Model Based Testing of Embedded Systems

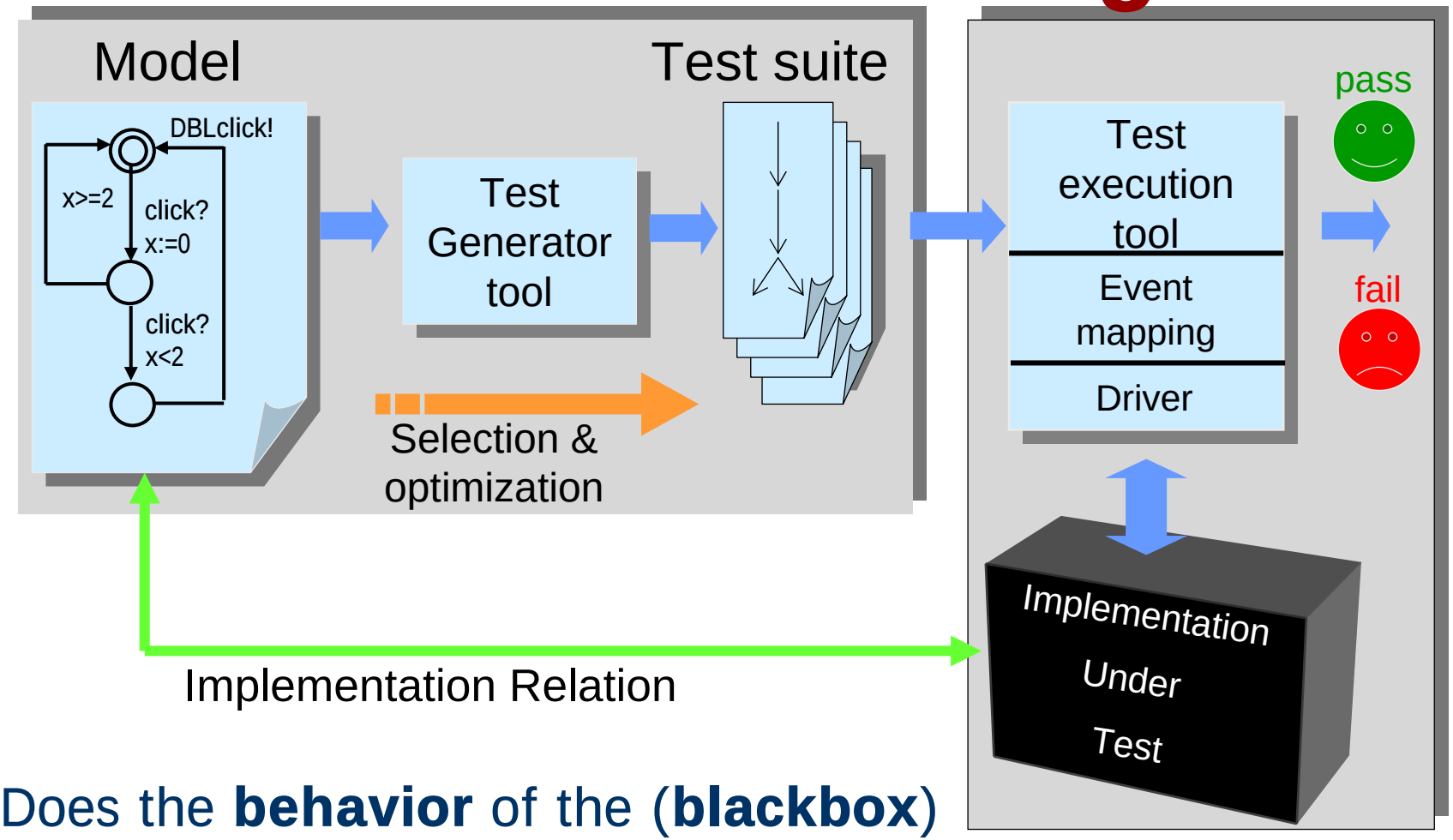
Brian Nielsen

Arne Skou

{bnielsen | ask}@cs.auc.dk

BRICS
Basic Research
in Computer Science

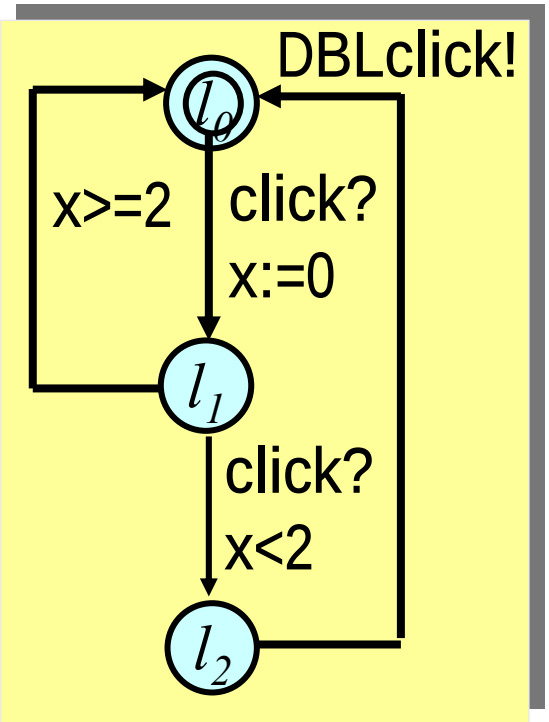
Automated Model Based Conformance Testing



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

Specification Based Testing

A Specification



Test cases

Click!
Wait 1.5
click!
DBLClick?

(pass)

Click!
Wait 5
click!
DBLClick?

(fail)

Click!
Wait 0.1
click!
DBLClick?

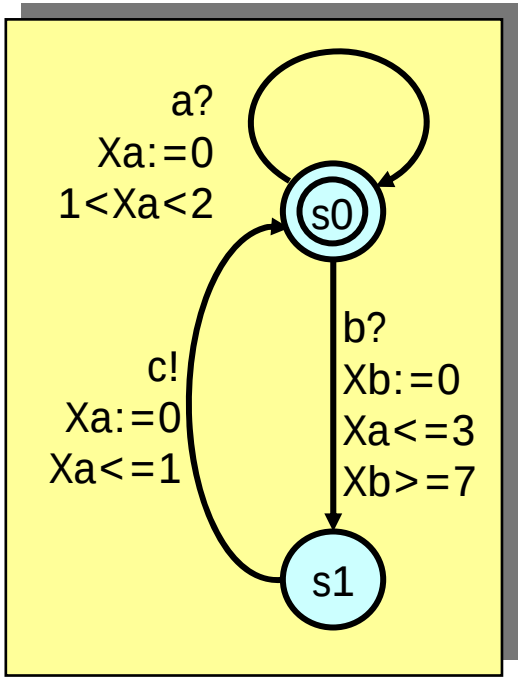
(pass)

Timed Automaton = FSM + Clocks (dense) + Guards + Resets

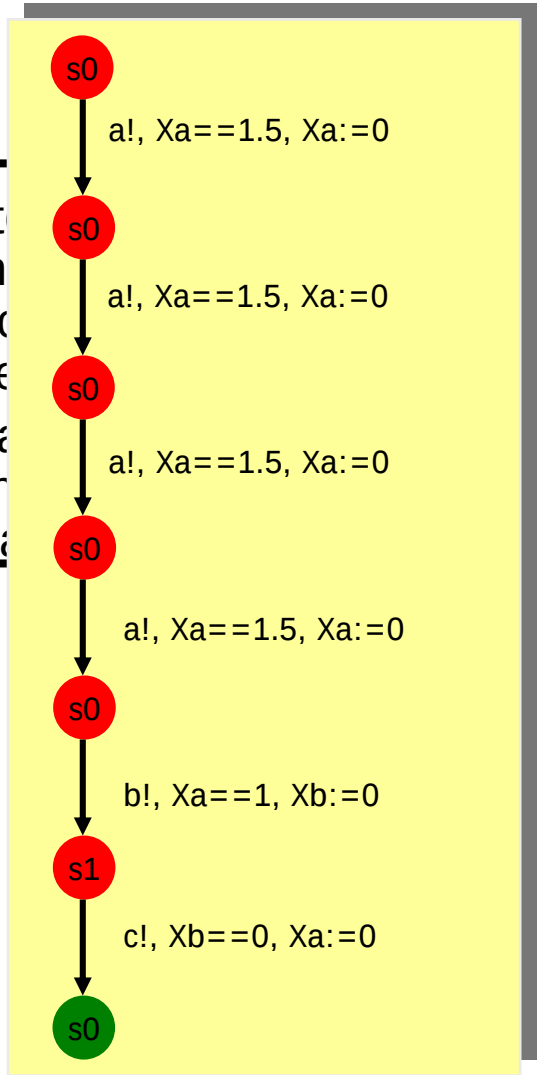
- $x_1 \sim c$, $x_1 - x_2 \sim c$, where $\sim \in \{<,\leq,\geq,>\}$
- $x:=c$
- Semantic state: $(l,\bar{u})$ $\quad (l_1, x = 1.17)$

Exercises

Simple TA-specification



Generated test case



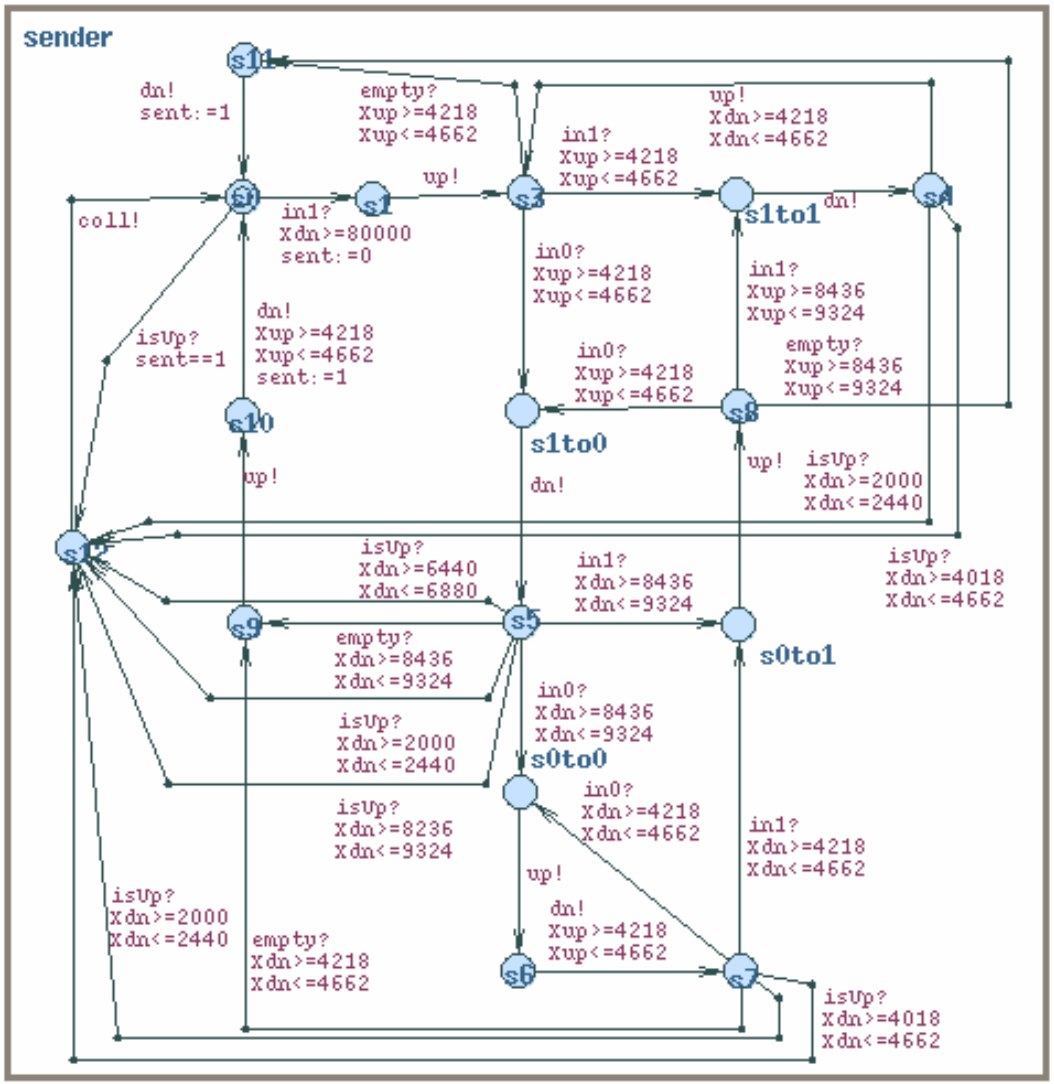
1) comput
trace th
that a "c
produce

2) How ma
you thin
necessa

12 test cases are generated!

How do we cope with real-life specs?

Philips Sender with collision detection

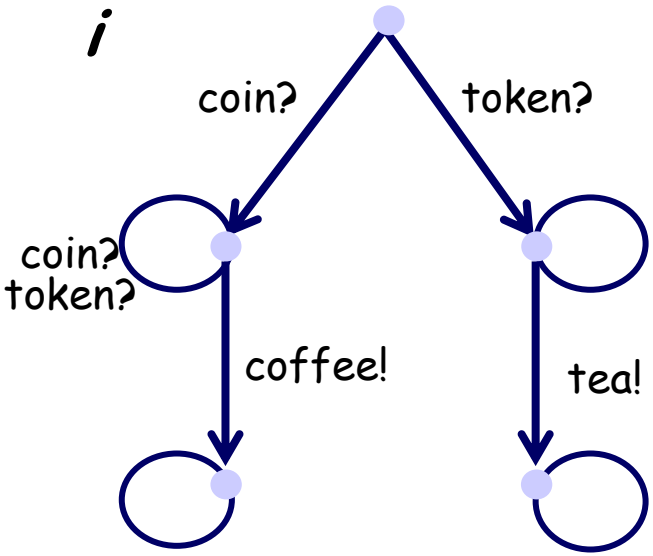



```
config
  system sender;
  int sent;
  observable in1, in0, up, dn, empty, isUp, coll;
```

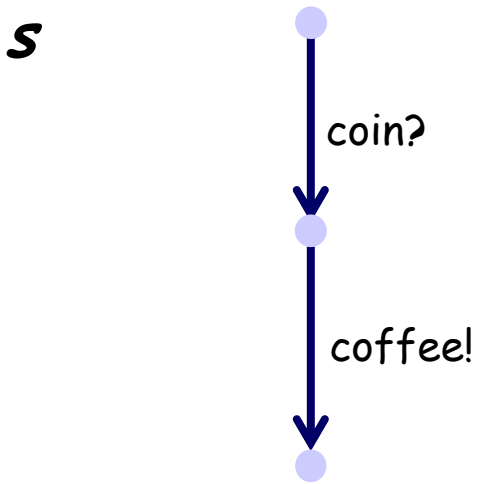
Correctness Criteria

BRICS
Basic Research
in Computer Science

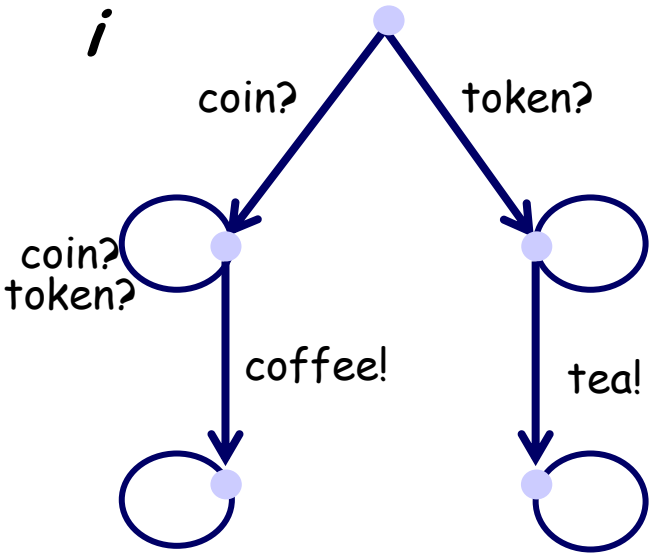
I conforms-to S ??



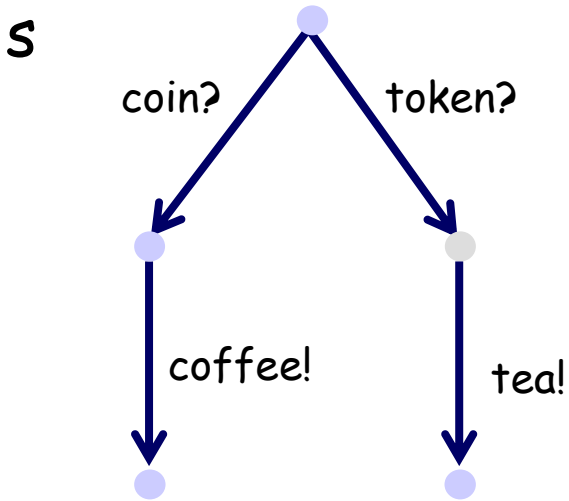
ioco



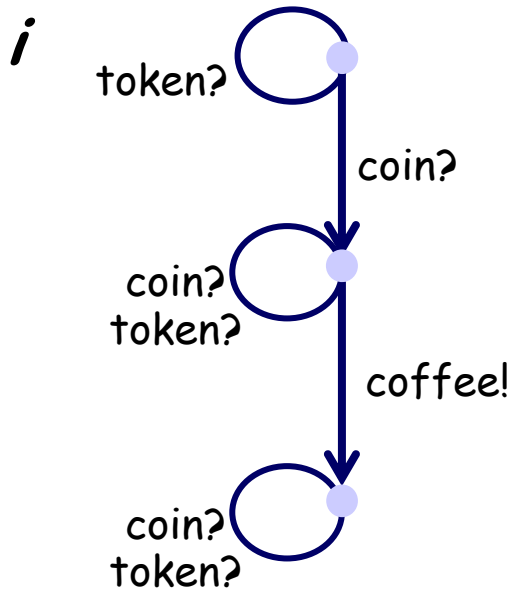
I conforms-to S ??



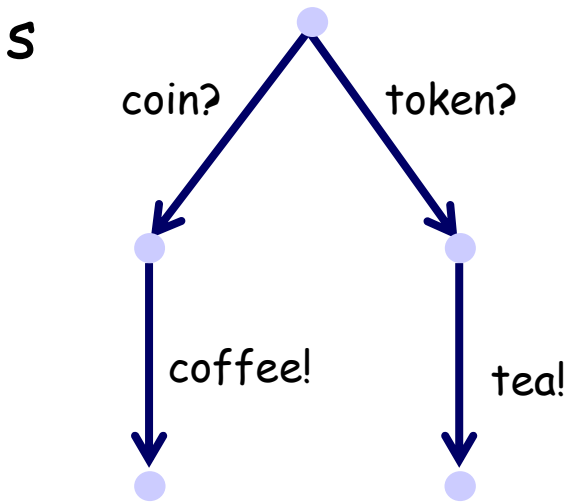
ioco



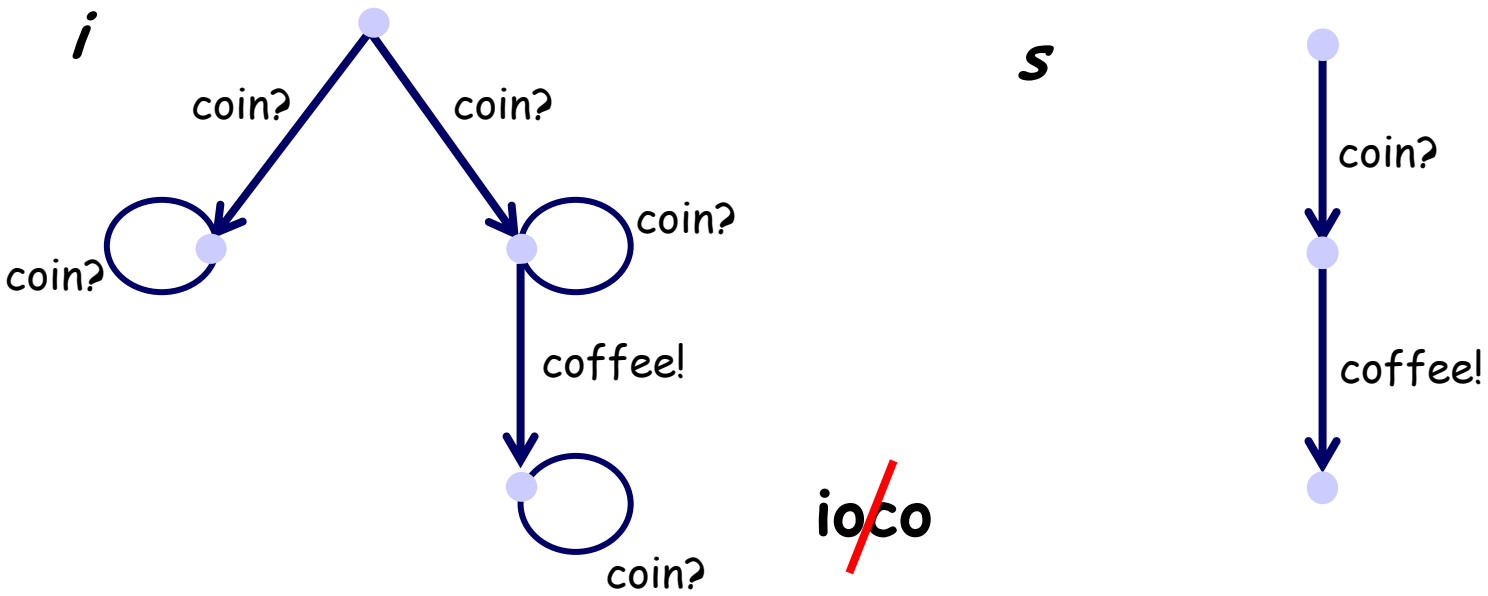
I conforms-to S ??



~~ioco~~



I conforms-to S ??



