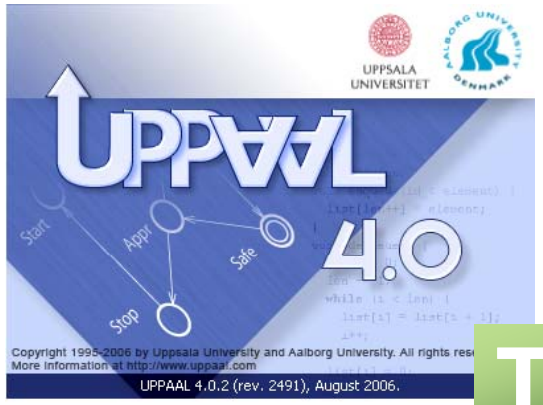


TRON

Real Time Testing

using UPPAAL

With

Shuhao Li, Mariius Mikucionis, Brian Nielsen, Arne Skou, Paul Pettersson, Jacob Illum Rasmussen

AALBORG UNIVERSITY DENMARK

Overview

- Introduction
- Off-line Test Generation
  - Controllable Timed Automata
  - Observable Timed Automata

CLASSIC CORA TIGA

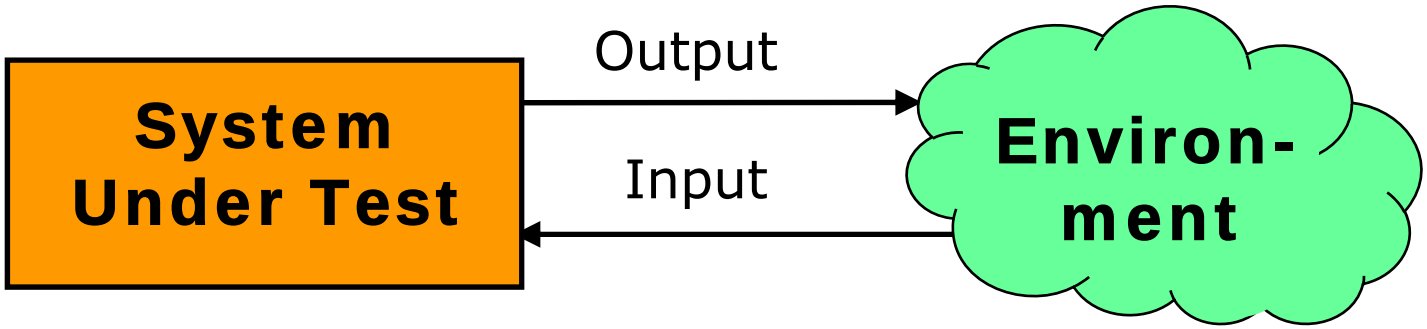
- On-line Test Generation

TRON

- Conclusion and Future Work

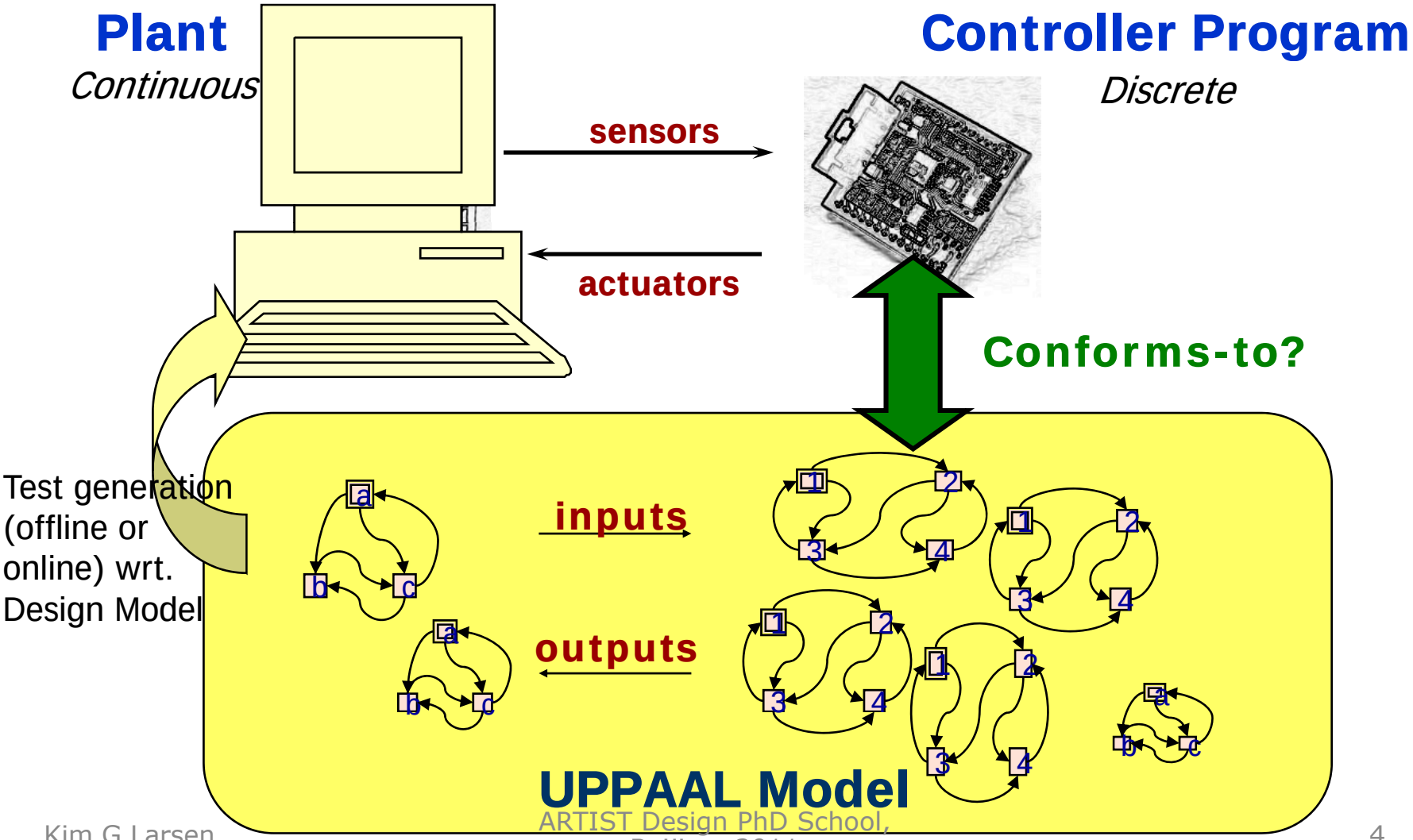
Testing

- Primary validation technique used in industry
  - In general avg. 10-20 errors per 1000 LOC
  - 30-50 % of development time and cost in embedded software
- To find errors
- To determine risk of release
- Part of system development life-cycle



- Expensive, error prone, time consuming (for Real-Time Systems)
- UPPAAL model can be used to generate test specifications

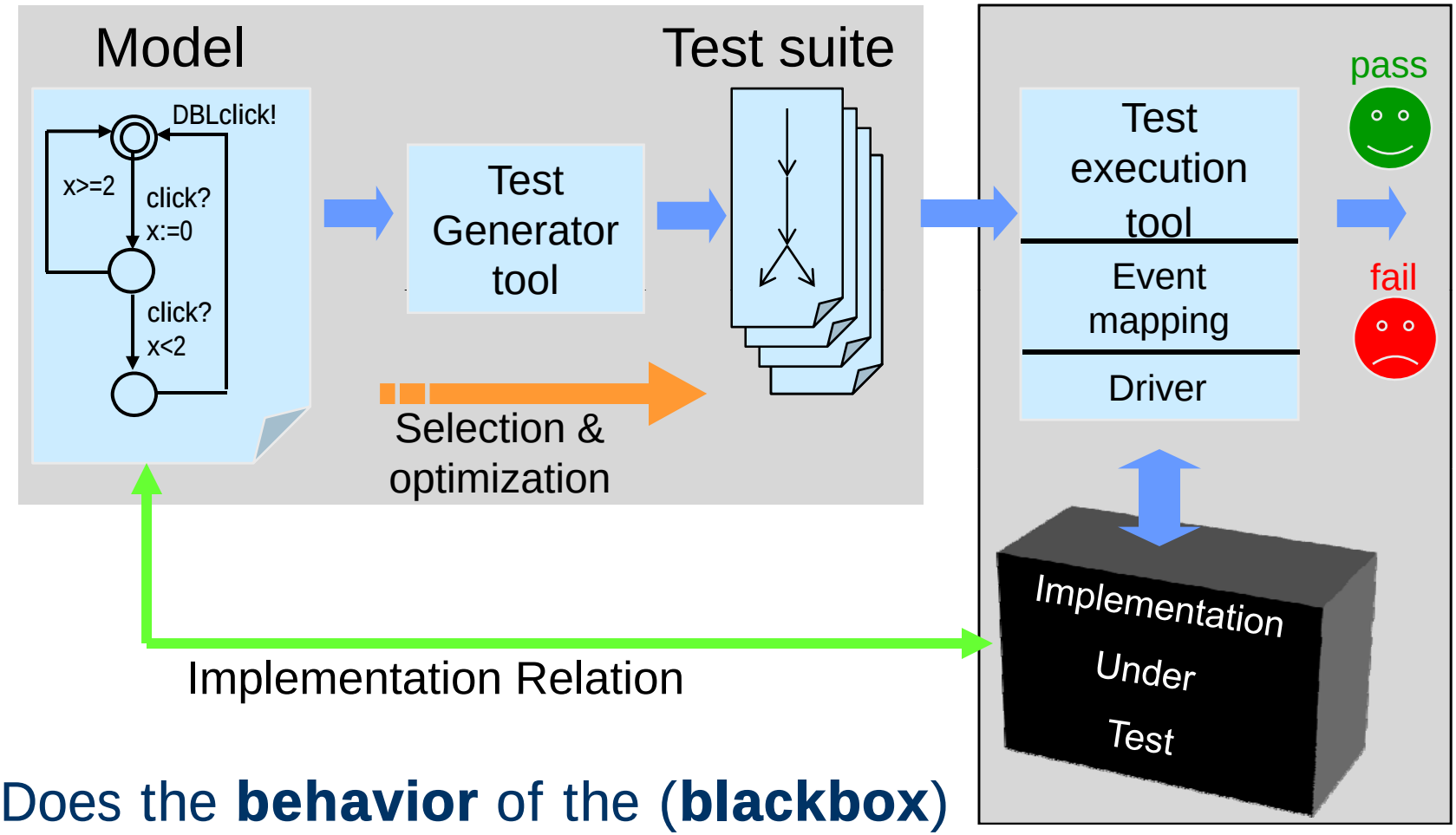
Real-time Model-Based Testing



Off-Line Test Generation

Controllable Timed Automata

Model Based Conformance Testing



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

Controllable Timed Automata

Input Enabled:
all inputs can always be accepted.

Assumption about model of SUT

Output Urgent:
enabled outputs will occur immediately.

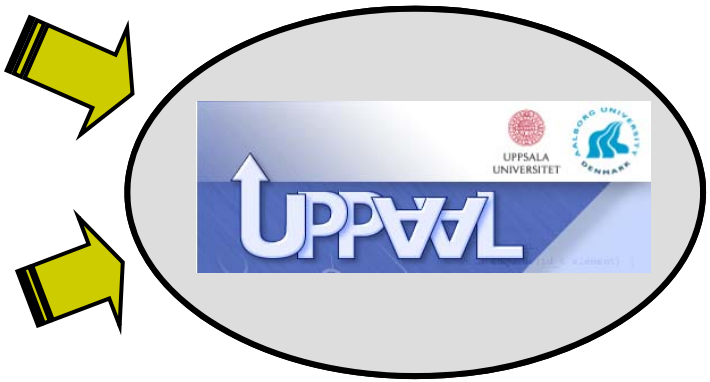
Determinism:
two transitions with same input/output leads to the same state.

Isolated Outputs:
if an output is enabled, no other output is enabled.

Test Generation
using Verification

myGearControl.xml

System model



Test purpose
Property

E<> Gear.Gear5

Trace
(witness)

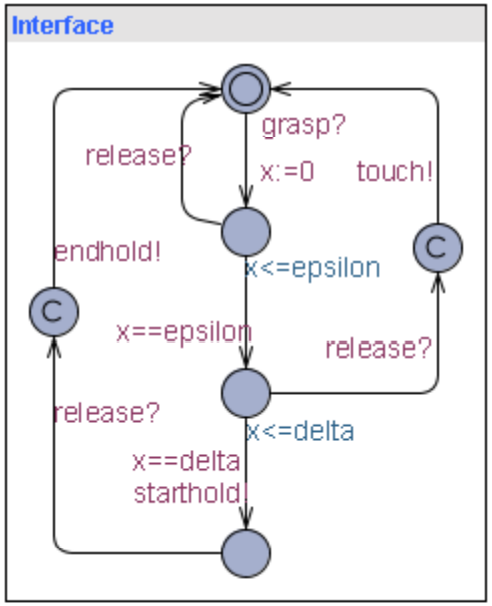
Some
Random
Shortest
Fastest

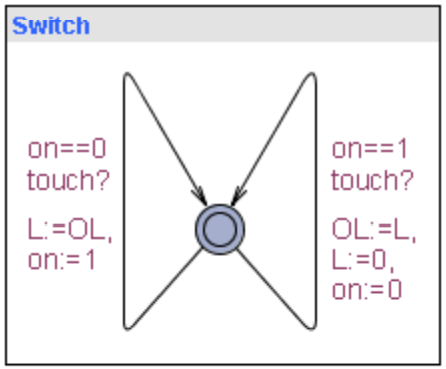
testGear5.trc

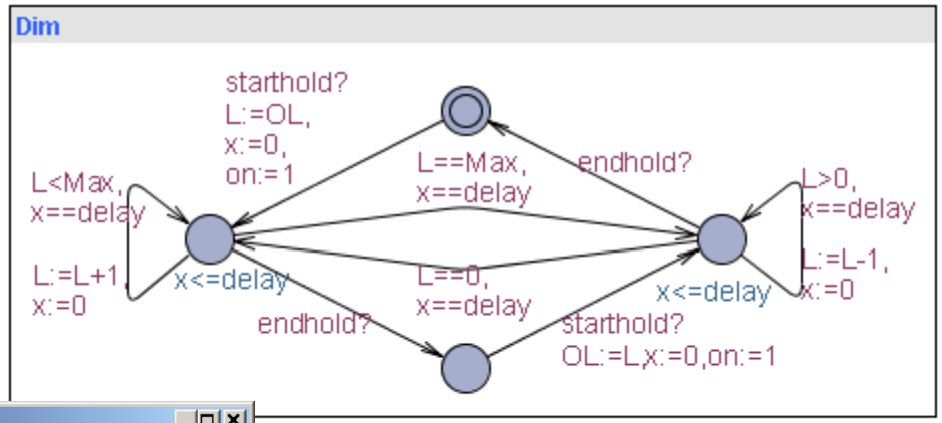
Use trace scenario as test case??!!

