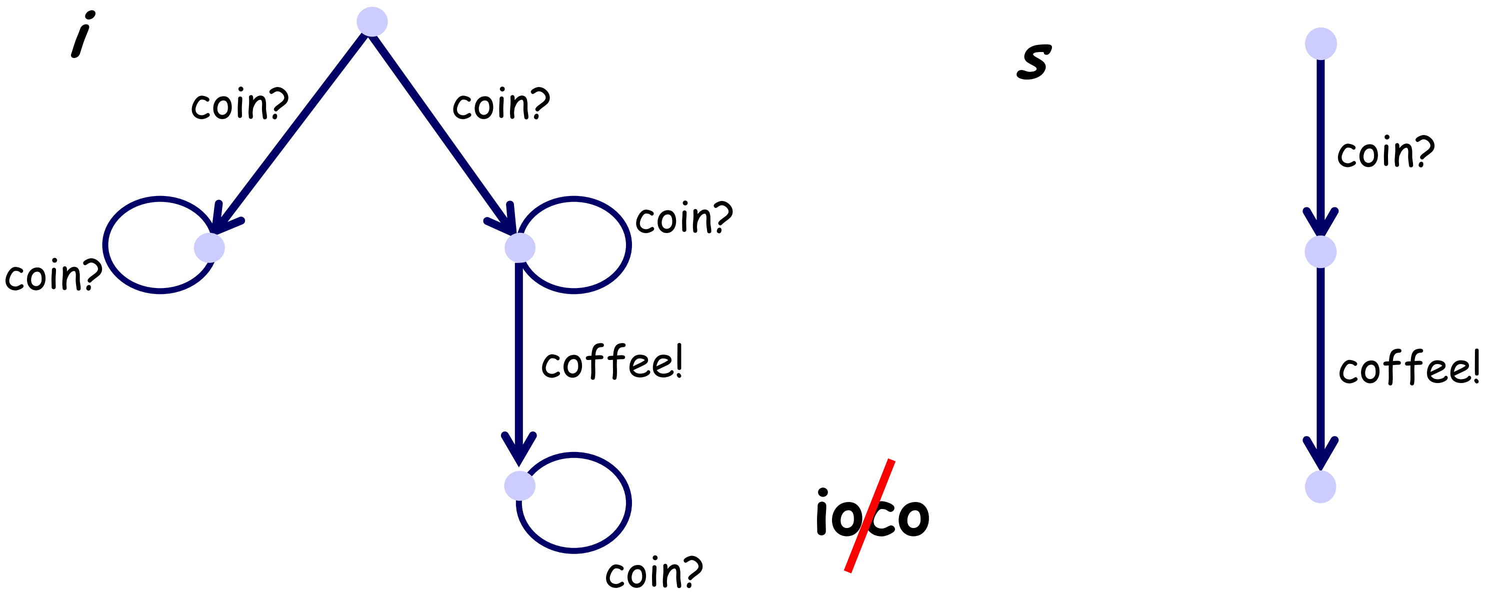
out **(** P **)** $= \{ o! \in L_U \mid p \xrightarrow{!o} ,\ p \in P \}$
$\cup \{ \delta \mid p \xrightarrow{\delta} p,\ p \in P \}$

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

[Jan Tretmans].

IOCO Example

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



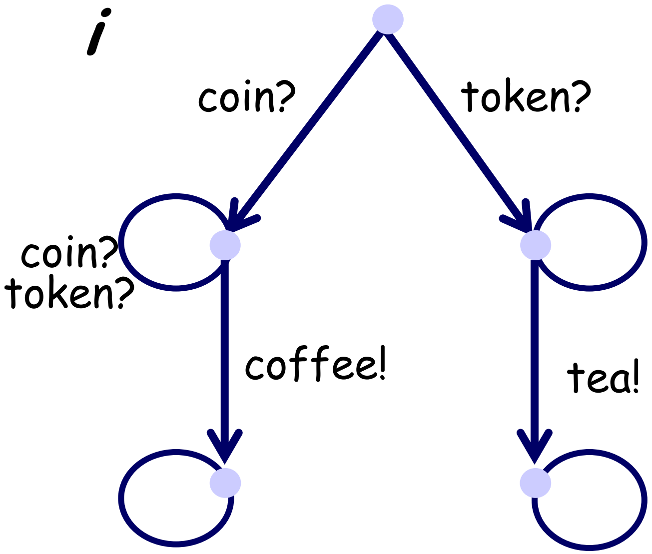
$\mathit{out}(i \ \textbf{after} \ \text{coin?}) = \{ \delta, \text{coffee!} \}$

$\mathit{out}(s \ \textbf{after} \ \text{coin?}) = \{ \text{coffee!} \}$

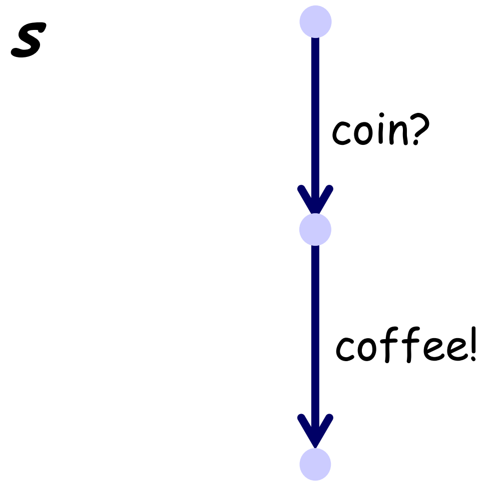
[Jan Tretmans].

IOCO Example

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



ioco



out (*i* **after** coin?) = { coffee! }

out (*i* **after** token?) = { tea! }

out (*s* **after** coin?) = { coffee! }

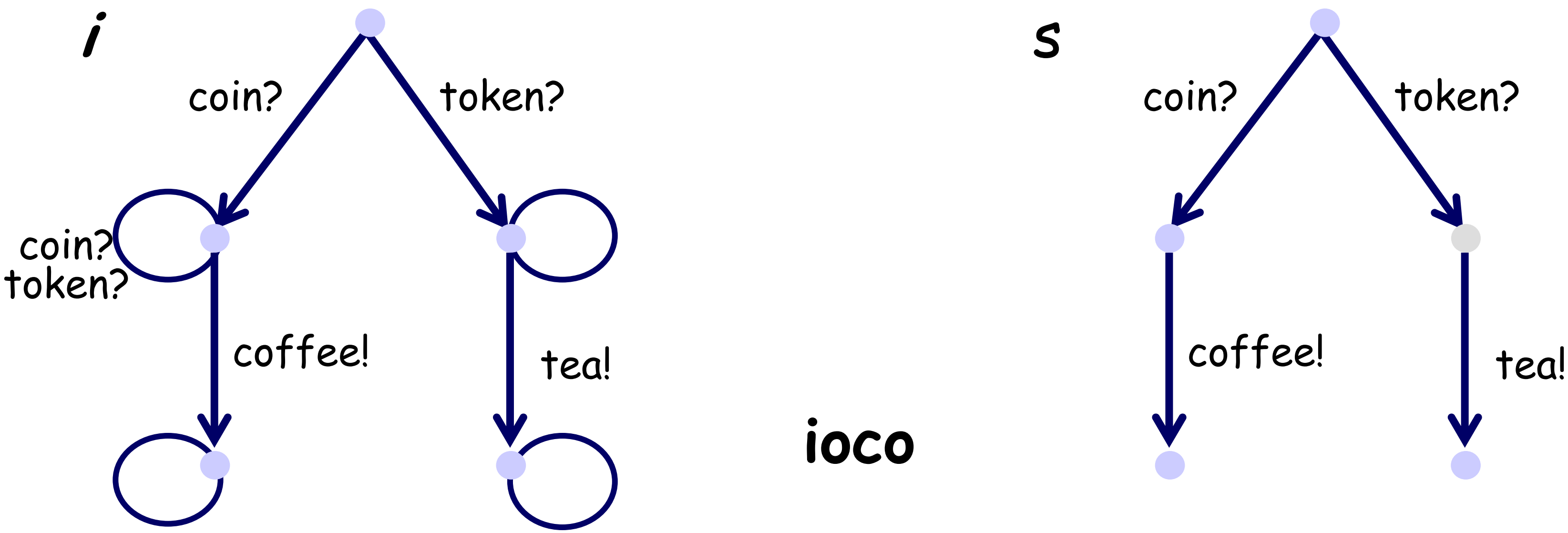
out (*s* **after** token?) = $\emptyset$

But token? $\notin$ *Straces* (*s*)

[Jan Tretmans].

Ioco

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



$\mathit{out}(i\ \mathbf{after}\ \text{coin?}) = \{\ \text{coffee!}\ \}$

$\mathit{out}(i\ \mathbf{after}\ \text{token?}) = \{\ \text{tea!}\ \}$

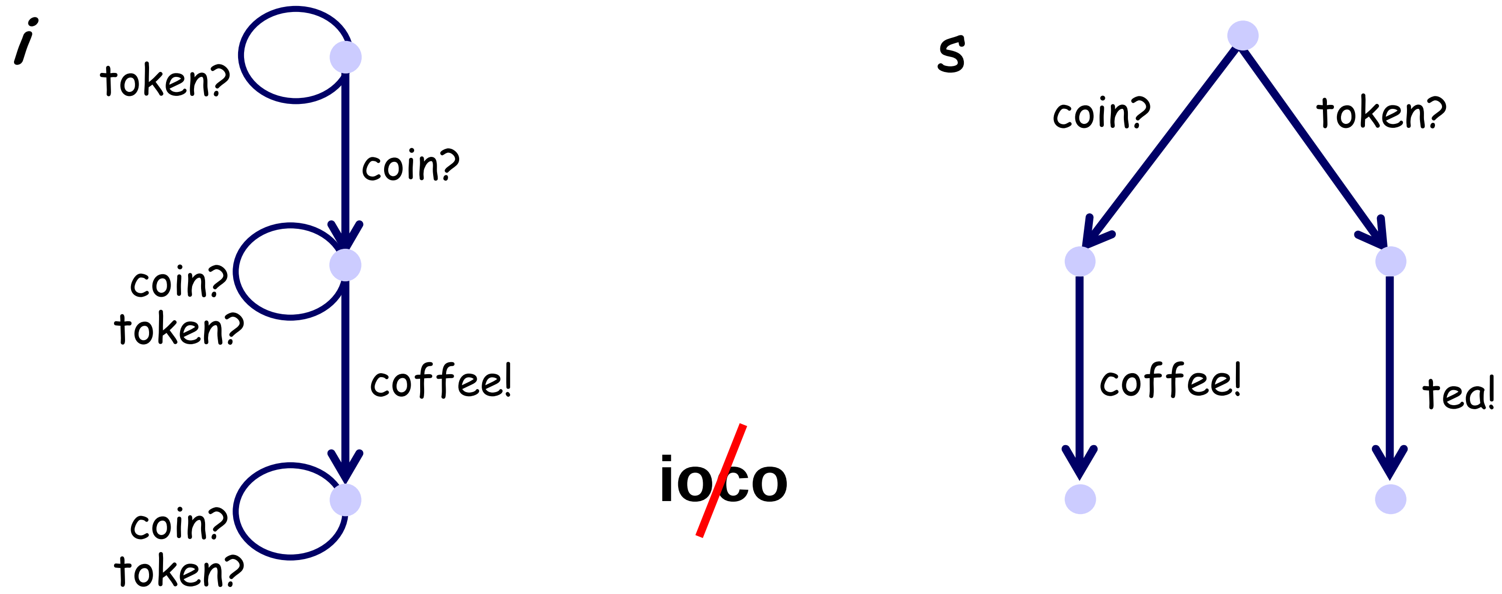
$\mathit{out}(s\ \mathbf{after}\ \text{coin?}) = \{\ \text{coffee!}\ \}$

$\mathit{out}(s\ \mathbf{after}\ \text{token?}) = \{\ \text{tea!}\ \}$

[Jan Tretmans].

Ioco

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



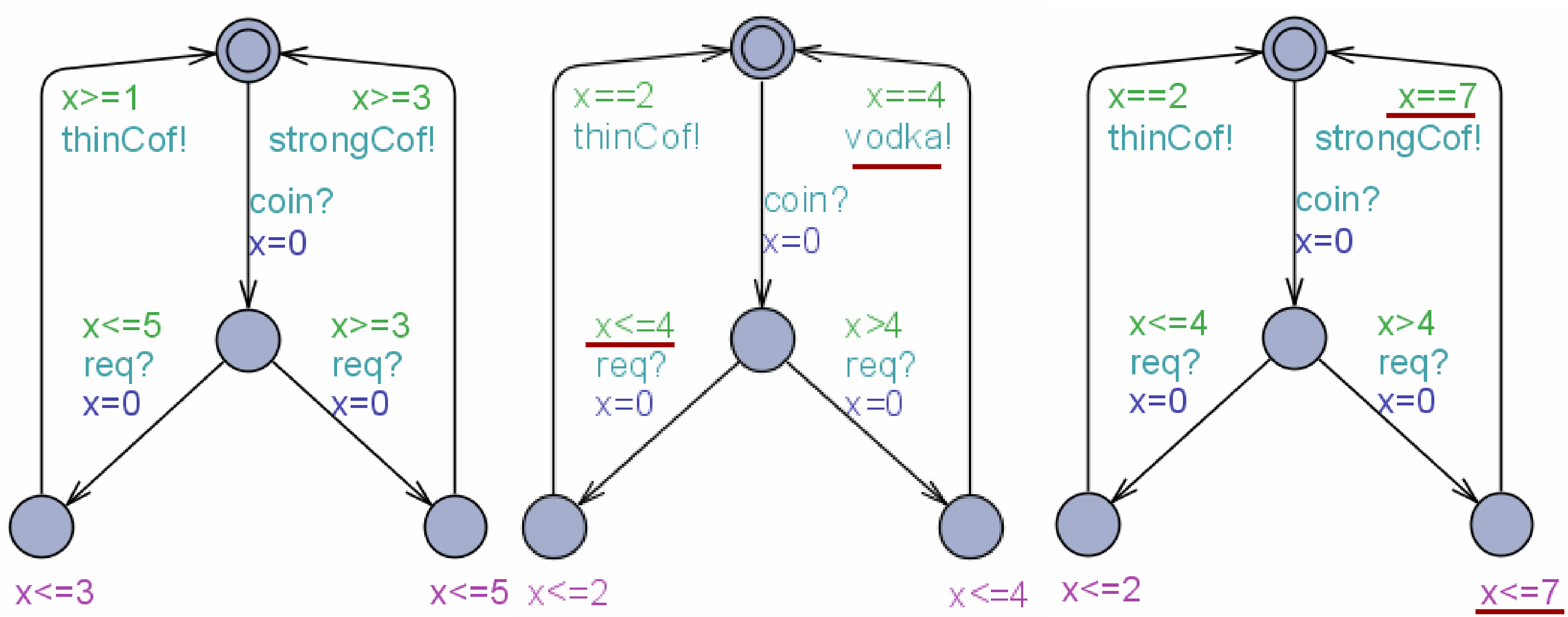
out (*i* **after** token?) = { δ }

out (*s* **after** token?) = { tea! }

[Jan Tretmans].