Implementation Relation ioco

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

$$p\ \mathbf{after}\ \sigma \;=\; \{\, p' \mid p \overset{\sigma}{\Longrightarrow} p' \,\}$$

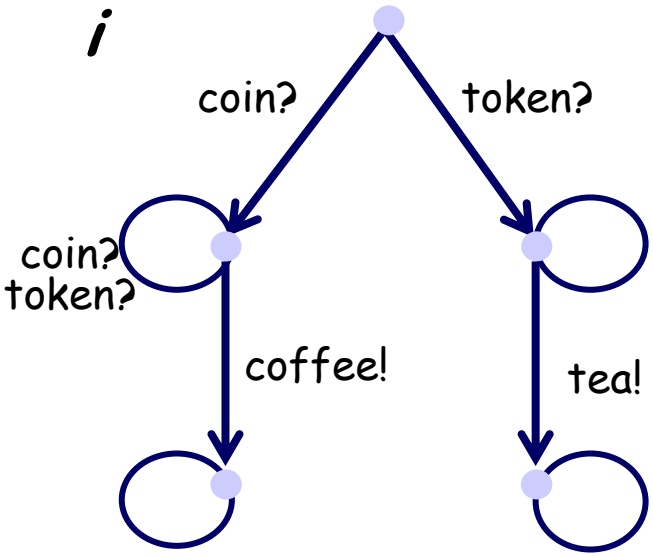
$$p \xrightarrow{\delta} p \;=\; p \xrightarrow{L_U} p \;=\; \forall\ x! \in L_U \cup \{\tau\} : p \not\xrightarrow{x!}$$

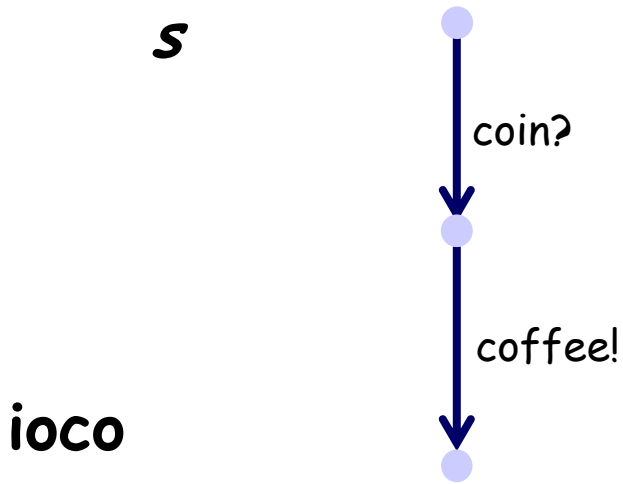
$$\mathit{out}(P) \;=\; \{\, x! \in L_U \mid p \xrightarrow{!x} ,\ p \in P \,\} \cup \{\, \delta \mid p \xrightarrow{\delta} p,\ p \in P \,\}$$

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

Implementation Relation ioco

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





ioco

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

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

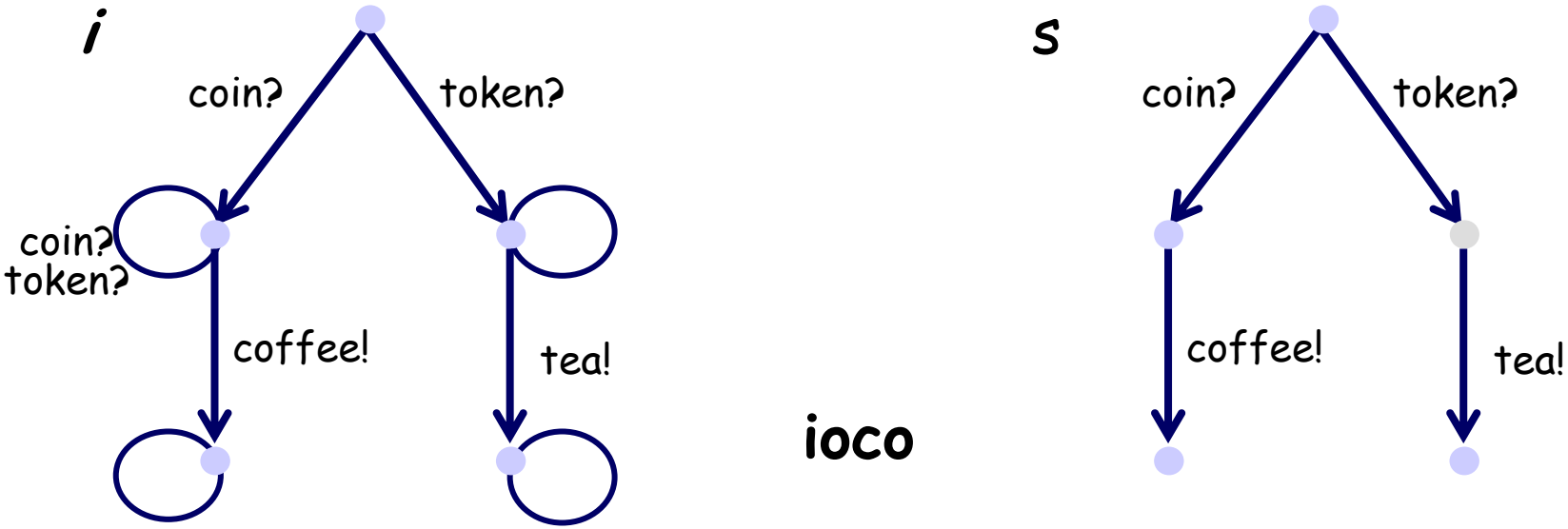
$\mathit{out}(s \textbf{ after } \text{coin?}) = \{ \text{coffee!} \}$

$\mathit{out}(s \textbf{ after } \text{token?}) = \varnothing$

But $\text{token?} \notin \mathit{Straces}(s)$

Implementation Relation ioco

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



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

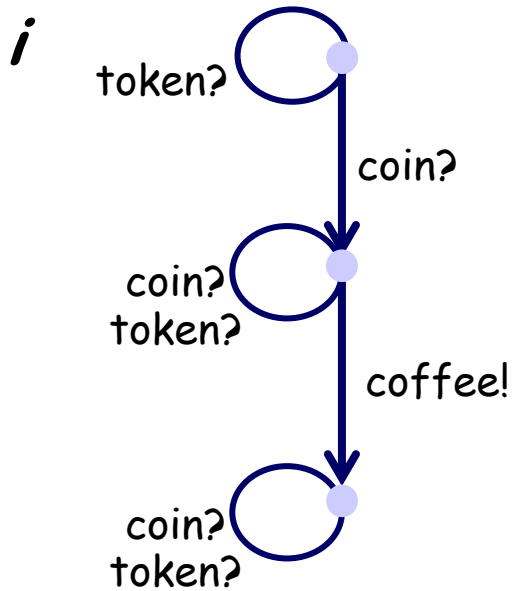
out (*i* **after** token?) = { tea! }

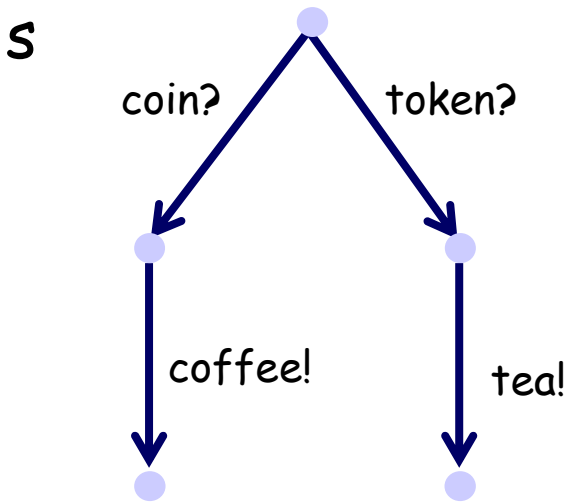
out (*s* **after** coin?) = { coffee! }

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

Implementation Relation ioco

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



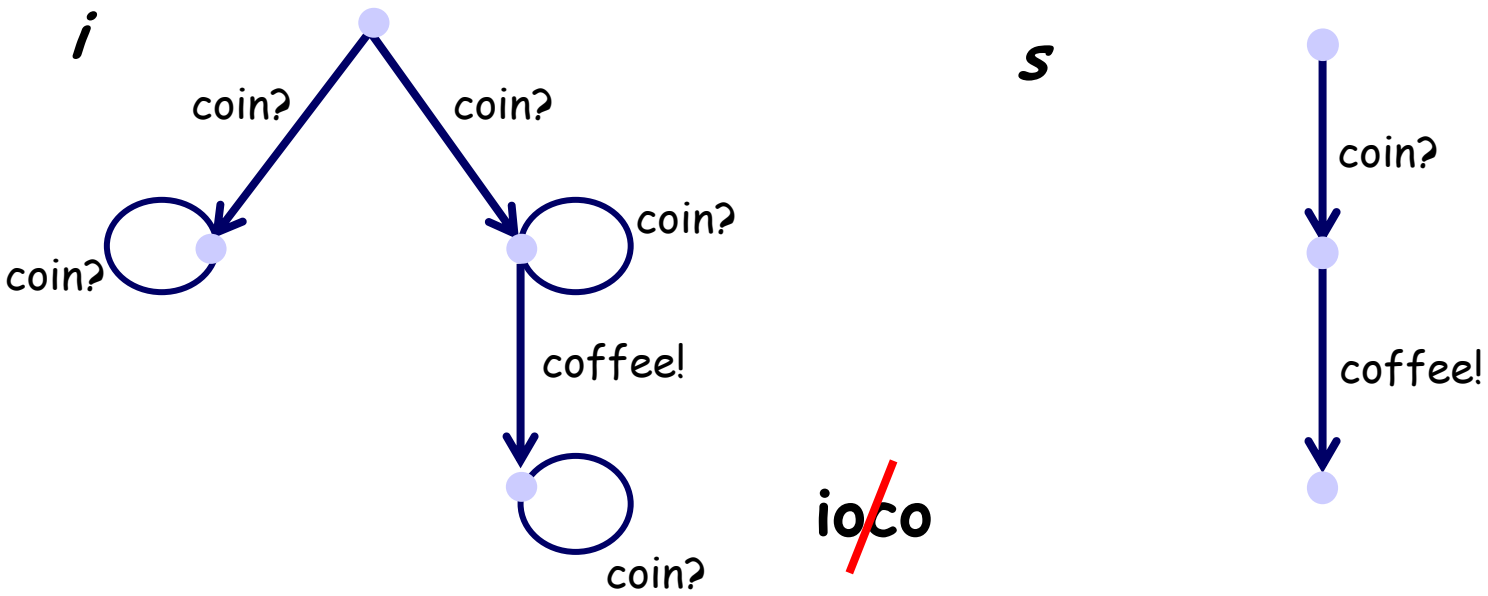


$\mathit{out}(i \textbf{ after } \text{token?}) = \{ \delta \}$

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

Implementation Relation ioco

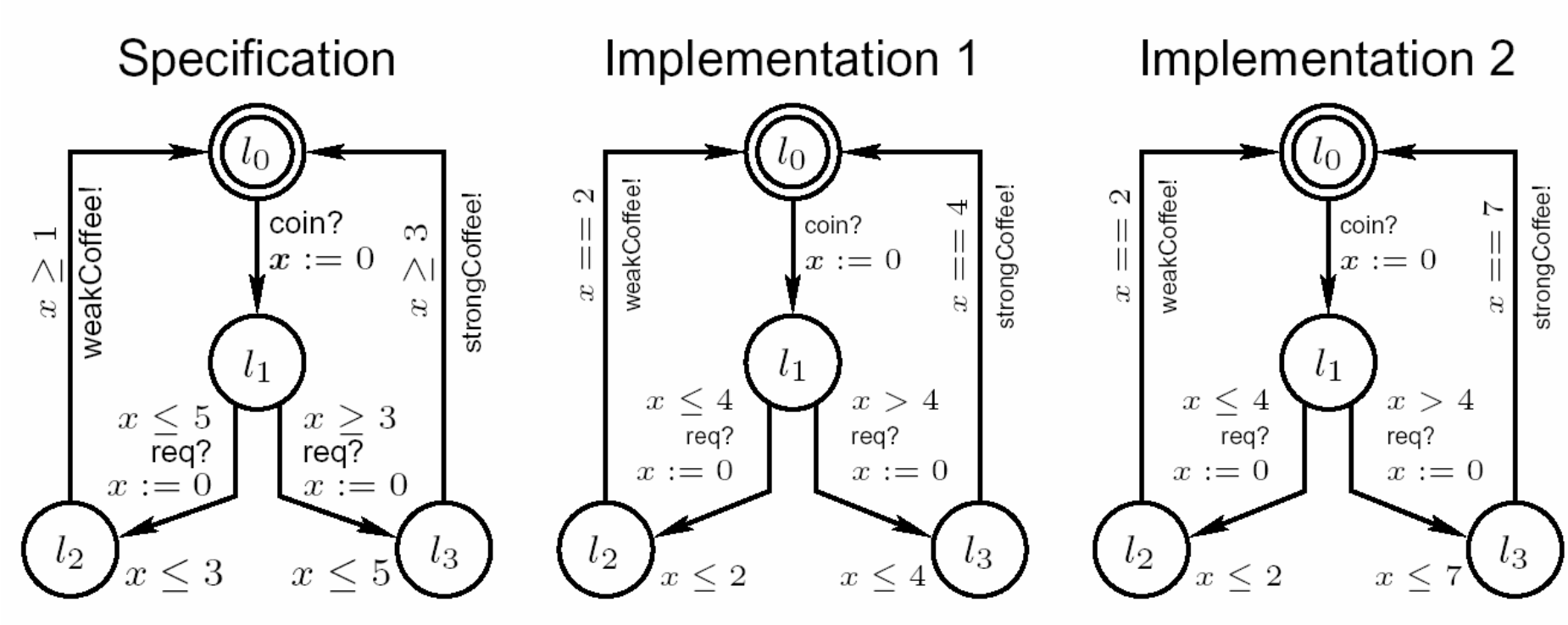
$$i \textbf{ ioco } s \;=_{def}\; \forall \sigma \in \mathit{Straces}(s) : \; \mathit{out}(i \textbf{ after } \sigma) \subseteq \mathit{out}(s \textbf{ after } \sigma)$$



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

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

Timed Conformance??



Example Traces

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

- c?.2.r?.2.weakC
- c?.5.r?.7

I1 **rt-ioco** S

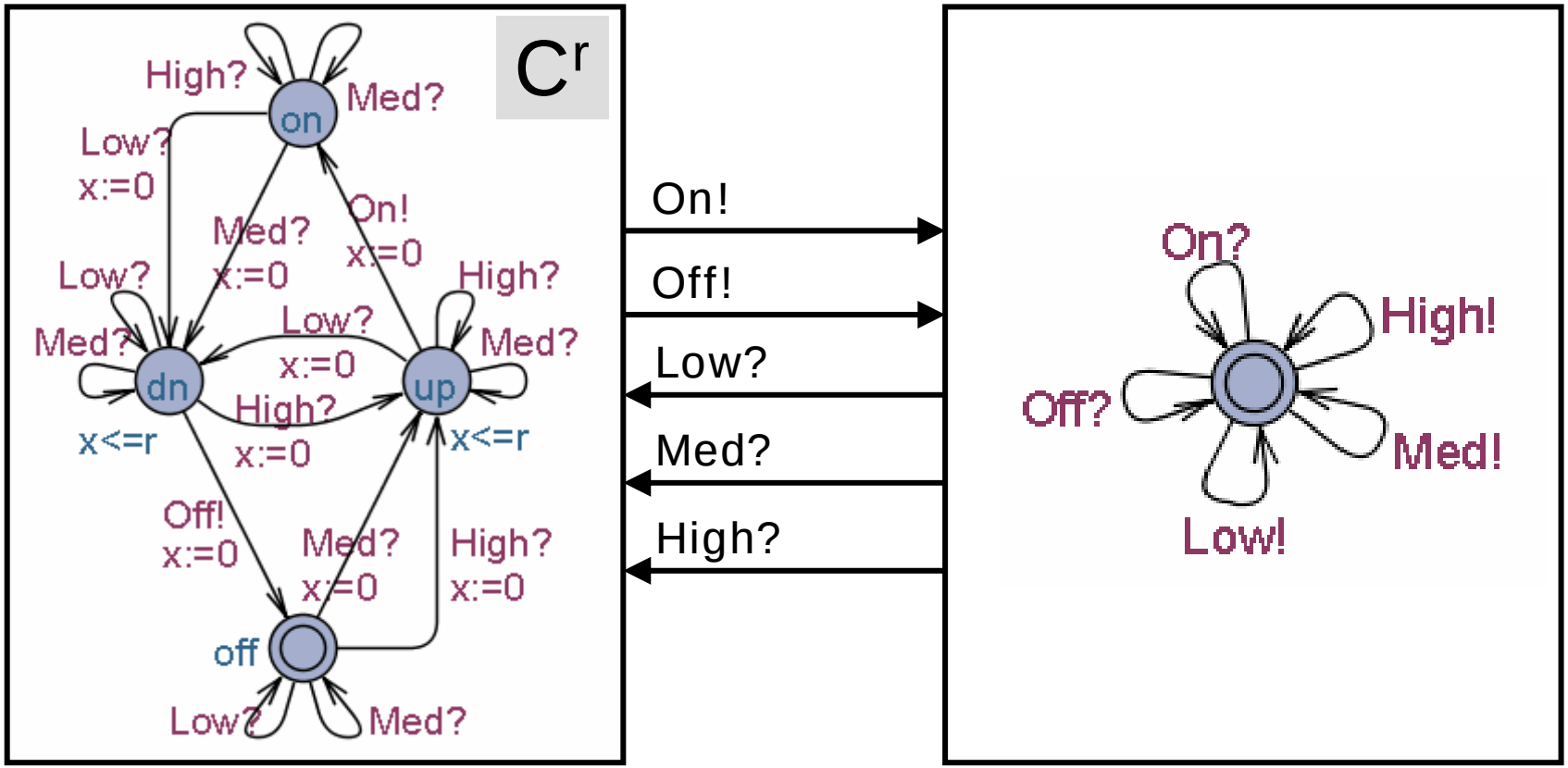
I2 ~~**rt-ioco**~~ S

Real-Time Conformance

- **TTr**(s): the set of *timed traces* from s
  - eg.: σ = `grasp?·50·release?·50`
- **Out**(s after σ) = possible outputs and delays after σ
  - eg. **Out** `(interface after grasp?.220)={touch!,0..270}`
- **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**
  - **never produces illegal output, and**
  - **always produces required output in time**

Sample Cooling Controller

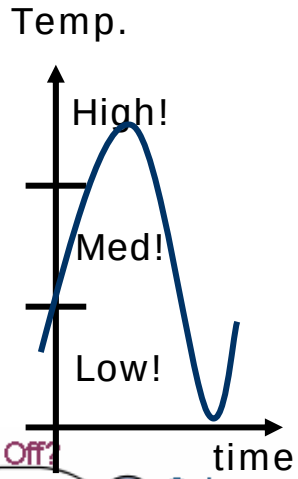


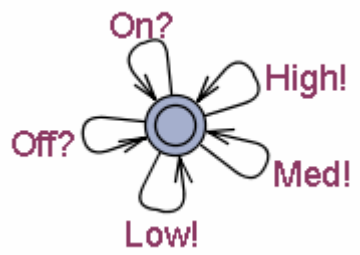
IUT-model

Env-model

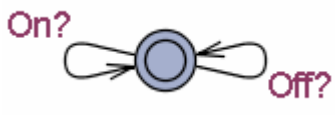
Env. Modeling

- Realism and Guiding
  - 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)

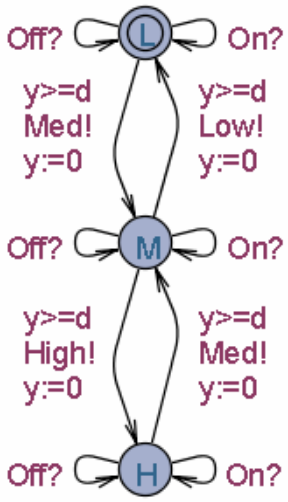




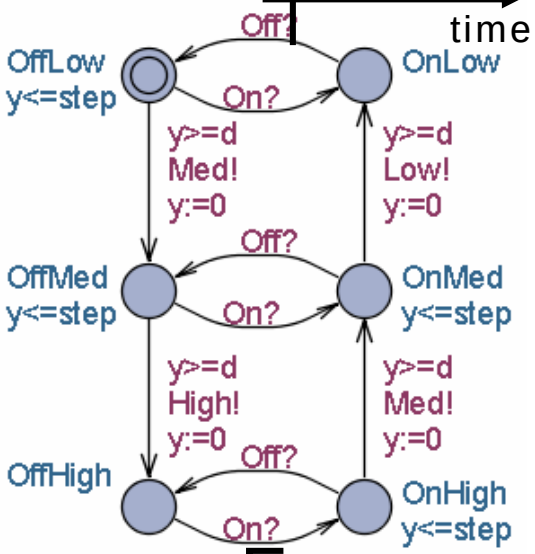
E_M



E_L



E_1

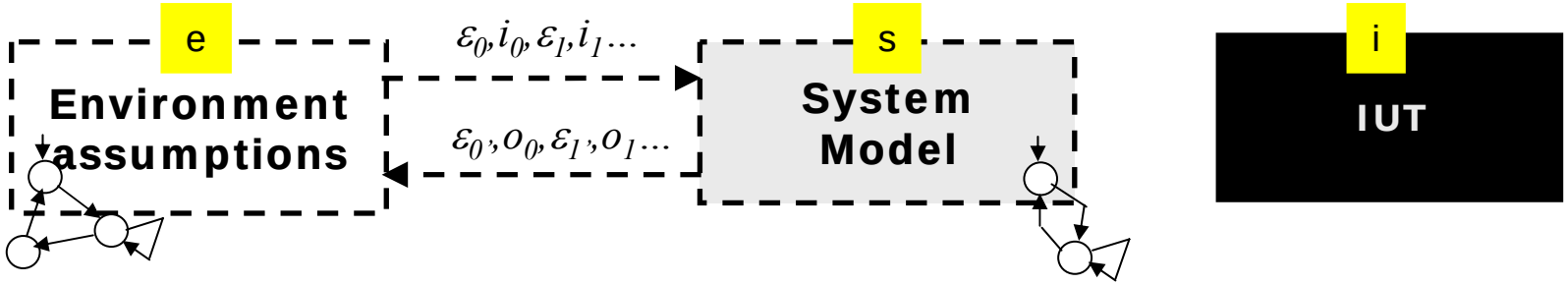


E_2

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

Implementation relation

Relativized real-time io-conformance



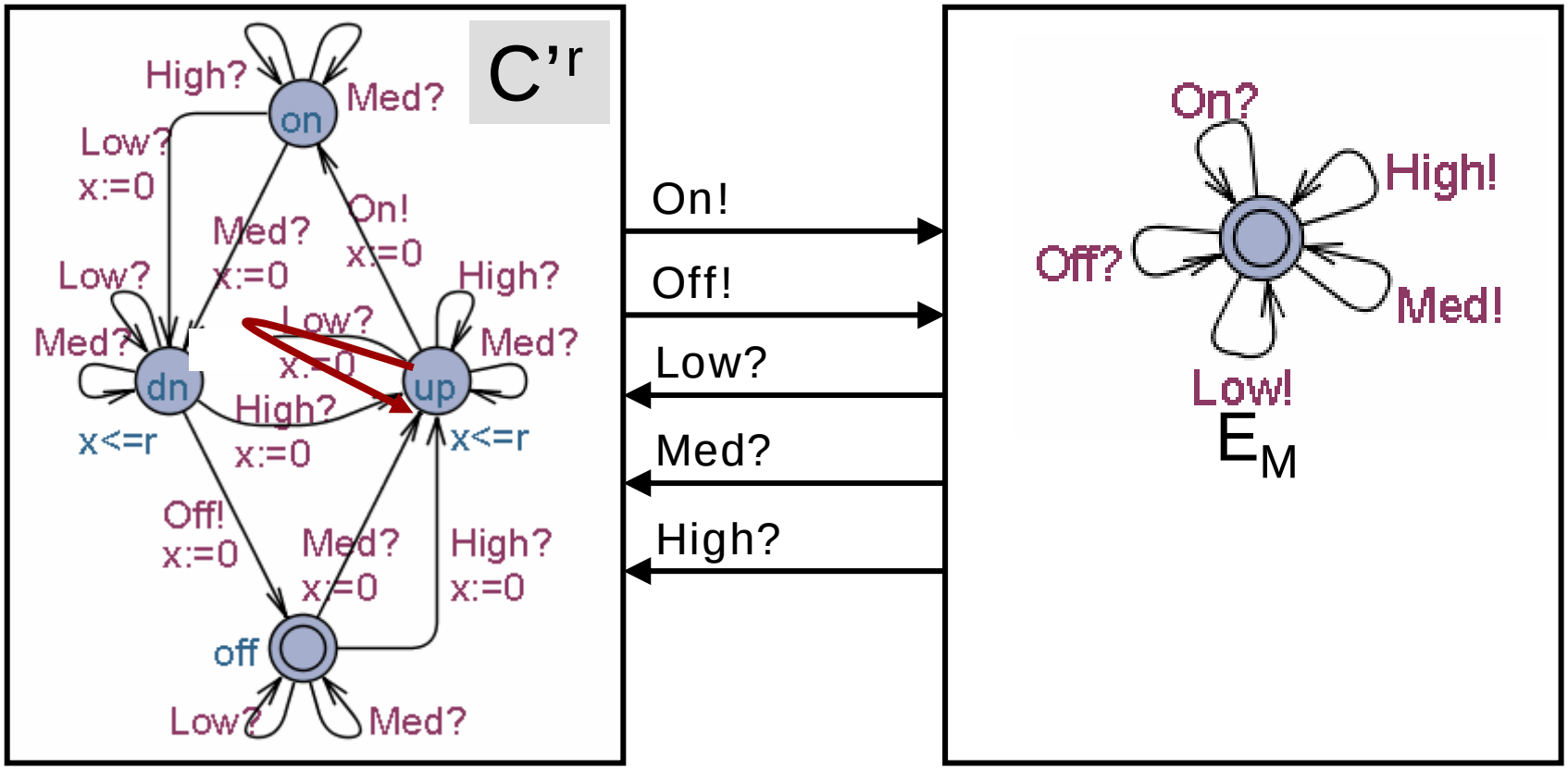
- **Let** *P* **be a set of states**
- **TTr**(*P*): the set of *timed traces* from states in *P*
- *P* **after** σ = the set of states reachable after timed trace σ
- **Out**(*P*) = possible outputs and delays in *P*

- 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 relevant environment behaviors**
  - **never produces illegal output, and**
  - **always produces required output in time**
- **~timed trace inclusion**