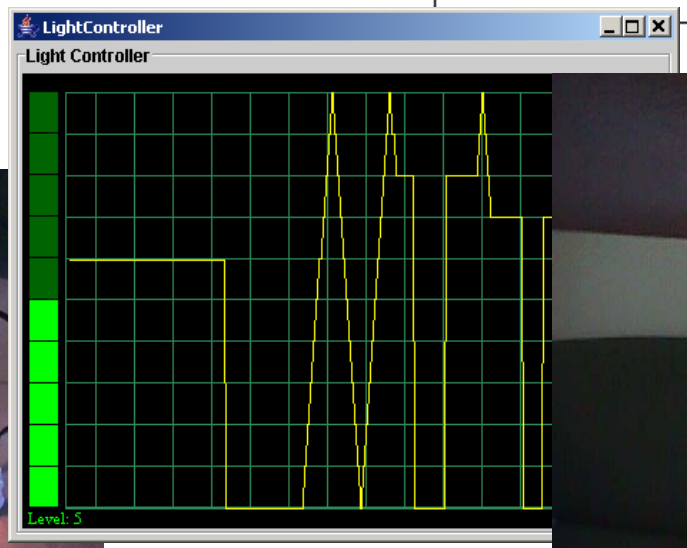
Example

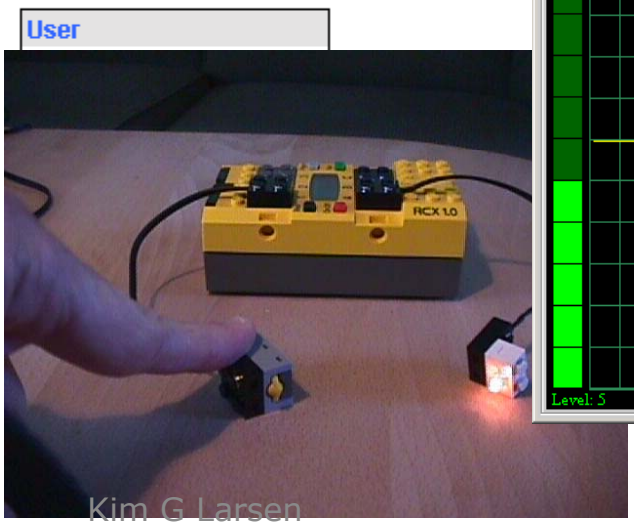
Light Controller





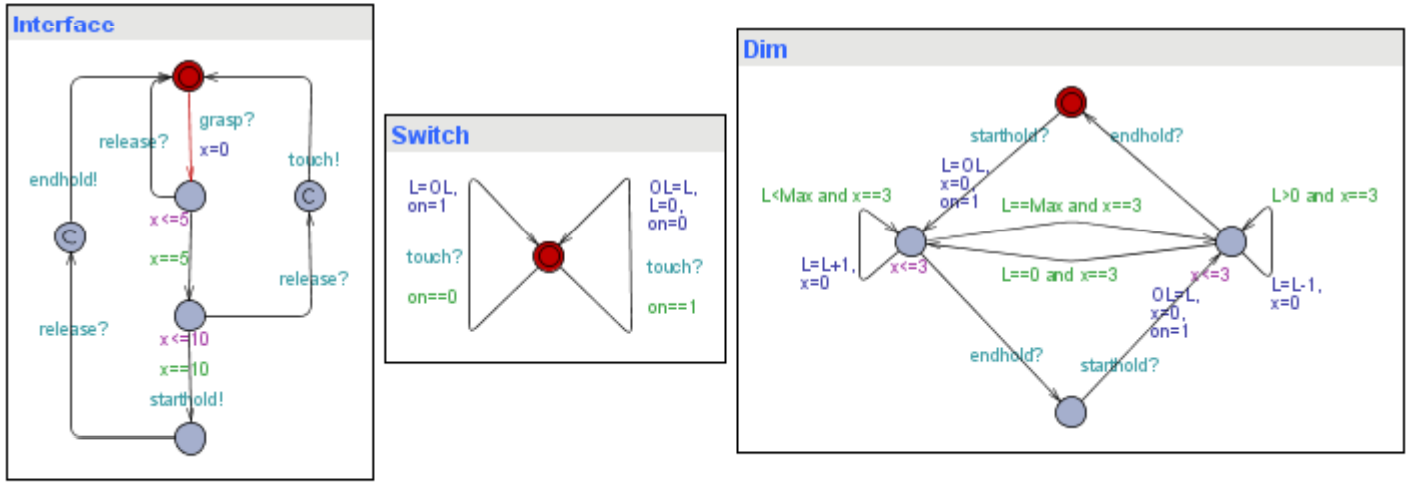






Test Purposes

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



TP: Check that the light can become bright:

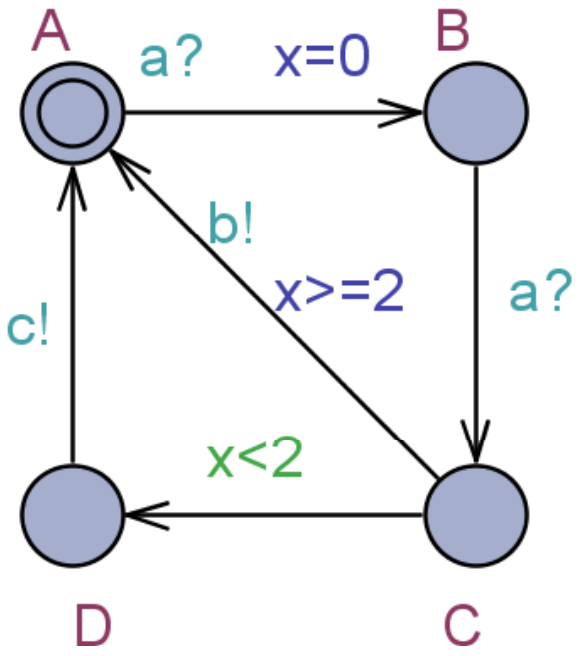
E<> L==10

```
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);
```

Shortest - and fastest –test !

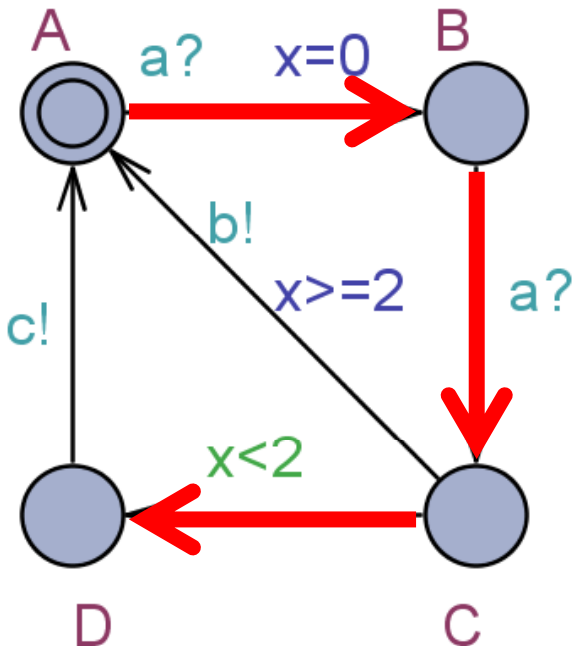
Coverage Based Test Generation

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



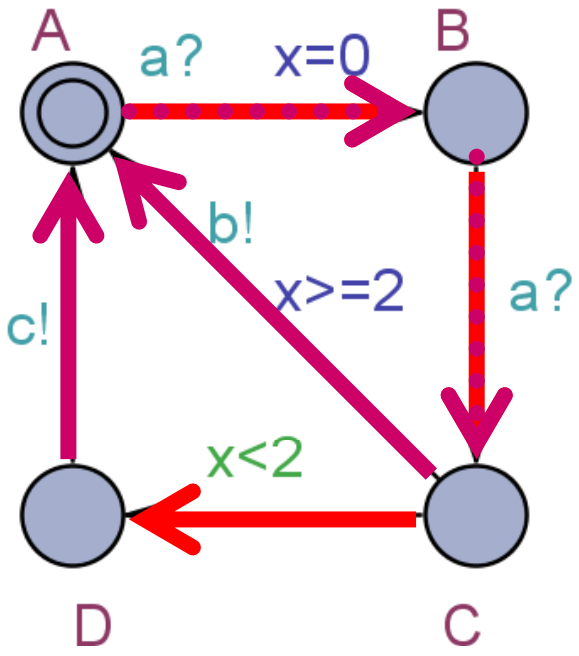
Coverage Based Test Generation

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



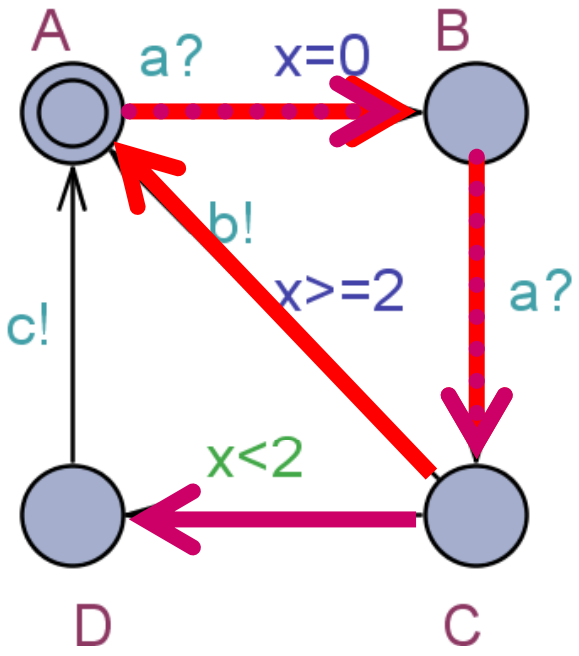
Coverage Based Test Generation

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



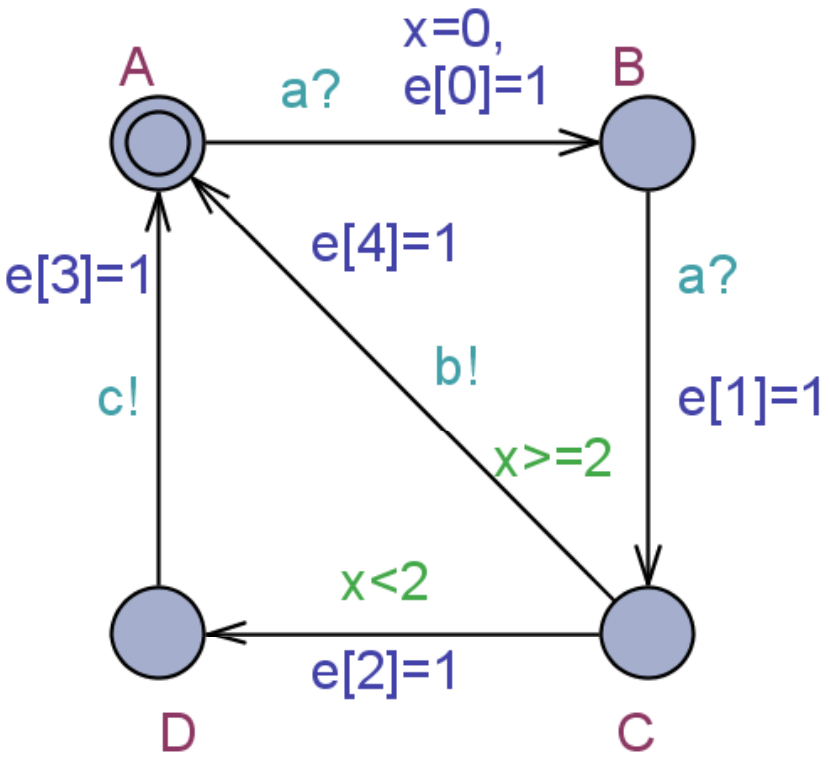
Coverage Based Test Generation

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



Edge Coverage

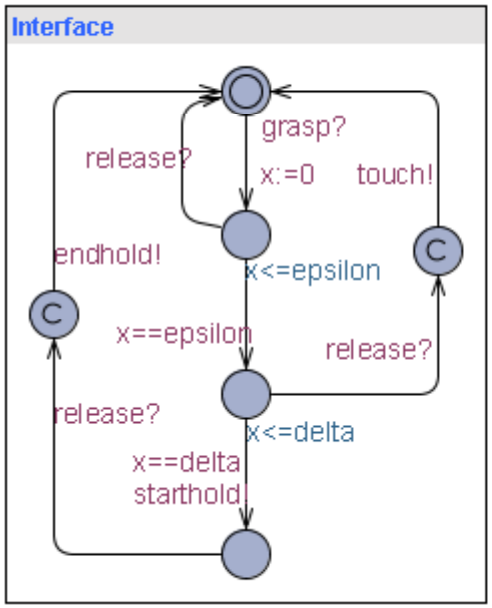
- Test sequence traversing all edges
- Encoding:
  - Enumerate edges
    $e_0, \ldots, e_n$
  - Add auxiliary variable **e[i]** for each edge
  - Label each edge **e[i]:=1**