Timed Conformance??

Specification | Implementation 1 | Implementation 2



Example Traces

- c?.2.r?.2.weakC
- c?.5.r?.4.strongC

- c?.5.r?.7

I1 **rt-ioco** S
I2 **rt-ioco** S

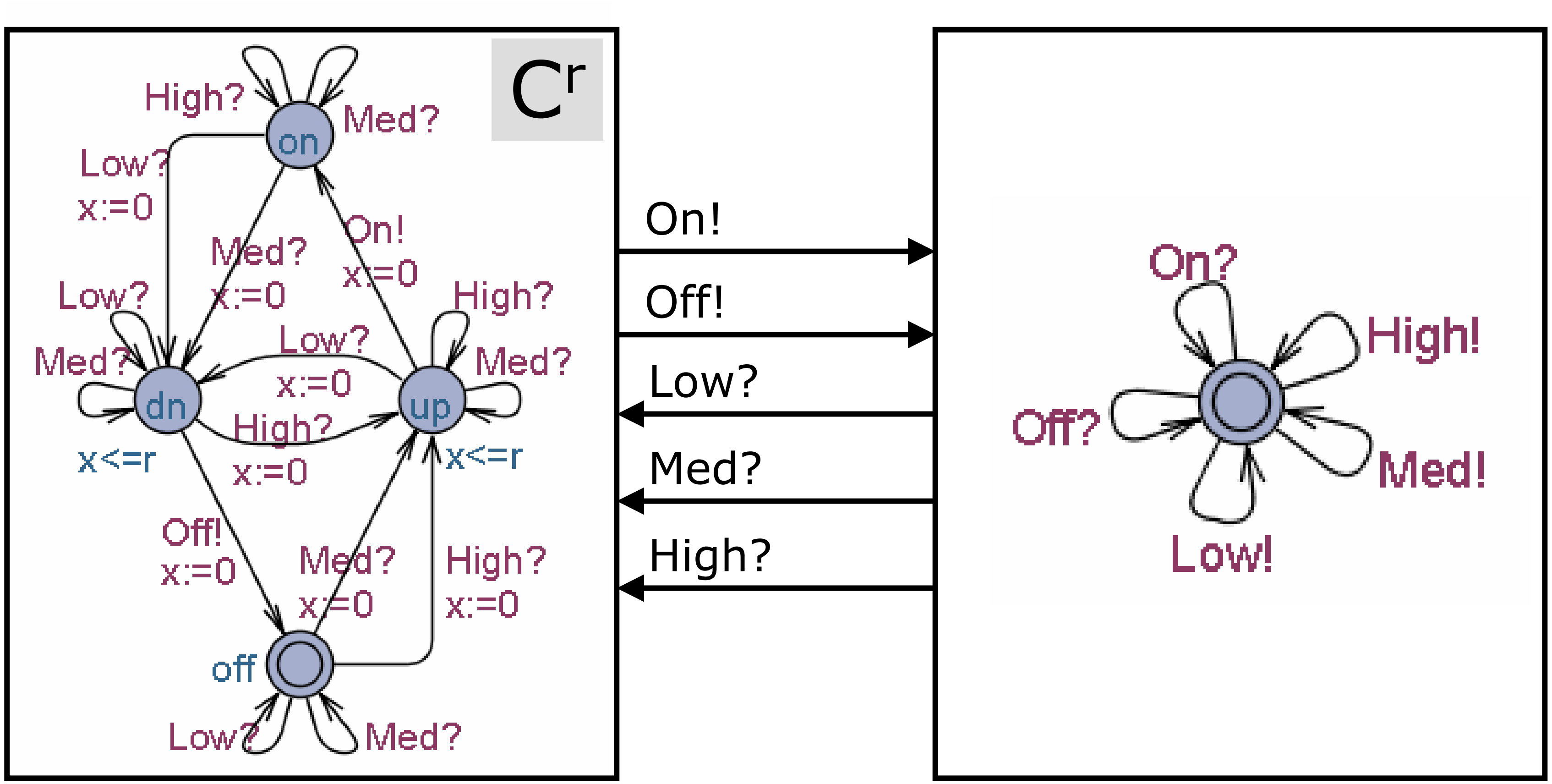
Timed Conformance

- Let **I, S** be timed I/O LTS, P a set of states
- **TTr**(P): the set of *timed traces* from P
  - eg.: σ = coin?.5.req?.2.weakCoffee!.9.coin?
- **Out**(P **after** σ) = possible ***outputs*** and ***delays*** after σ
  - eg. out ({l_2,x=1}): {weakCoffee,0...2}

- **I rt-ioco S =def**
  - $\forall\sigma \in$ **TTr(S): Out(I after** σ**)** $\subseteq$ **Out(S after** σ**)**
  - **TTr(I)** $\subseteq$ **TTr(s)**

- **Intuition**
  - **no illegal output is produced and**
  - **required output is produced (at right time)**

Sample Cooling Controller



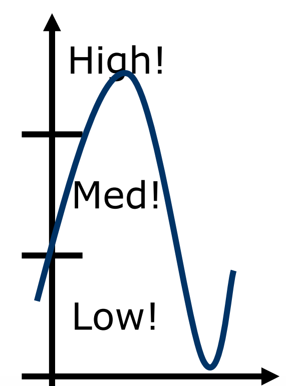
IUT-model

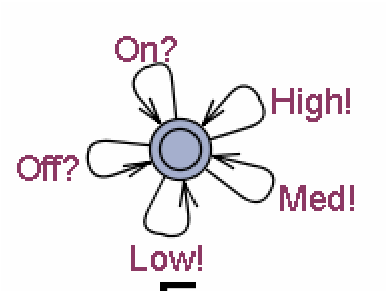
Env-model

- When T is high (low) switch on (off) cooling within r secs.
- When T is medium cooling may be either on or off (impl freedom)

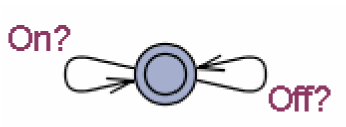
Environment Modeling

- E_M Any action possible at any time
- E_1 Only realistic temperature variations
- E_2 Temperature never increases when cooling
- E_L No inputs (completely passive)

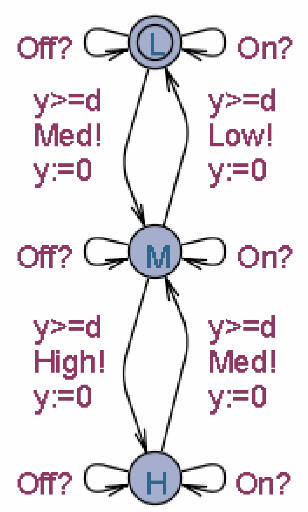




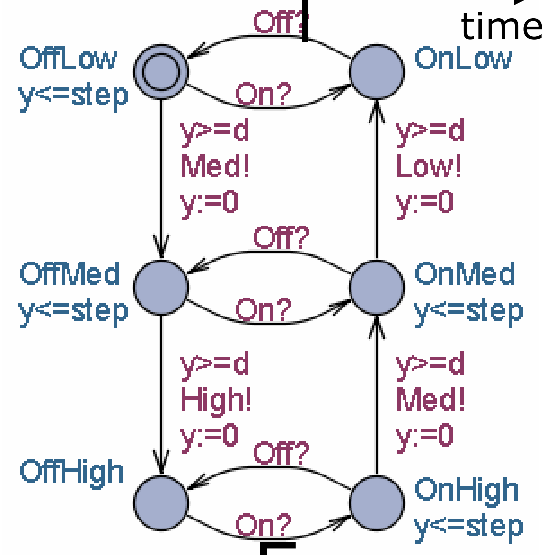
E_M



E_L



E_1

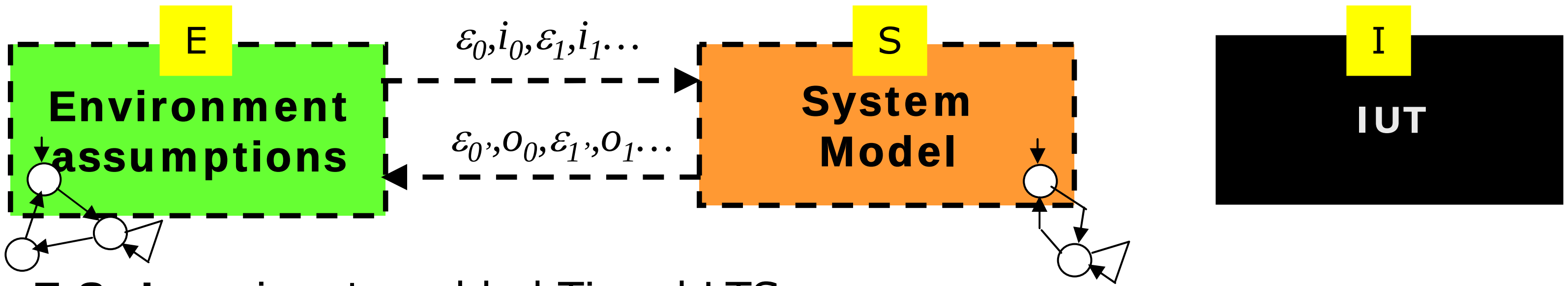


E_2

$$E_L \sqsubseteq E_2 \sqsubseteq E_1 \sqsubseteq E_M$$

Implementation relation

Relativized real-time io-conformance



- **E,S, I** are input enabled Timed LTS
- **Let** *P* **be a set of states**
- **TTr**(*P*): the set of *timed traces* from states in *P*
- *P* **after** σ = the set of states reachable after timed trace σ
- **Out**(*P*) = possible outputs and delays from states in *P*

- I rt-ioco$_E$ S $=_{def}$
  $\forall\sigma \in$ TTr(E): Out((E,I) after σ) $\subseteq$ Out((E,S) after σ)

- I rt-ioco$_E$ s iff TTr(I) $\cap$ TTr(E) $\subseteq$ TTr(S) $\cap$ TTr(E)

- **Intuition, for all assumed environment behaviors, the IUT**
  - **never produces illegal output, and**
  - **always produces required output in time**
- **~timed trace inclusion**

An Algorithm

Aalborg University Denmark

BRICS Basic Research in Computer Science

CISS Center for Indlejrede Software Systemer

IDEA: State-set tracking

- Dynamically compute all potential states that the model M can reach after the timed trace $\varepsilon_0, i_0, \varepsilon_1, o_1, \varepsilon_2, i_2, o_2, \ldots$ [Tripakis] Failure Diagnosis
- $Z = M$ ***after*** $(\varepsilon_0, i_0, \varepsilon_1, o_1, \varepsilon_2, i_2, o_2)$
- *If $Z = \emptyset$ the IUT has made a computation not in model:* ***FAIL***
- *i is a relevant input in Env iff $i \in EnvOutput(Z)$*

(Abstract) Online Algorithm

Algorithm *TestGenExe (S, E, IUT, T)* **returns** {**pass**, **fail**}
Z := {(s0, e0)}.
while Z ≠ ∅ ∧ #iterations ≤ T **do either** randomly:

1. // offer an input

    if *EnvOutput(Z)* ≠ ∅
        randomly choose *i* ∈ *EnvOutput(Z)*
        send i to IUT
        Z := Z *After* i

2. // wait d for an output

    randomly choose d ∈ *Delays(Z)*
    wait (for d time units or output o at d′ ≤ d)
    if o occurred **then**
            Z := Z *After* d′
            Z := Z *After* o // may become ∅ (⇒fail)
    else
            Z := Z *After* d // no output within d delay

3. *restart*:

    Z := {(s0, e0)}, **reset** IUT //reset and restart

if Z = ∅ **then return fail else return pass**

(Abstract) Online Algorithm

Algorithm *TestGenExe (S, E, IUT, T)* **returns** {**pass**, **fail**)

Z := {(s0, e0)}.

while Z ≠ ∅ ∧ #iterations ≤ T **do either** randomly:

1. // offer an input

    if *EnvOutput(Z)* ≠ ∅
    randomly choose *i* ∈ *EnvOutput(Z)*
    send i to I
    Z := Z

2. // wait d fo

    randomly c
    wait (for d
    if o occurre
    Z := Z A
    Z := Z *After* o // may become ∅ (⇒fail)
    else
    Z := Z *After* d // no output within d delay

3. *restart*:

    Z := {(s0, e0)}, **reset** IUT //reset and restart

if Z = ∅ **then return fail else return pass**

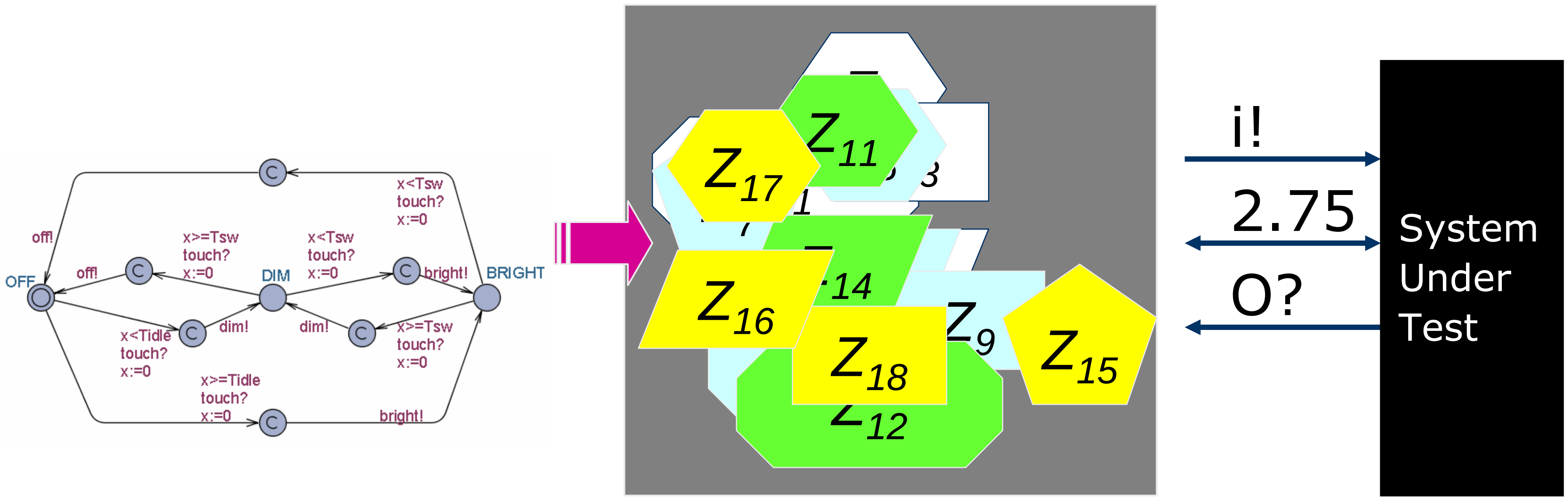
- Sound
- Complete (as T → ∞)