Sample Cooling Controller



IUT

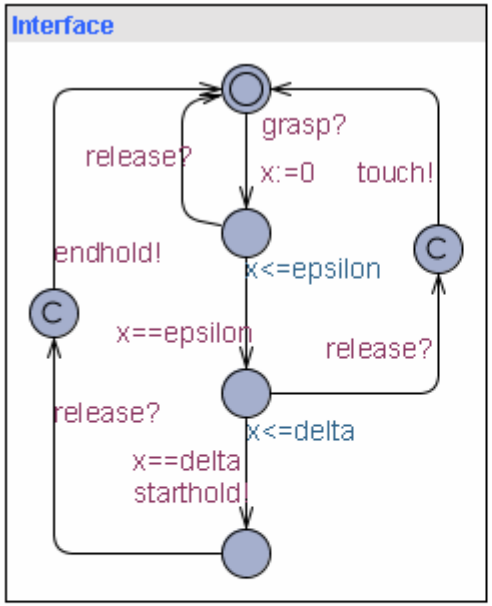
Env-model

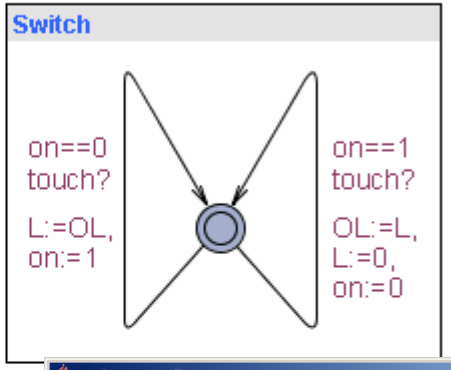
C'^r ~~rt-ioco~~ $_{E_M}$ C^r

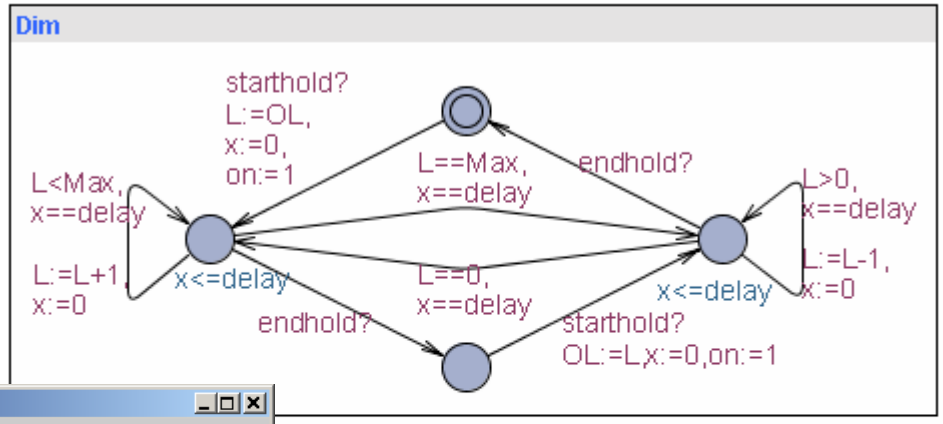
Test Generation

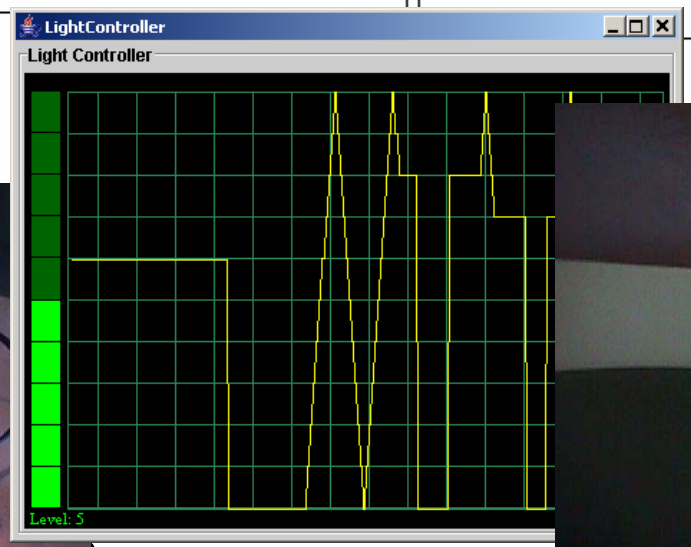
BRICS
Basic Research
in Computer Science

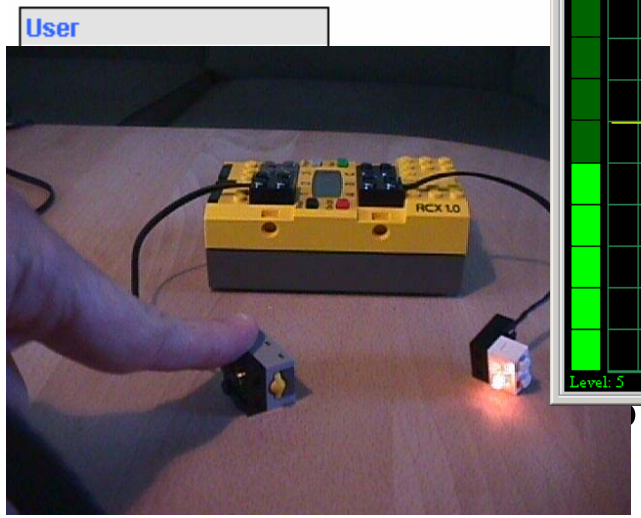
Touch-sensitive Light-Controller











Test Generation Principles

- Test purpose based generation
  - “test that light can become max”
  - Formalize purpose
  - Use model to compute sequence that meets purpose (inputs and expected outputs)
- Model Coverage
  - State-coverage
  - Transition
  - Def-Use pairs
  - ...
- Randomized model interpretation
- Fault-Models
- ONLINE (randomized) testing

Time Optimal Real-Time Test Generation using UPPAAL

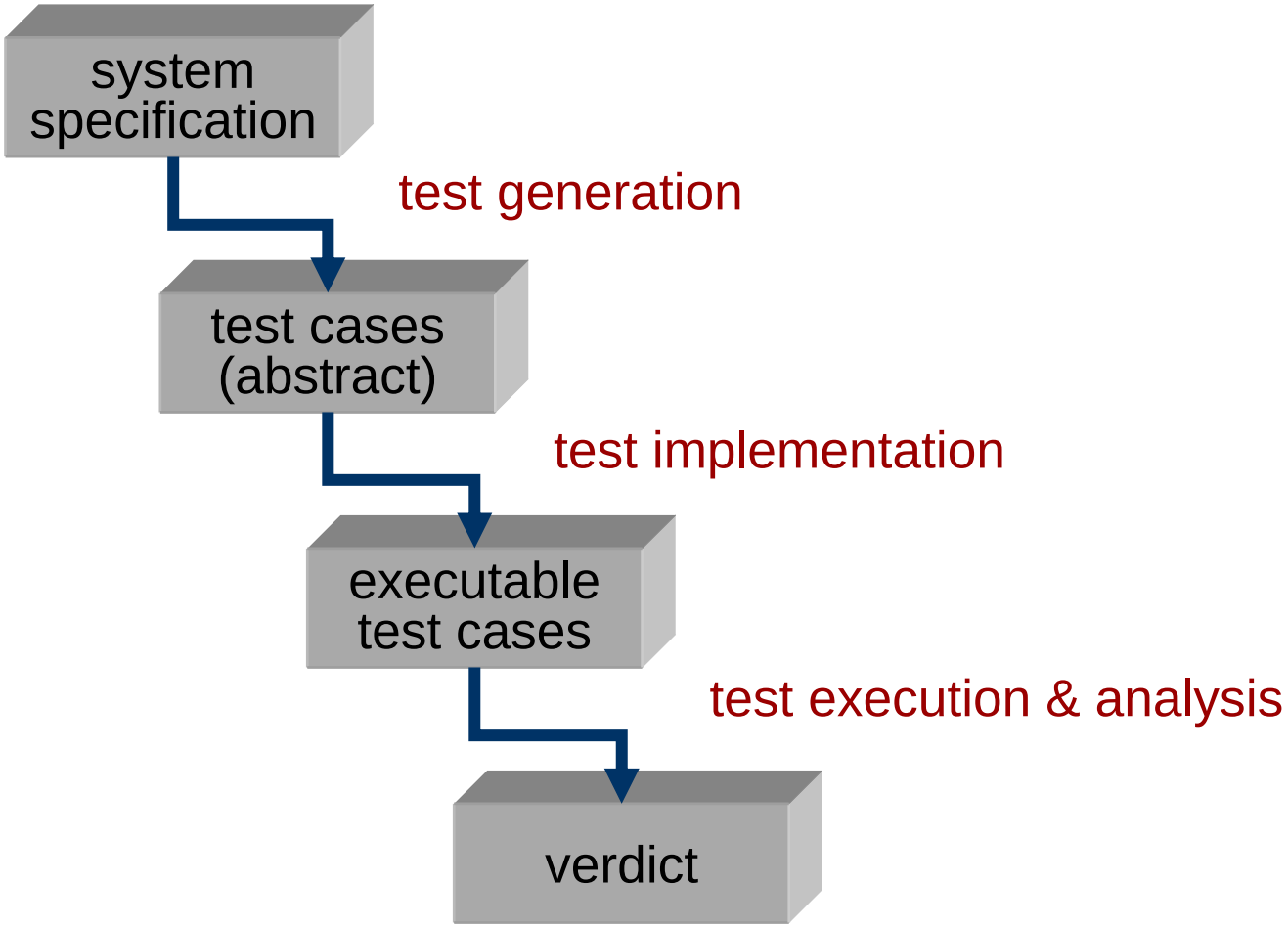
Anders Hessel,
Paul Pettersson
Uppsala University
Sweden

Kim Larsen,
Brian Nielsen,
Arne Skou
Aalborg University
Denmark

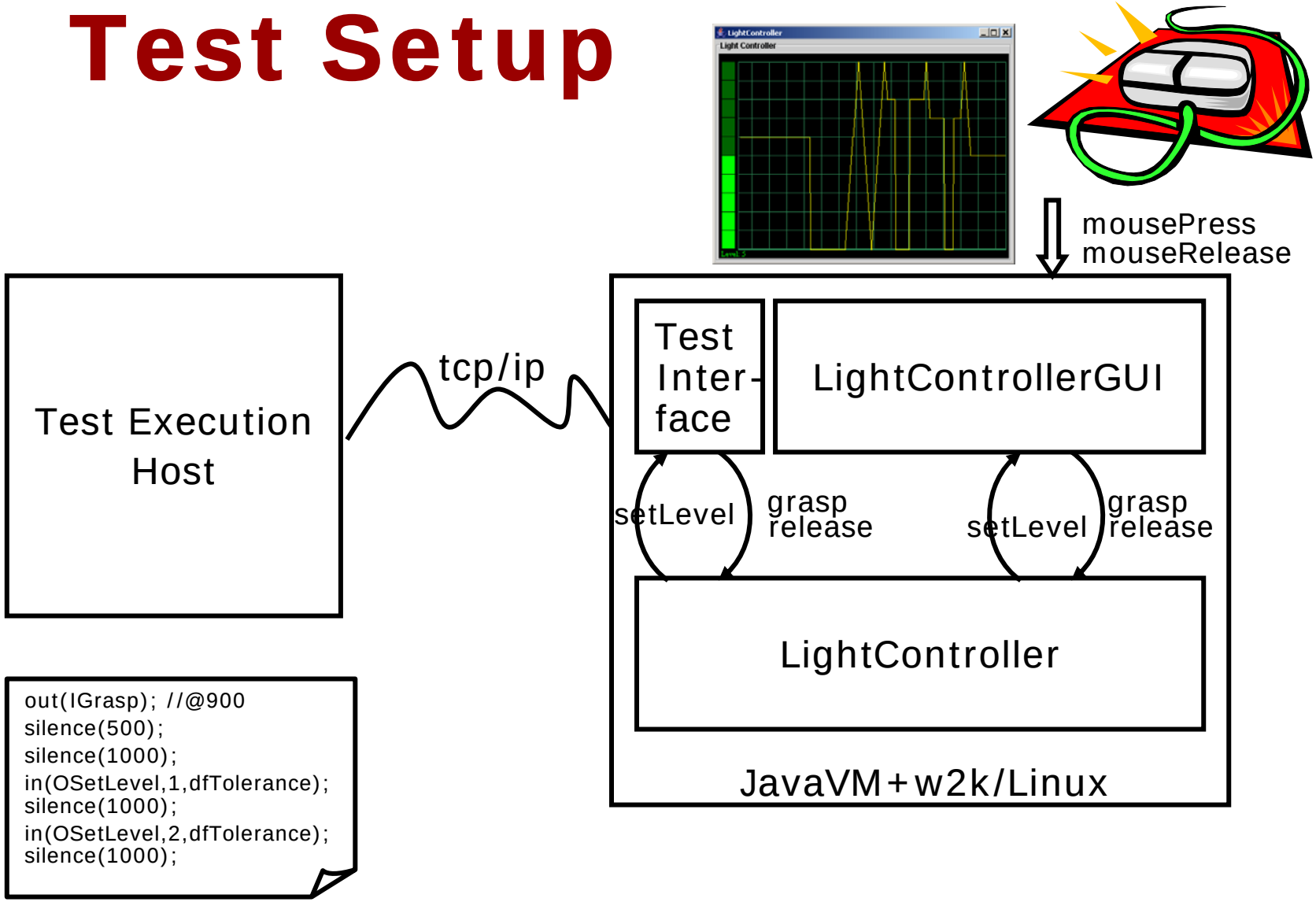
BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Testing Process



Test Setup


LightController
Light Controller
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);

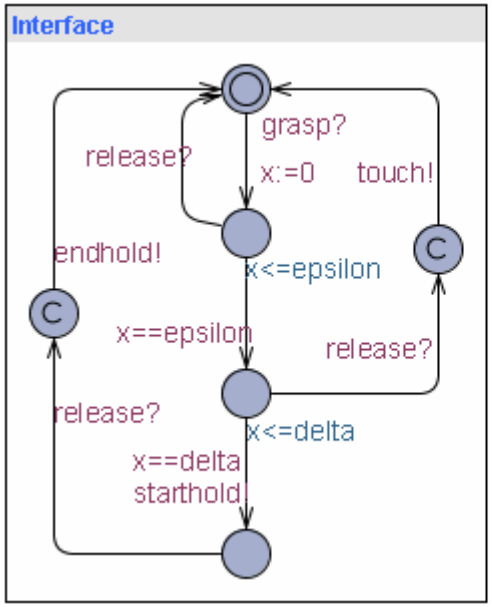
  send eventNo to IUT at now();
- void **silence**(int msDelay);

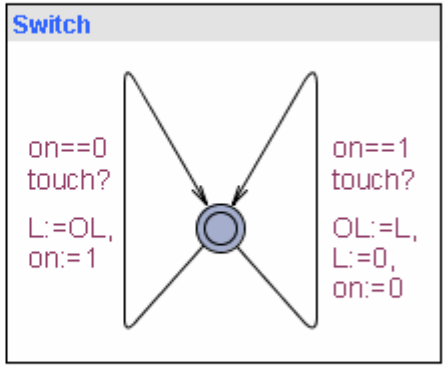
  expect no outputs for msDelay: otherwise fail
- void **in** (int eventNo,int par, int msTolerance);

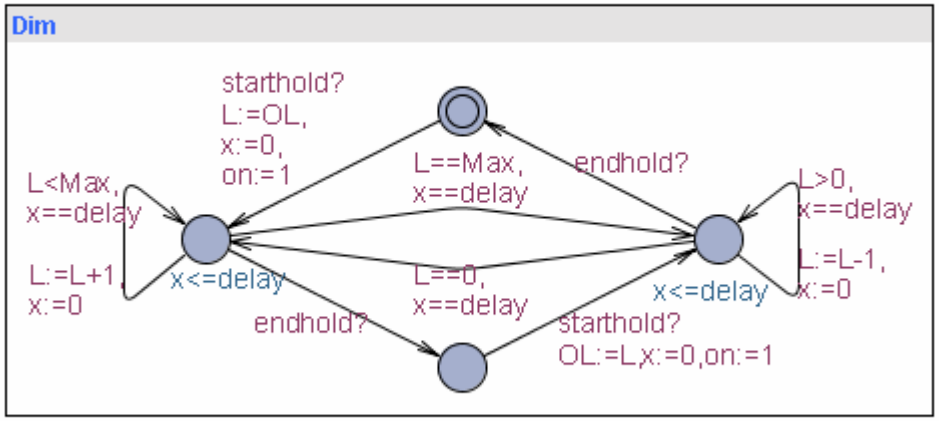
  expect input event(par) before now()+msTolerance otherwise fail
- void **at**(int eventNo, int par, int msTime, int msTolerance);

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

Timed Tests







EXAMPLE test cases for **Interface**

- Epsilon = 200ms
- Delta = 500ms

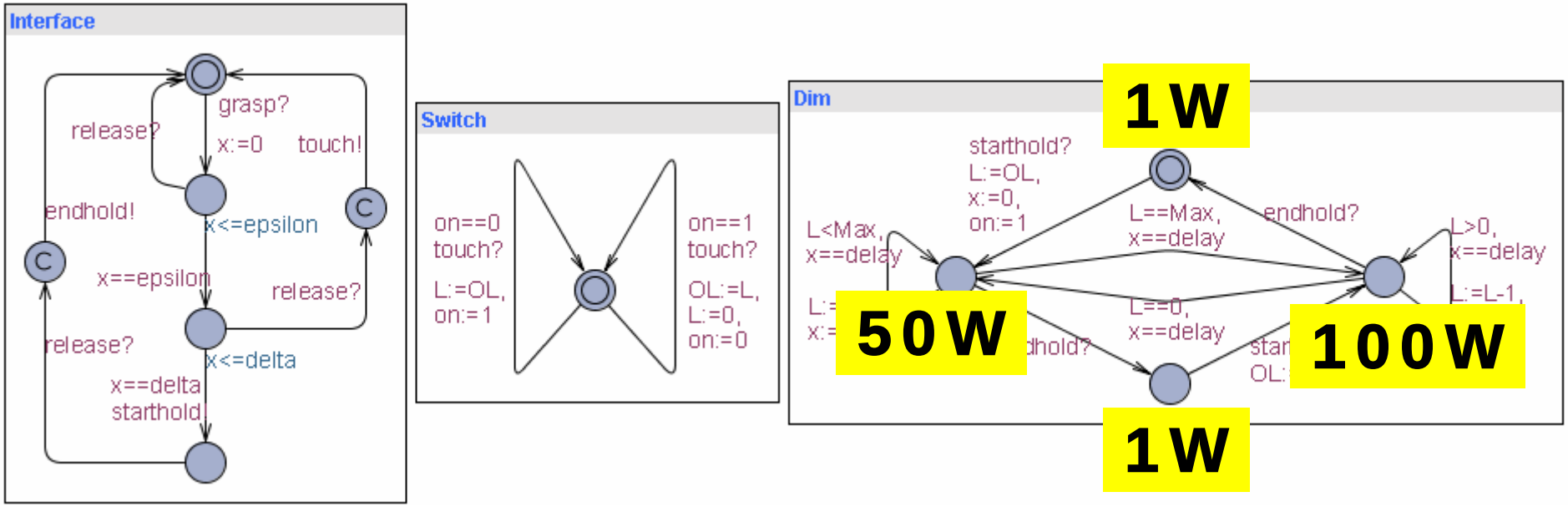
0·grasp!·210·release!·touch?.PASS

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

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

INFINITELY MANY SEQUENCES!!!!!!

Optimal Tests

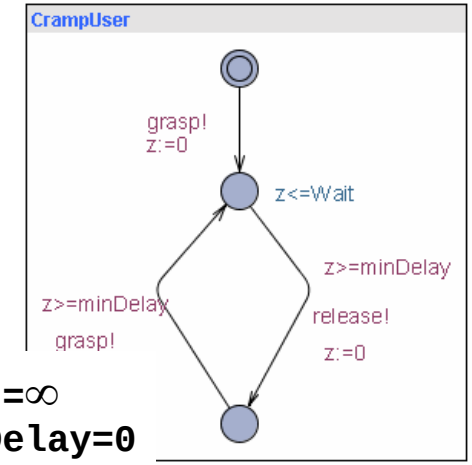


- ***Shortest*** test for max light??
- ***Fastest*** test for max light??
- ***Fastest*** edge-covering test suite??
- Least ***power*** consuming test??

Test Purposes 1

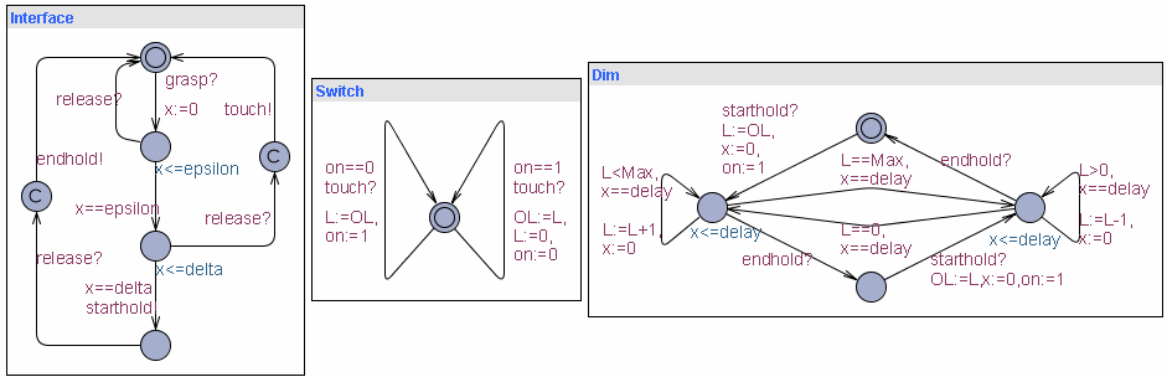
A specific test objective (or observation) the tester wants to make on SUT

Environment model



Wait=∞
minDelay=0

System model



TP1: Check that the light can become bright:

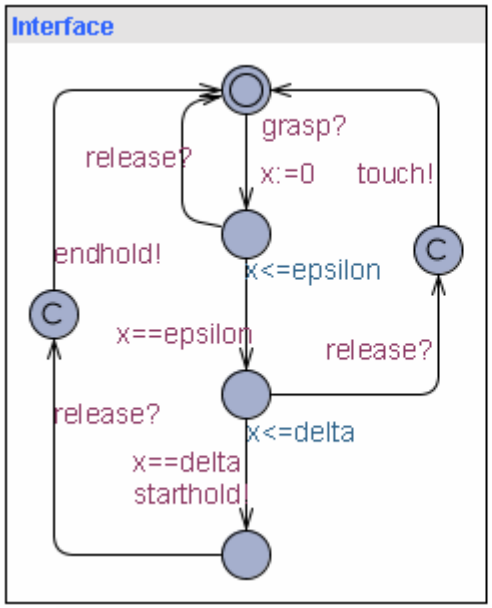
E<> L==10

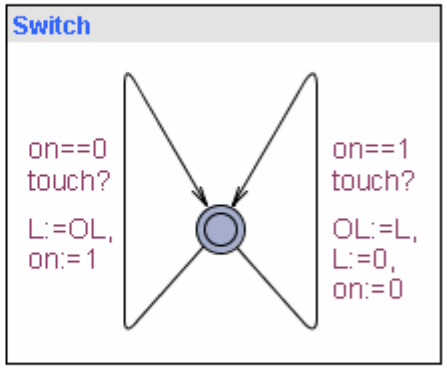
- *Shortest* (and *fastest*) Test:

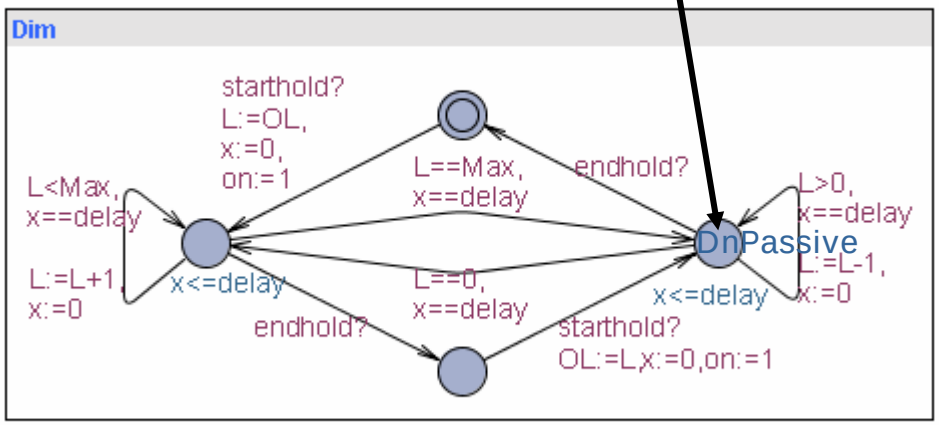
```
out(IGrasp);silence(500);in(OSetLevel,0);silence(1000);
in(OSetLevel,1);silence(1000);in(OSetLevel,2); silence(1000);
in(OSetLevel,3);silence(1000);in(OSetLevel,4);silence(1000);
in(OSetLevel,5);silence(1000);in(OSetLevel,6);silence(1000);
in(OSetLevel,7);silence(1000);in(OSetLevel,8);silence(1000);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

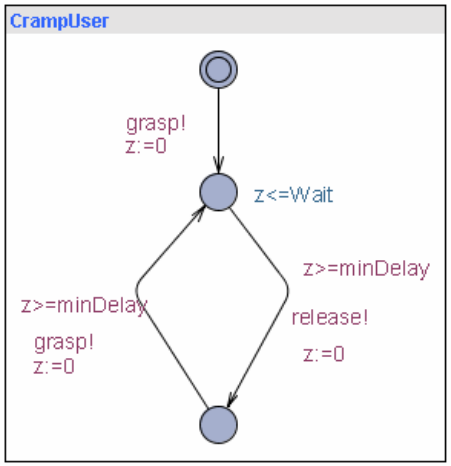
Test Purposes 2

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







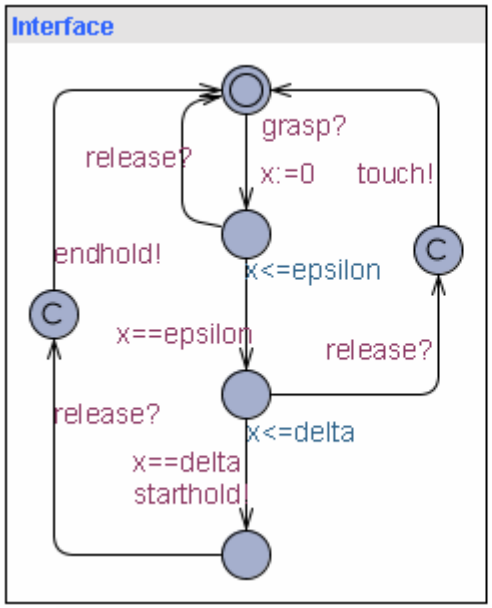


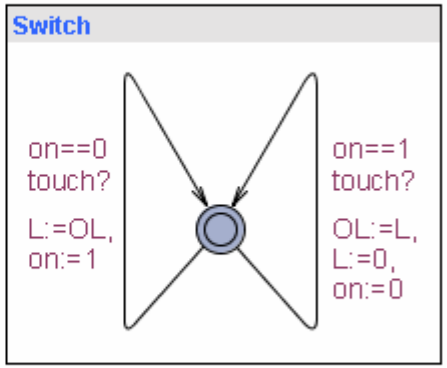
- If delay=1000
- *Shortest* (and *fastest*) Test:

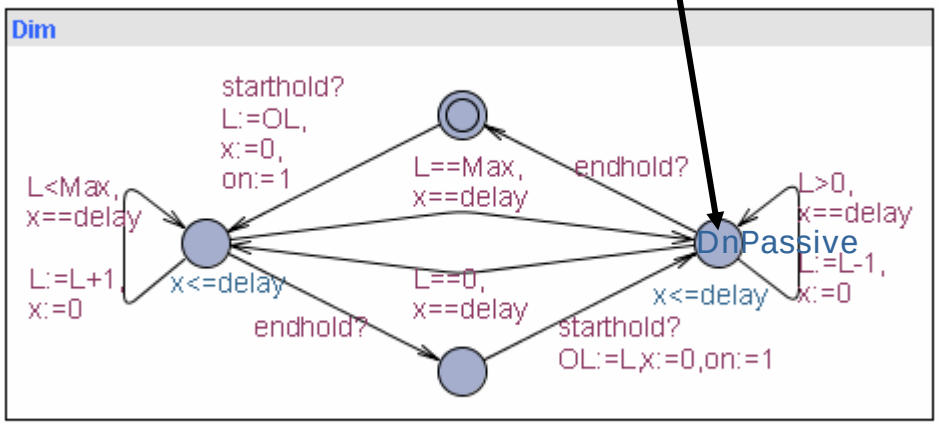
```
out(IGrasp);
silence(500);
in(OSetLevel,0);
out(IRelease);
out(IGrasp);
silence(500);
```

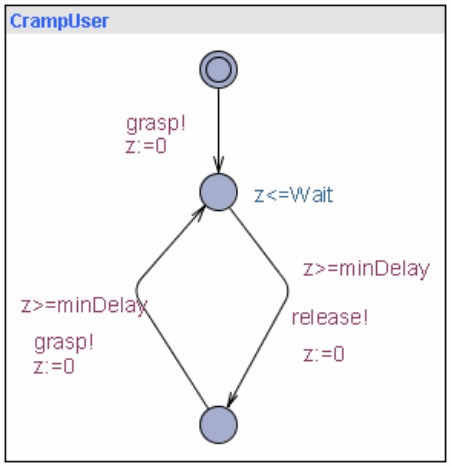
Test Purposes 2

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









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

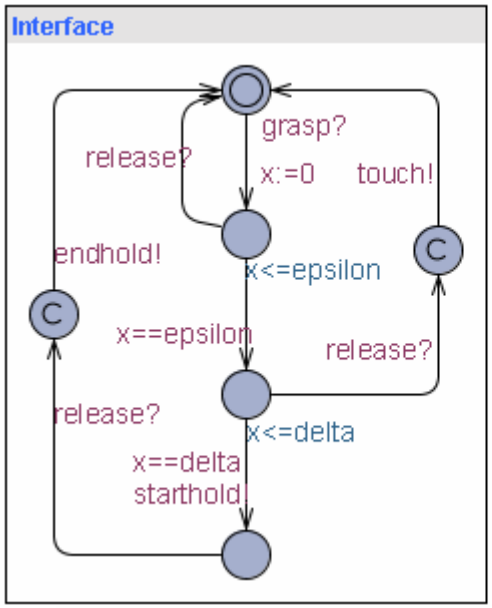
```
out(IGrasp);
silence(500);
in(OSetLevel,0);
out(IRelease);
out(IGrasp);
silence(500);
```

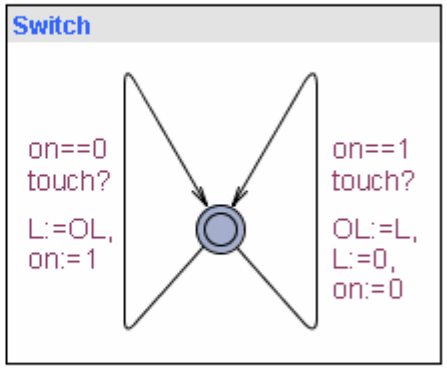
- *Fastest* Test:

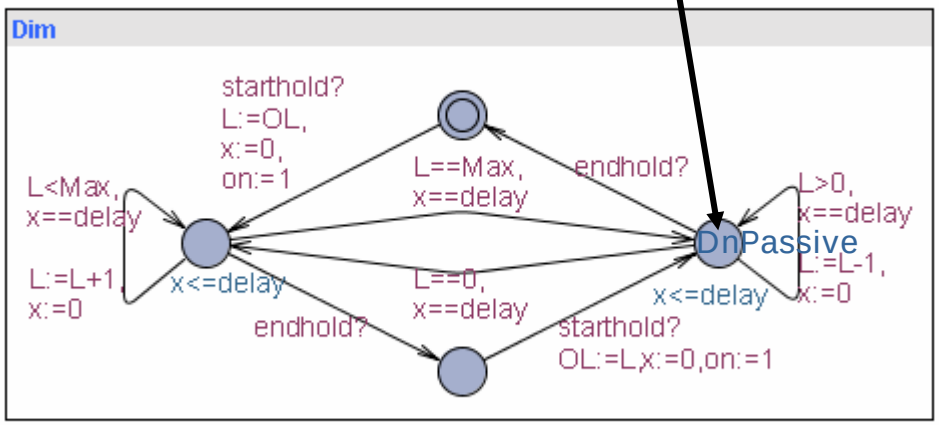
```
out(IGrasp);silence(500);in(OSetLevel,0);silence(40);
in(OSetLevel,1);silence(40);in(OSetLevel,2); silence(40);
in(OSetLevel,3);silence(40);in(OSetLevel,4); silence(40);
in(OSetLevel,5);silence(40);in(OSetLevel,6); silence(40);
in(OSetLevel,7);silence(40);in(OSetLevel,8); silence(40);
in(OSetLevel,9);silence(40);in(OSetLevel,10);silence(40);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

Test Purposes 2

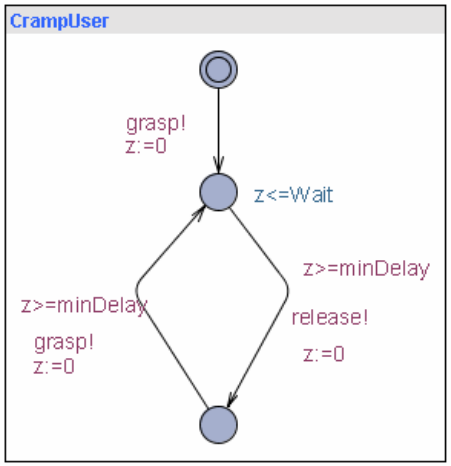
TP2: Check that controller can enter location ‘DnPassive’:
E<> Dim.DnPassive







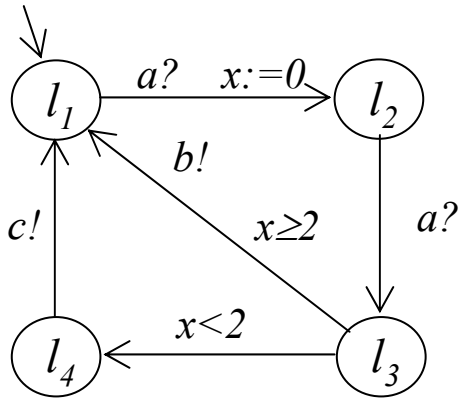
- If Wait=1500 and minDelay=400?



Ask a tool

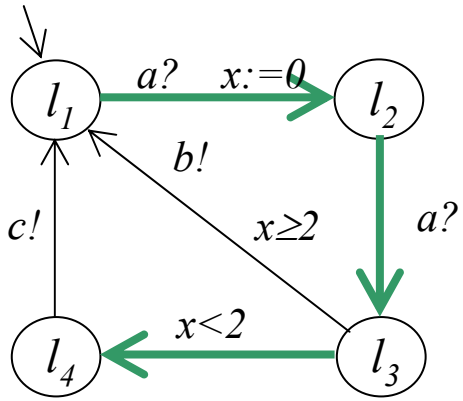
Coverage Based Test Generation

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



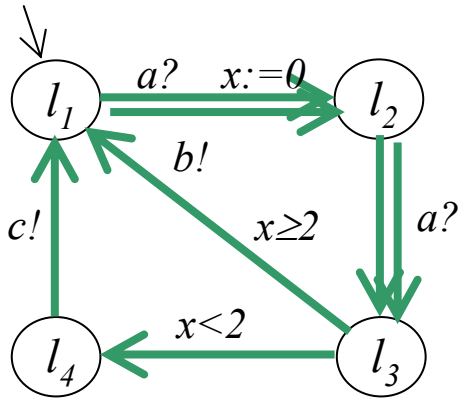
Coverage Based Test Generation

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



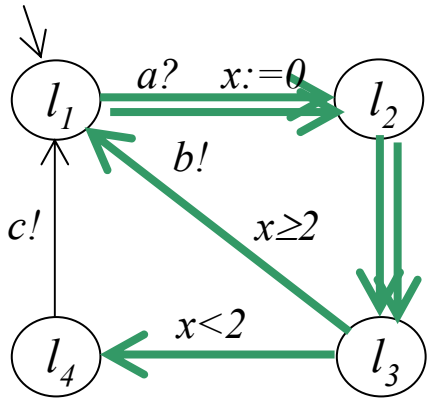
Coverage Based Test Generation

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



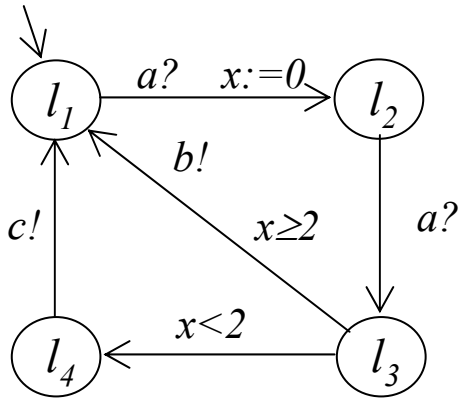
Coverage Based Test Generation

- Multi purpose testing
- Cover measurement
- Examples:
  - Location Coverage,
  - Edge Coverage,
  - **Definition / Use Pair Coverage**



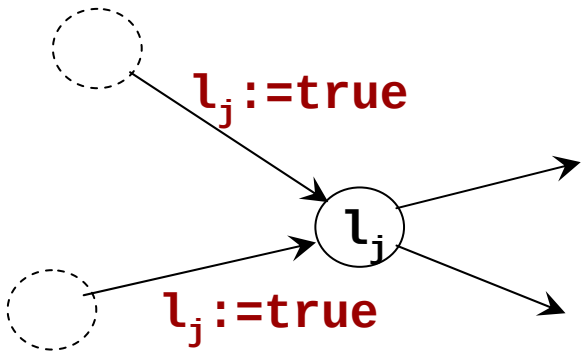
Coverage Based Test Generation

- Multi purpose testing
- Cover measurement
- Examples:
  - Locations coverage,
  - Edge coverage,
  - Definition/use pair coverage
  - All Definition/Use pairs
- Generated by min-cost reachability analysis of annotated graph



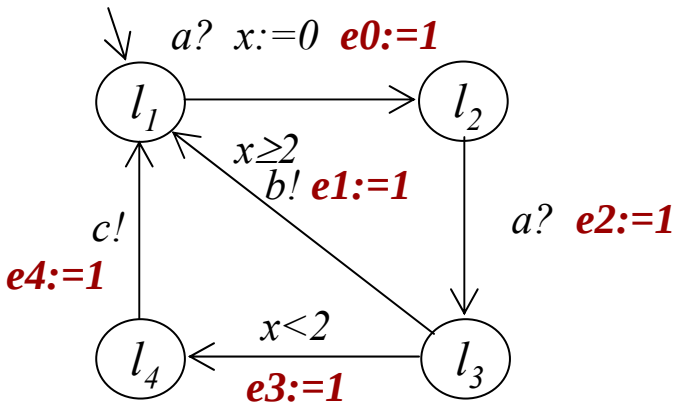
Location Coverage

- Test sequence traversing all locations
- Encoding:
  - Enumerate locations $l_0, \ldots, l_n$
  - Add an auxiliary variable l_i for each location
  - Label each ingoing edge to location i l_i:=true
  - Mark initial visited l_0:=true
- Check: EF(l_0=true $\wedge$ … $\wedge$ l_n=true)



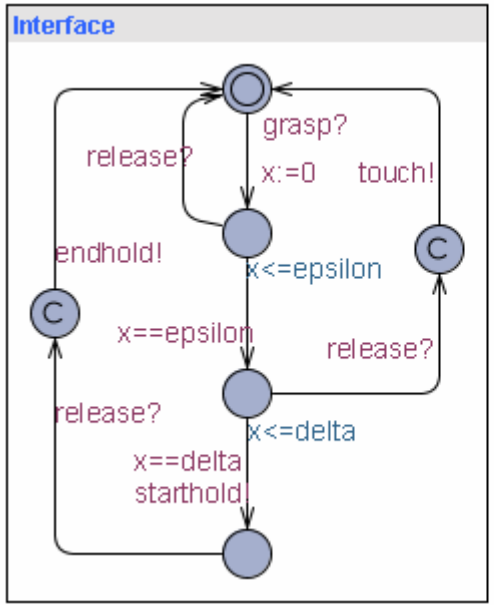
Edge Coverage

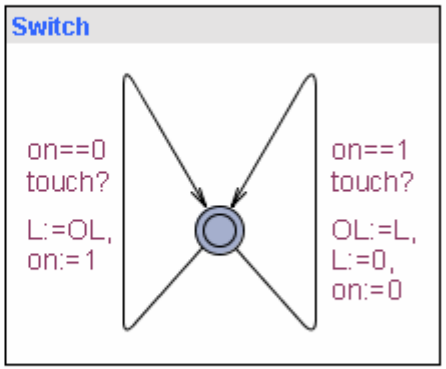
- Test sequence traversing all edges
- Encoding:
  - Enumerate edges $e_0, \ldots, e_n$
  - Add auxiliary variable e_i for each edge
  - Label each edge $e_i := true$
- Check: $EF(\ e_0 = true \wedge \ldots \wedge e_n = true\ )$