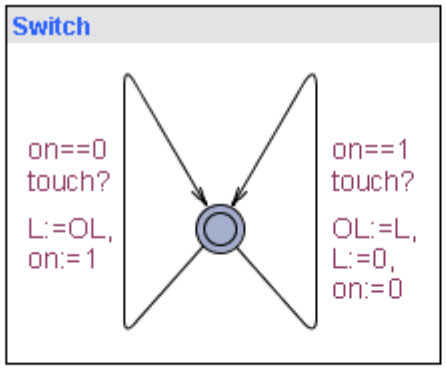


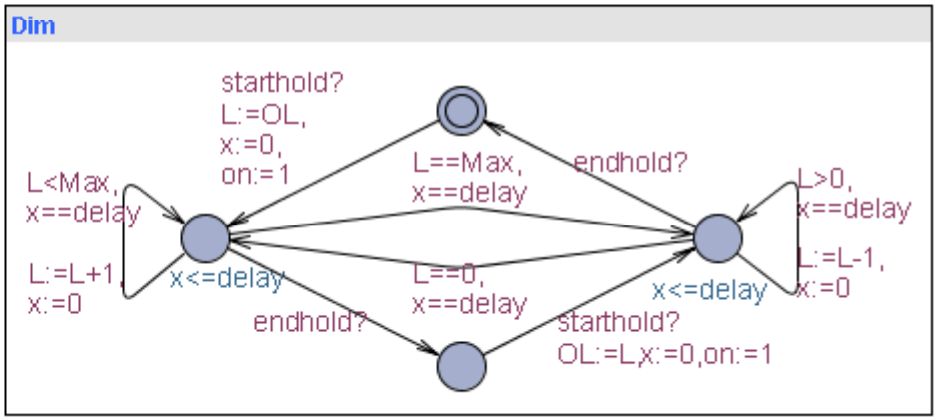
- Check:
  E<>(e[0]=1 ∧ … ∧ e[n]=1)

Fastest Edge Coverage

Time=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);
```

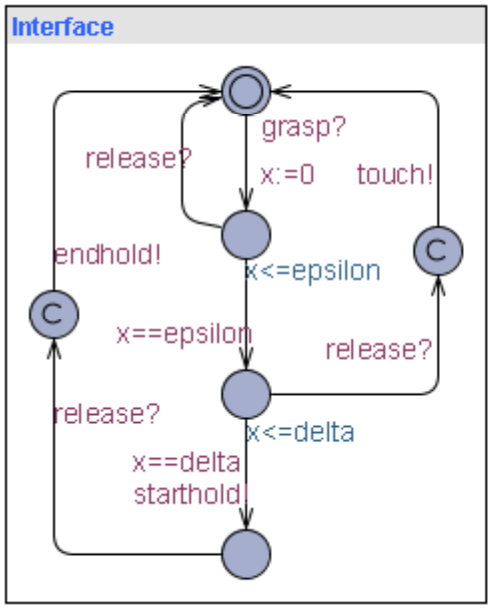
Page 1

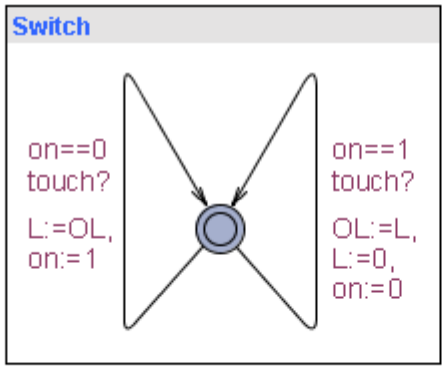
```
//13
out(IGrasp); //@900         // Bring dimmer  PassiveDn->ActiveDN->
silence(500);//hold         // ActiveUP+increase to level 10
silence(1000); in(OSetLevel,1);
silence(1000); in(OSetLevel,2);
silence(1000); in(OSetLevel,3);
silence(1000); in(OSetLevel,4);
silence(1000); in(OSetLevel,5);
silence(1000);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

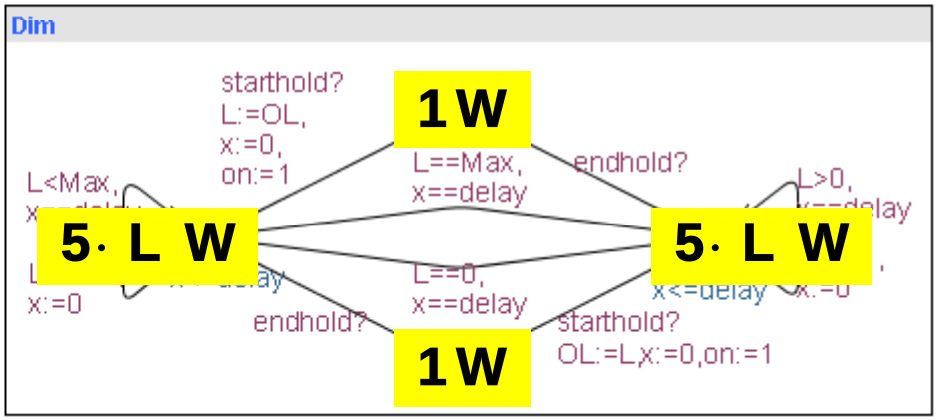
Page 2

Power-Optimal Edge Coverage

Cost=320 J








```
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);
```

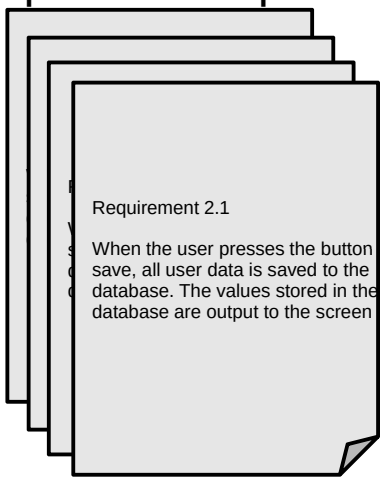
Page 1

```
//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

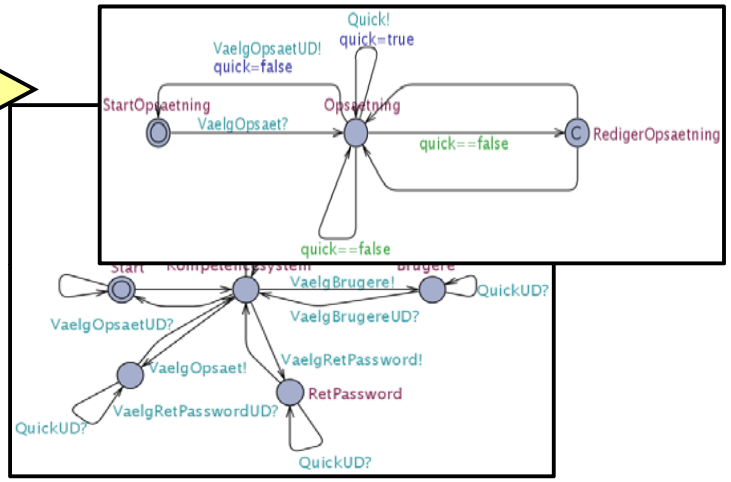
V-PLUS: Model-based GUI Testing for Automatic or Manual Execution

Requirement spec.:

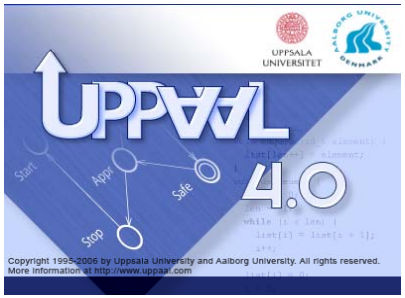


Create Models

scott/tiger GROUP



Generate Covering Test Suite:

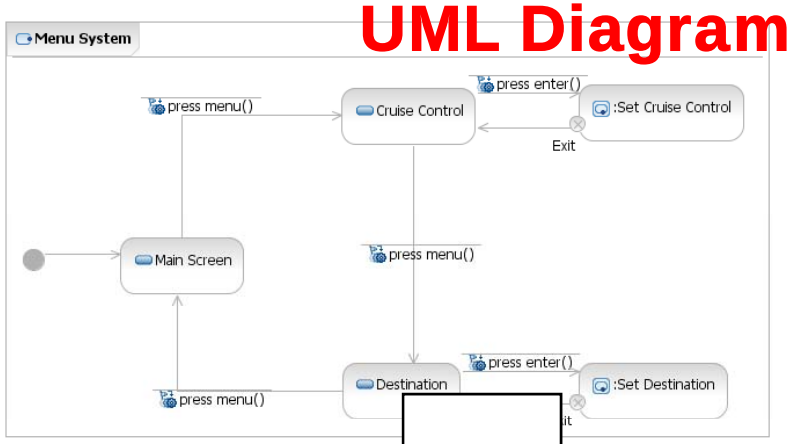


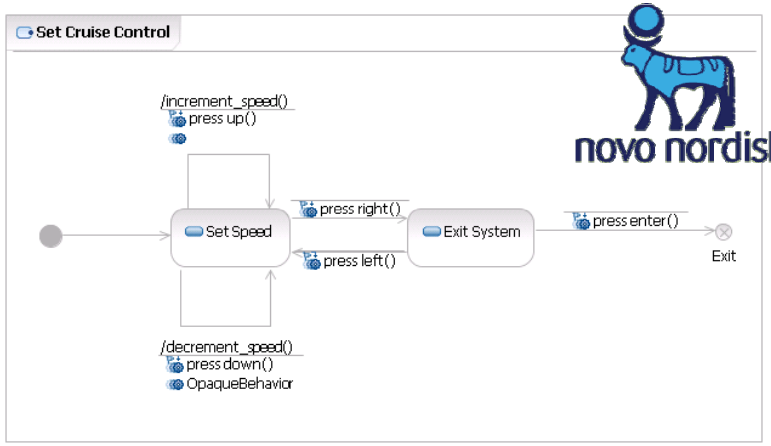
Output:

- Excel
- TestDrive Gold
- Quality Center/ QTP

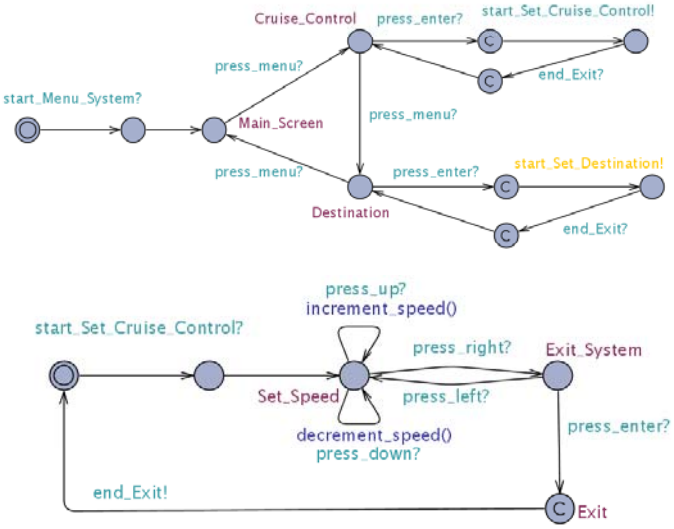
Test Generation from UML Statecharts

UML Diagrams





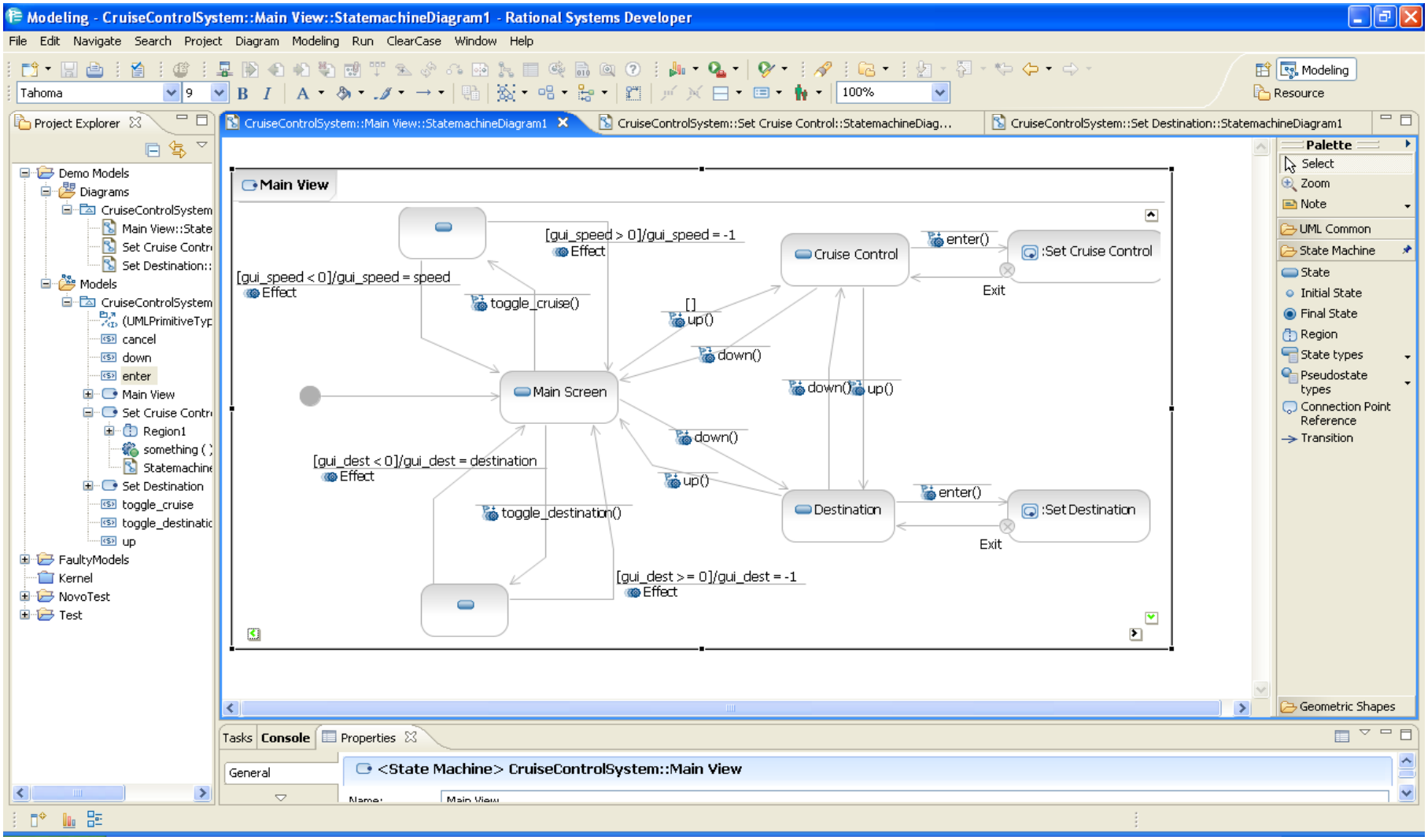
Uppaal Models



Custom Scripts