(Under some technical assumptions)

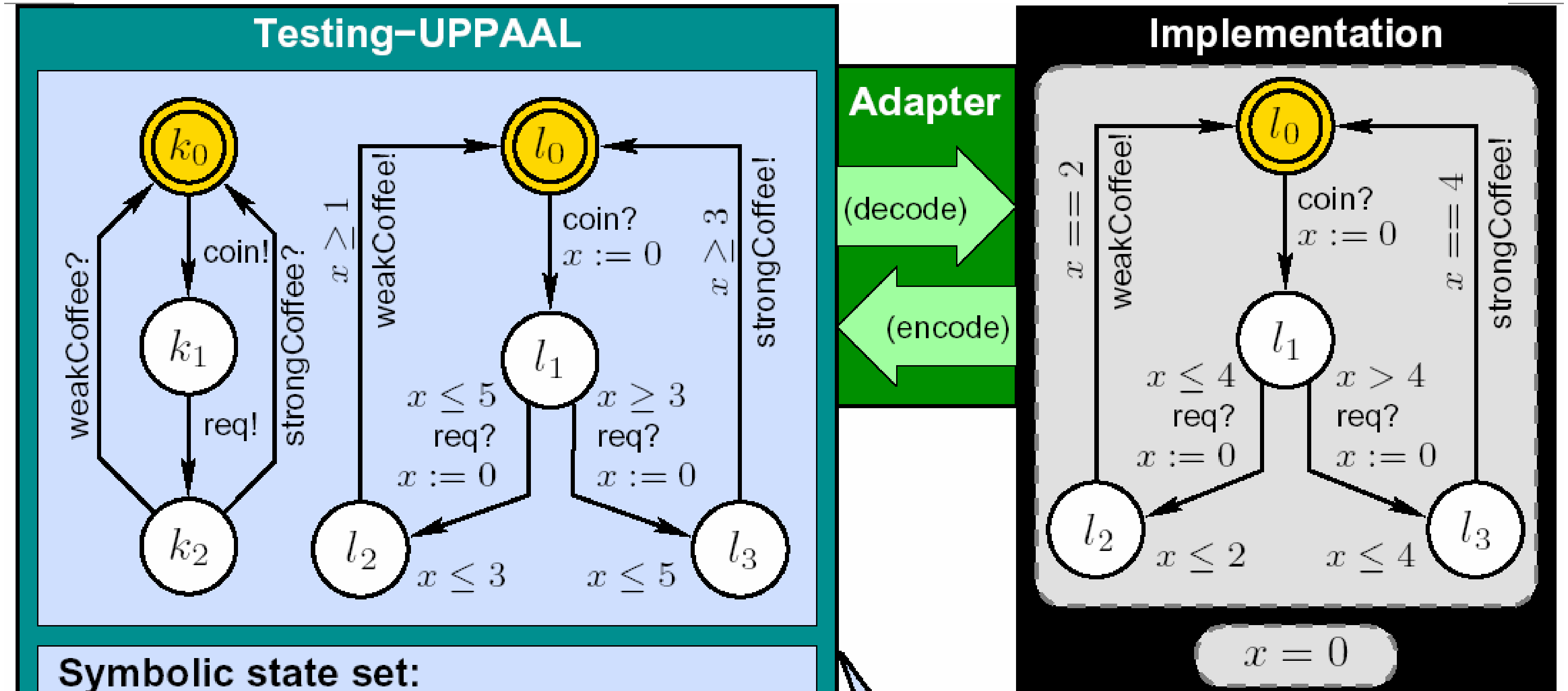
Real-time Online

Specification
TA-network

State-set explorer:
maintain and analyse a *set* of *symbolic* states (zones) in real time!



Online Testing Example



Symbolic state set:

$\{\langle k_0 l_0, 0 \leq x \leq 0 \rangle\}$

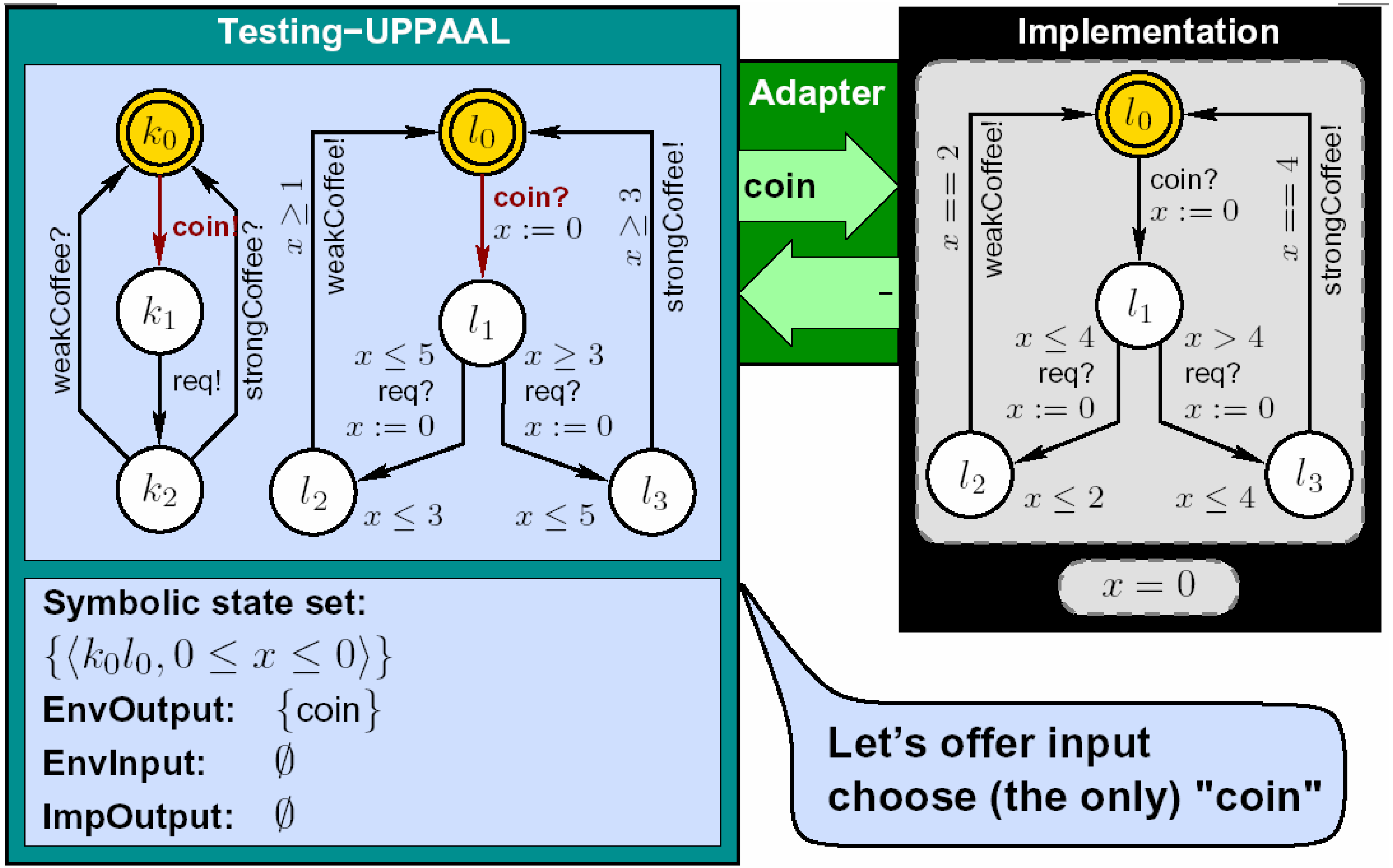
EnvOutput: $\{$coin$\}$

EnvInput: $\emptyset$

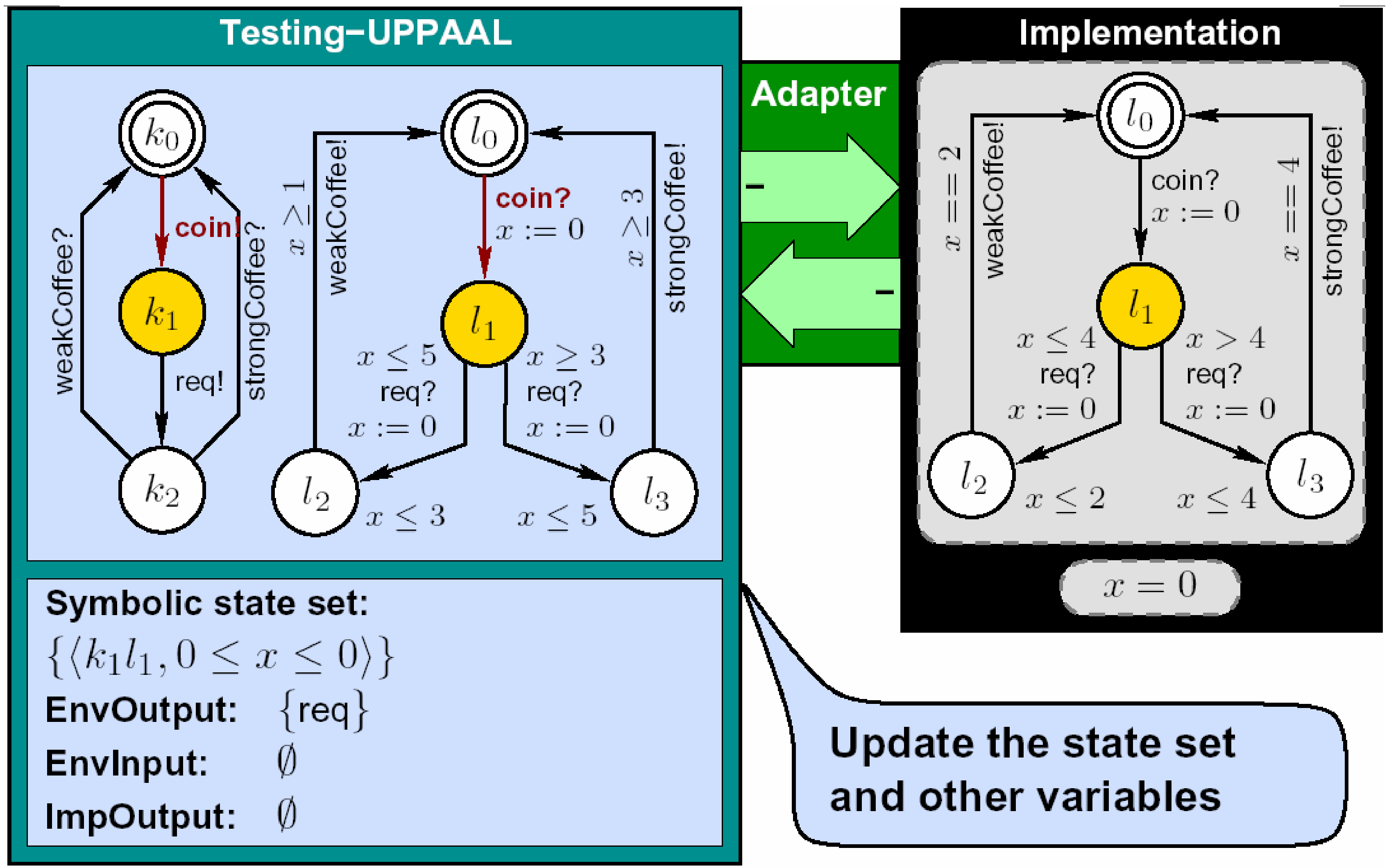
ImpOutput: $\emptyset$

Wait for output (delay) or offer input?