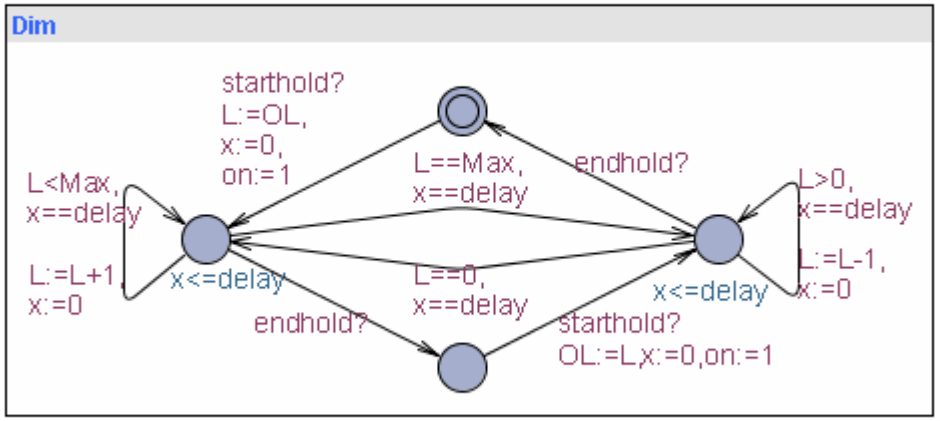


Fastest Edge Coverage

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

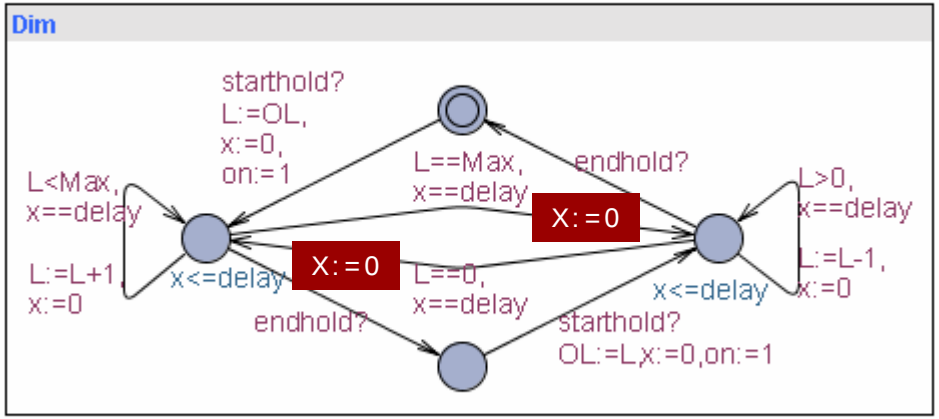
```
//13
out(IGrasp); //@900          // Bring dimmer  PassiveDn->ActiveDN->
silence(500);//hold          // ActiveUP+increase to level 10
silence(1000); in(OSetLevel,1);
silence(1000); in(OSetLevel,2);
silence(1000); in(OSetLevel,3);
silence(1000); in(OSetLevel,4);
silence(1000); in(OSetLevel,5);
silence(1000);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

Mutants

- 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 incorrect additional delay in dimmer as if x:=0 was on ActiveUP ↔ActiveDN transitions

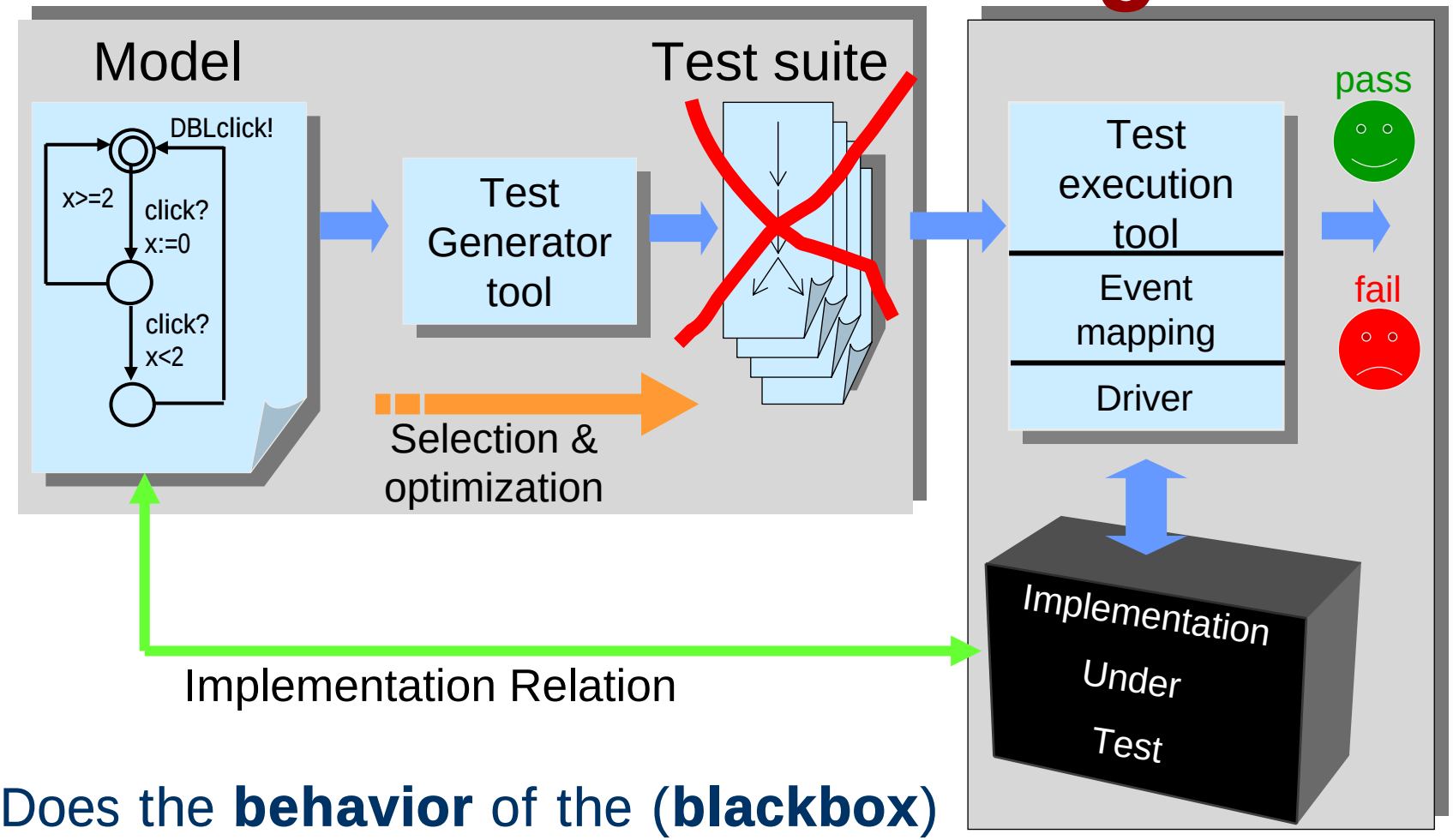
Outcome

Description	Test#	M0	M1	M2
MaxLevel	1	pass	pass	pass
Short ActiveDn	4	pass	pass	pass
Resume	5	Pass	Fail	pass
Edge Cov.	3	Pass	Fail	Fail

Online Testing Of Real-Time Systems

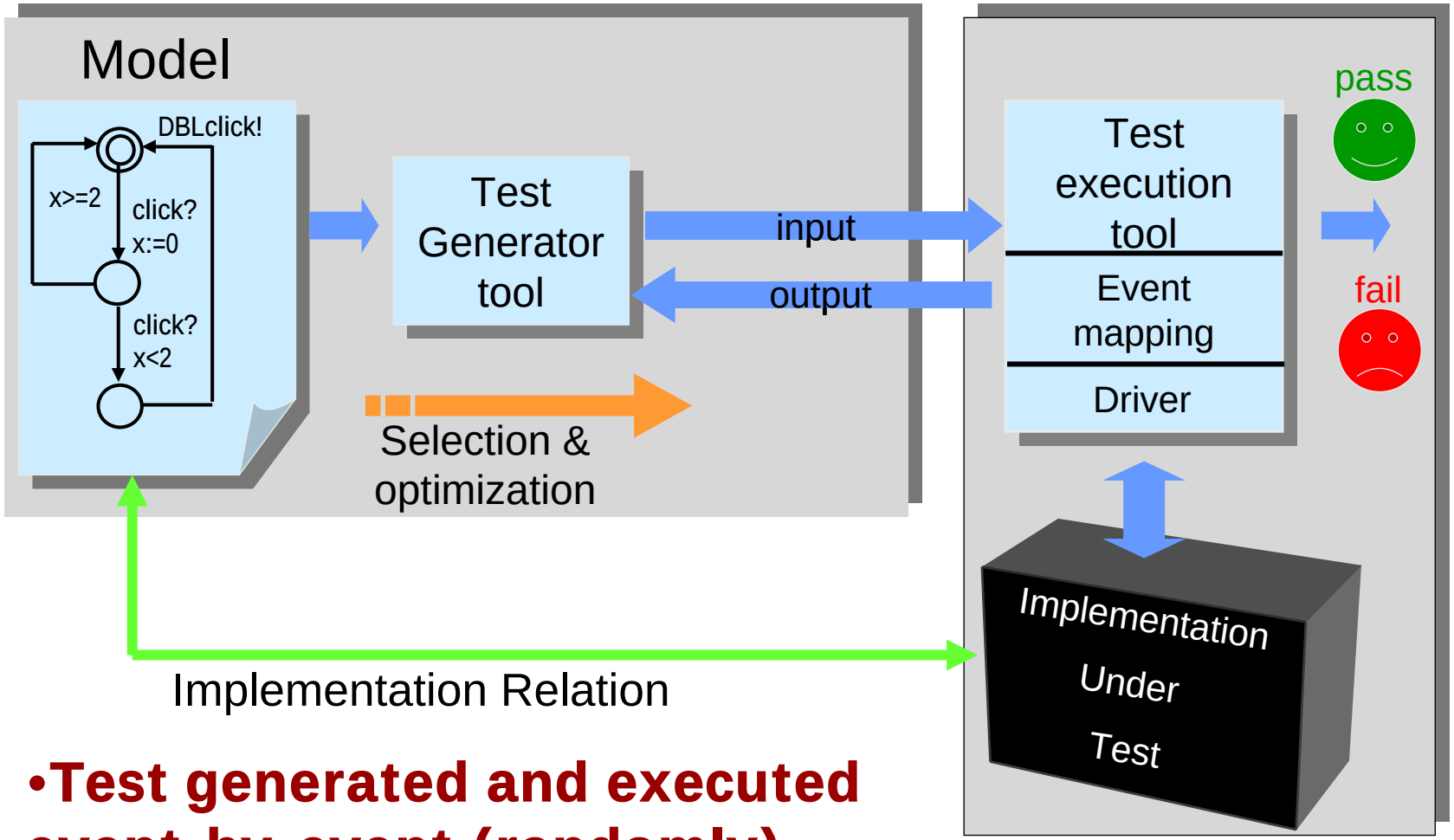
BRICS
Basic Research
in Computer Science