```
//Begin Test
assert_view_state('Menu::Main Screen');
user_press(MENU);
assert_view_state('Menu::Cruise Control');
user_press(ENTER);
assert_view_state('CC::Adjust speed');
user_press(DOWN);
user_press(RIGHT);
assert_view_state('CC::Exit Selected');
user_press(ENTER);
assert_view_state('Menu::Cruise Control');
user_press(MENU);
assert_view_state('Menu::Destination');
user_press(MENU);
assert_view_state('Menu::Main Screen');
user_press(MENU);
assert_view_state('Menu::Cruise Control');
user_press(ENTER);
assert_view_state('CC::Adjust speed');
user_press(UP);
assert_variable_value('CC::speed',20);
user_press(DOWN);
assert_variable_value('CC::speed',10);
user_press(UP);
assert_variable_value('CC::speed',20);
user_press(RIGHT);
assert_view_state('CC::Exit Selected');
user_press(LEFT);
assert_view_state('CC::Adjust speed');
//End Test
```

An Industrial Tool Chain...



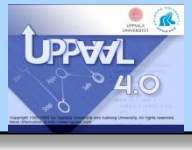
An Industrial Tool Chain..

Build functional model as **UML state machines** using Rational Systems Developer from IBM

XMI

Generate edge coverage test using **Yggdrasil**

XMI-to-TA



Output tests from Yggdrasil to ***JavaScript***

(QTP, Selenium, Ruby)

Execute scripts

w Jacob Illum Rasmussen, Arne Skou
Formal Methods 2009

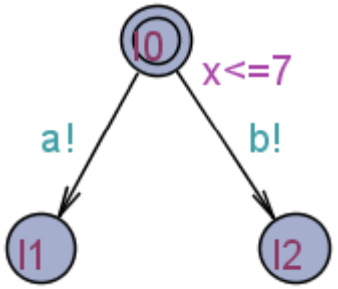