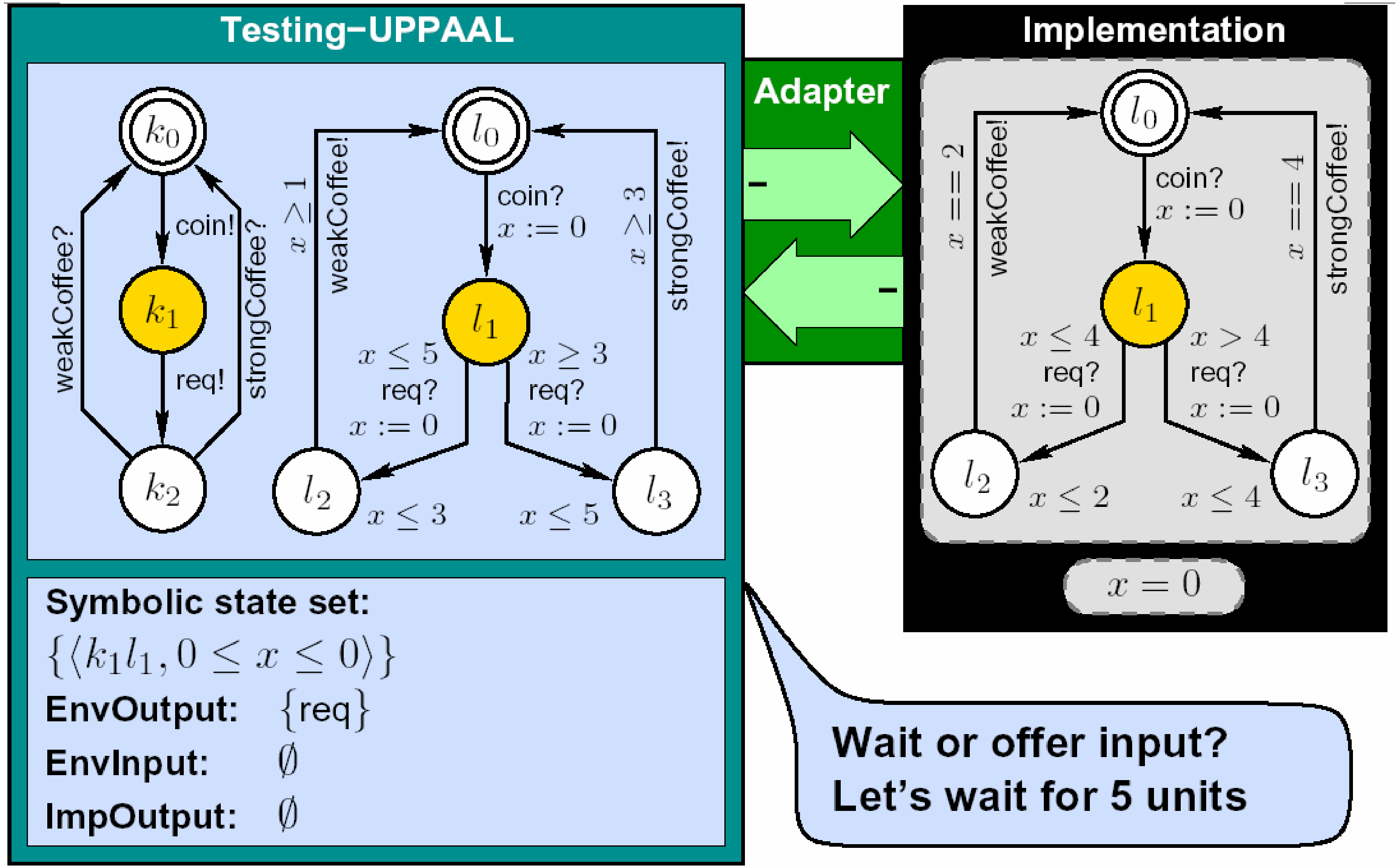
Online Testing



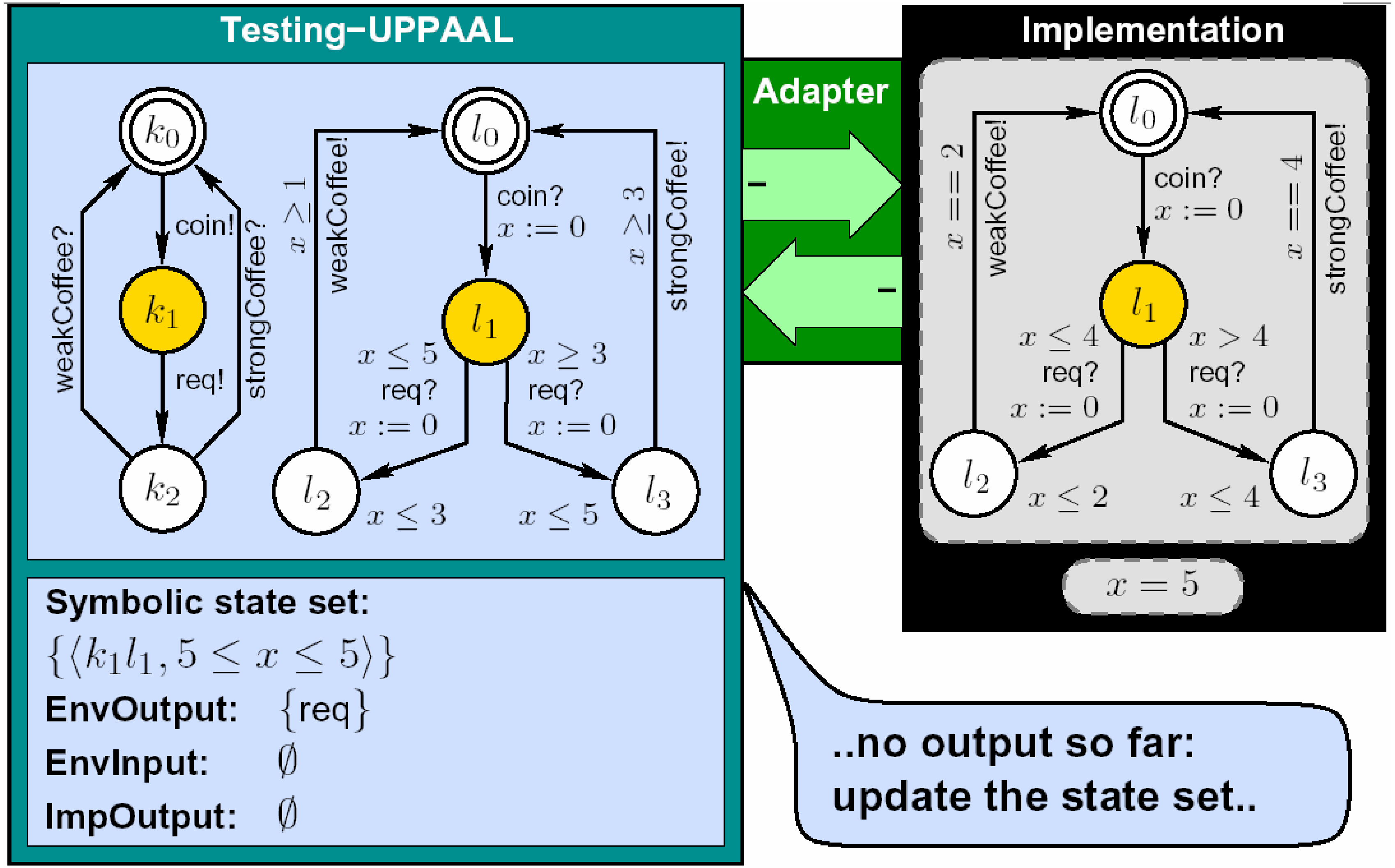
Online Testing



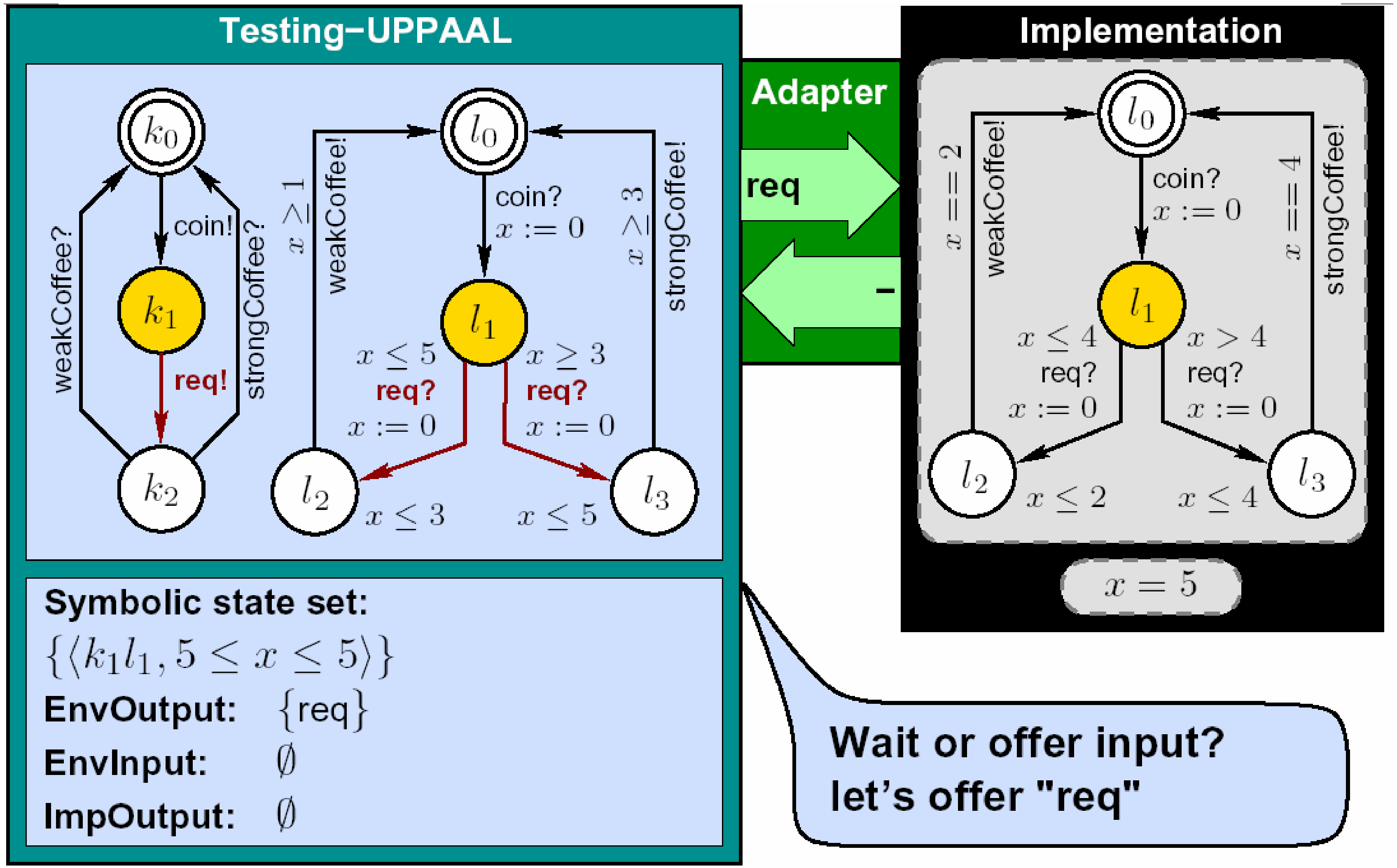
Online Testing



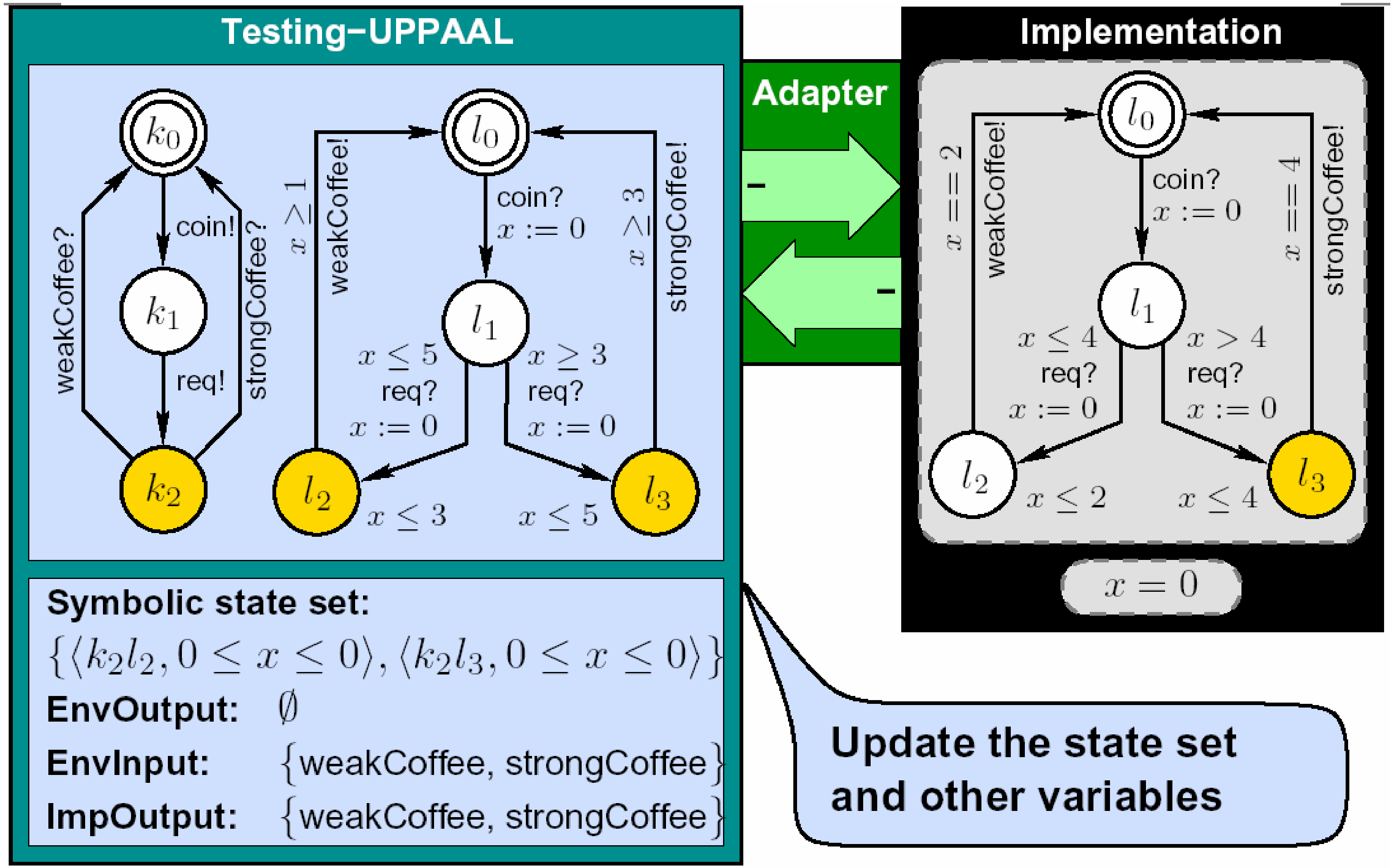
Online Testing



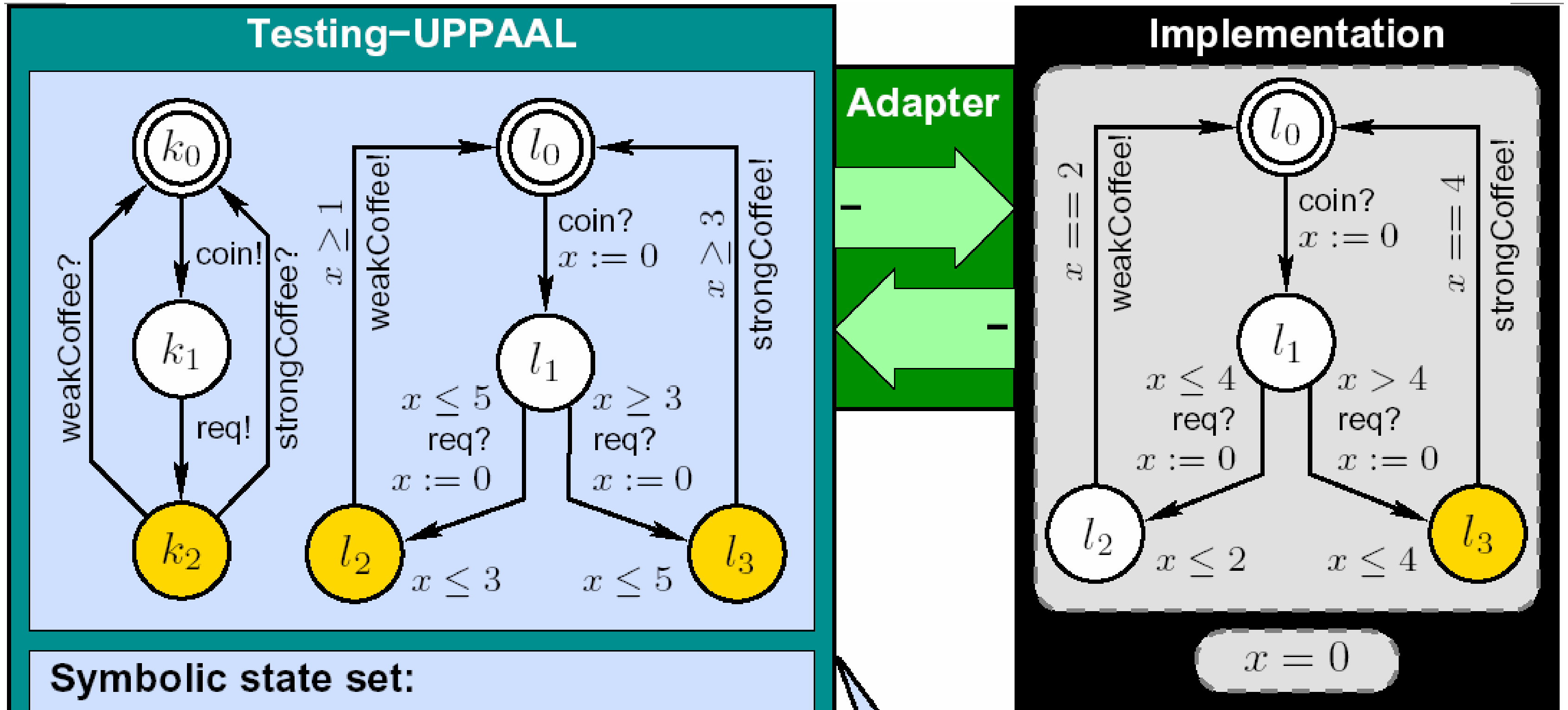
Online Testing



Online Testing



Online Testing



Symbolic state set:

$\{\langle k_2 l_2, 0 \leq x \leq 0\rangle, \langle k_2 l_3, 0 \leq x \leq 0\rangle\}$

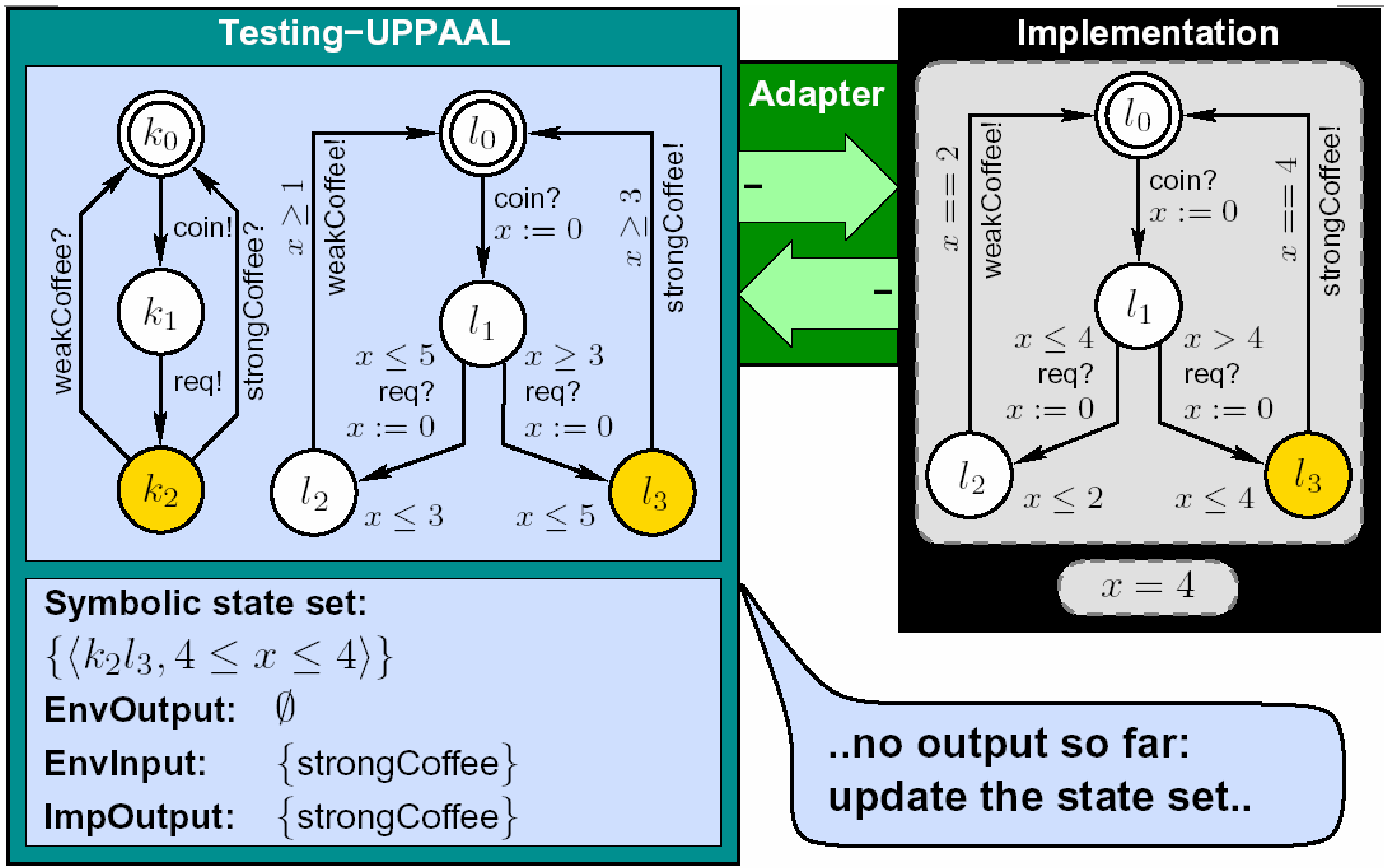
EnvOutput: $\emptyset$

EnvInput: {weakCoffee, strongCoffee}

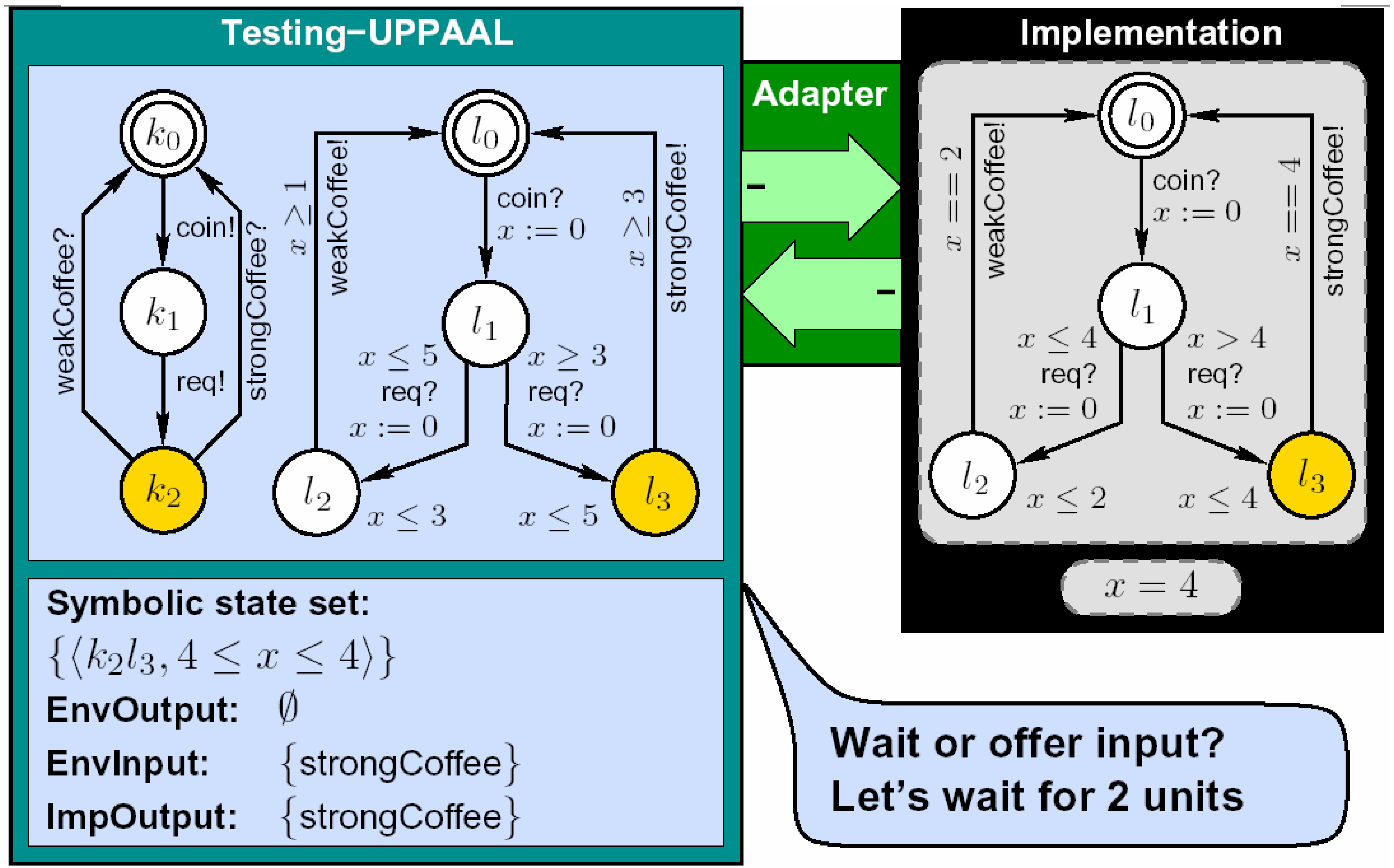
ImpOutput: {weakCoffee, strongCoffee}

Wait or offer input?
Let's wait for 4 units

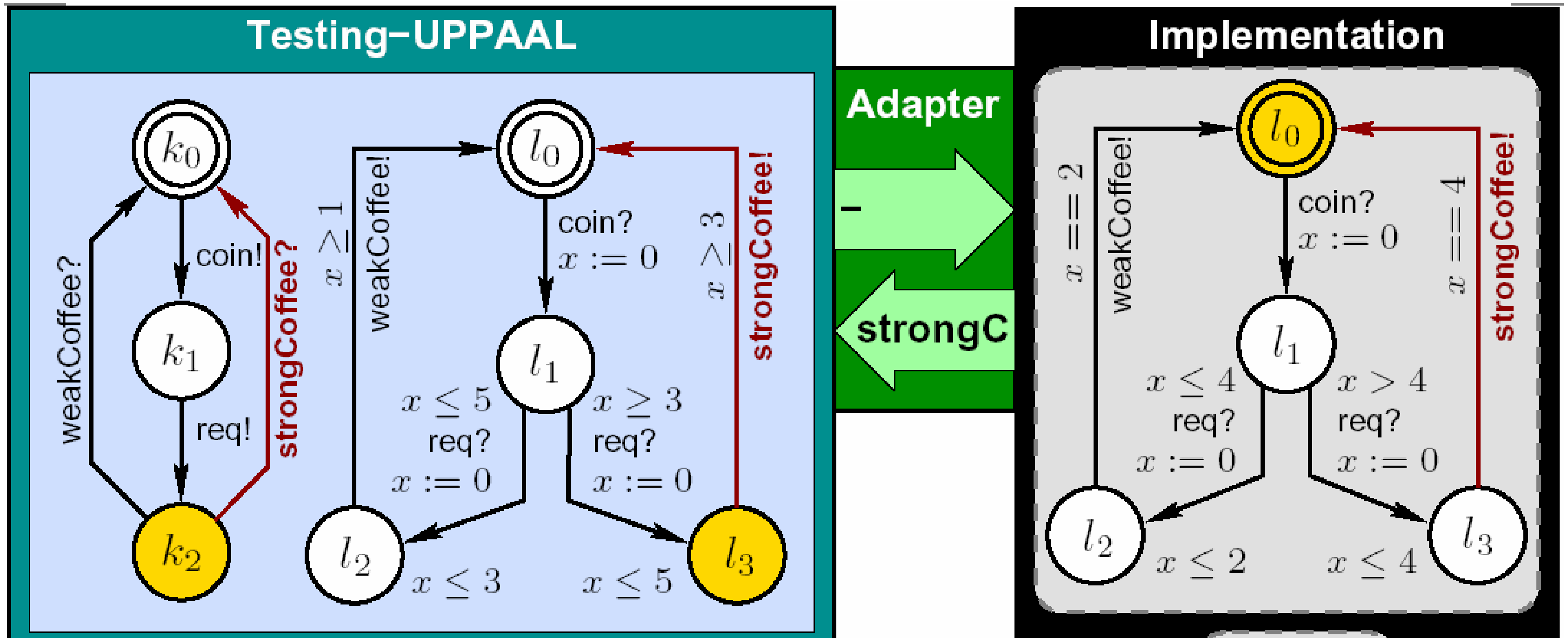
Online Testing



Online Testing



Online Testing



Symbolic state set:

$\{\langle k_2 l_3, 4 \le x \le 4\rangle\}$

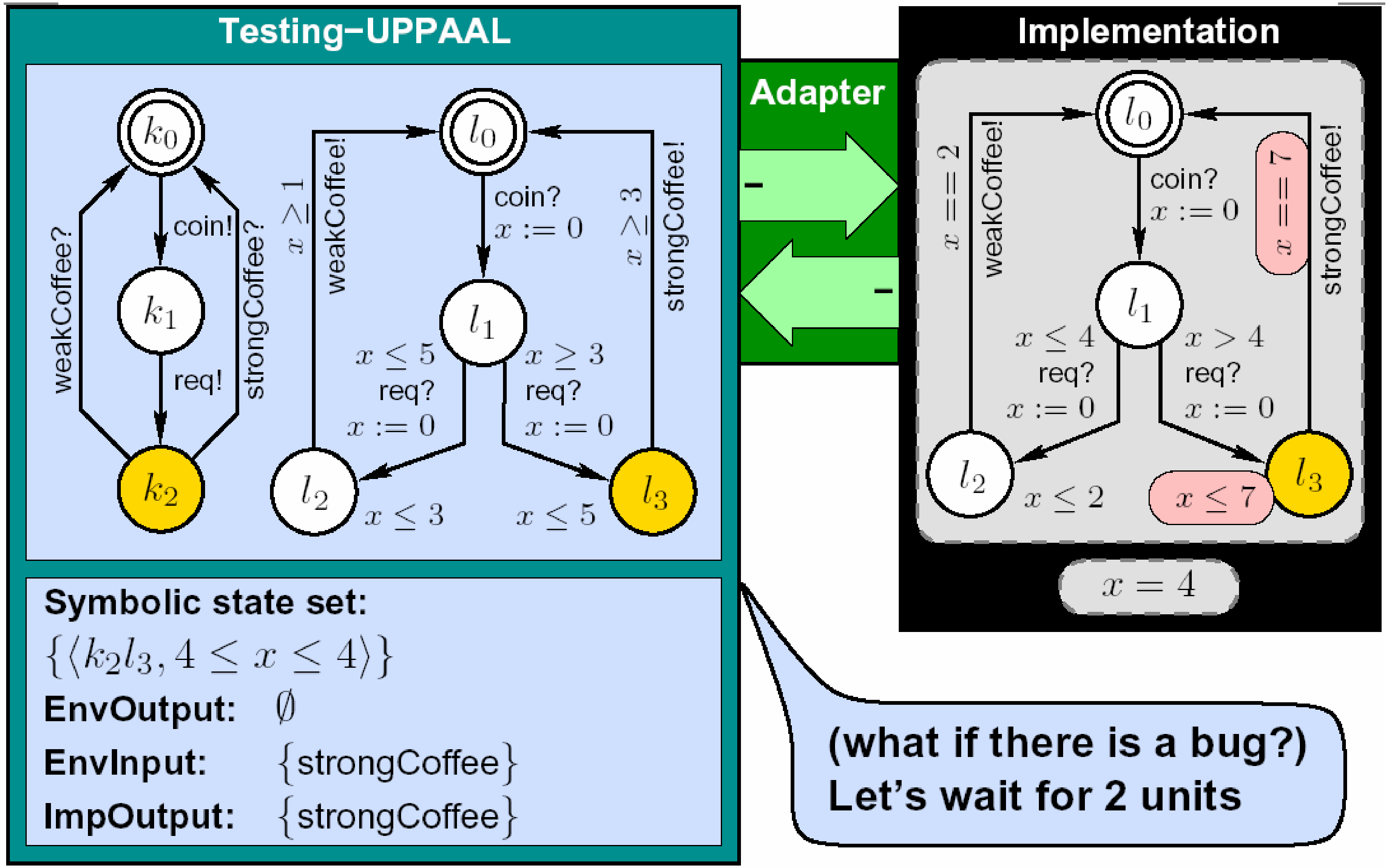
EnvOutput: $\emptyset$

EnvInput: {strongCoffee}

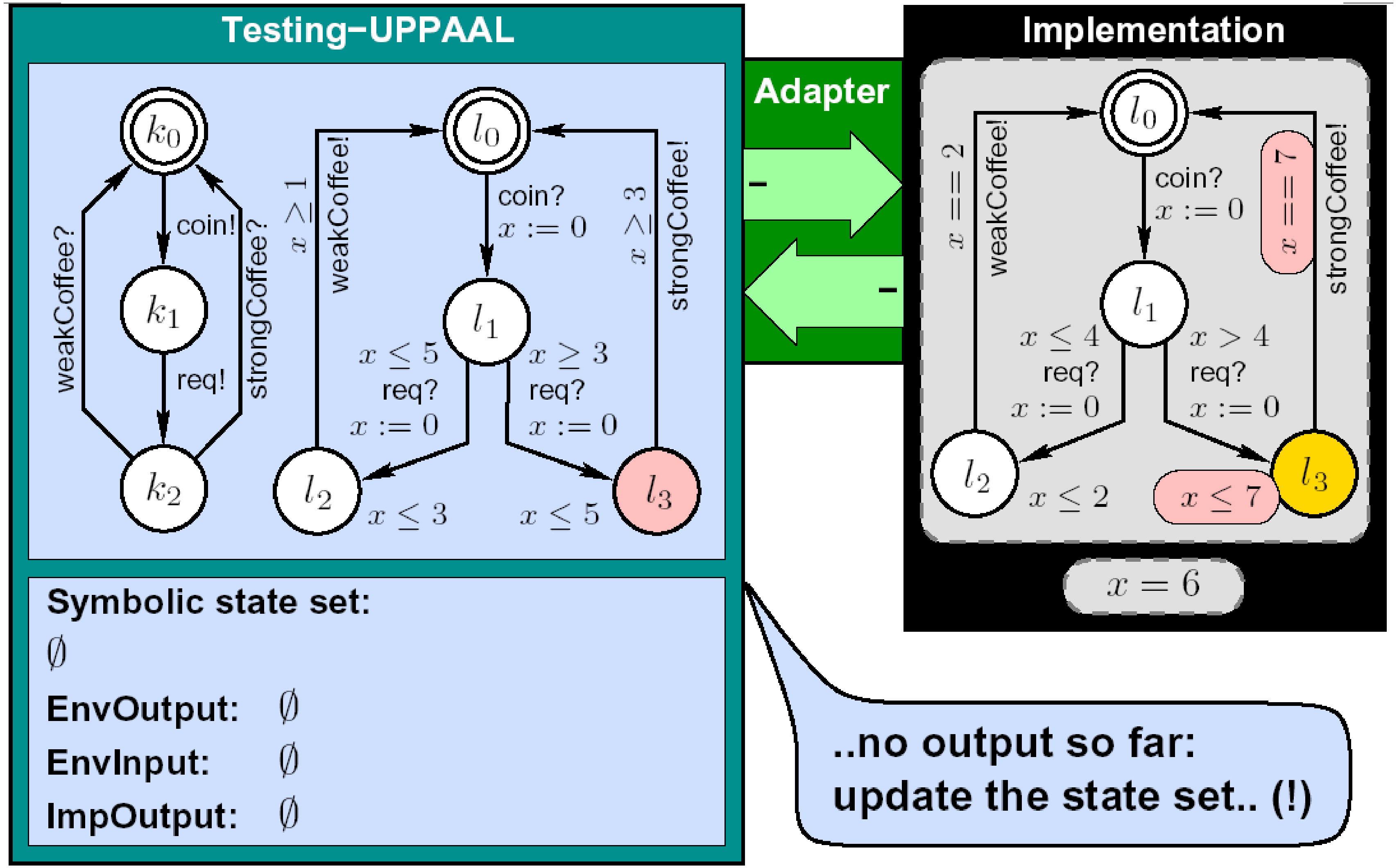
ImpOutput: {strongCoffee}

got output after 0 delay: update the state set

Online Testing



Online Testing



Danfoss EKC Case

Electronic Cooling Controller

Sensor Input

- air temperature sensor
- defrost temperature sensor
- (door open sensor)

Keypad Input

- 2 buttons (~40 user settable parameters)

Output Relays

- compressor relay
- defrost relay
- alarm relay
- (fan relay)

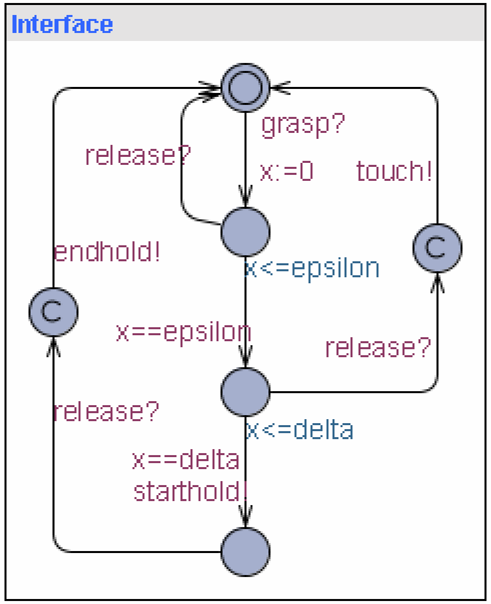
Display Output

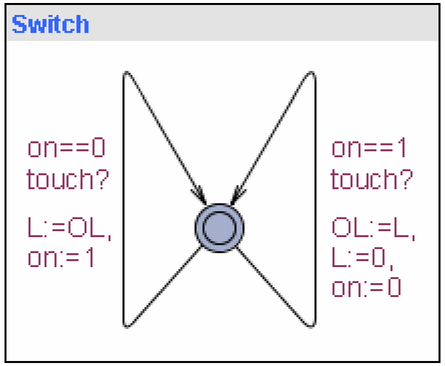
- alarm / error indication
- mode indication
- current calculated temperature

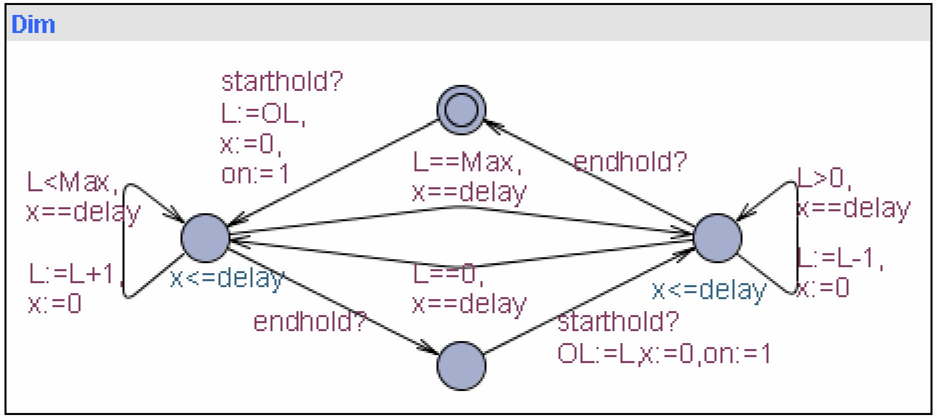
- Optional real-time clock or LON network module

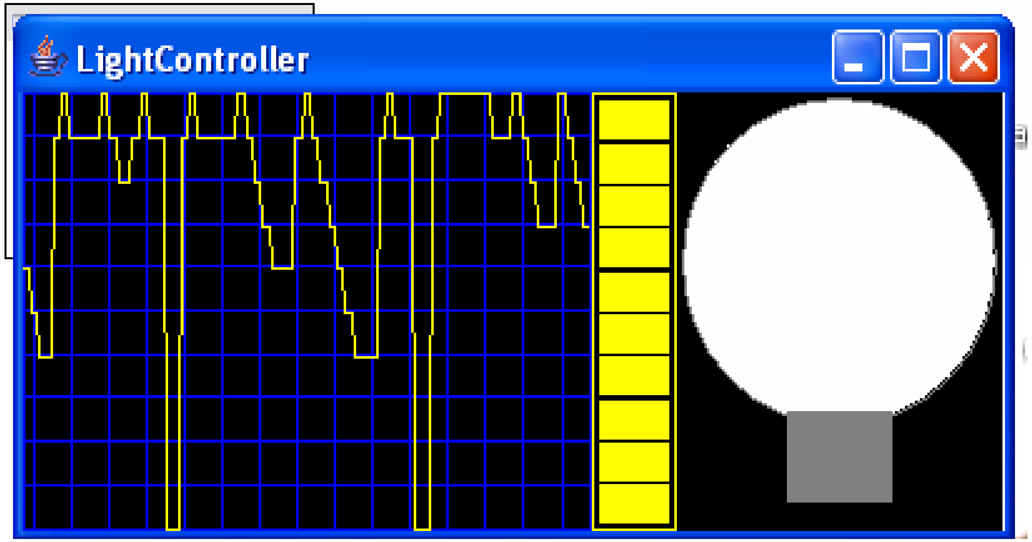
Demo

Touch-sensitive Light-Controller

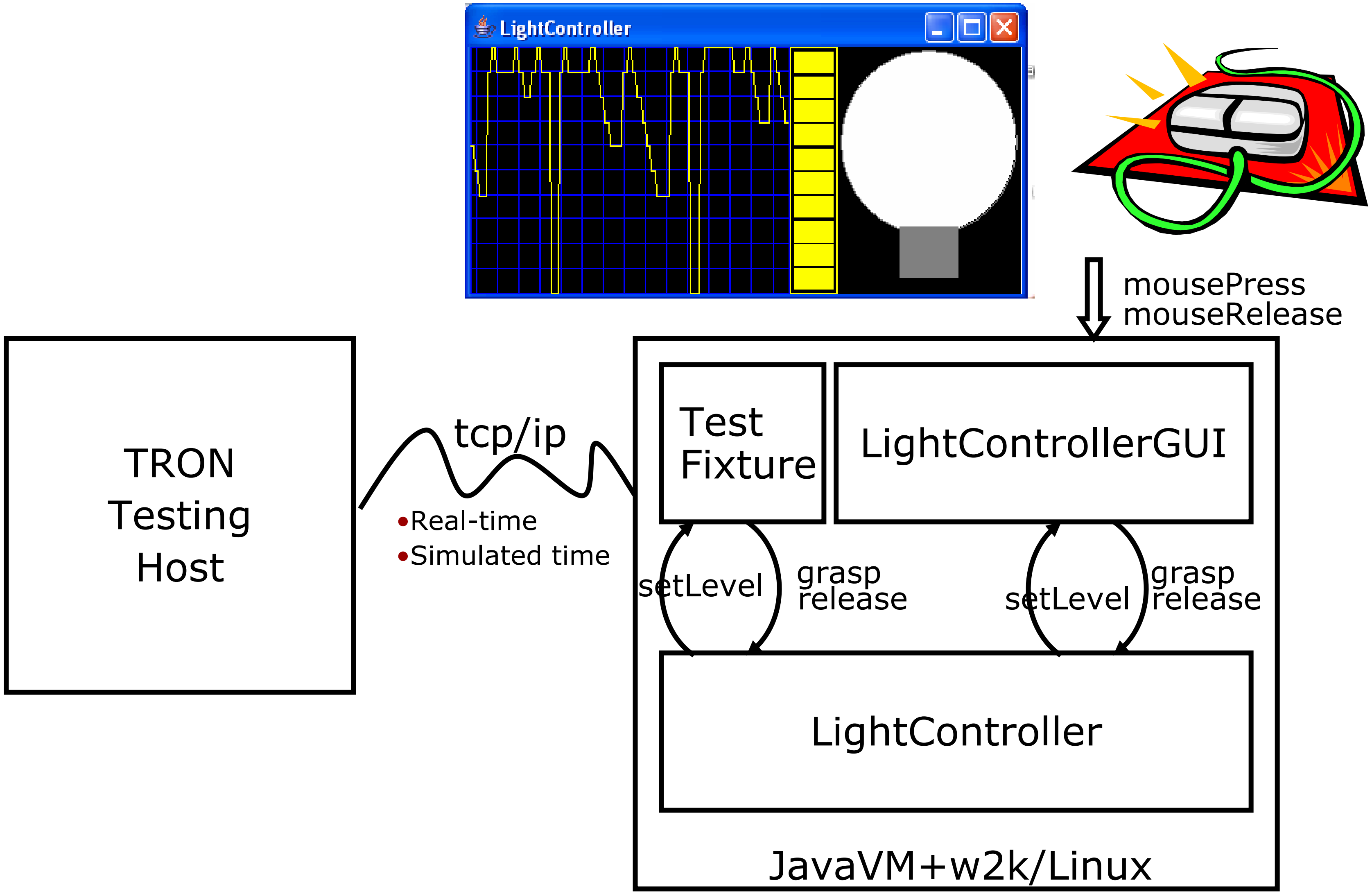








Test Setup


LightController
mousePress
mouseRelease
TRON
Testing
Host
tcp/ip
•Real-time
•Simulated time
Test
Fixture
LightControllerGUI
setLevel
grasp
release
setLevel
grasp
release
LightController
JavaVM+w2k/Linux