Automated Model Based Conformance Testing



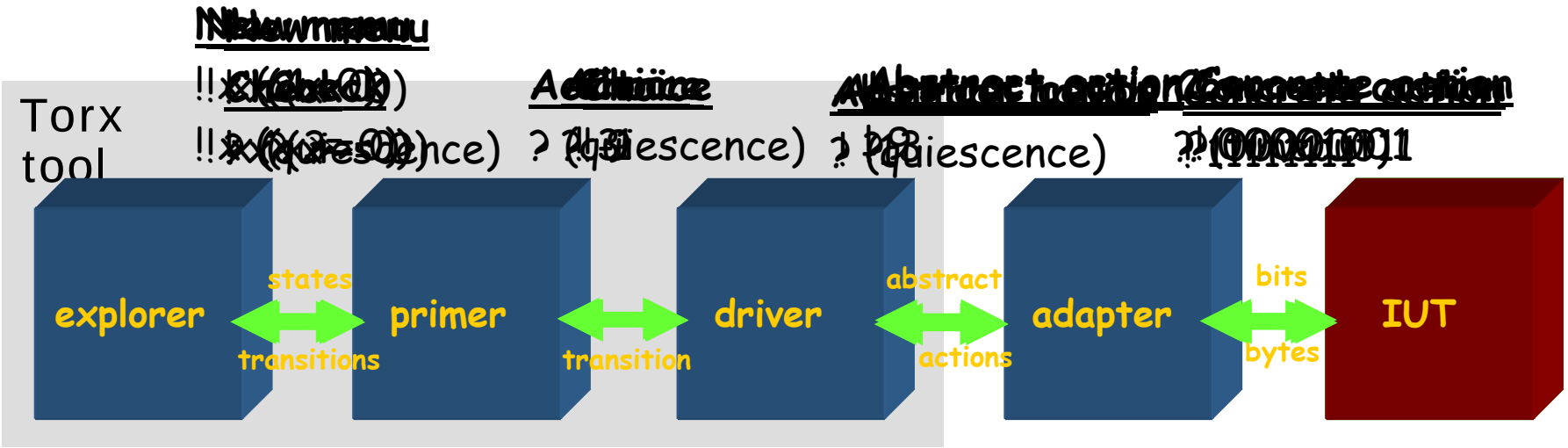
Does the **behavior** of the (**blackbox**) implementation ***comply*** to that of the specification?

Online Testing

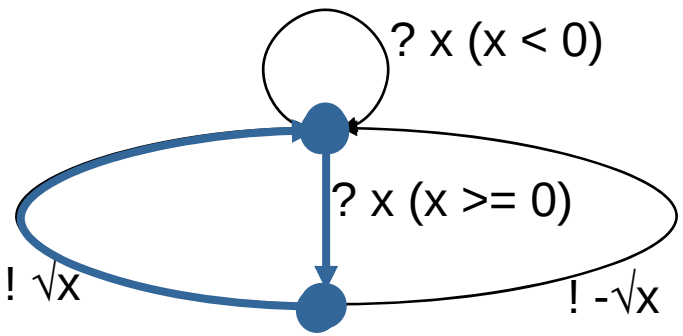


- **Test generated and executed event-by-event (randomly)**
- **A.K.A on-the-fly testing**

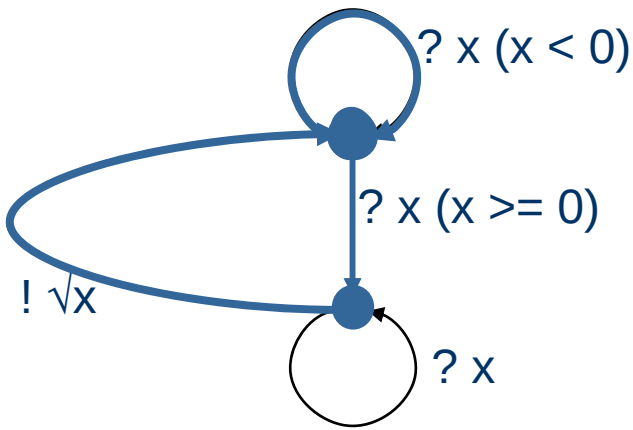
On-The-Fly Testing



specification



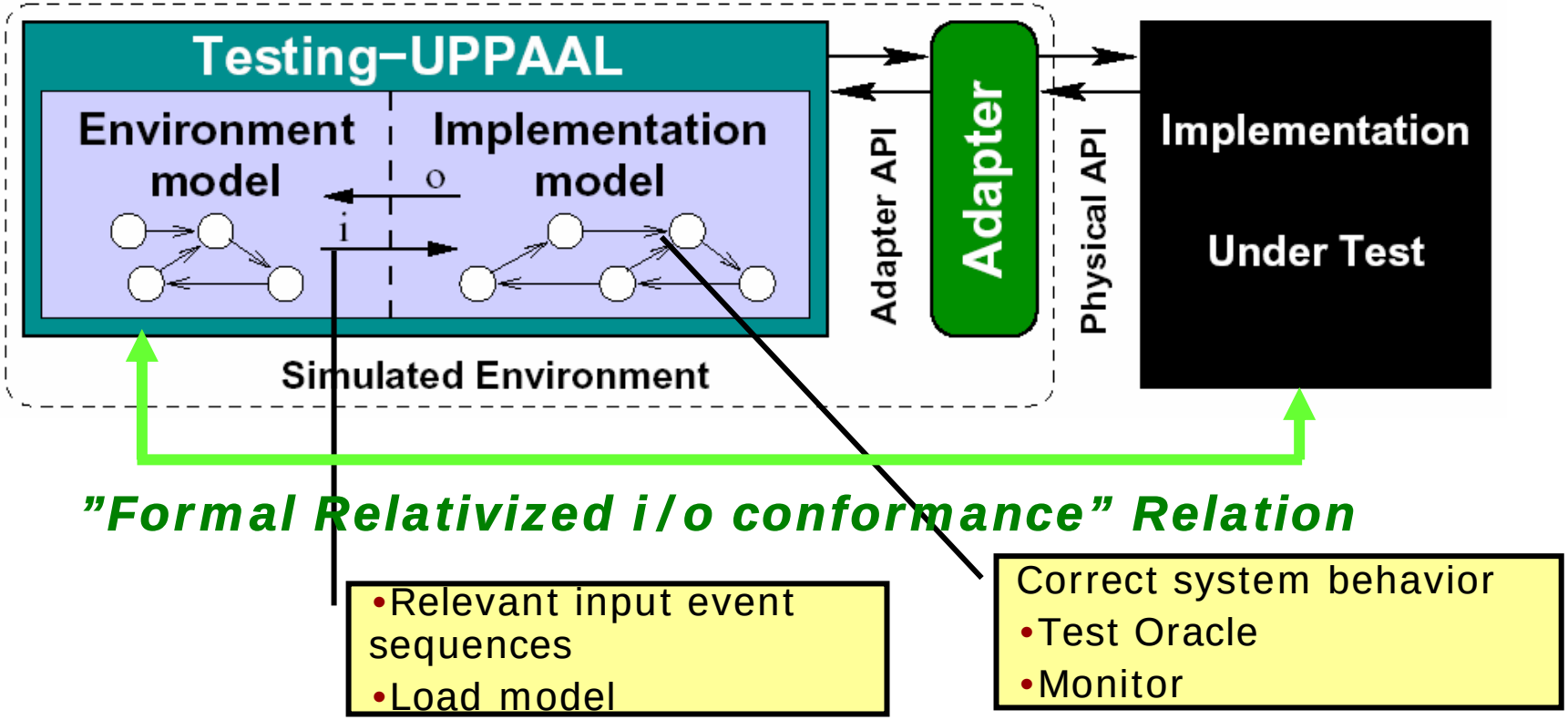
implementation



[Jan Tretmans].

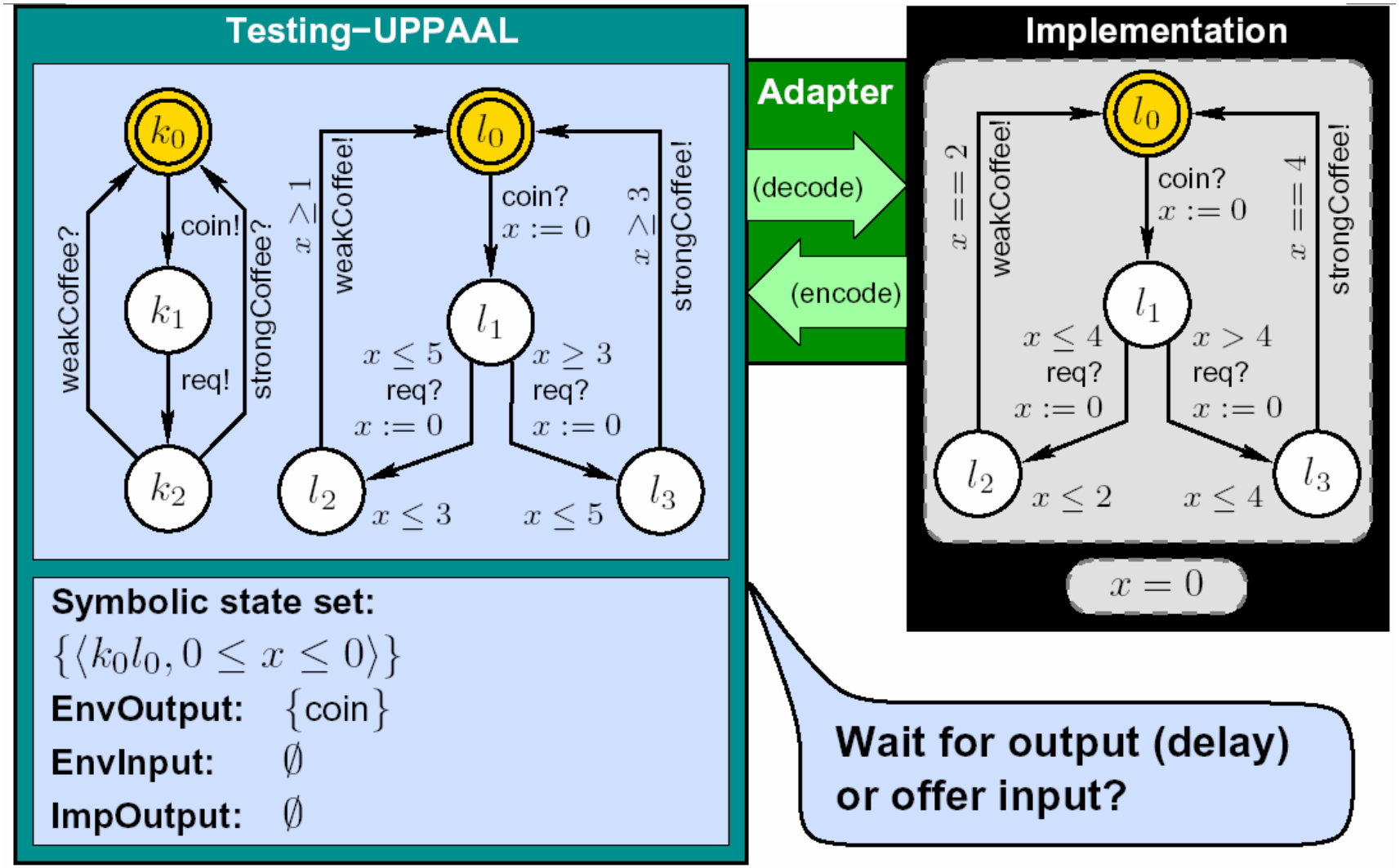
Our Framework

- ***UppAal Timed Automata*** *Network: Env || IUT*

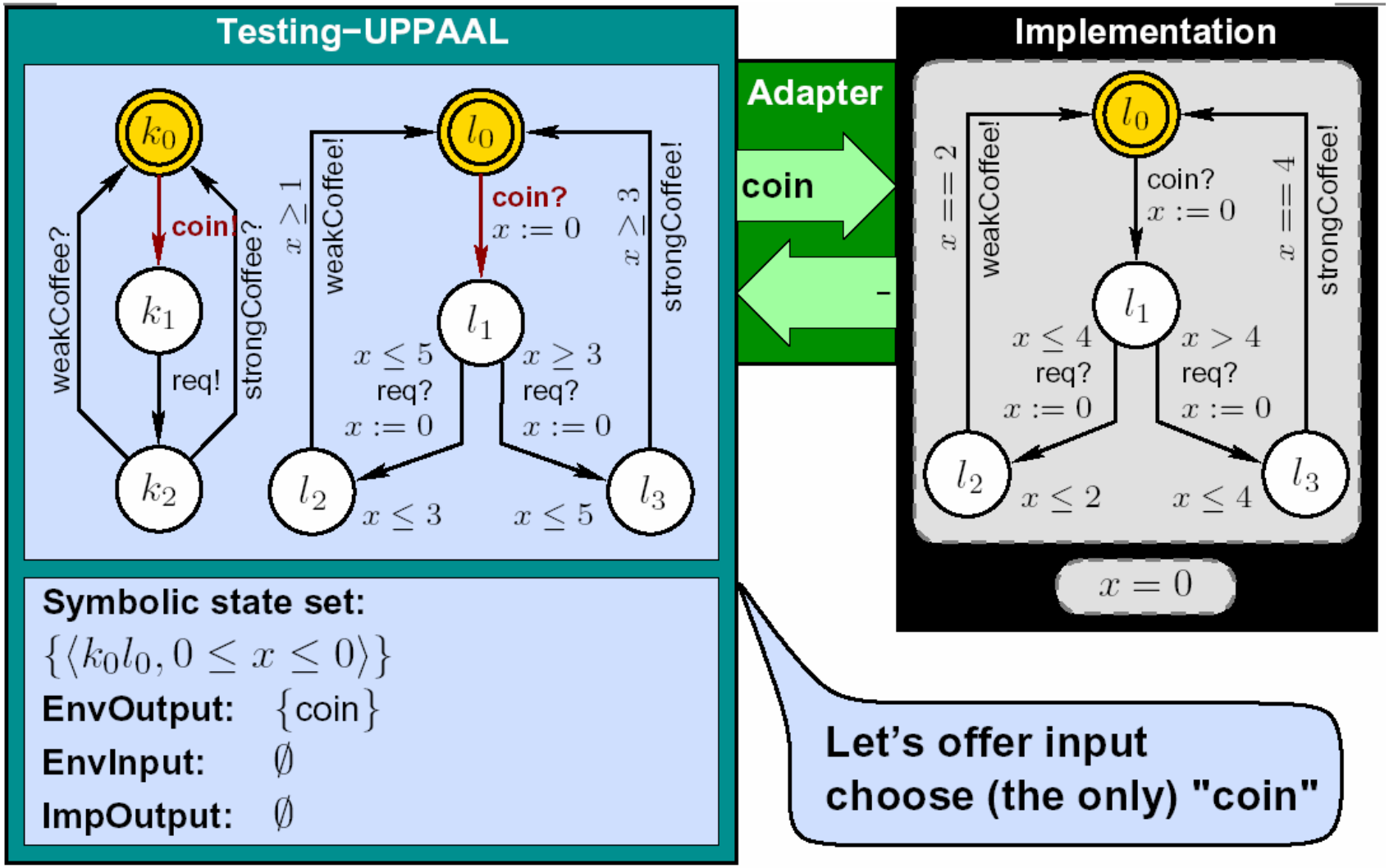


- *Complete and sound algorithm*
- Efficient symbolic reachability algorithms
- **UppAal-TRON:** Testing Real-Time Systems Online
- Release 1.3 http://www.cs.aau.dk/~marius/tron/

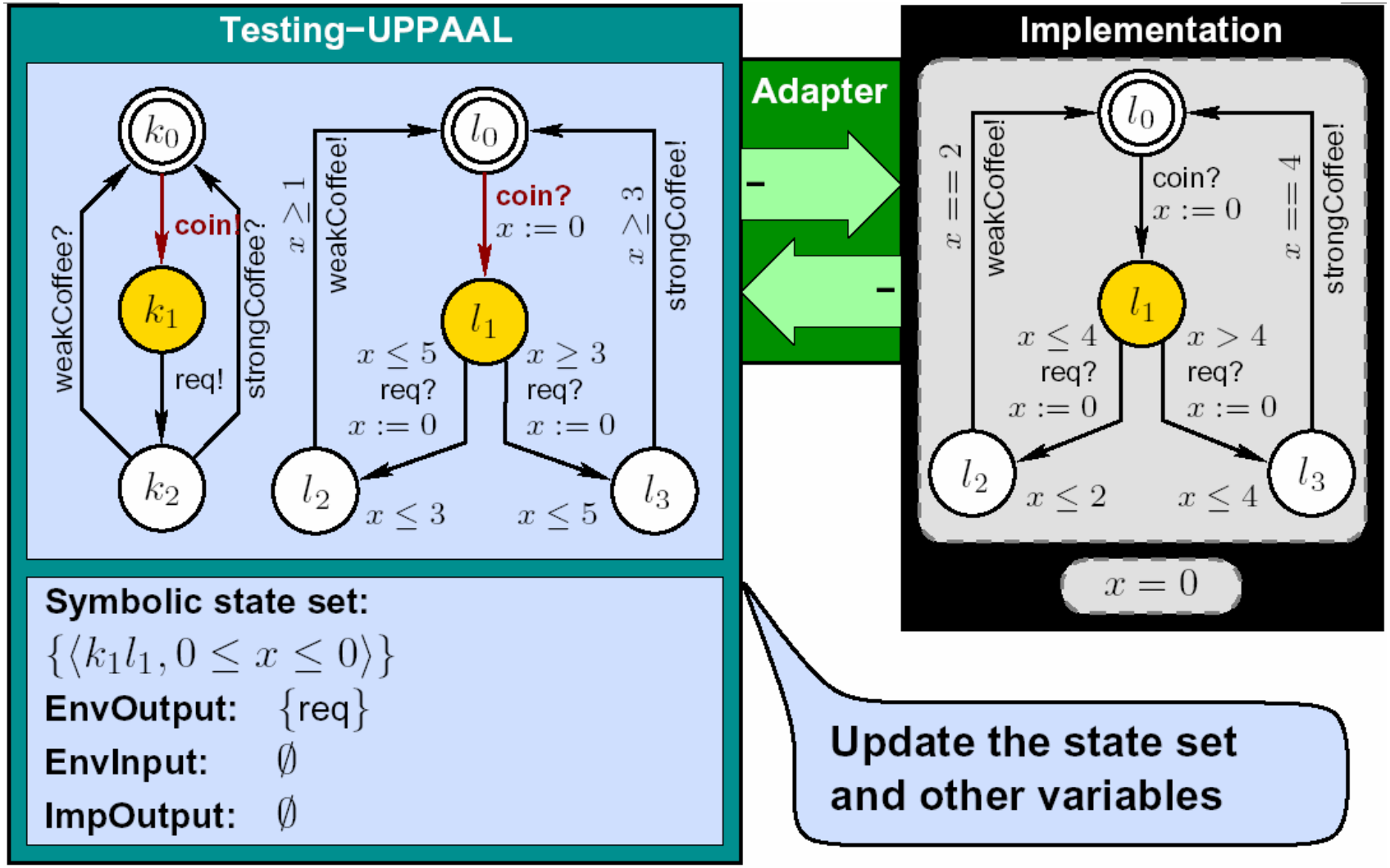
Online Testing



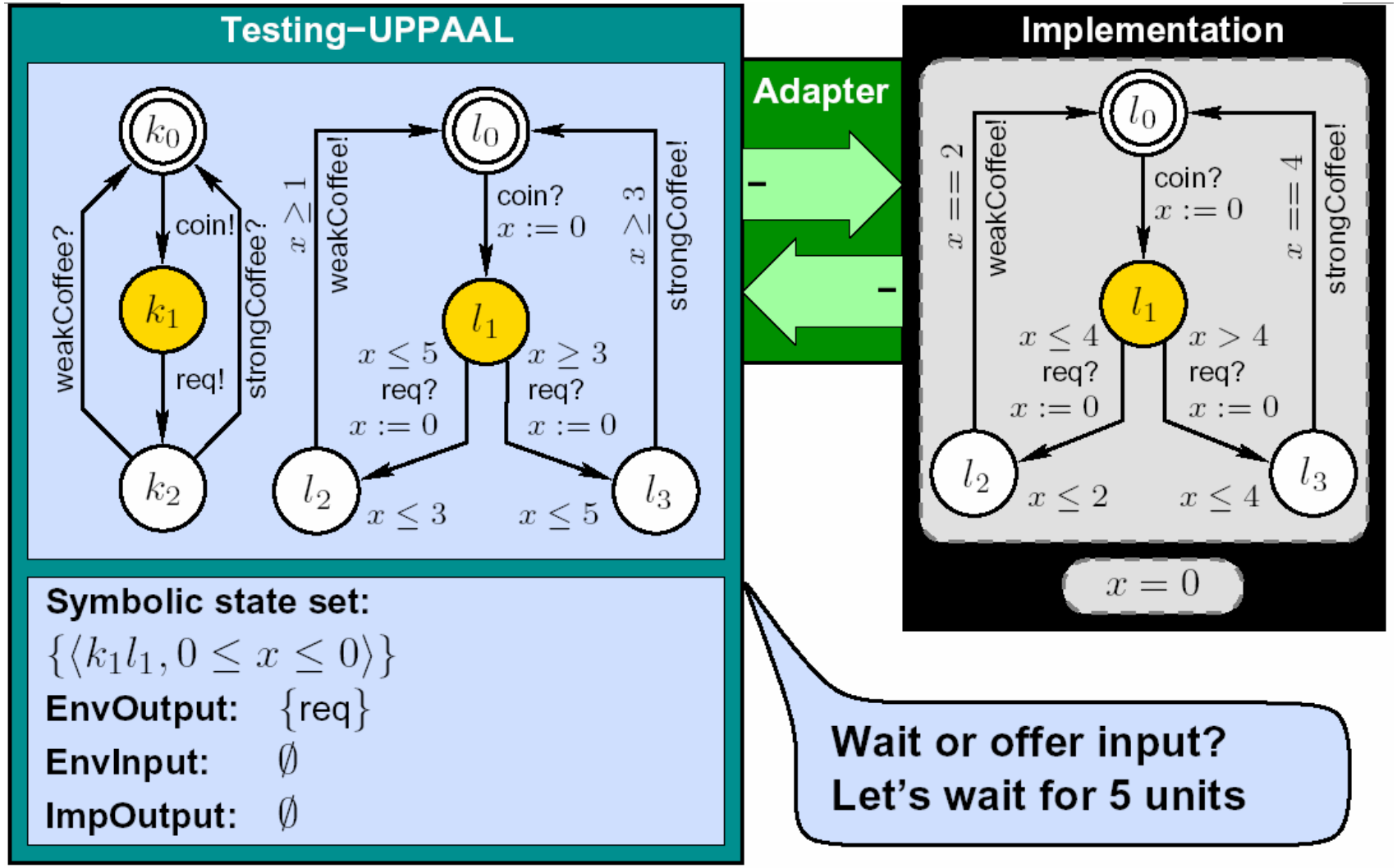
Online Testing



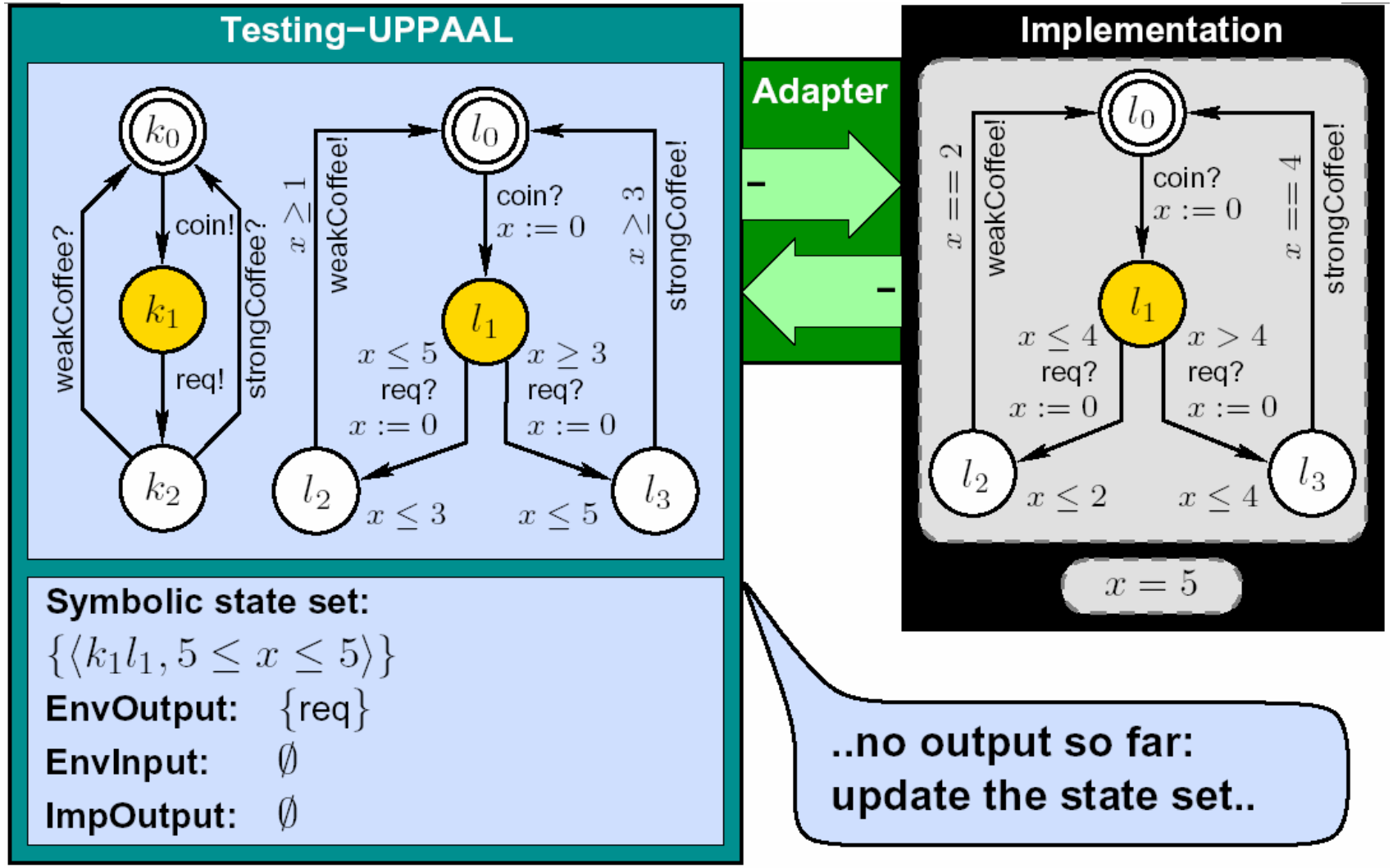
Online Testing



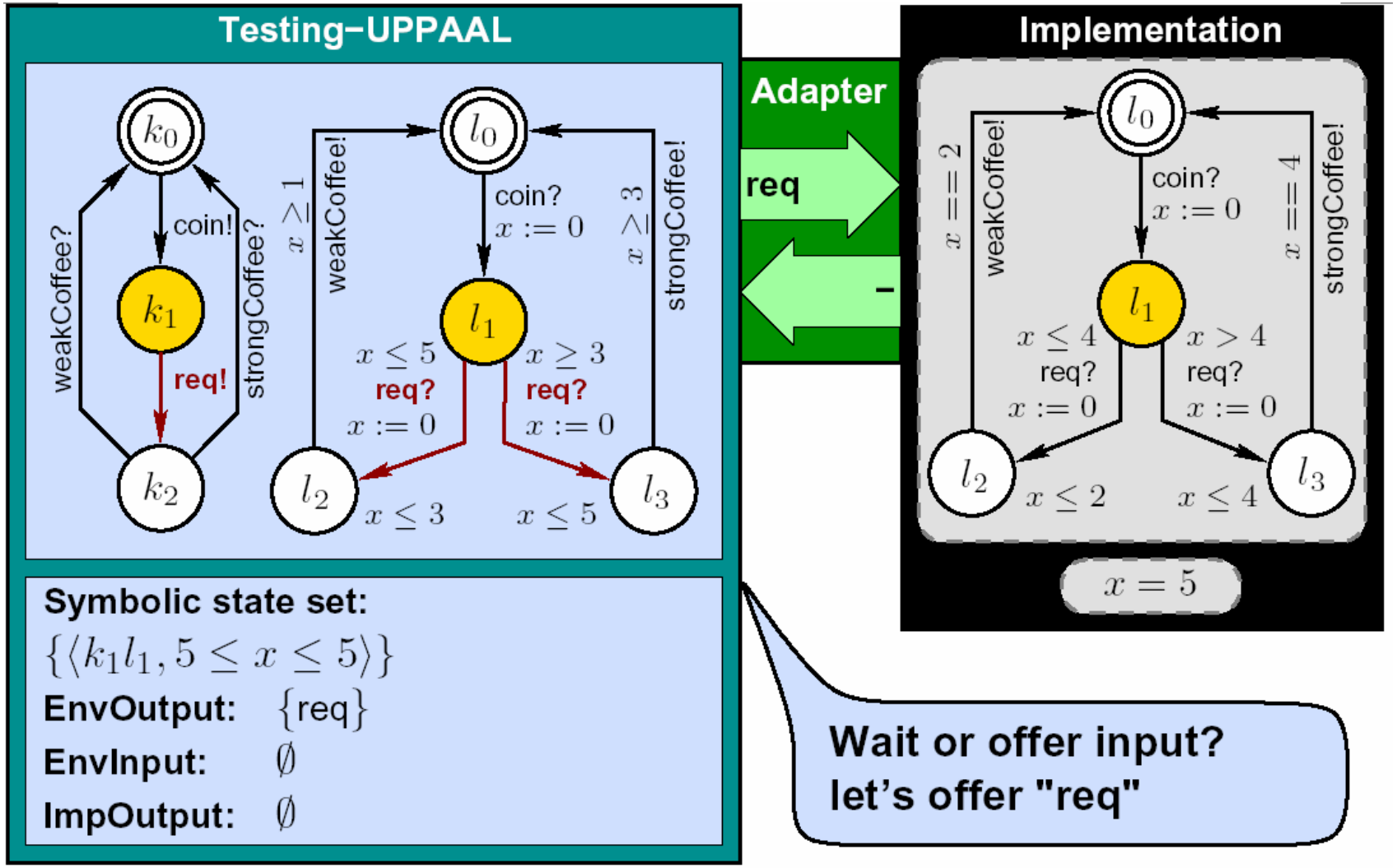