GUI testing

Off-Line Test Generation

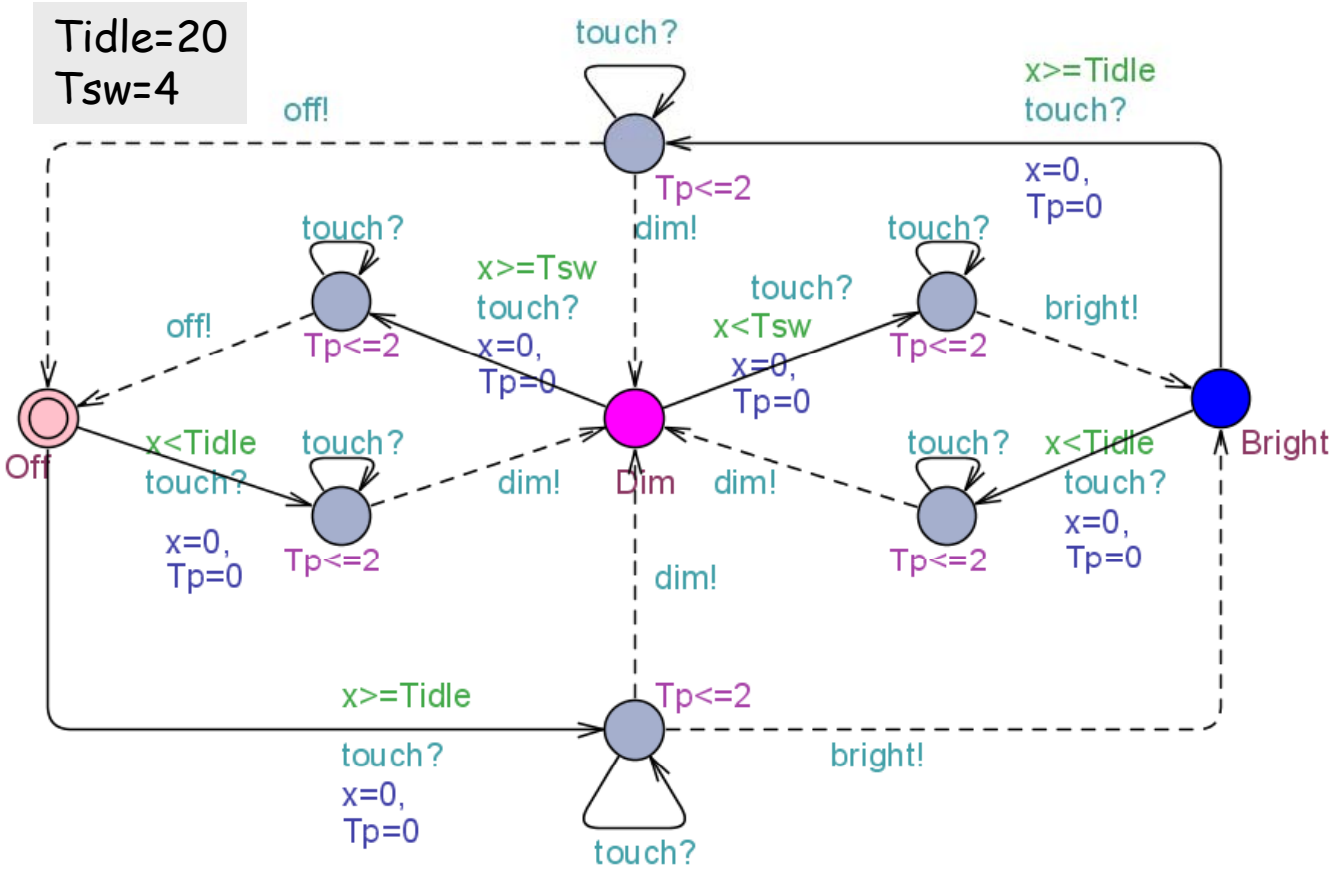
Observable Timed Automata

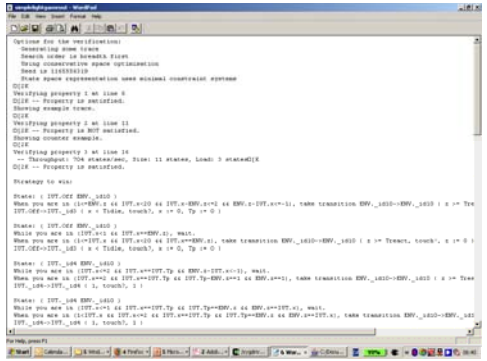
Observable Timed Automata

- **Determinism**:
  two transitions with same input/output leads to the same state
- **Input Enabled**:
  all inputs can always be accepted
- **Time Uncertainty of outputs**:
  timing of outputs uncontrollable by tester
- **Uncontrollable output**:
  IUT controls which enabled output will occur in what order



Timed Games and Test Generation

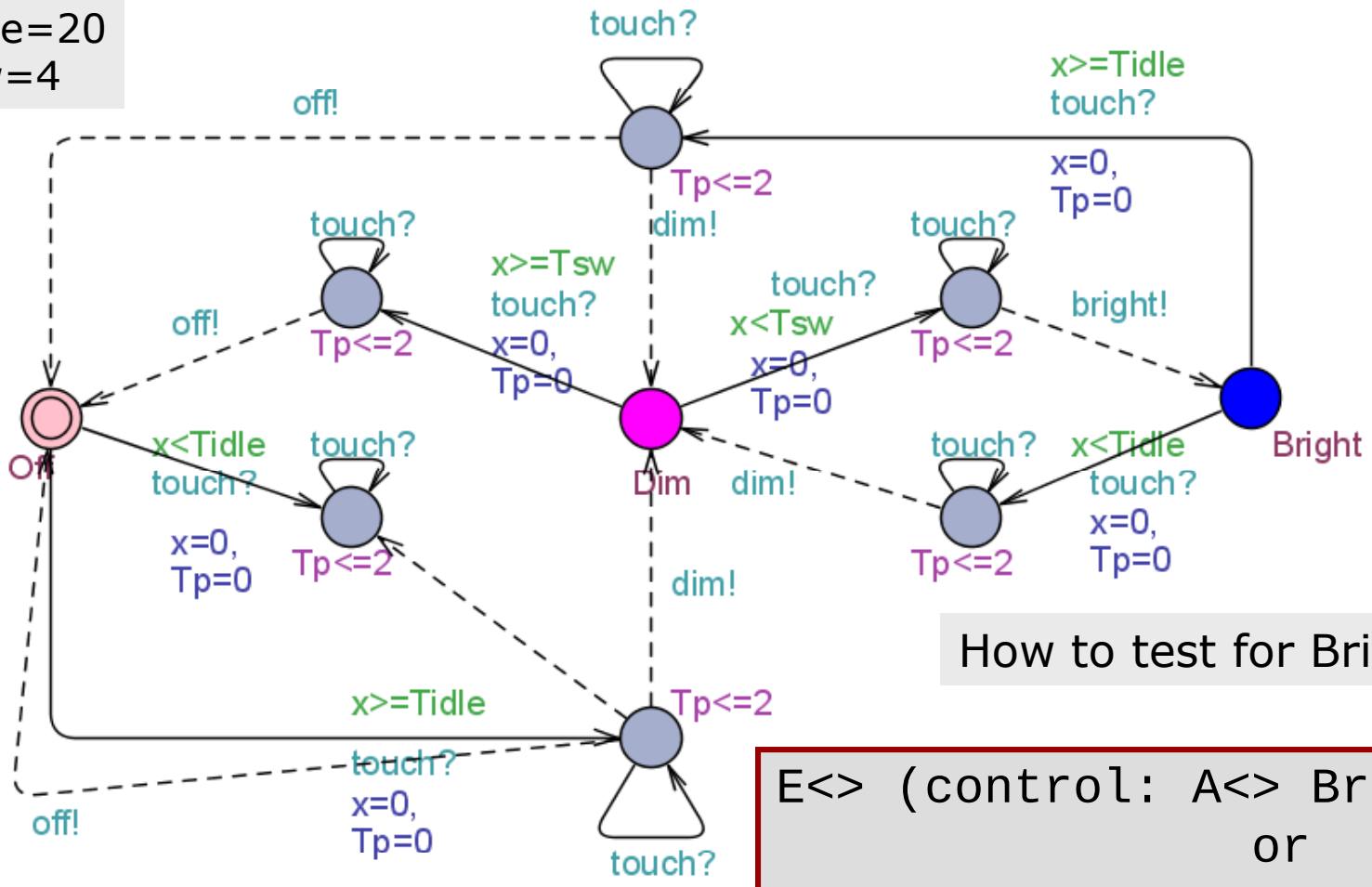




Off-line test-case generation =
Compute winning strategy for reaching **Bright**
Assign verdicts st. lost game means IUT not conforming

A trick light control

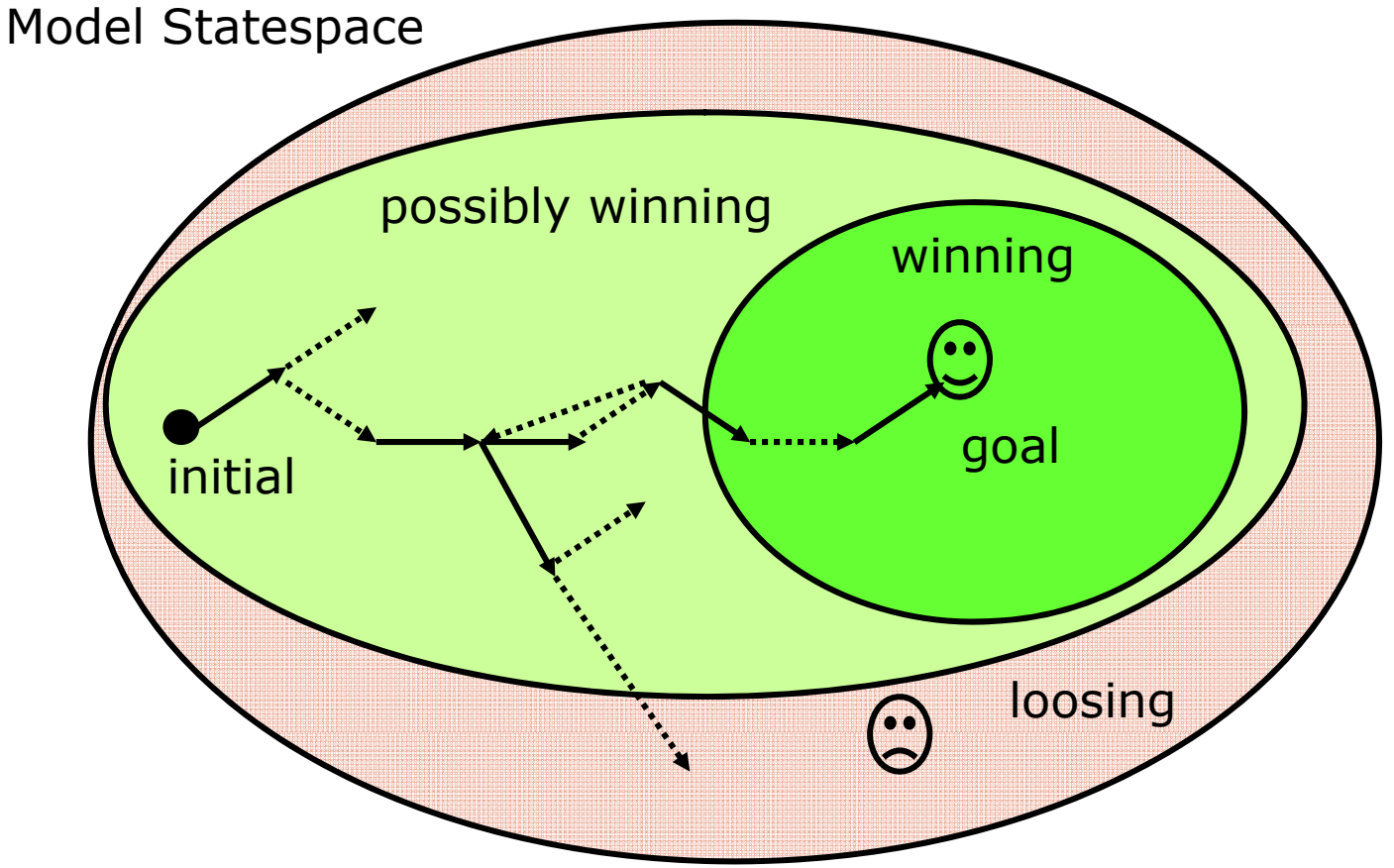
Tidle=20
Tsw=4



How to test for Bright ?

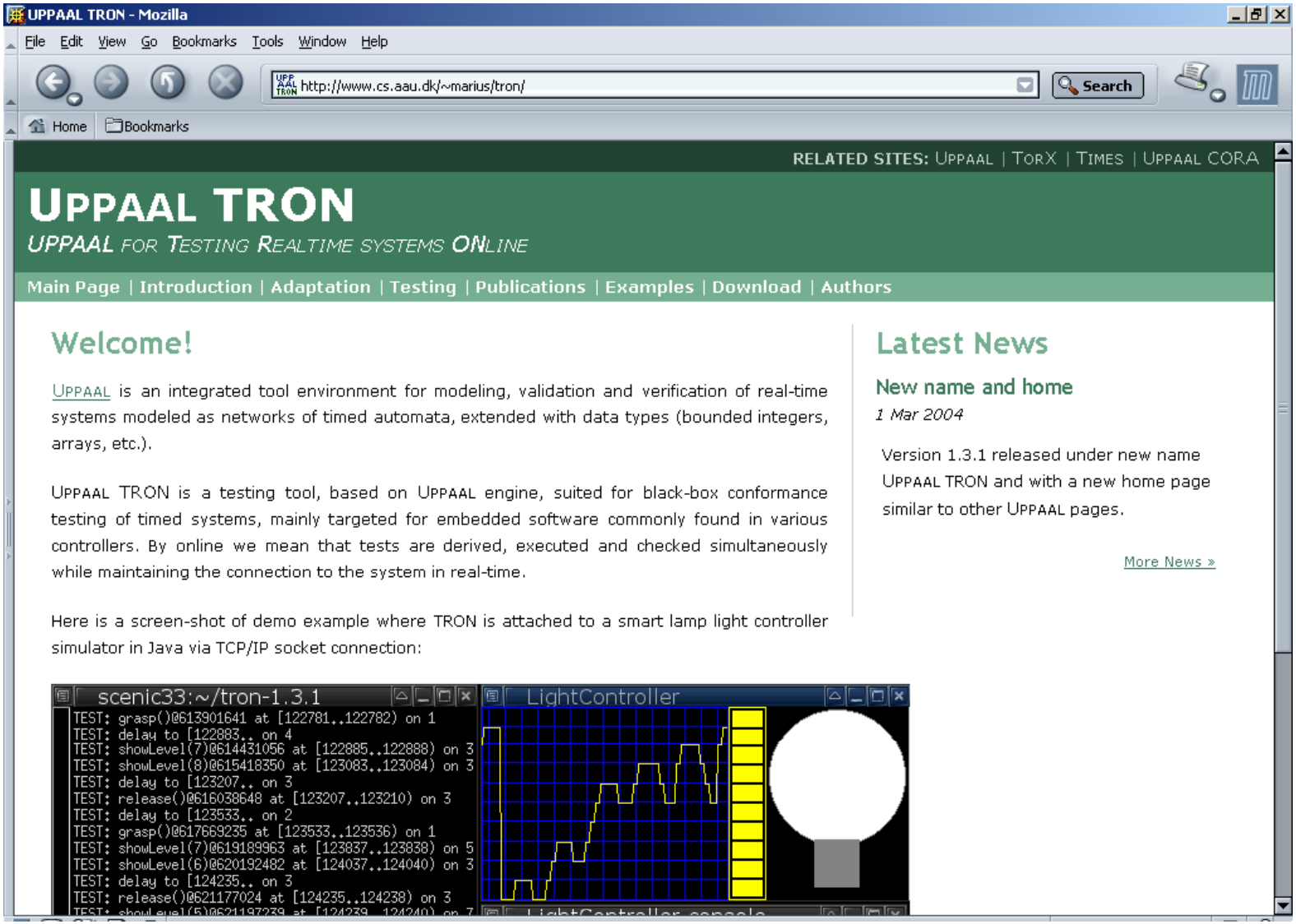
```
E<> (control: A<> Bright)
                or
<<c,u>> ♢(<<c>> ♢ Bright)
```

Cooperative Strategies

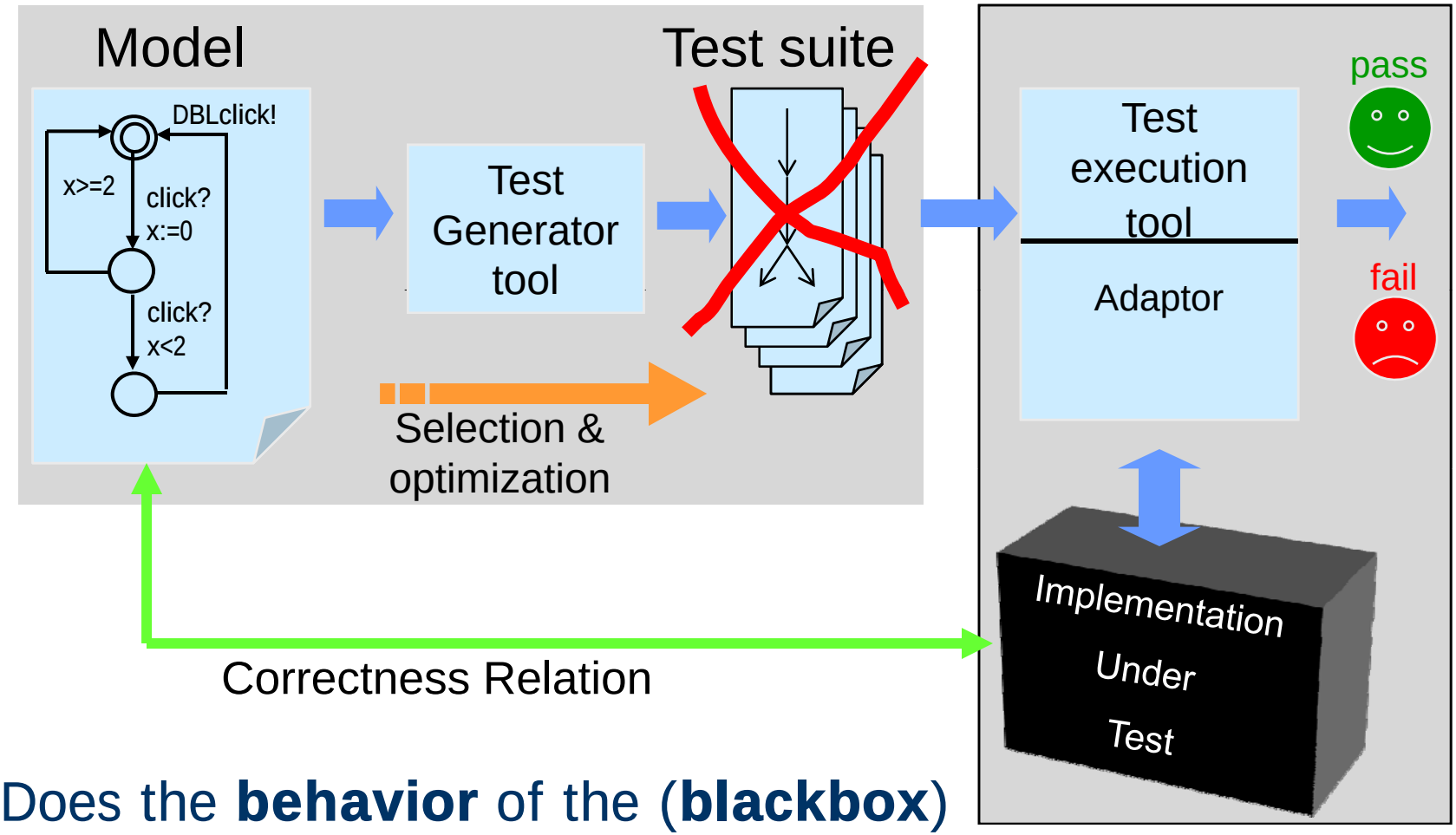


- Play the game (execute test) while time available or game is lost
- Possibly using ranomized online testing

On-Line Testing

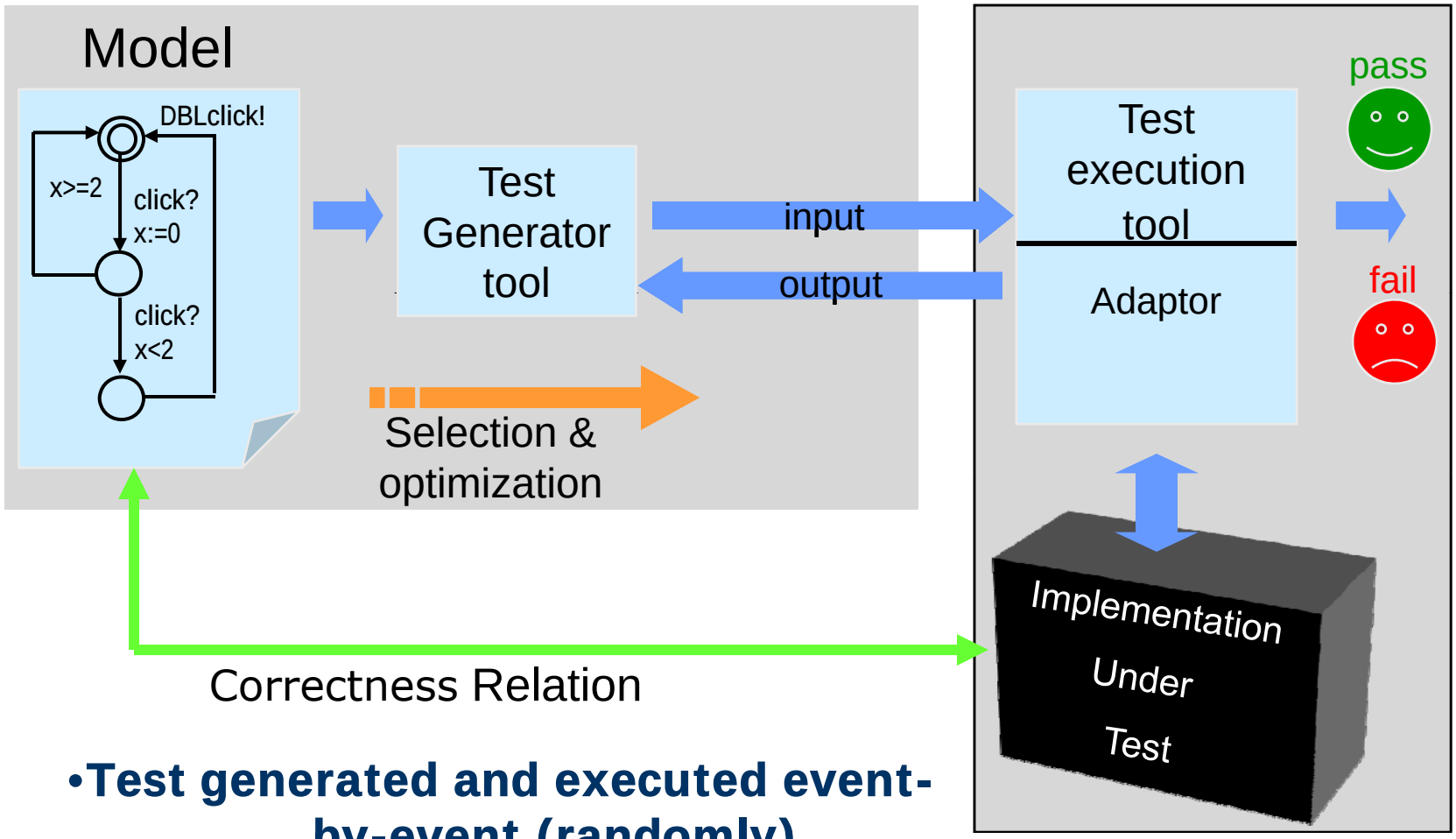

UPPAAL TRON - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www.cs.aau.dk/~marius/tron/
Search
Home
Bookmarks