Mutants

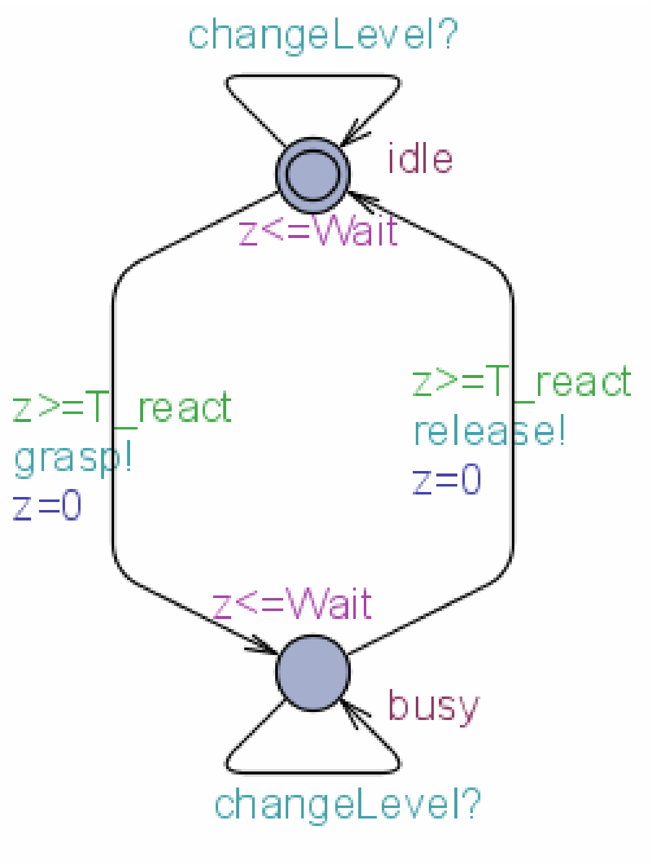
- Mutant: Non-conforming program version with a seeded error
  - M1 incorrectly implements switch

```
synchronized public void handleTouch() {
  if(lightState==lightOff) {
    setLevel(oldLevel);
    lightState=lightOn;
  }
   else { //was missing
 if(lightState==lightOn){
   oldLevel=level;
   setLevel(0);
   lightState=lightOff;
}
```

  - M2 violates a deadline

Environments

- Fast user has Treact=2
- Slow has Treact 550 > delta
  - (cannot trigger touch)

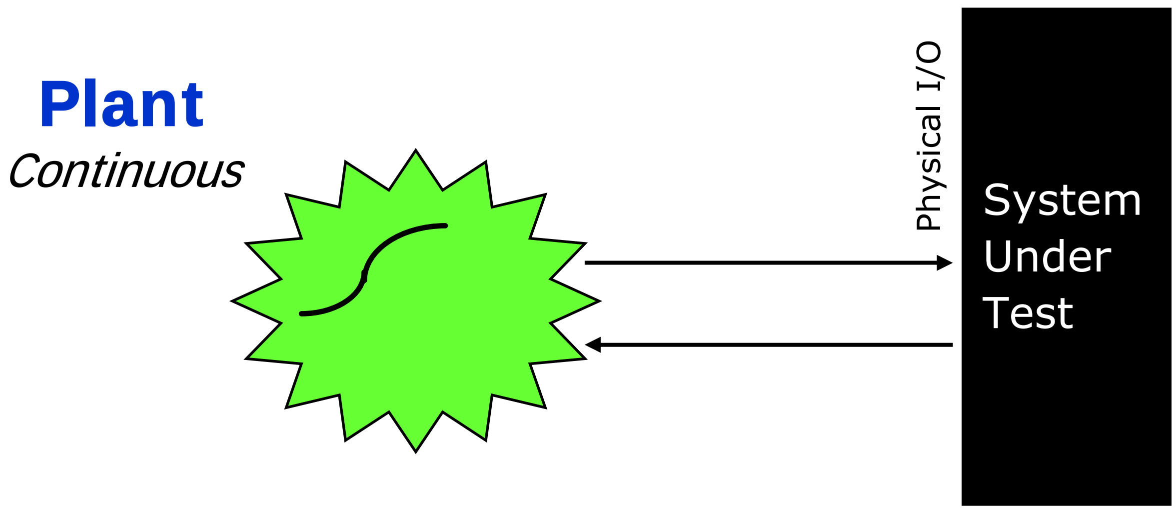


Testing, Monitoring, Simulation, Environment Emulation

BRICS
Basic Research in Computer Science

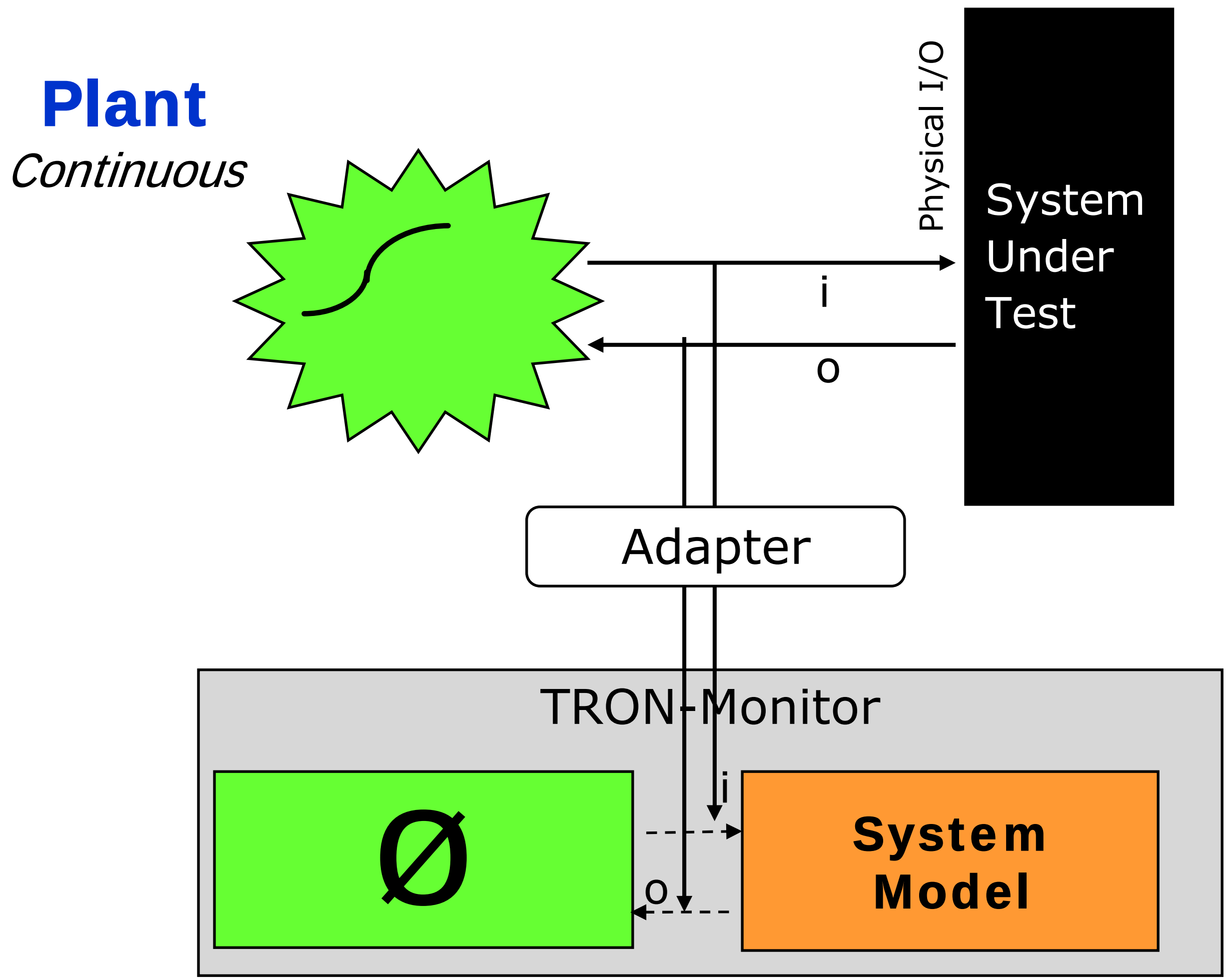
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Real System