Online Testing



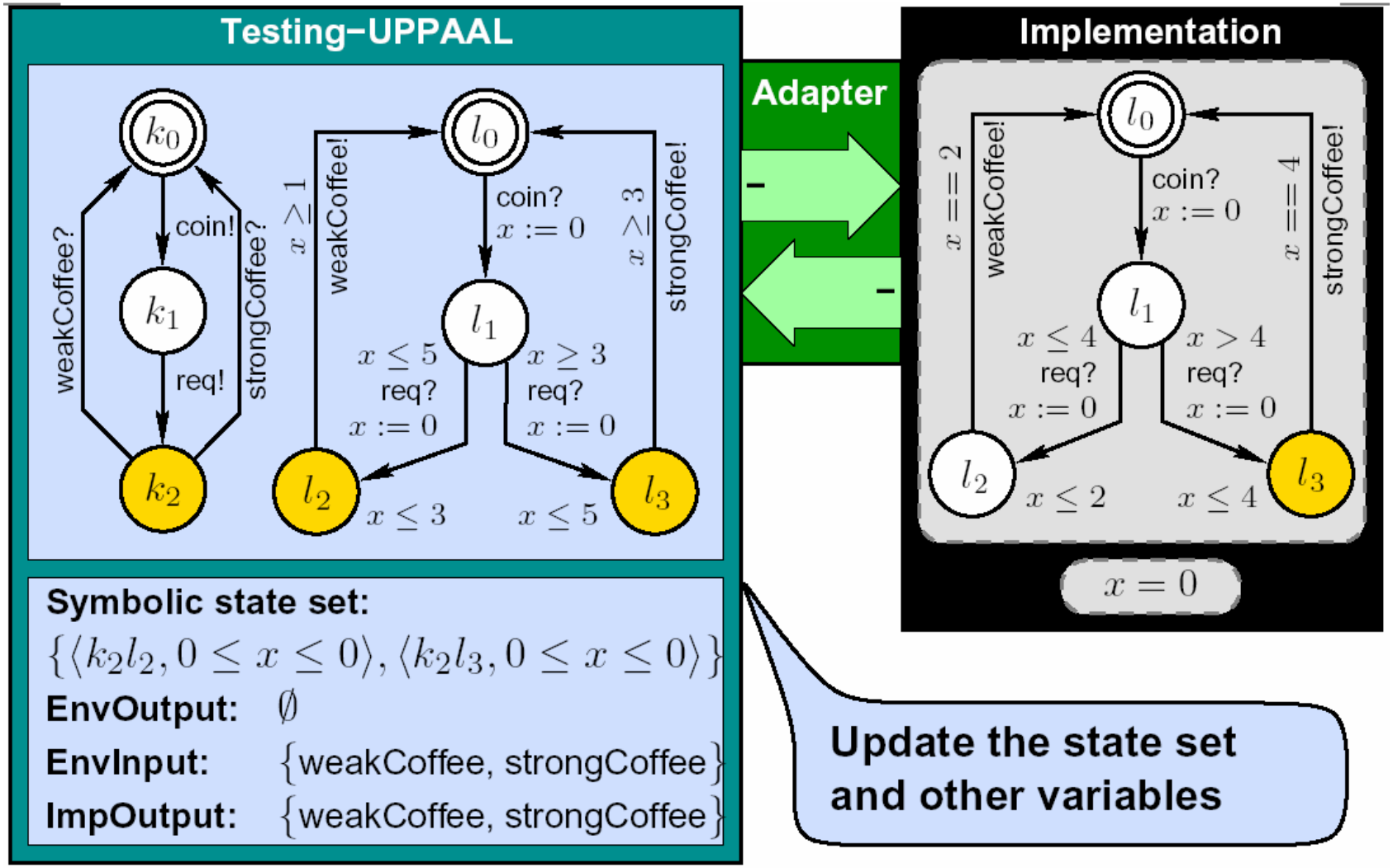
Online Testing



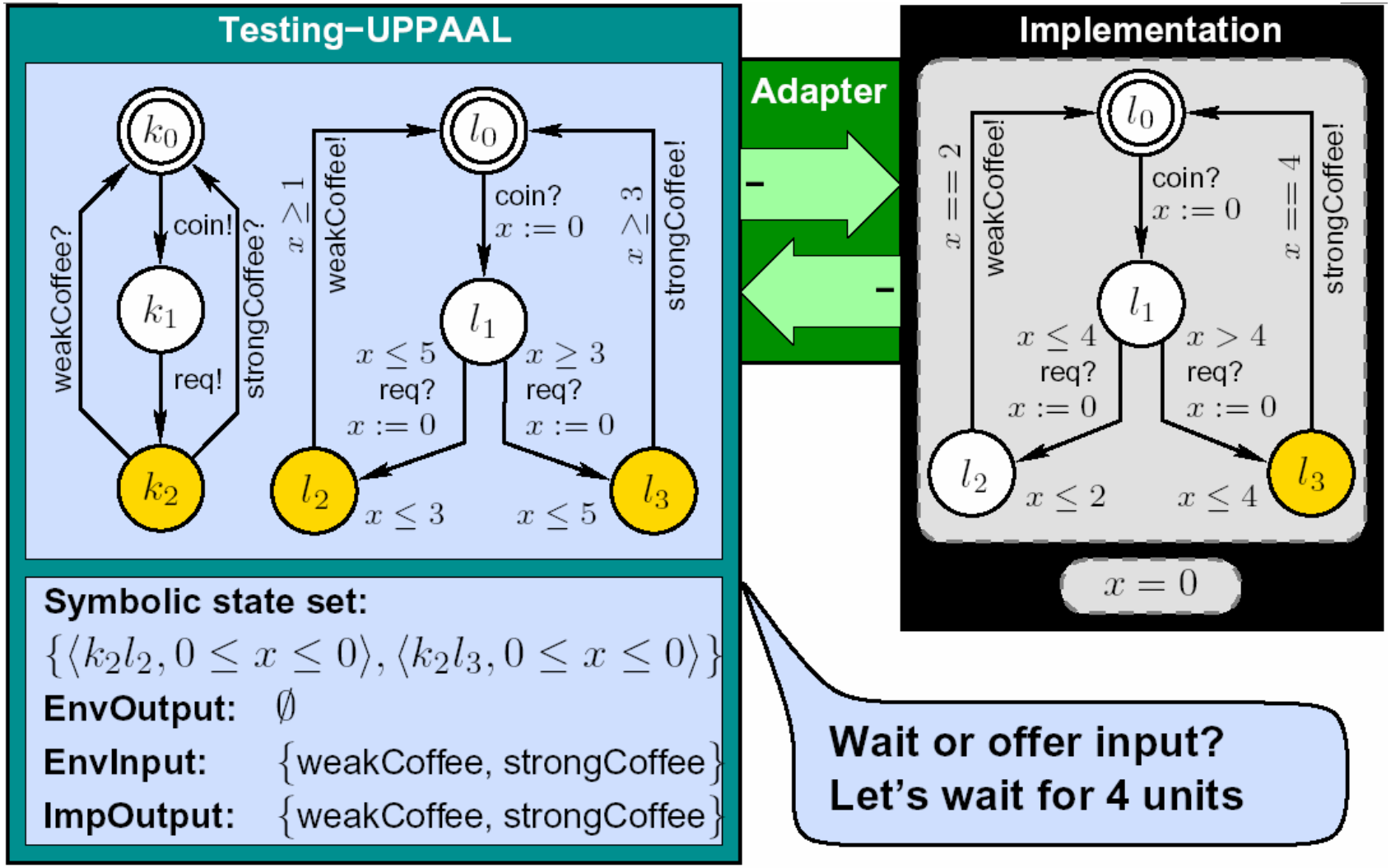
Online Testing



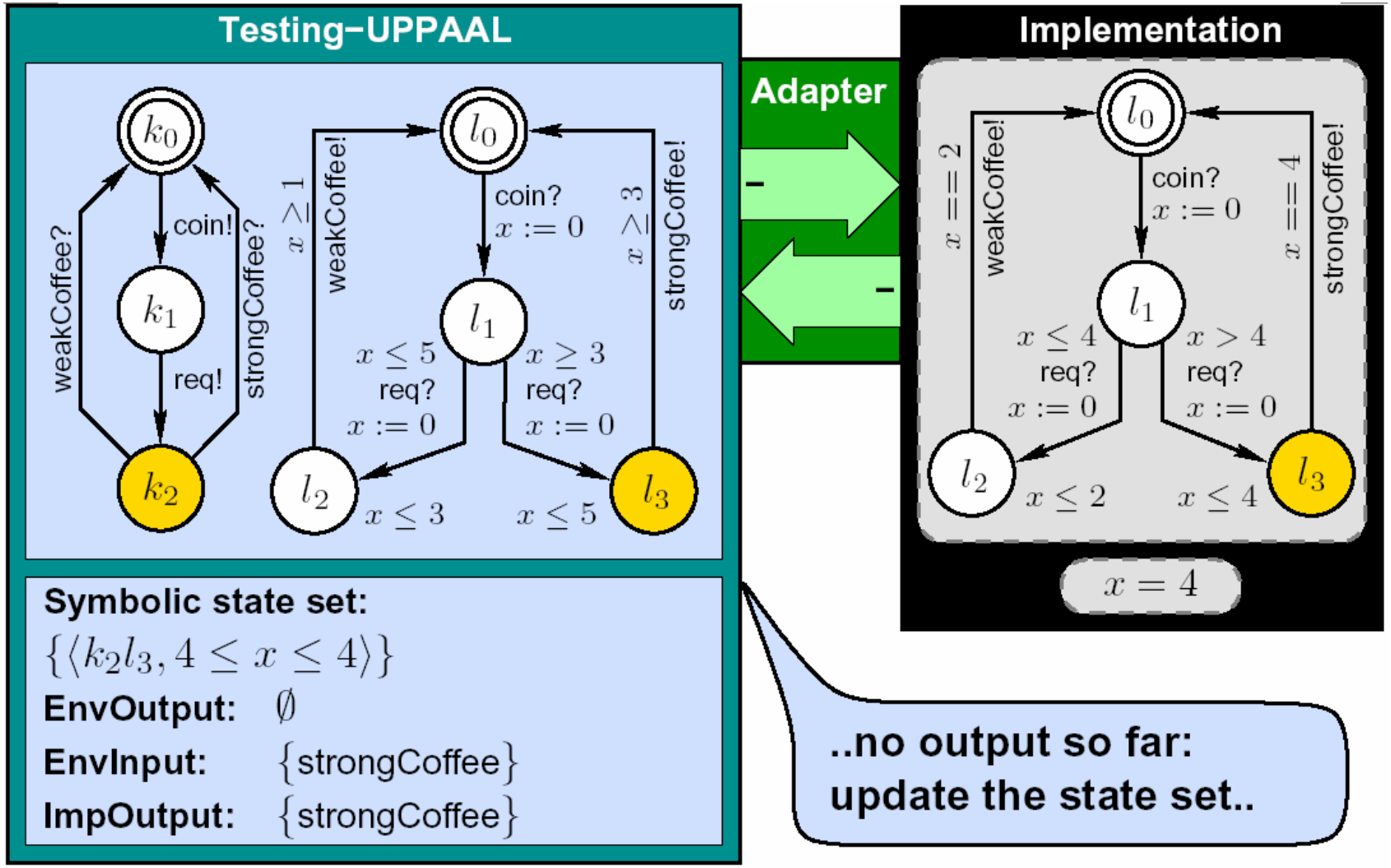
Online Testing



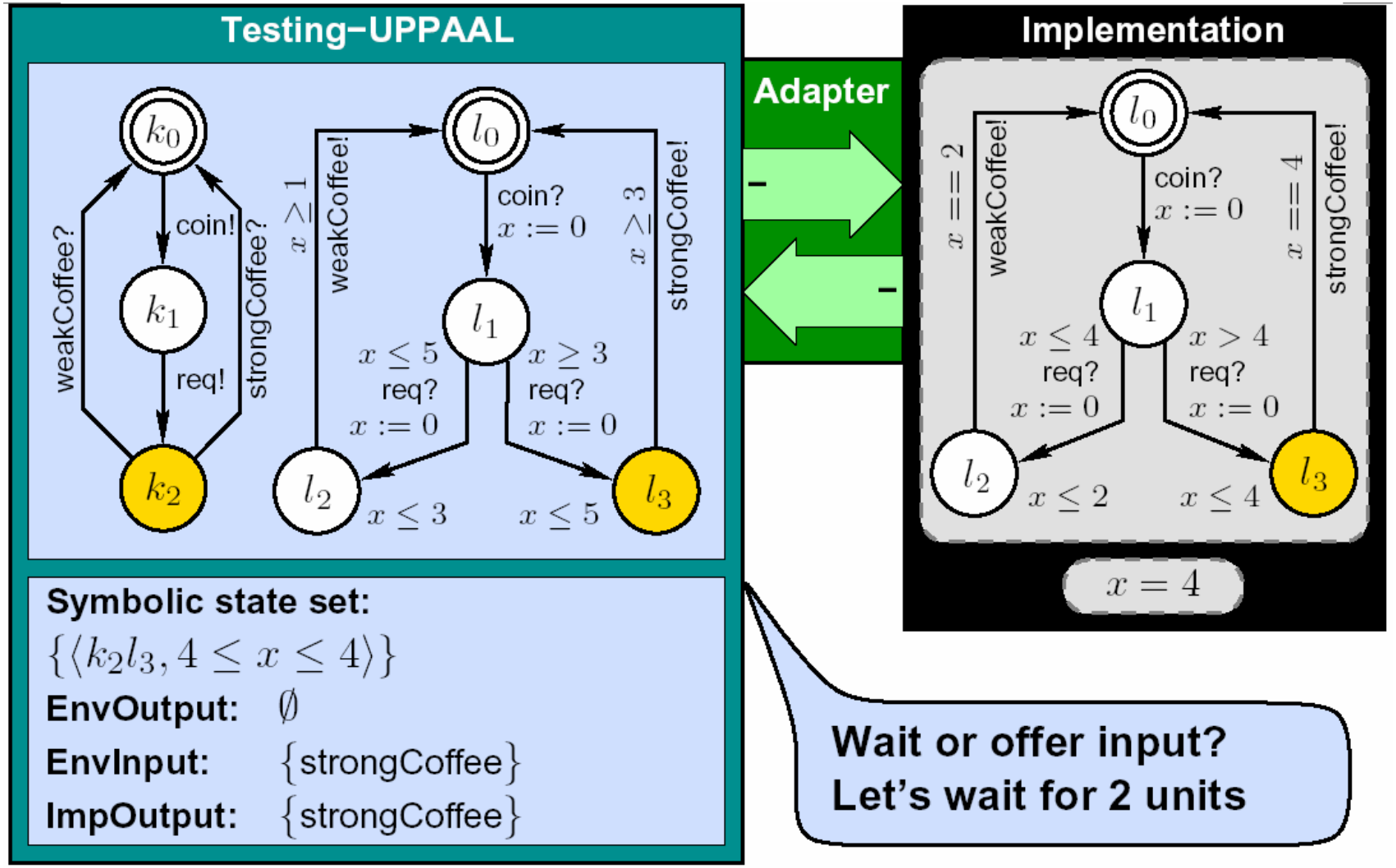
Online Testing



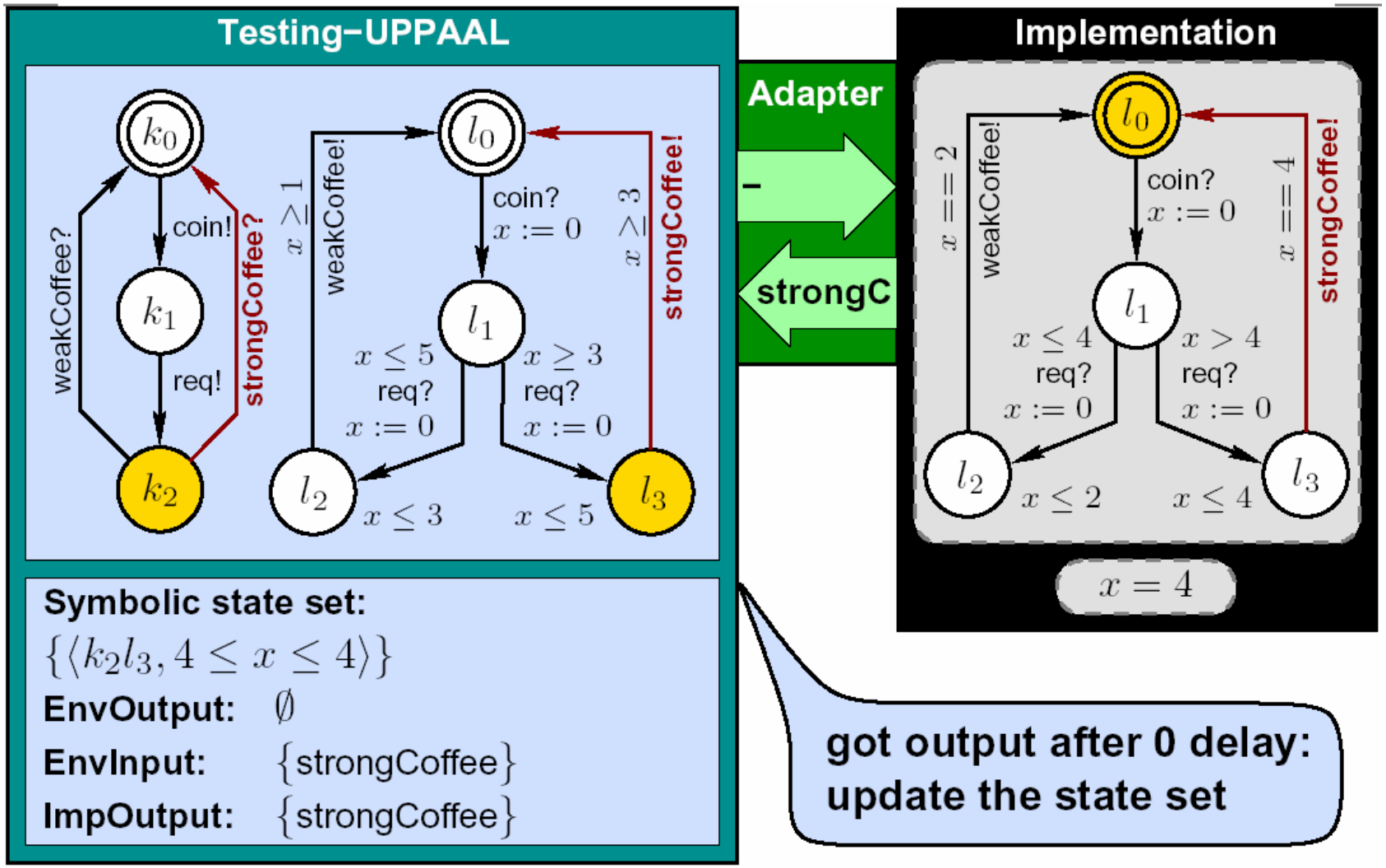
Online Testing



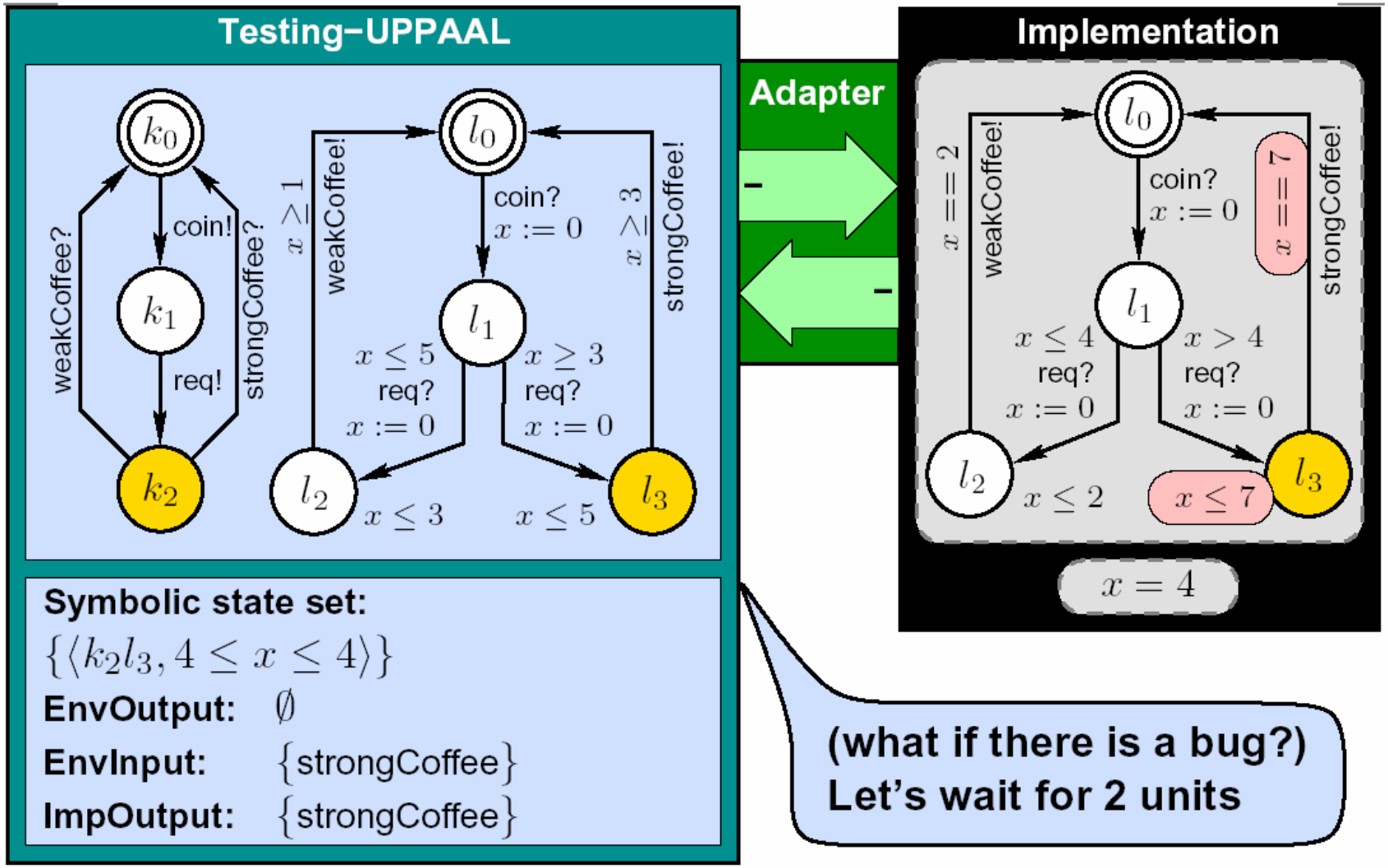
Online Testing



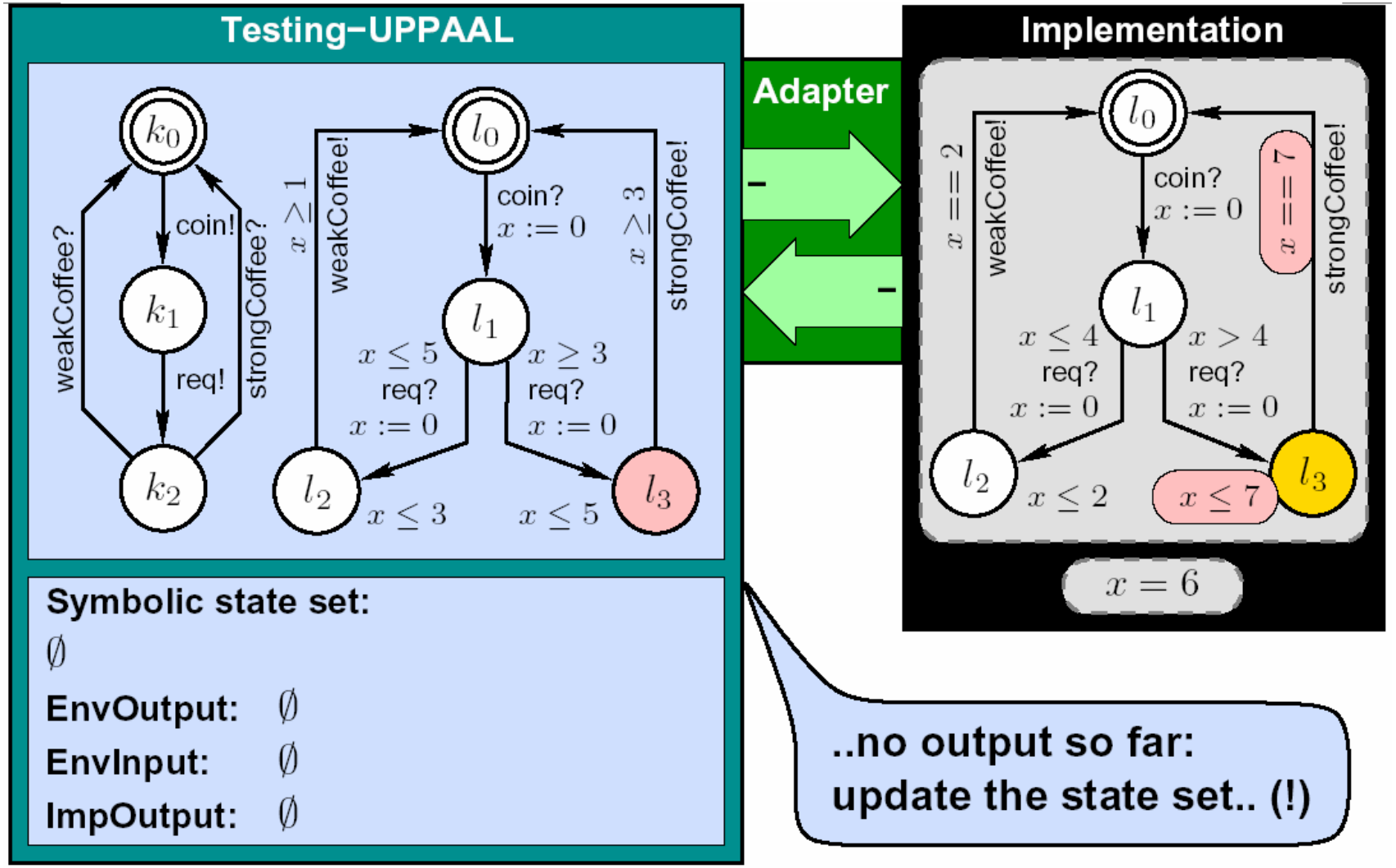
Online Testing



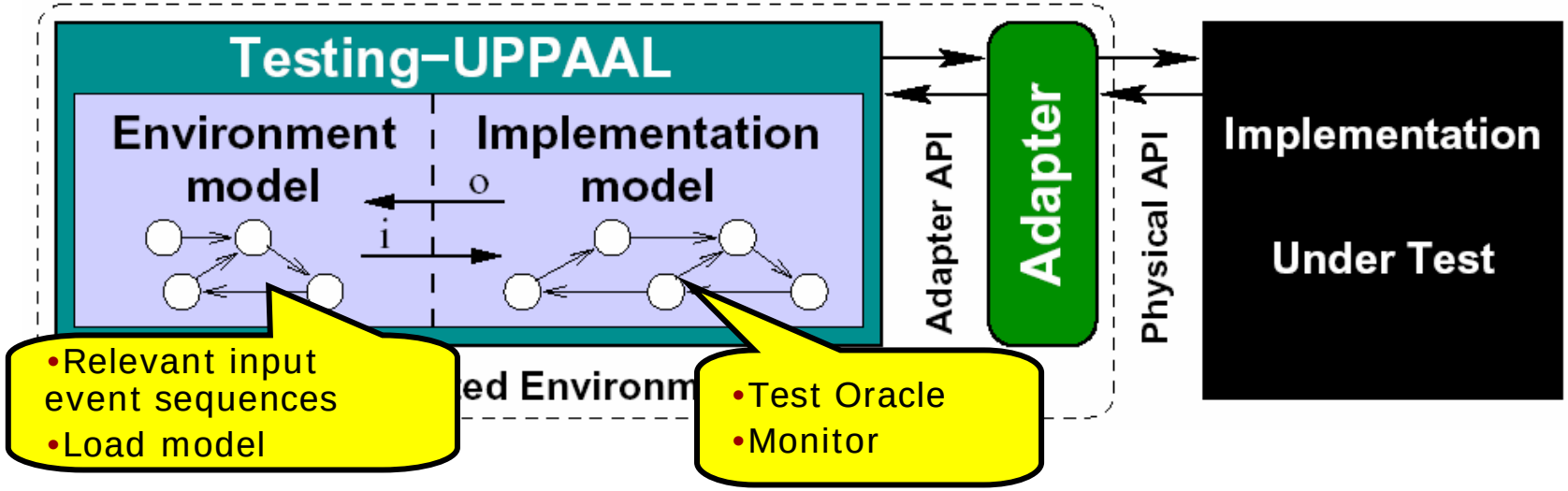
Online Testing



Online Testing



Test Specification



User Supplied Test Specification

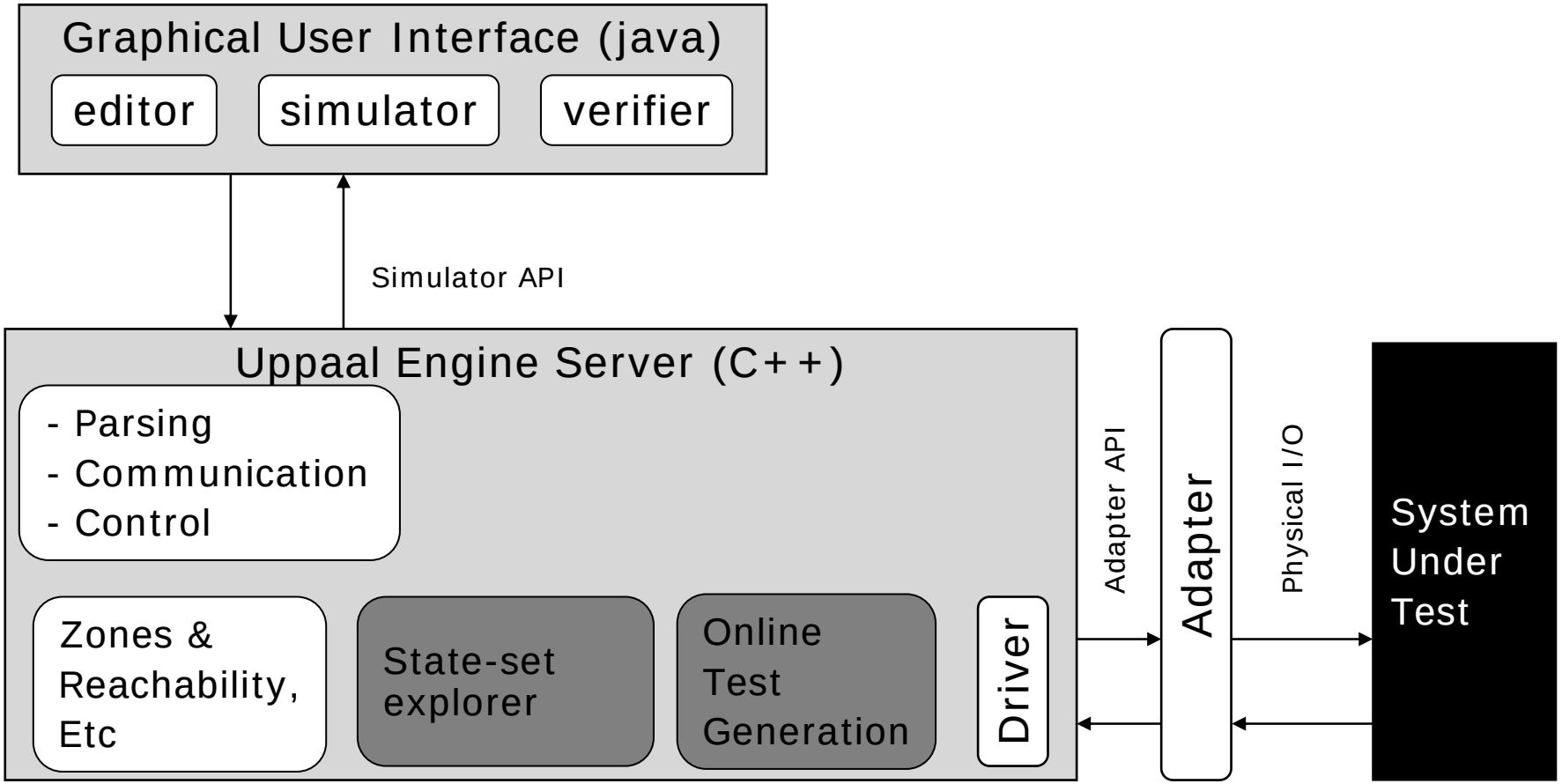
- *Closed TA Network* partitioned into *Env* and *IUT*.
  - IUT model weakly input input enabled
  - Model of *Environment*
- Designate *observable input* and *output* actions.
- Specify amount of real time per one time-unit in model.

Online Algorithm

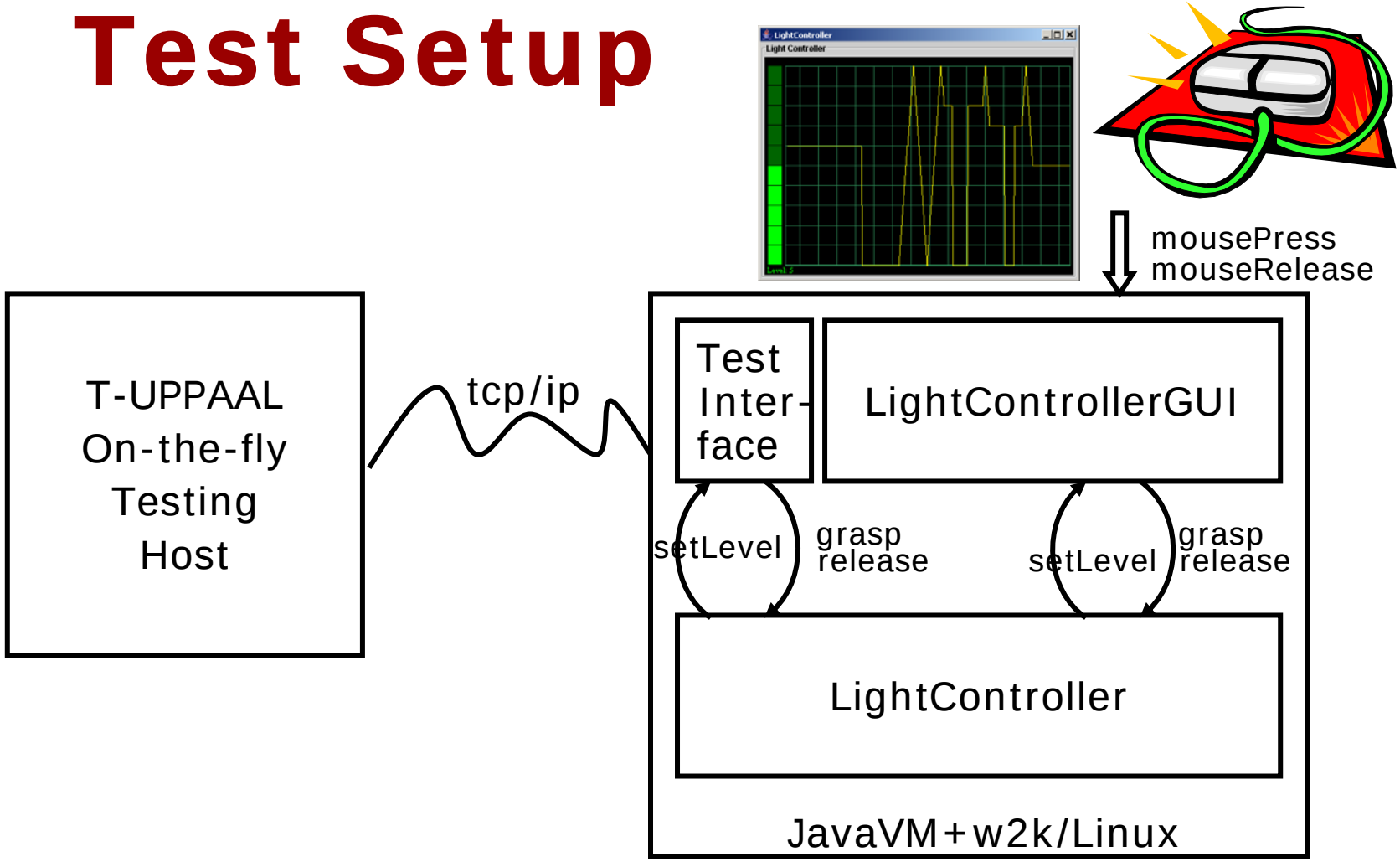
Algorithm *TestGenExec (TestSpec)* **returns** {**pass**, **fail**, **inconclusive**)