RELATED SITES: UPPAAL | TORX | TIMES | UPPAAL CORA
UPPAAL TRON
UPPAAL FOR TESTING REALTIME SYSTEMS ONLINE
Main Page | Introduction | Adaptation | Testing | Publications | Examples | Download | Authors
Welcome!
UPPAAL is an integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types (bounded integers, arrays, etc.).
UPPAAL TRON is a testing tool, based on UPPAAL engine, suited for black-box conformance testing of timed systems, mainly targeted for embedded software commonly found in various controllers. By online we mean that tests are derived, executed and checked simultaneously while maintaining the connection to the system in real-time.
Here is a screen-shot of demo example where TRON is attached to a smart lamp light controller simulator in Java via TCP/IP socket connection:
Latest News
New name and home
1 Mar 2004
Version 1.3.1 released under new name UPPAAL TRON and with a new home page similar to other UPPAAL pages.
More News »
scenic33:~/tron-1.3.1
TEST: grasp()@613901641 at [122781..122782) on 1
TEST: delay to [122883.. on 4
TEST: showLevel(7)@614431056 at [122885..122888) on 3
TEST: showLevel(8)@615418350 at [123083..123084) on 3
TEST: delay to [123207.. on 3
TEST: release()@616038648 at [123207..123210) on 3
TEST: delay to [123533.. on 2