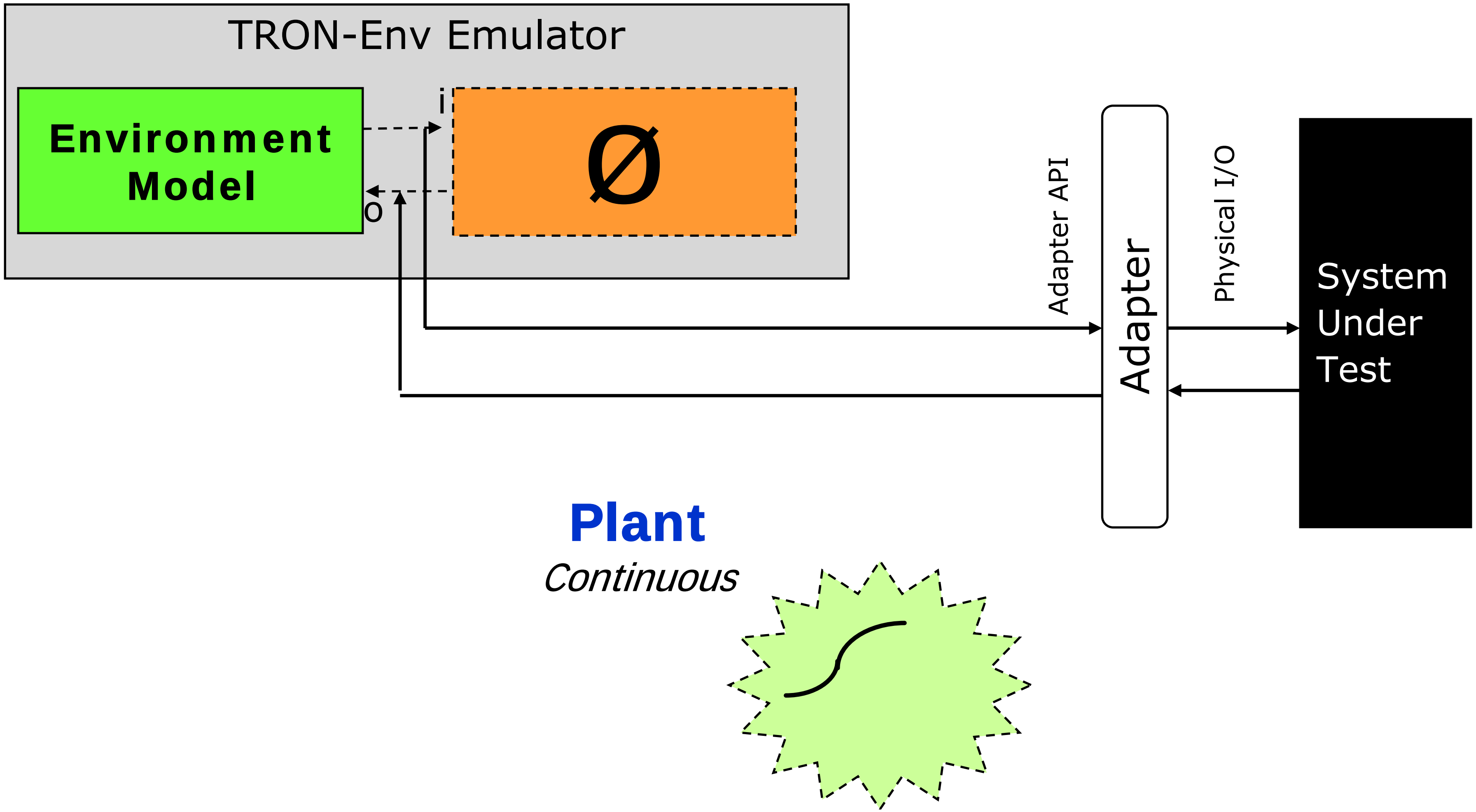
Monitoring

- Passively listen and check observed trace
- Aka Passive Testing

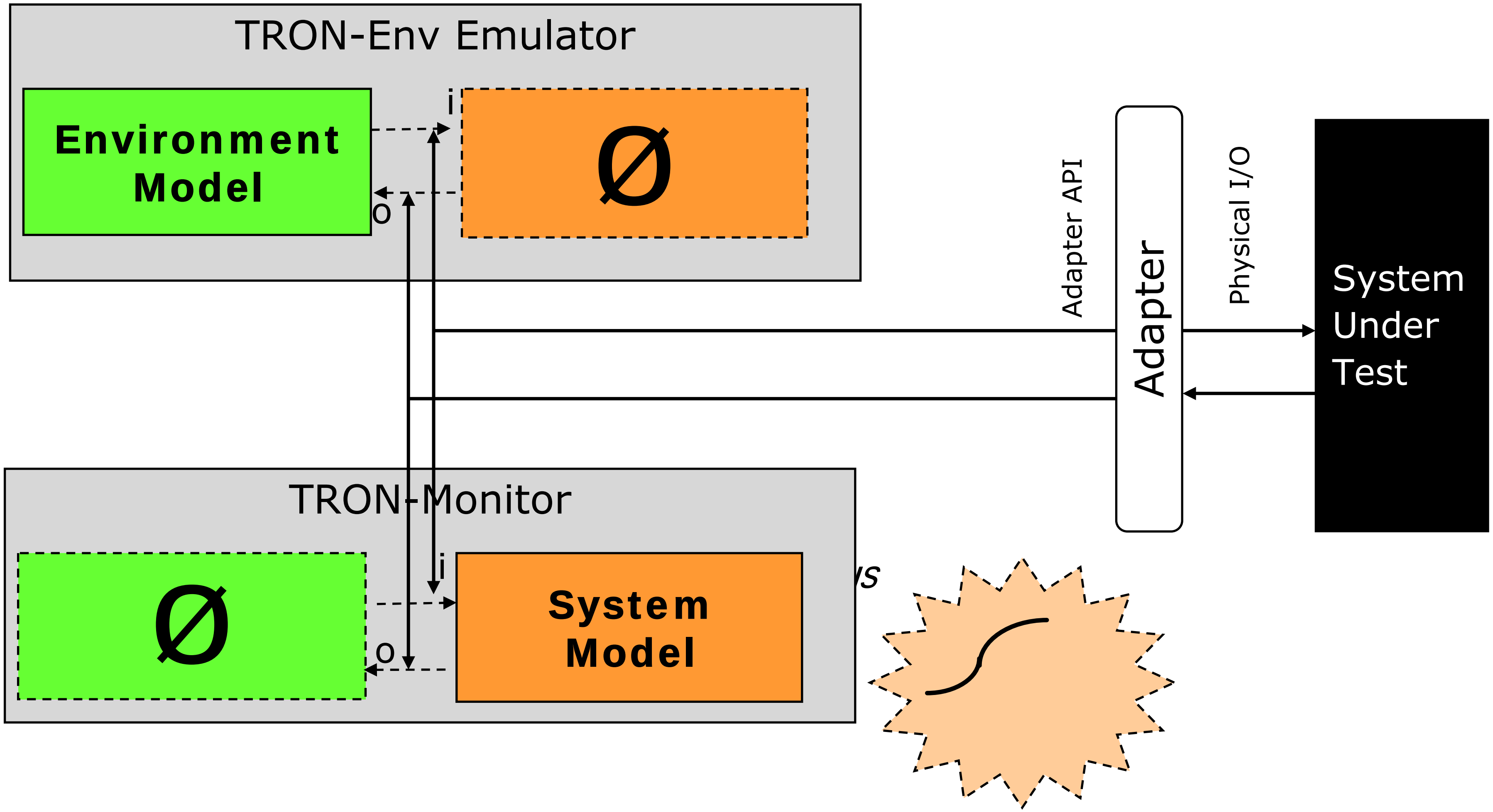


Environment Emulation

- Do not evaluate IUT behavior

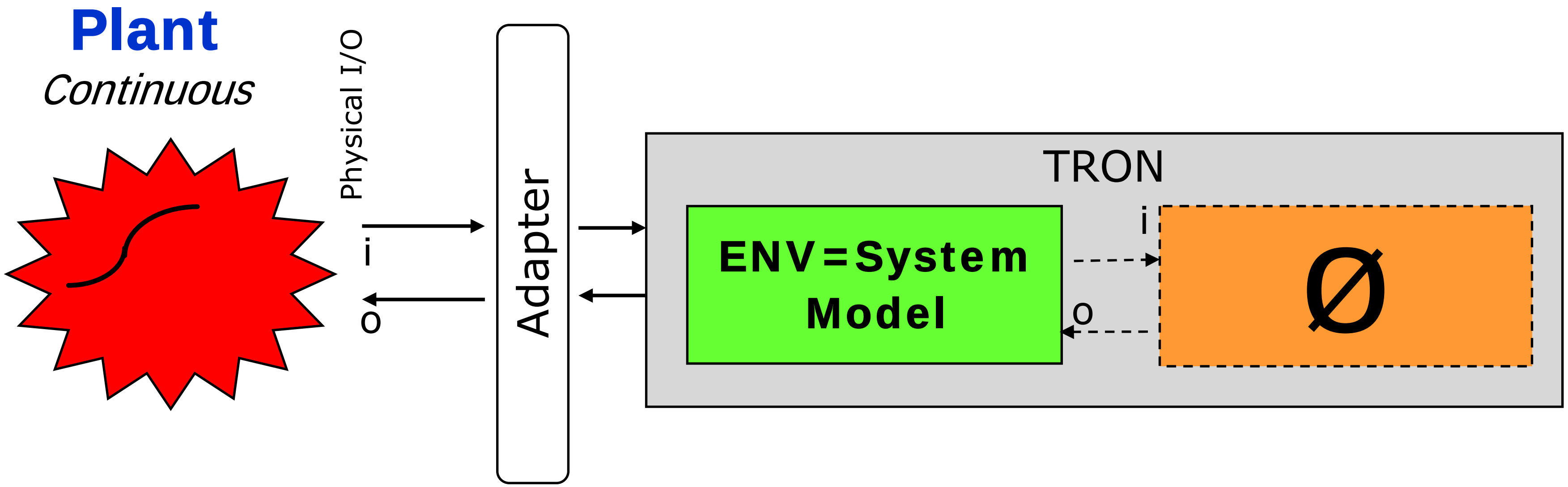


Testing = Environment Emulation + Monitoring

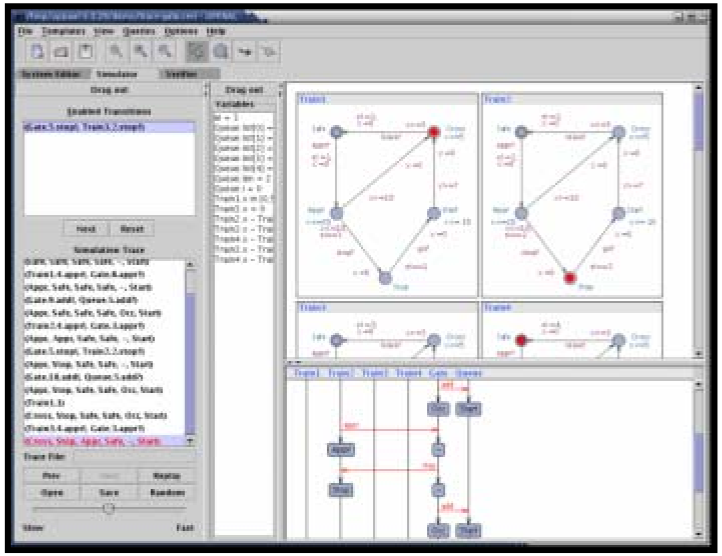


Simulator / prototype

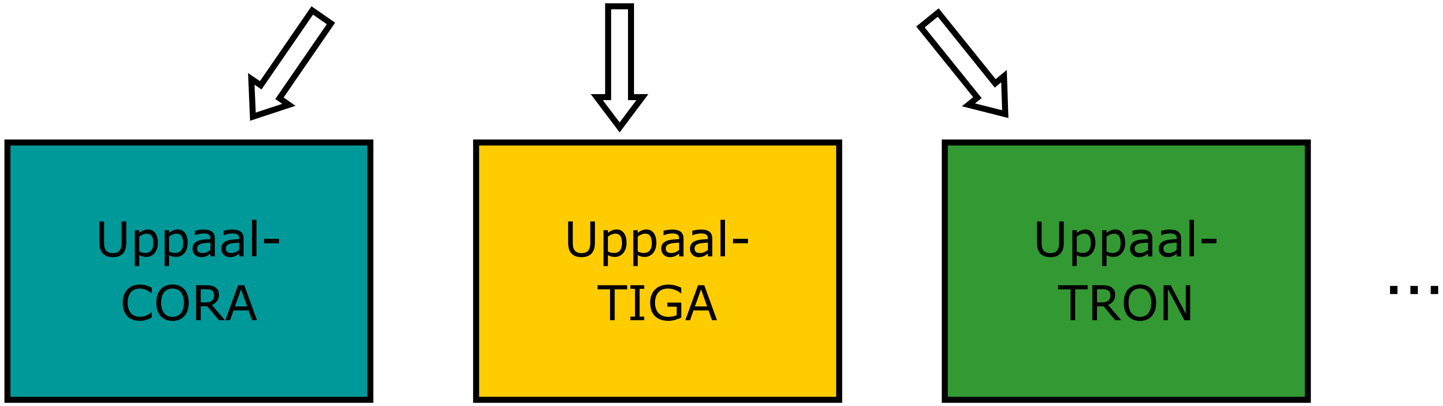
- Use implementation model as environment
- Use TRON as interpreter



UPPAAL Tools



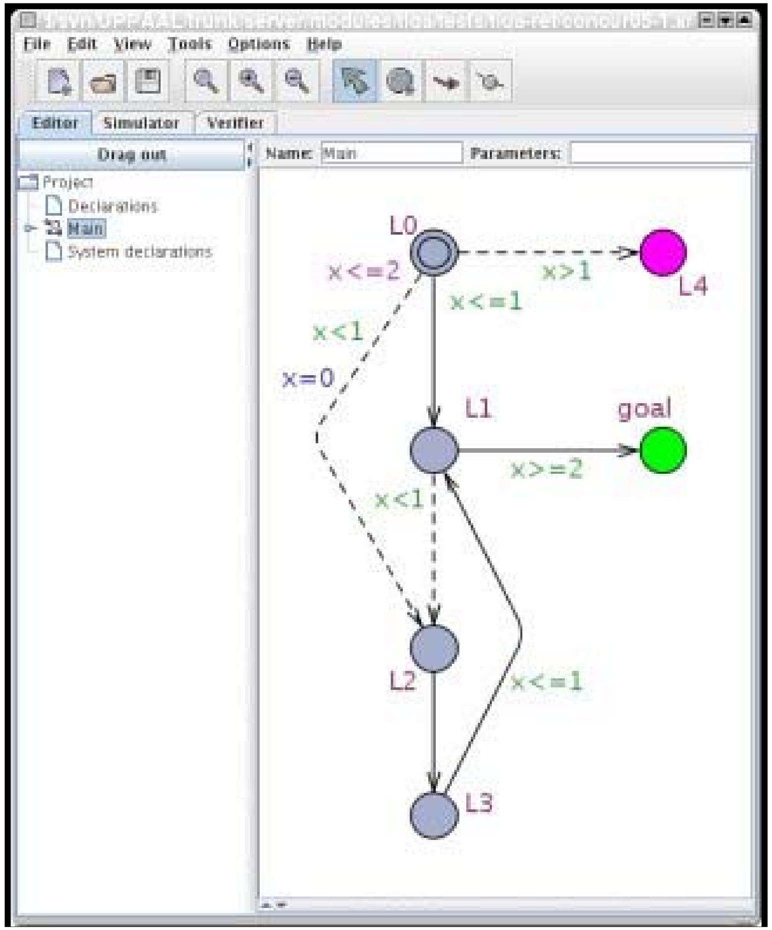
Efficient reachability analysis of network of timed automata



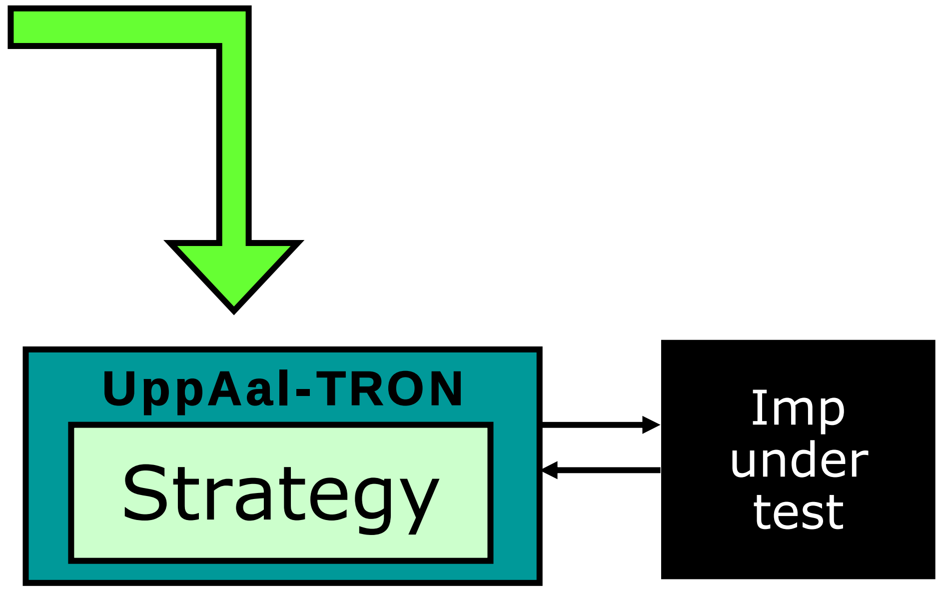
- TIGA: Timed games (reachability and safety)
- CORA: Cost Optimal reachability from priced TA
- TRON: Testing Real-time Online

Games and Testing

- UPPAAL-TIGA: analysis of timed game automata wrt. reachability and safety.
- Explicit observation objective, eg., get StrongCof



Possibly or Definitely (Winning) Strategy



Conclusions

- Explicit Environment Modeling
  - Realism and guiding
  - Separation of concerns
  - Modularity
  - Creative tool uses
  - Theoretical properties
- Real-Time Online testing from timed automata is feasible, but
  - Both theoretically and technically very challenging
  - Many open research issues

Related Work

- Formal Testing Frameworks
  - [Brinksma, Tretmans]
- Real-Time Implementation Relations
  - [Khoumsi'03, Briones'04]
- Symbolic Reachability analysis of Timed Automata
  - [Dill'89, Larsen'97,...]
- Online state-set computation
  - [Tripakis'02]
- Online Testing
  - [Tretmans'99, Peleska'02, Krichen'04]

END

Basic Research in Computer Science

CENTER FOR INDLEJREDE SOFTWARE SYSTEMER