```
S:={⟨l0,0⟩}, continueTesting:= true
While continueTesting do either
    1.  i:=ChooseAction(EnvOutput(S)) // Offer an input
        send i to SUT
        Z:=After(S,i)
    2.  δ=chooseDelay(S)              // Delay and wait for output
        Wait(δ)
            if o occurred after δ' ≤ δ then
                S:=After(S, δ')
                1.  if o ∉ ImpOutput(S) then return fail
                2.  if o ∉ EnvInput(S) then return inconclusive
                S:=After(S,o)
            else                      // no output within δ time
                S:=After(S, δ)
                if S=∅ then return fail
    3.  continueTesting := false      // terminate
return pass
```

T-UppAal: implementation



Test Setup


LightController
Light Controller
mousePress
mouseRelease
T-UPPAAL
On-the-fly
Testing
Host
tcp/ip
Test
Inter-
face
LightControllerGUI
setLevel
grasp
release
setLevel
grasp
release
LightController
JavaVM+w2k/Linux

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)

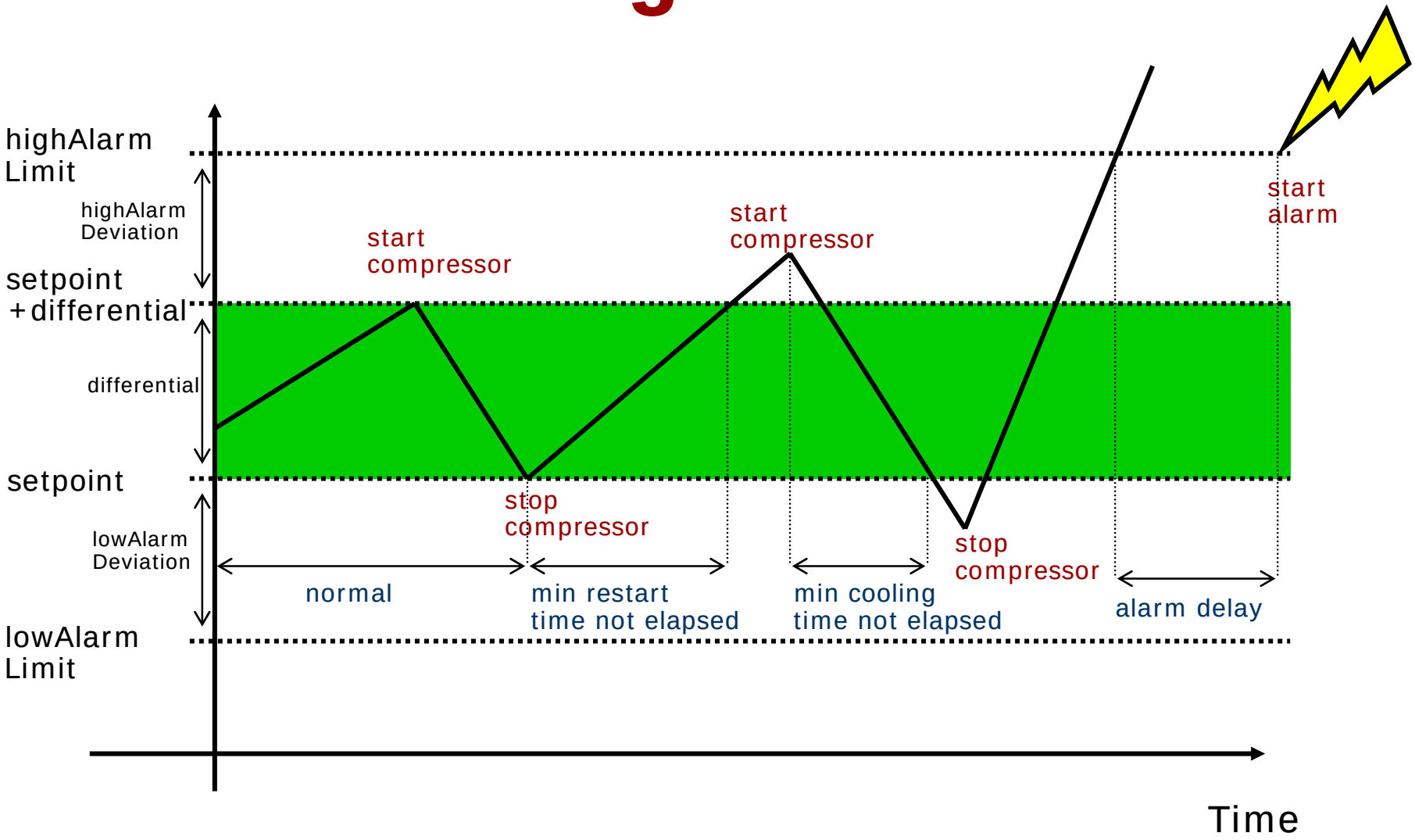
Display Output

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

- Optional real-time clock or LON network module

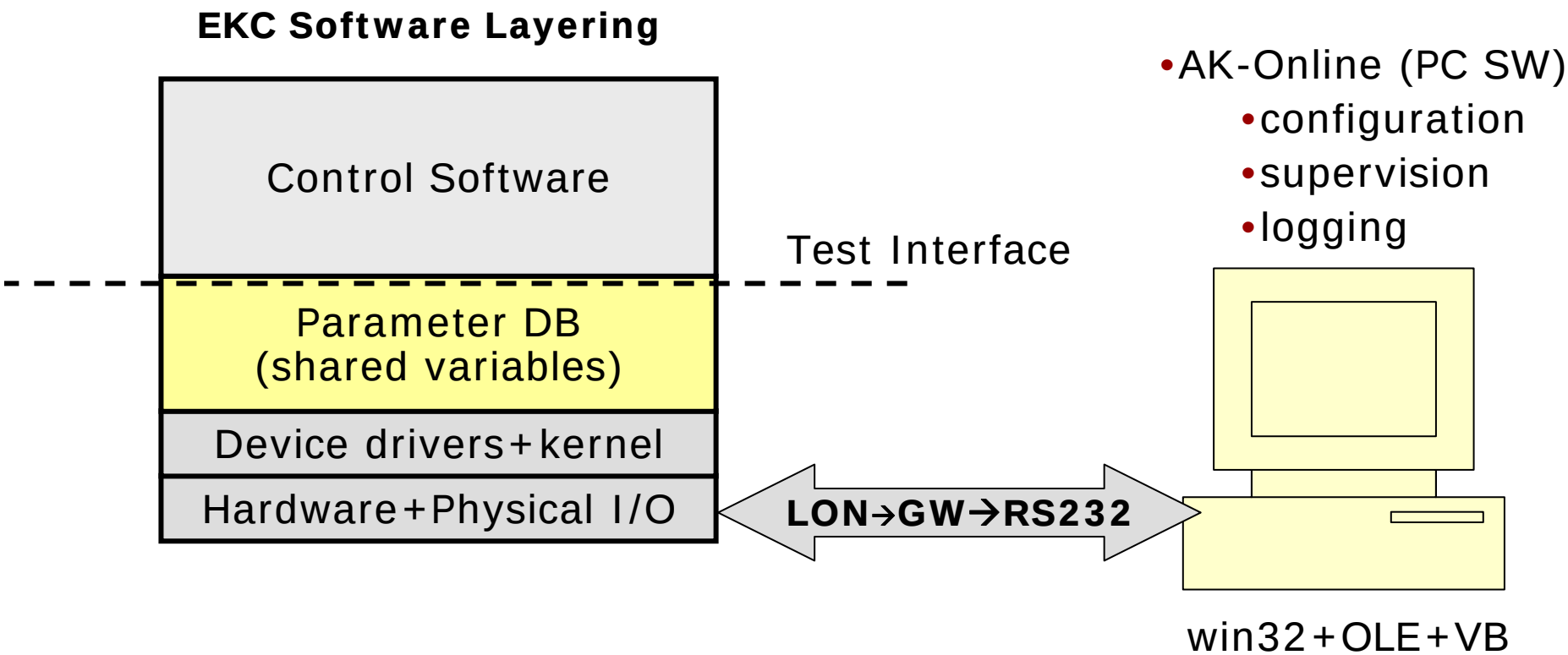
Industrial Cooling Plants

Basic Refrigeration Control

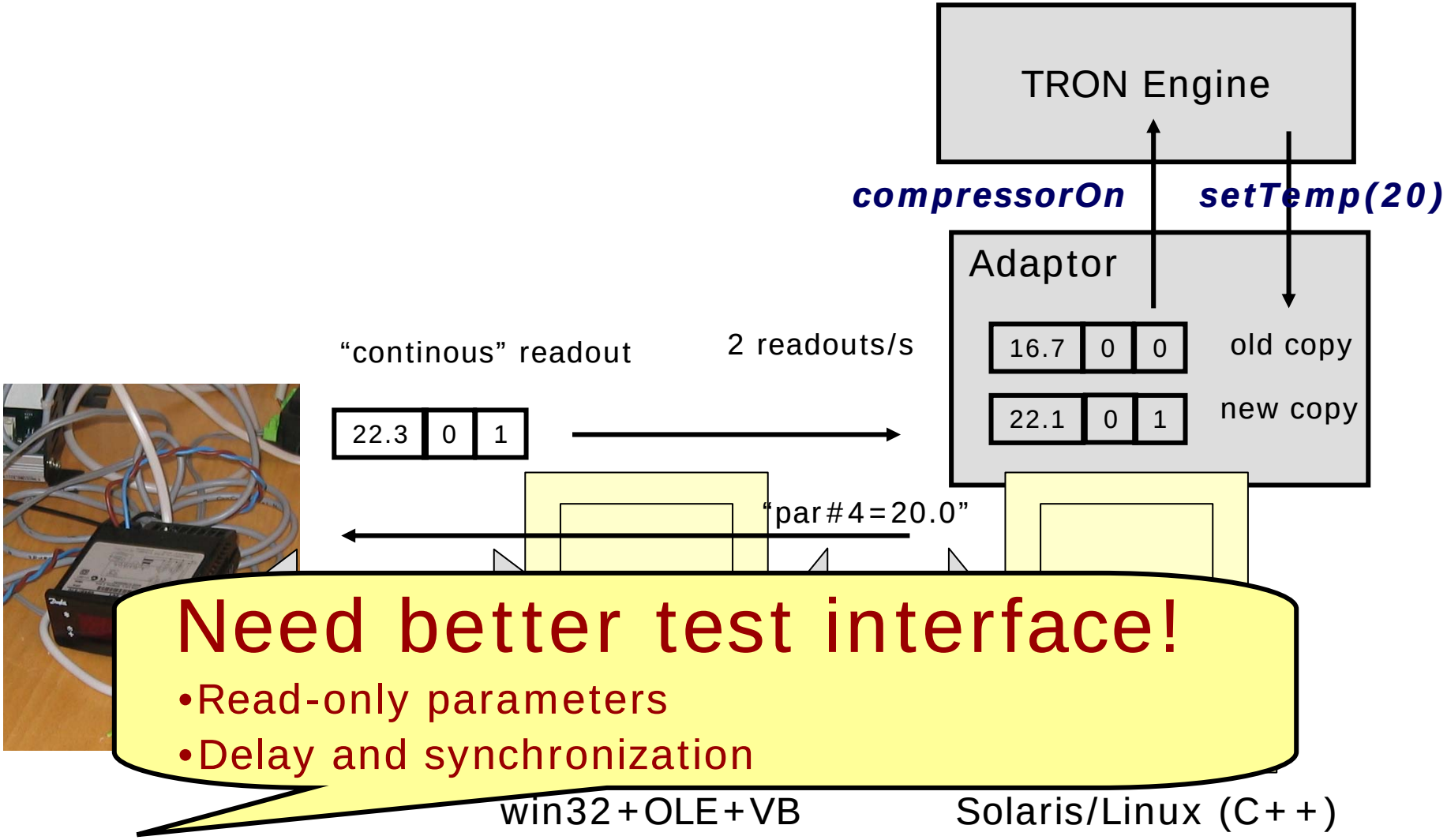


EKC Adaptation 1

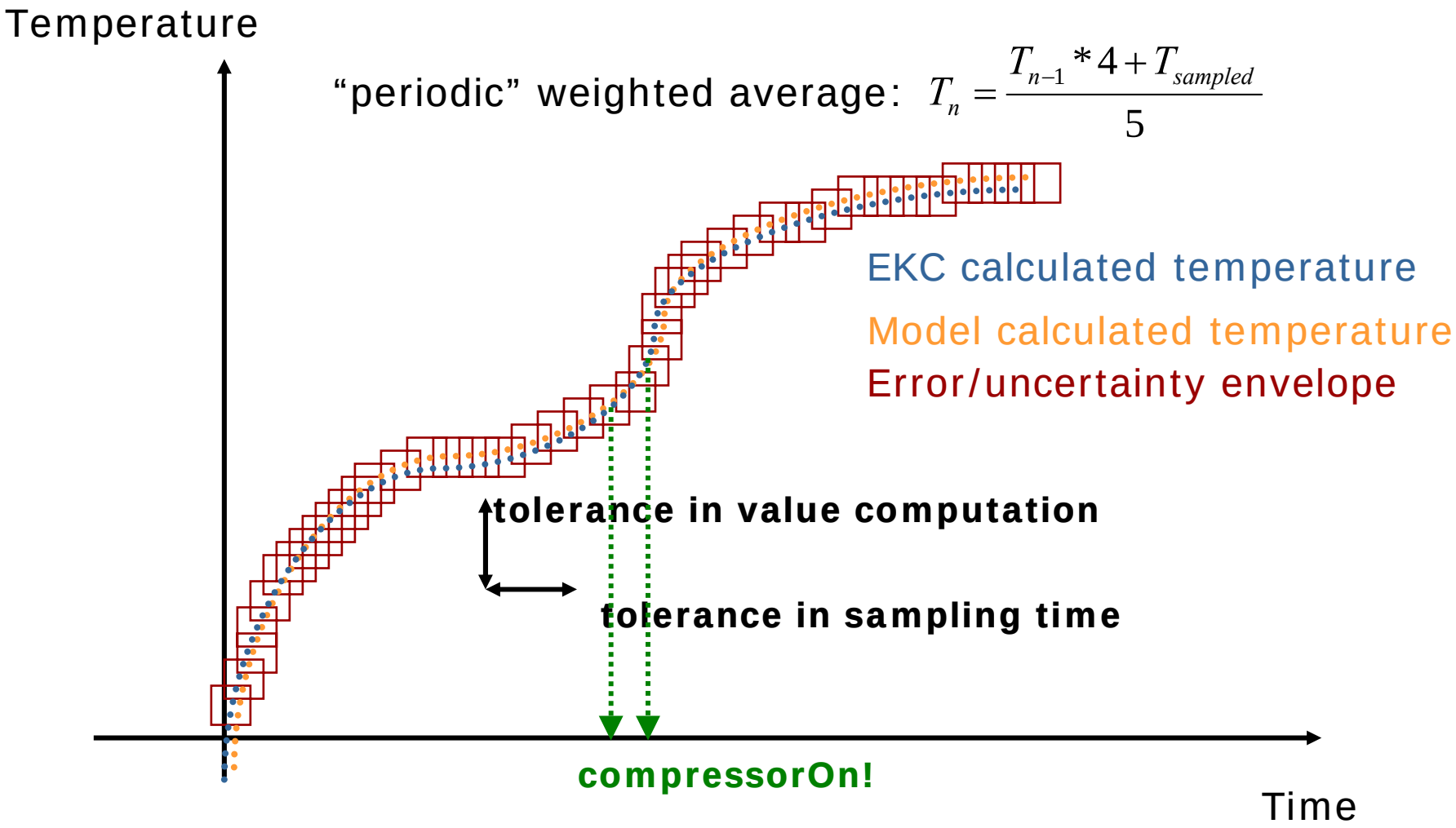
- **Read and write parameter "database"**
- **47 parameters**



EKC Adaptation 2

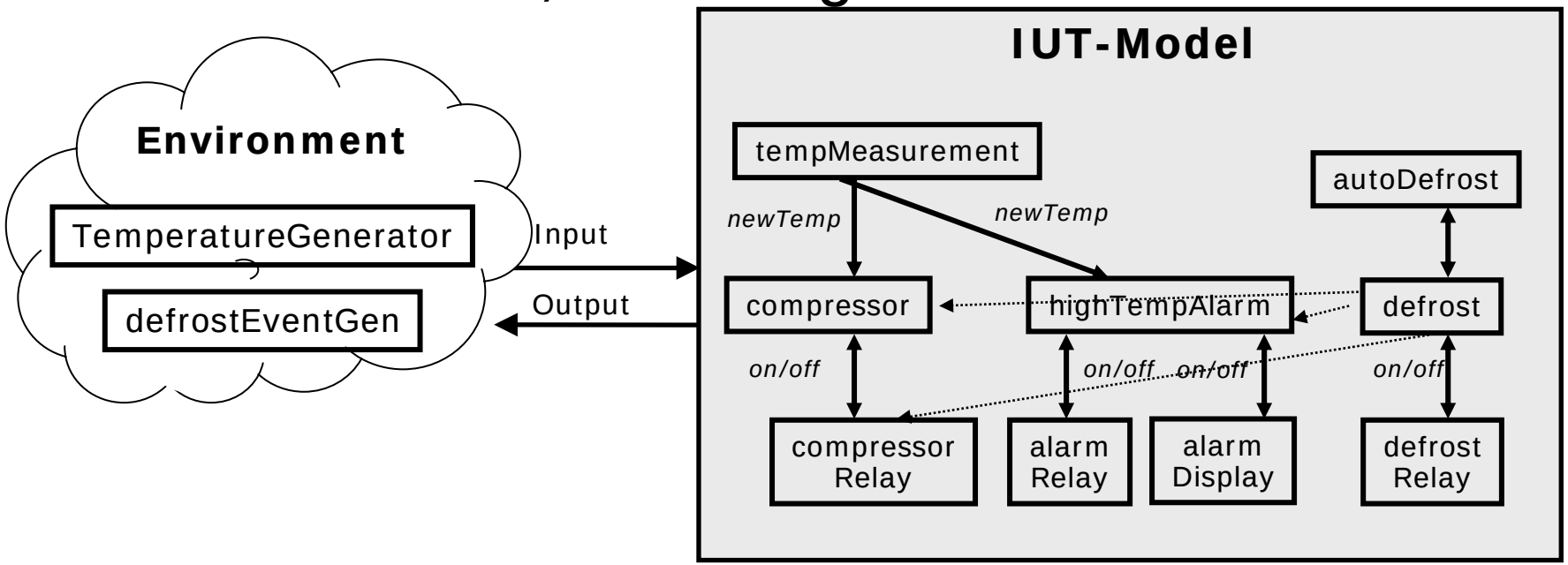


Temperature Tracking



Main Model Components

- 18 concurrent timed automata
- 14 clocks, 14 integers



Reverse Engineering

- Unclear and incomplete specifications
- Method of Working
  1. Formulate hypothesis model
  2. Test
  3. **FAIL**-verdict ⇒ Refine model
  4. **(PASS)** ⇒ Confirm with Danfoss
- Detects differences between actual and modeled behavior
- ***Indicates promising error-detection capability***
- 4 examples

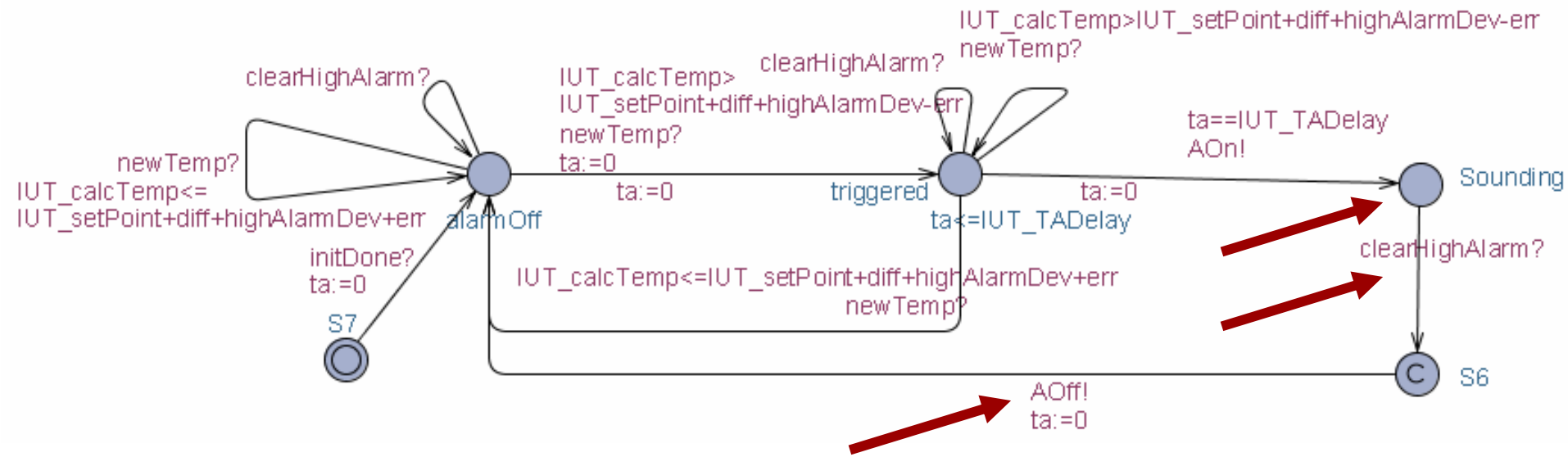
Ex1: Control Period

- Control actions issued when ”calculatedTemp” crosses thresholds

“periodic” weighted average: $$T_n = \frac{T_{n-1} * 4 + T_{sampled}}{5}$$

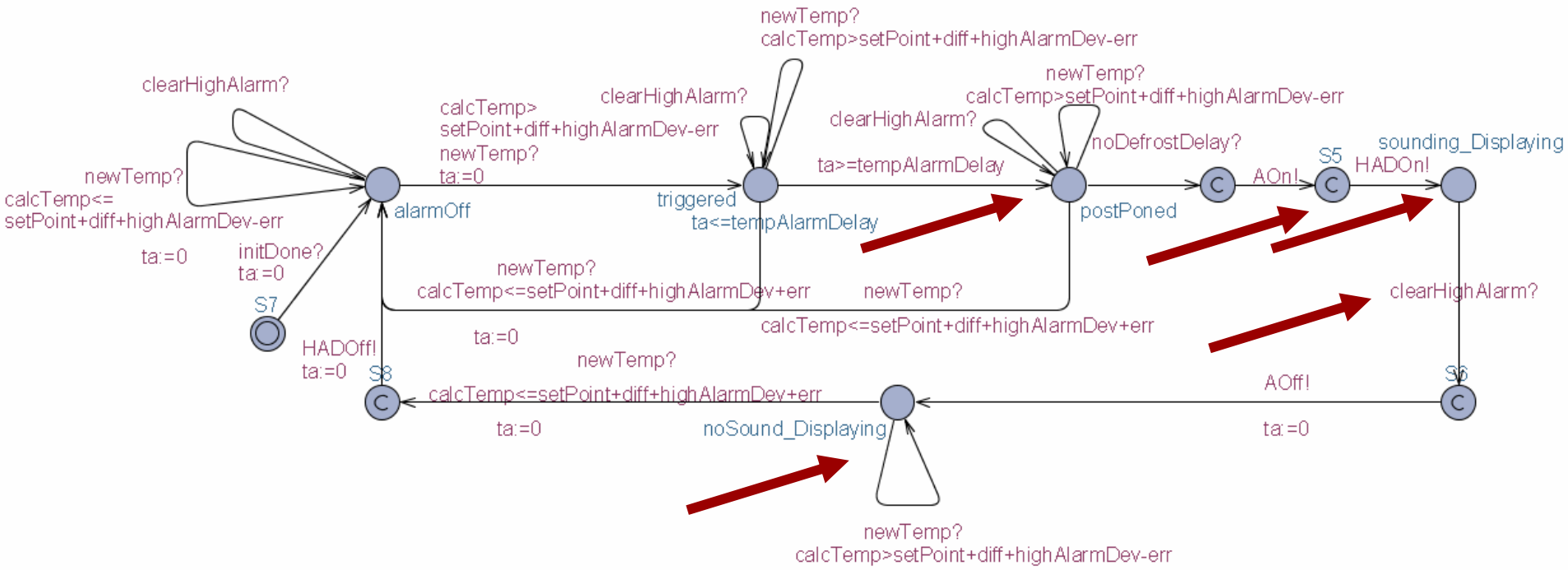
- No requirements on period given
- Tested to be 1.2 seconds

Ex2: High Alarm Monitor v1



Clearing the alarm do not switch off alarm state, only alarm relay

Ex2: High Alarm Monitor v2



- Add HighAlarmDisplay action
- Add location for “noSound, but alarmDisplaying”
- (Postpone alarms after defrosting)

Ex3: Defrosting and Alarms

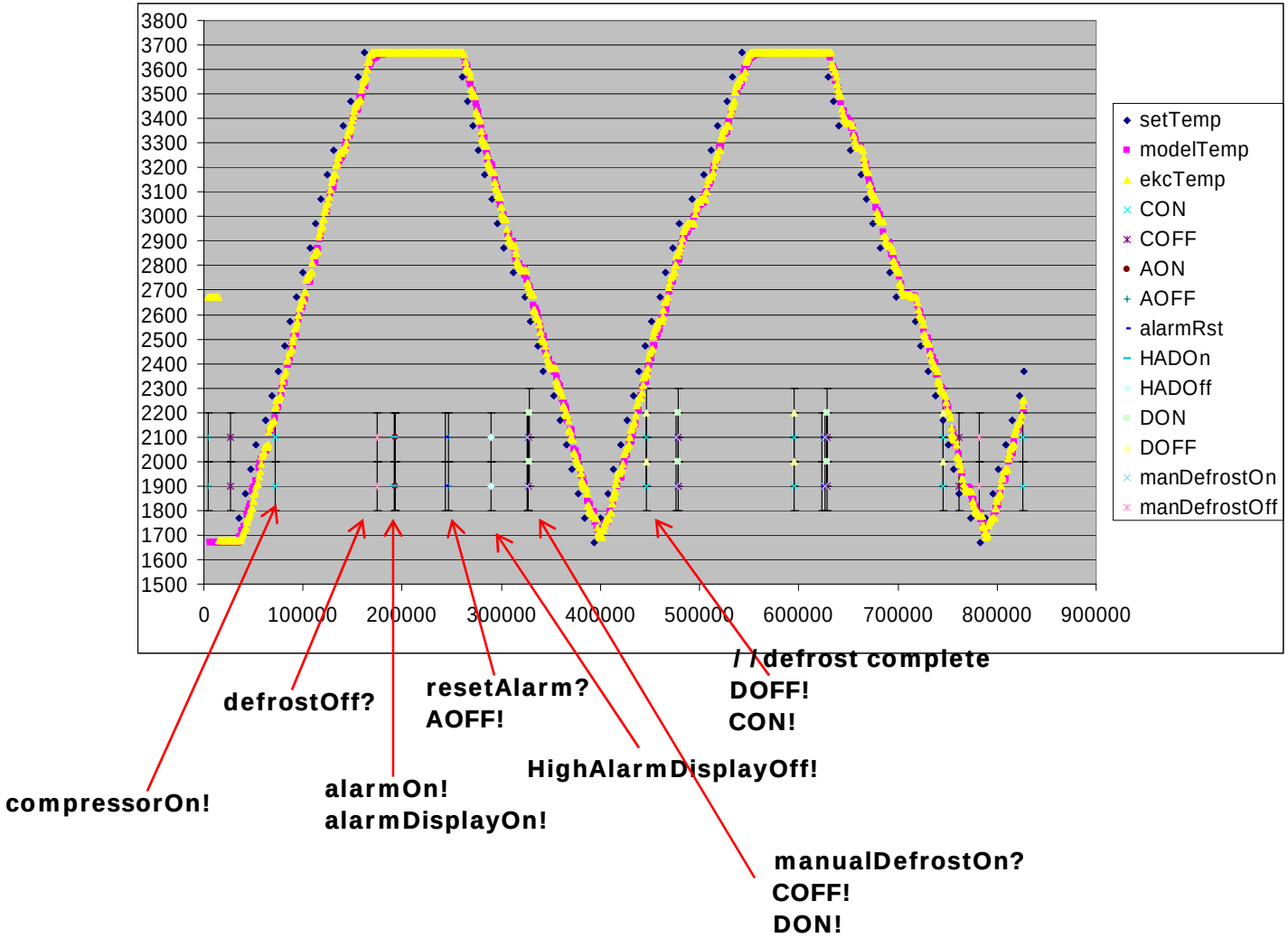
- When defrosting the temperature rises
- Postpone high temperature alarms during defrost
- System parameter alarmDelayAfterDefrost
- Several Interpretations
  1. Postpone alarmDelayAfterDefrost+alarmDelay after defrost?
  2. Postpone alarmDelayAfterDefrost+alarmDelay after highTemp detected?
  3. Postpone alarmdelayAfterDefrost until temperature becomes low; then use alarmDelay
- Option 3 applies!

Ex4: Defrost TimeTolerance

- Defrost relays engaged earlier and disengaged later than expected
- Assumed 2 seconds tolerance
- Defrosting takes long time
- Implementation uses a low resolution timer (10 seconds)

Example Test Run

(log visualization)



END

BRICS
Basic Research
in Computer Science