TEST: grasp()@617669235 at [123533..123536) on 1
TEST: showLevel(7)@619189963 at [123837..123838) on 5
TEST: showLevel(6)@620192482 at [124037..124040) on 3
TEST: delay to [124235.. on 3
TEST: release()@621177024 at [124235..124238) on 3
LightController

Automated Model Based Conformance Testing



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

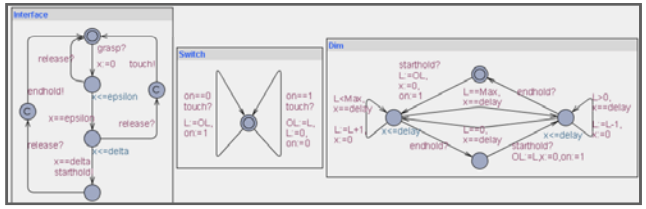
Online Testing

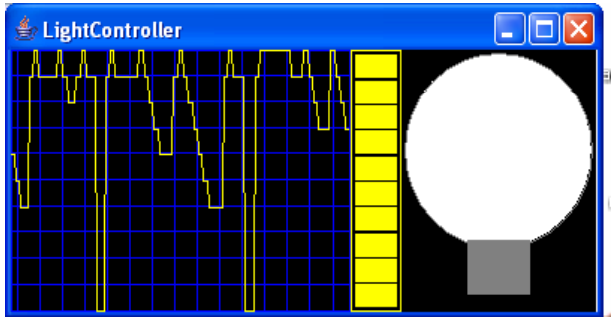


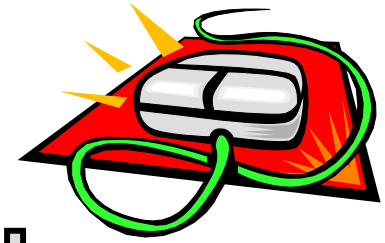
- **Test generated and executed event-by-event (randomly)**
- **A.K.A on-the-fly testing**

On-line Testing

Light Controller







mousePress
mouseRelease

UPPAAL^{TRON}
Testing
Host

tcp/ip

- Real-time
- Simulated time

Test Fixture

LightControllerGUI

setLevel

grasp
release

grasp
release

setLevel

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

An Algorithm

Algorithm 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*** $(\varepsilon_0, i_0, \varepsilon_1, o_1, \varepsilon_2, i_2, o_2)$

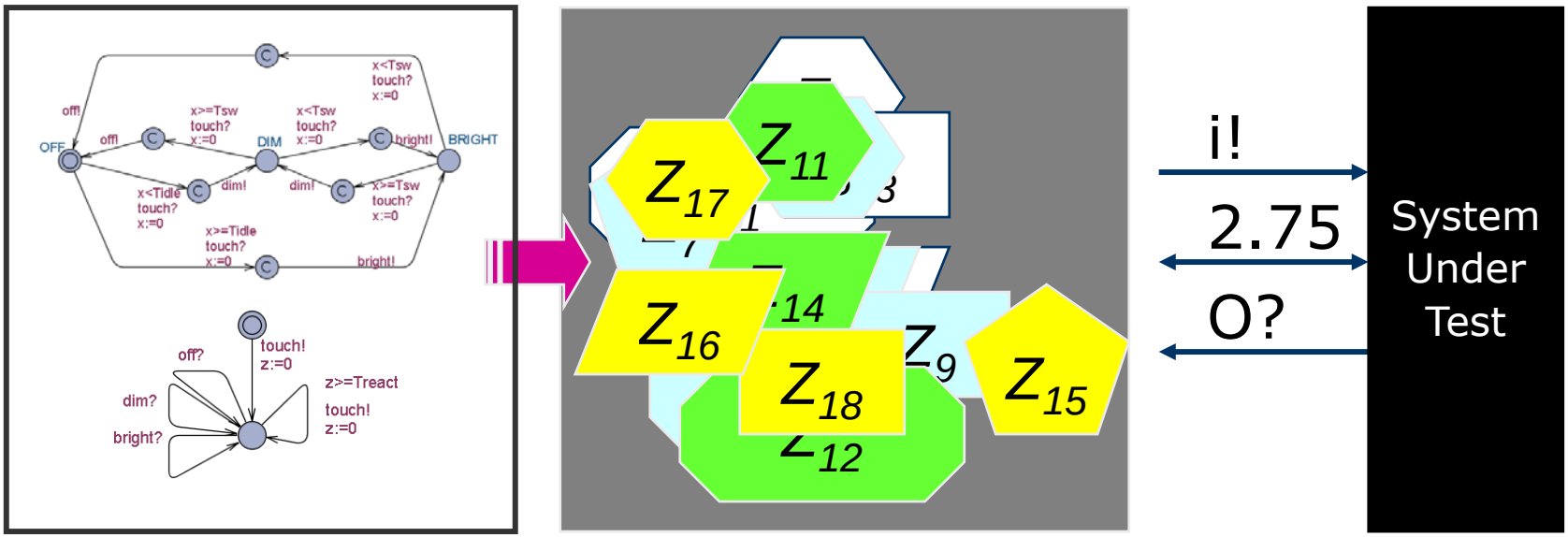
- *If* $Z = \emptyset$ *the IUT has made a computation not in model:* ***FAIL***

- *i is a relevant input in Env iff* $I \in EnvOutput(Z)$

Online State Estimation

Timed Automata Specification

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



(Abstract) Online Algorithm

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

Z := {(s0, e0)}.

while Z ≠ ∅ **and** #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
   Z := Z

2. // wait d fo
   randomly c
   wait (for d
   if o occurre
   Z :=
   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

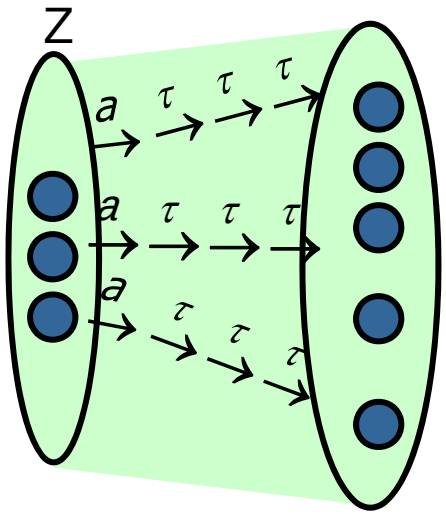
if Z = ∅ **then return fail else return pass**

- Sound
- Complete (as T → ∞)

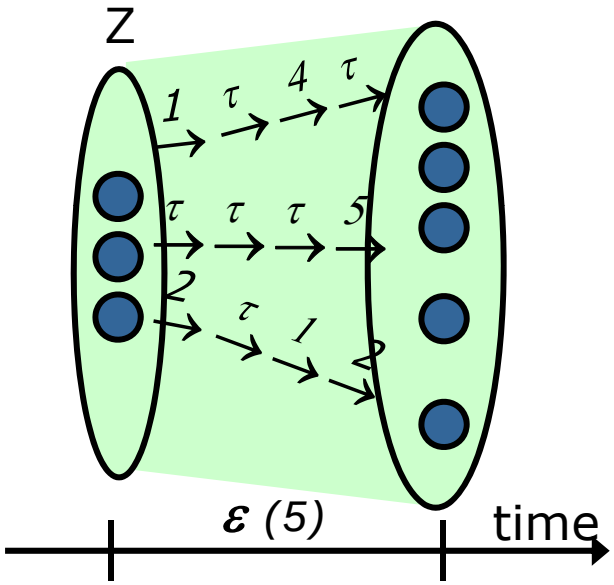
(Under some technical assumptions)

State-set Operations

Z* after *a: *possible states after* ***action a*** *(and* τ^**)*

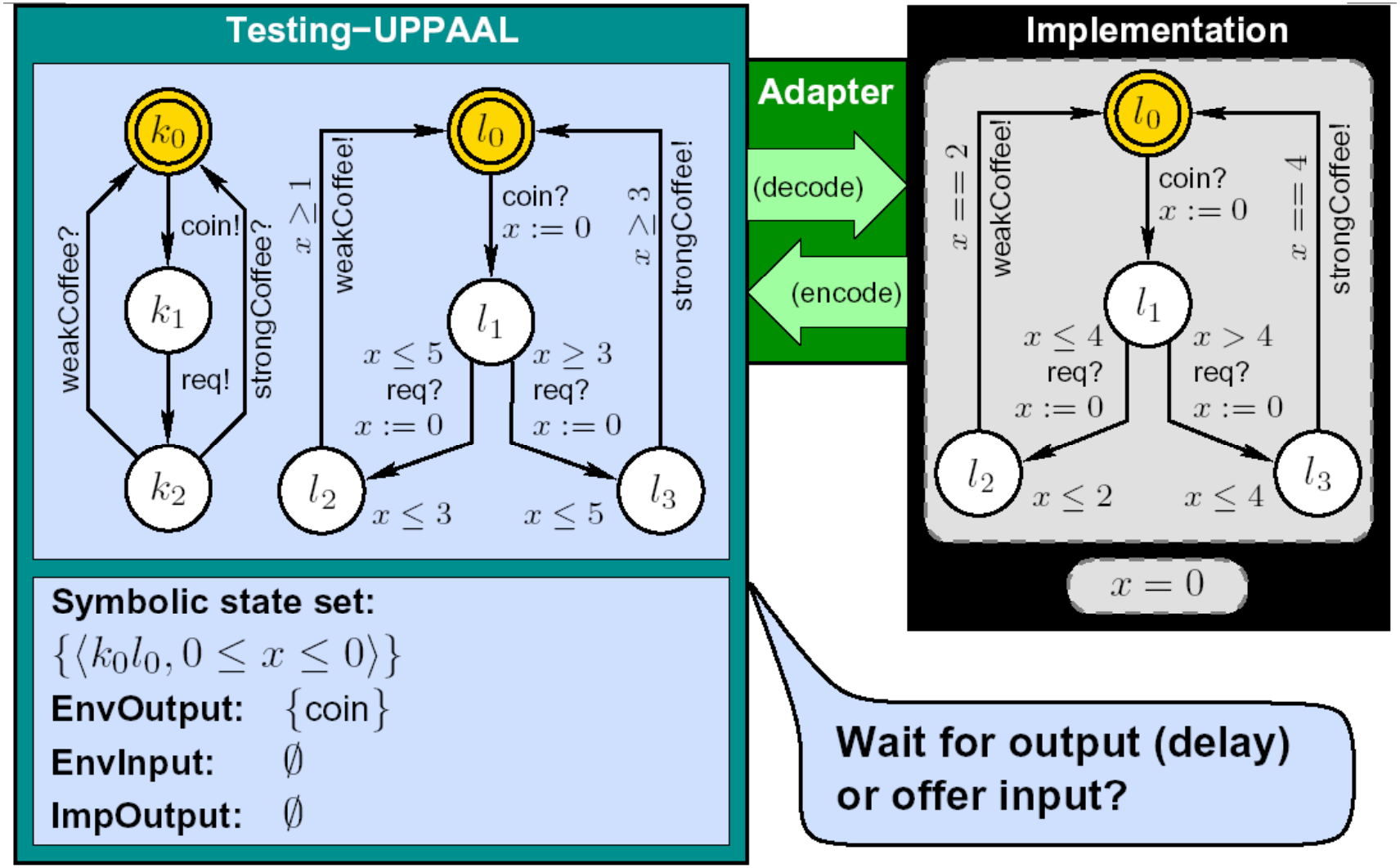


Z* after** ε :*possible states after* τ^* *and* ε_i *, totaling a* ***delay *of* ε

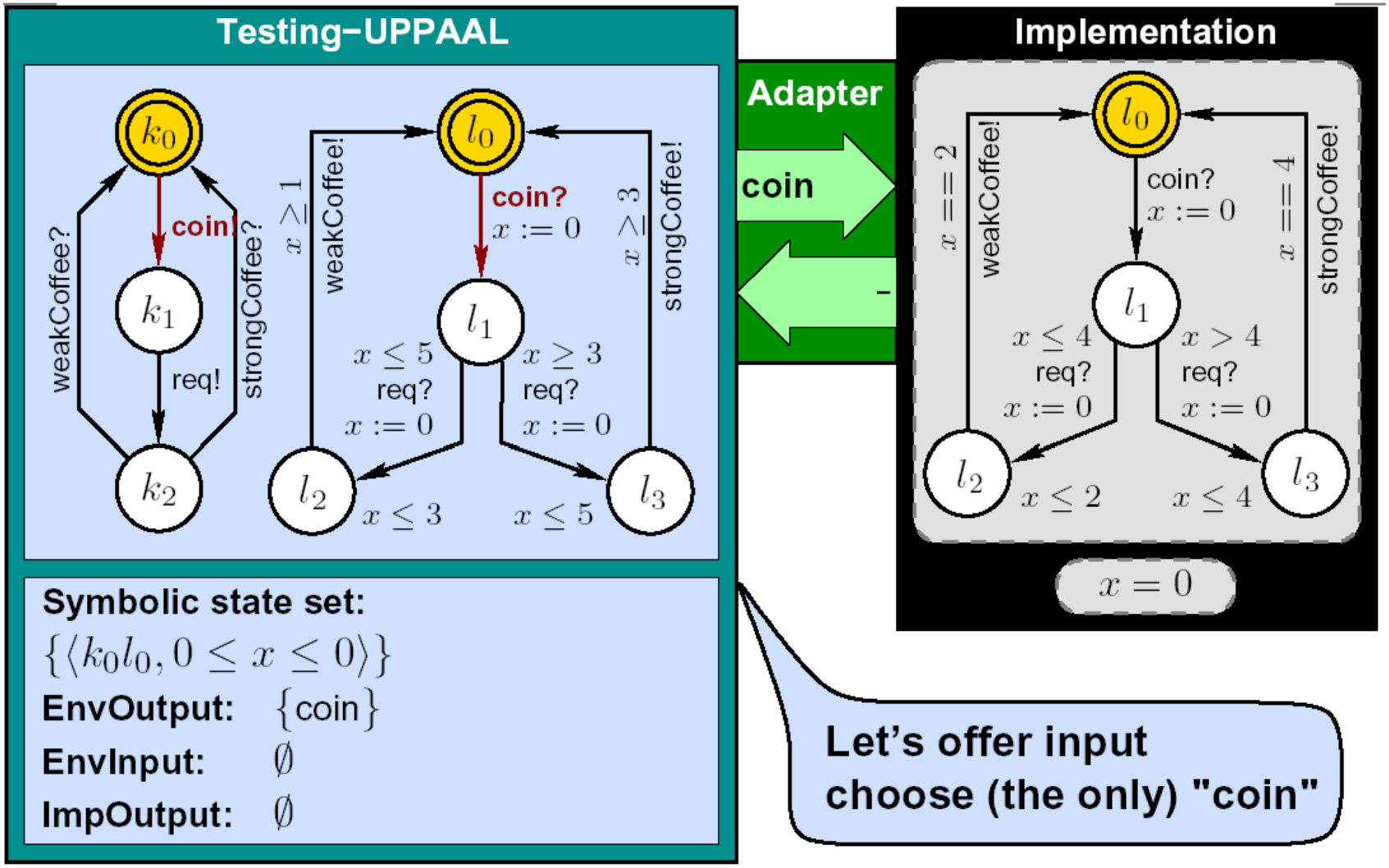


- Can be computed efficiently using the symbolic data structures and algorithms in Uppaal

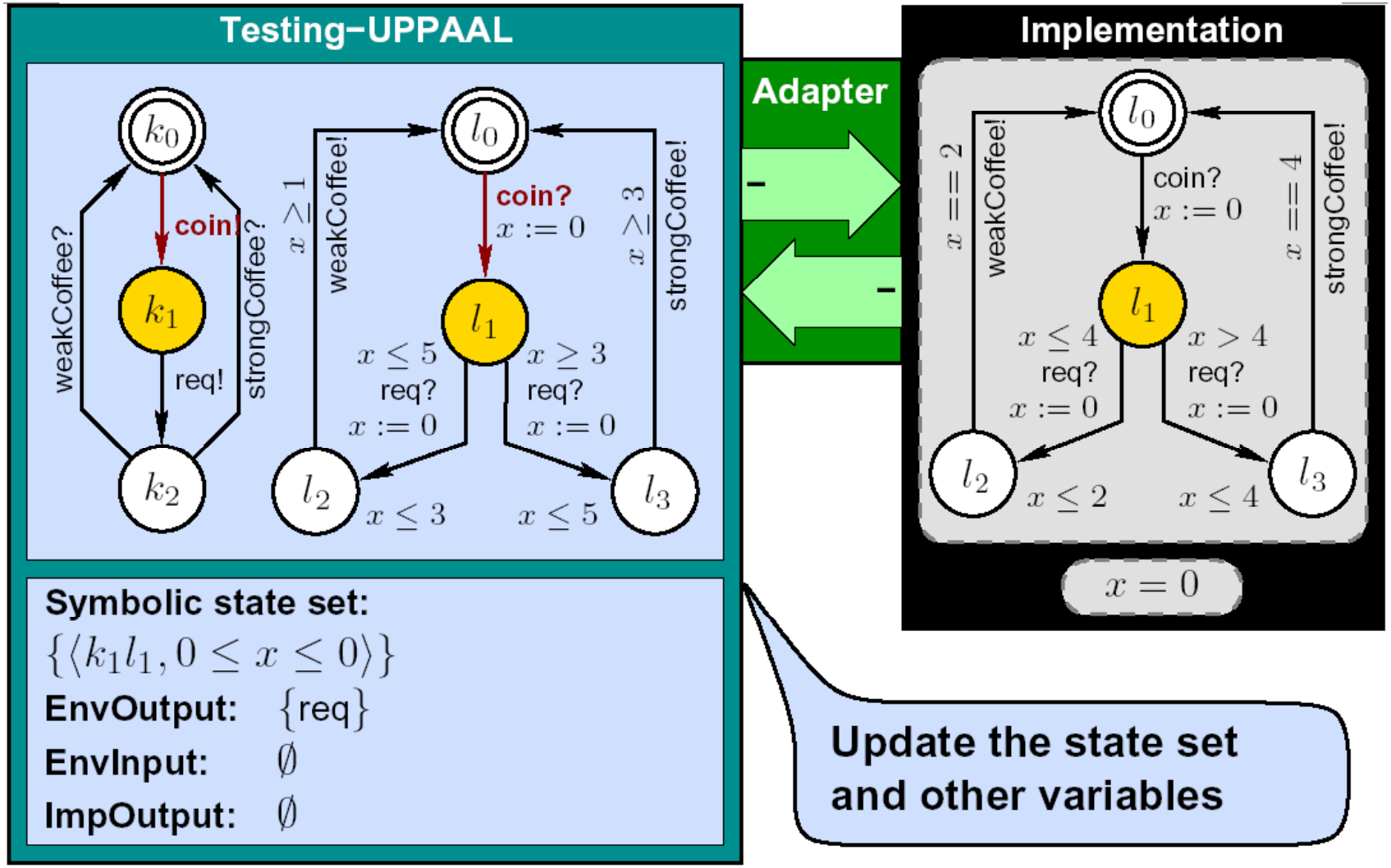
Online Testing Example



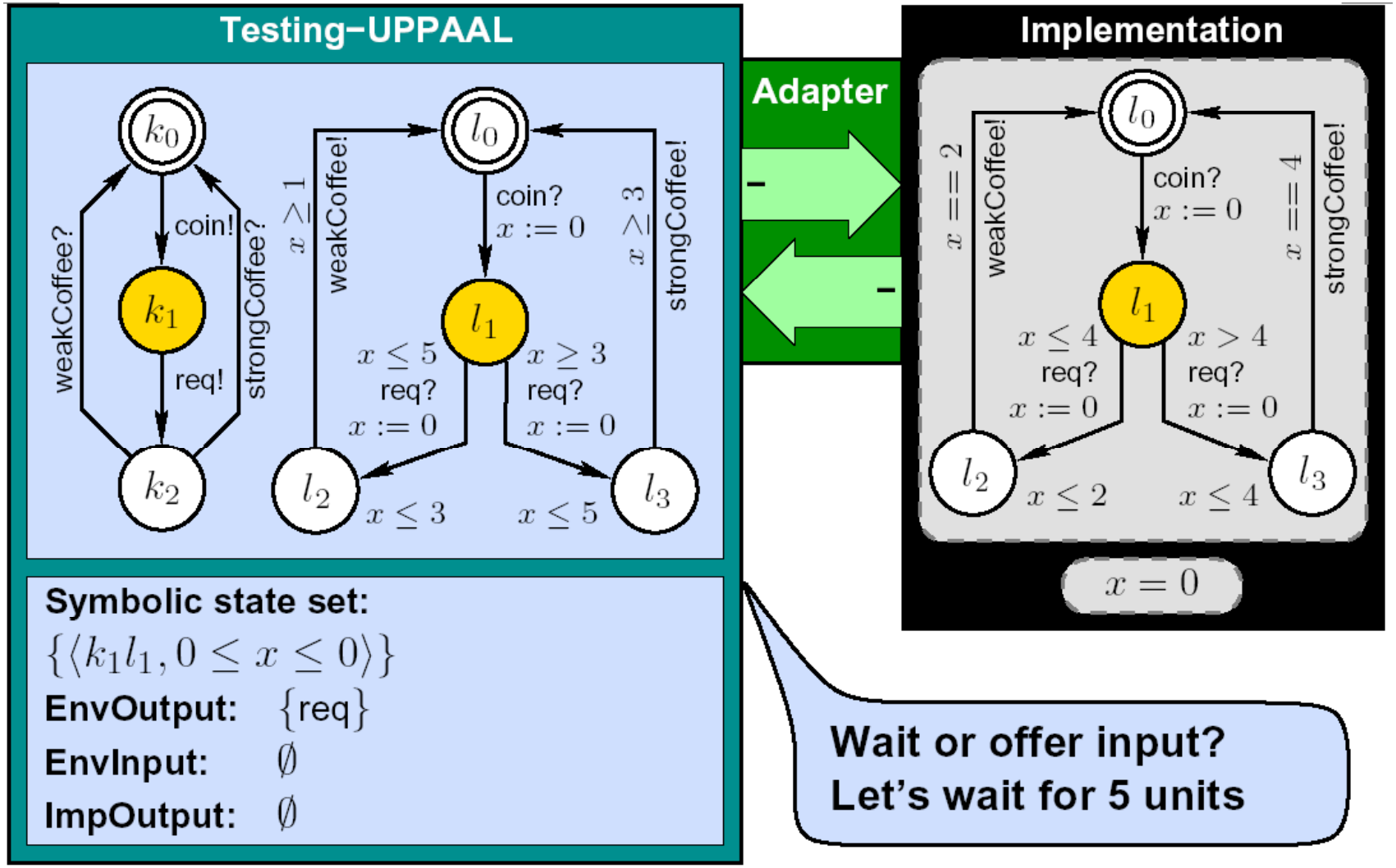
Online Testing



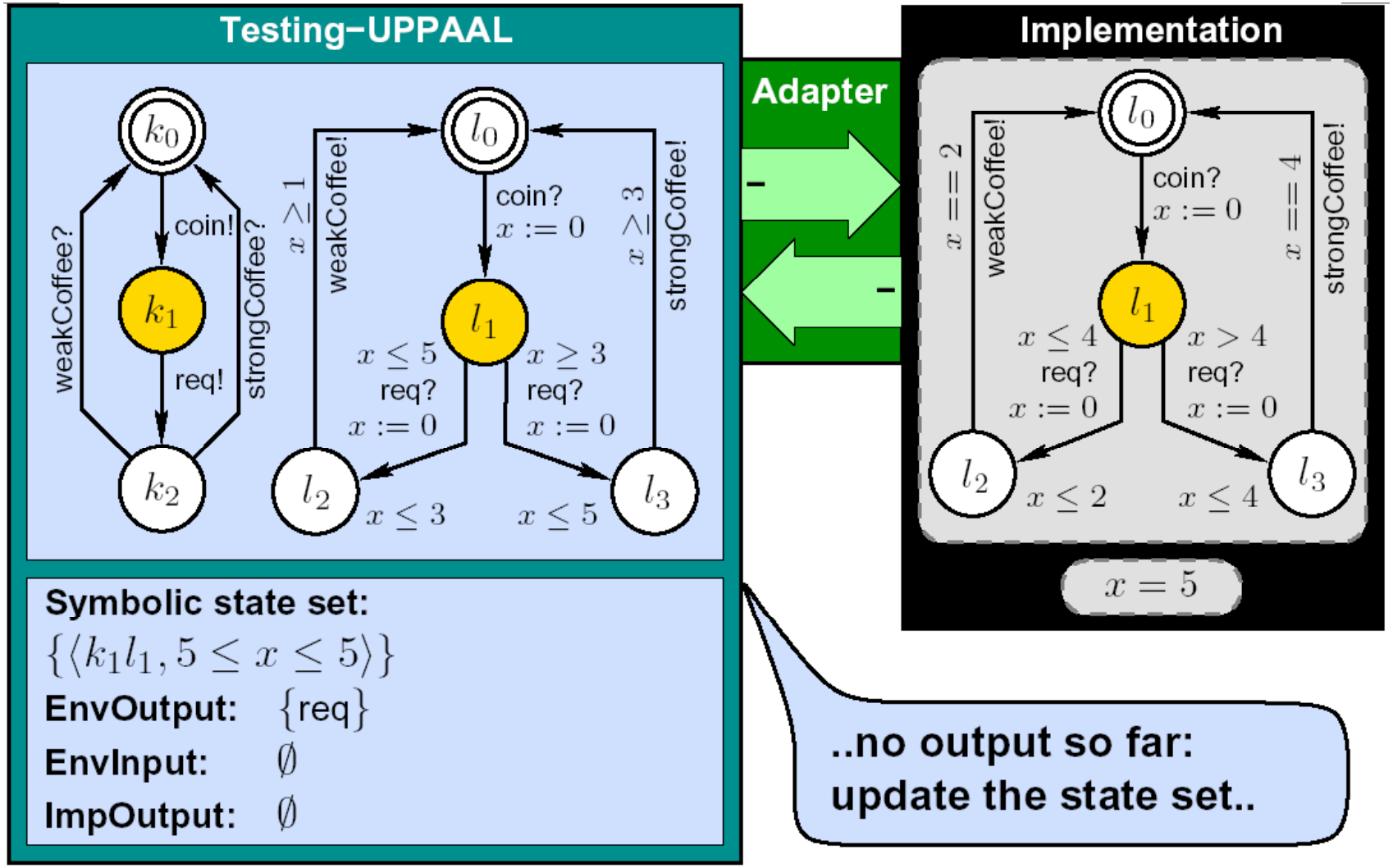
Online Testing



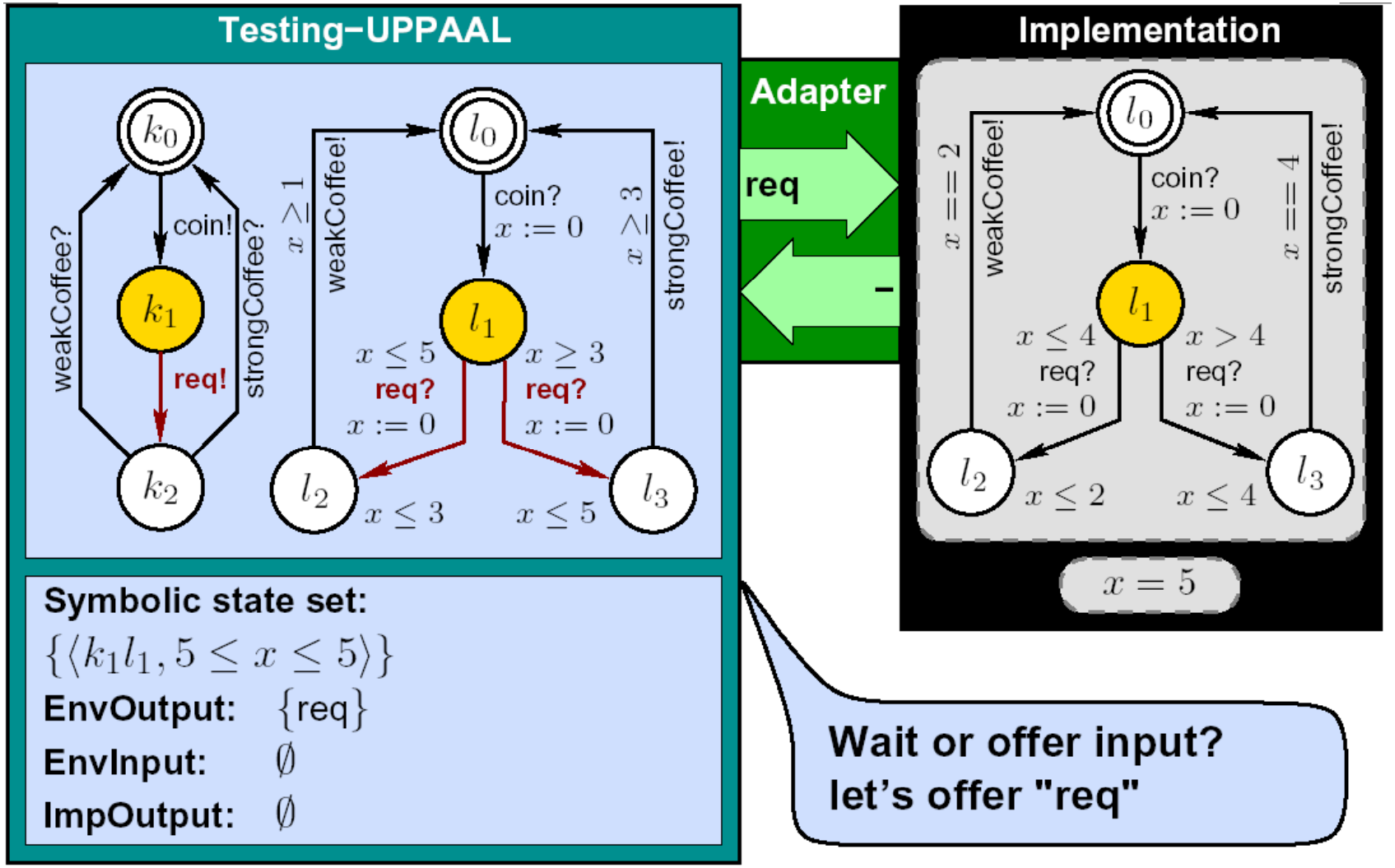
Online Testing



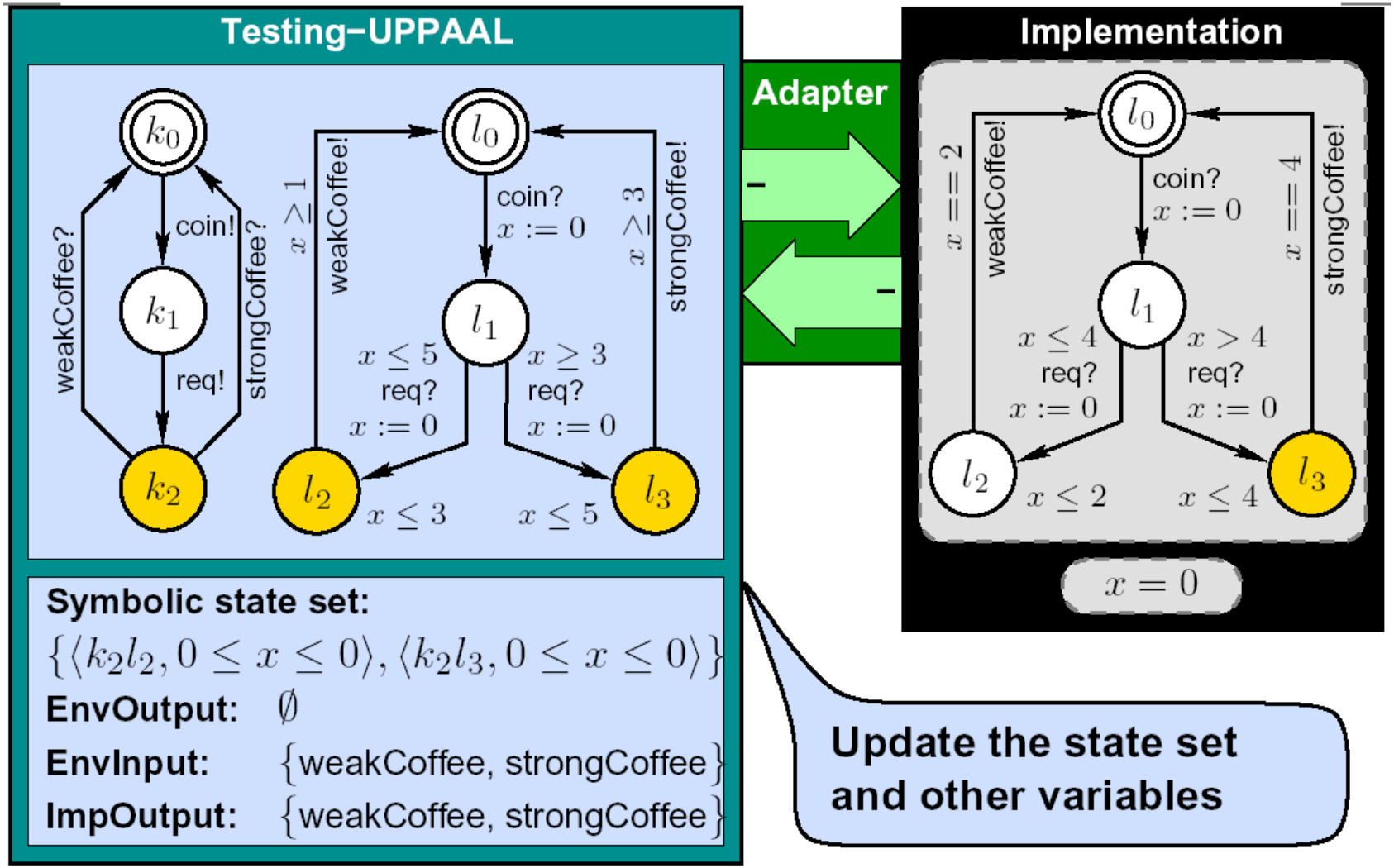
Online Testing



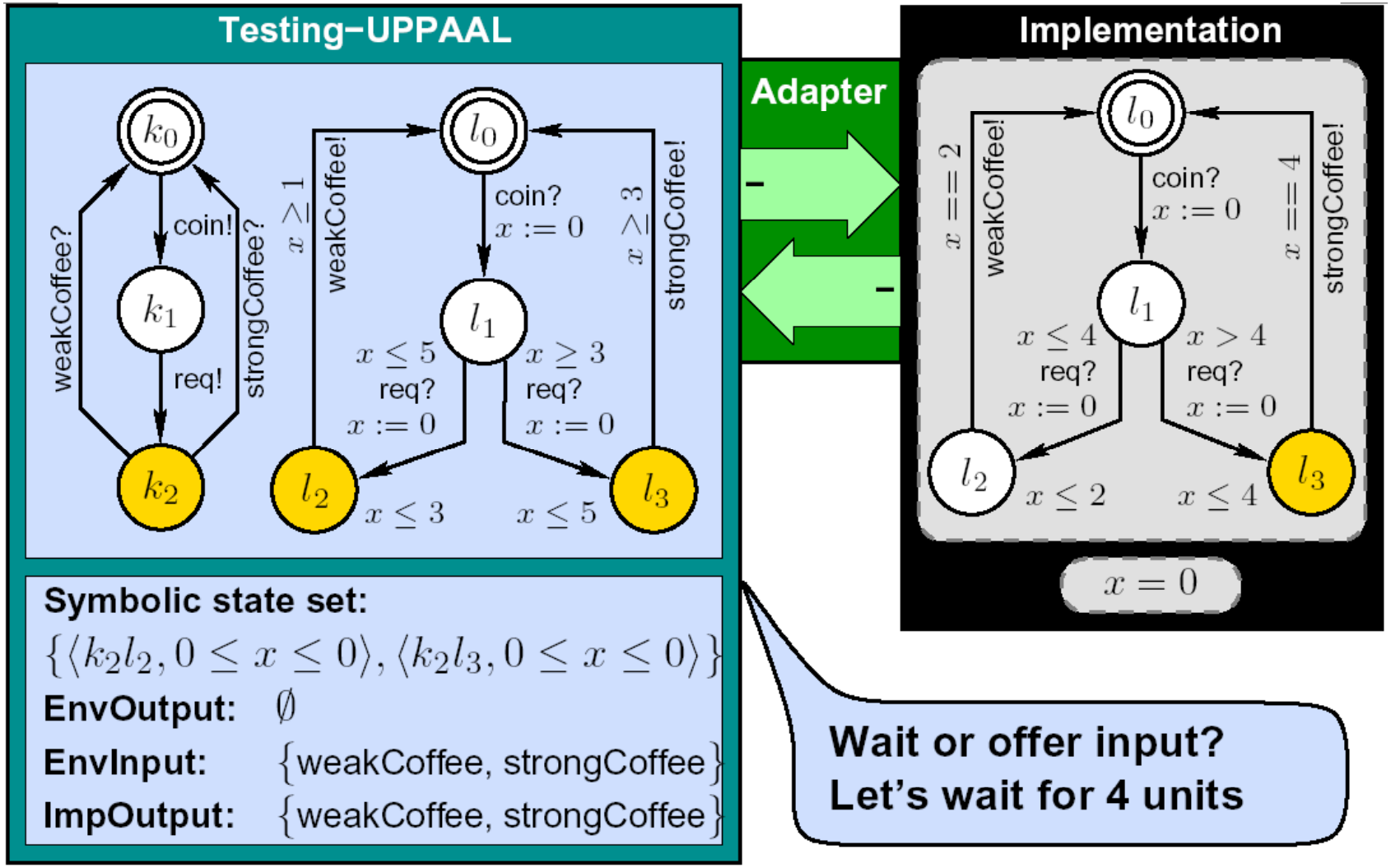
Online Testing



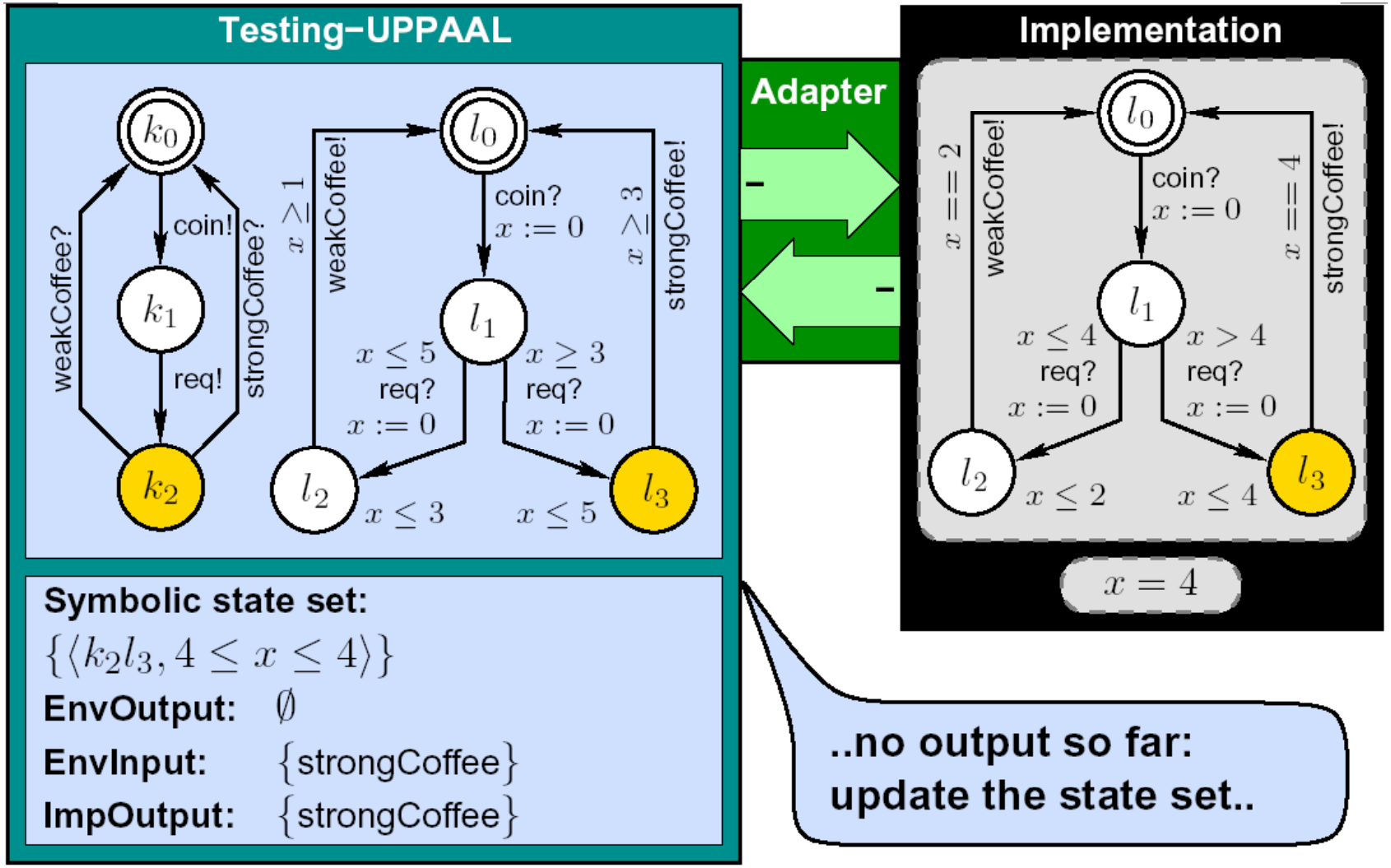
Online Testing



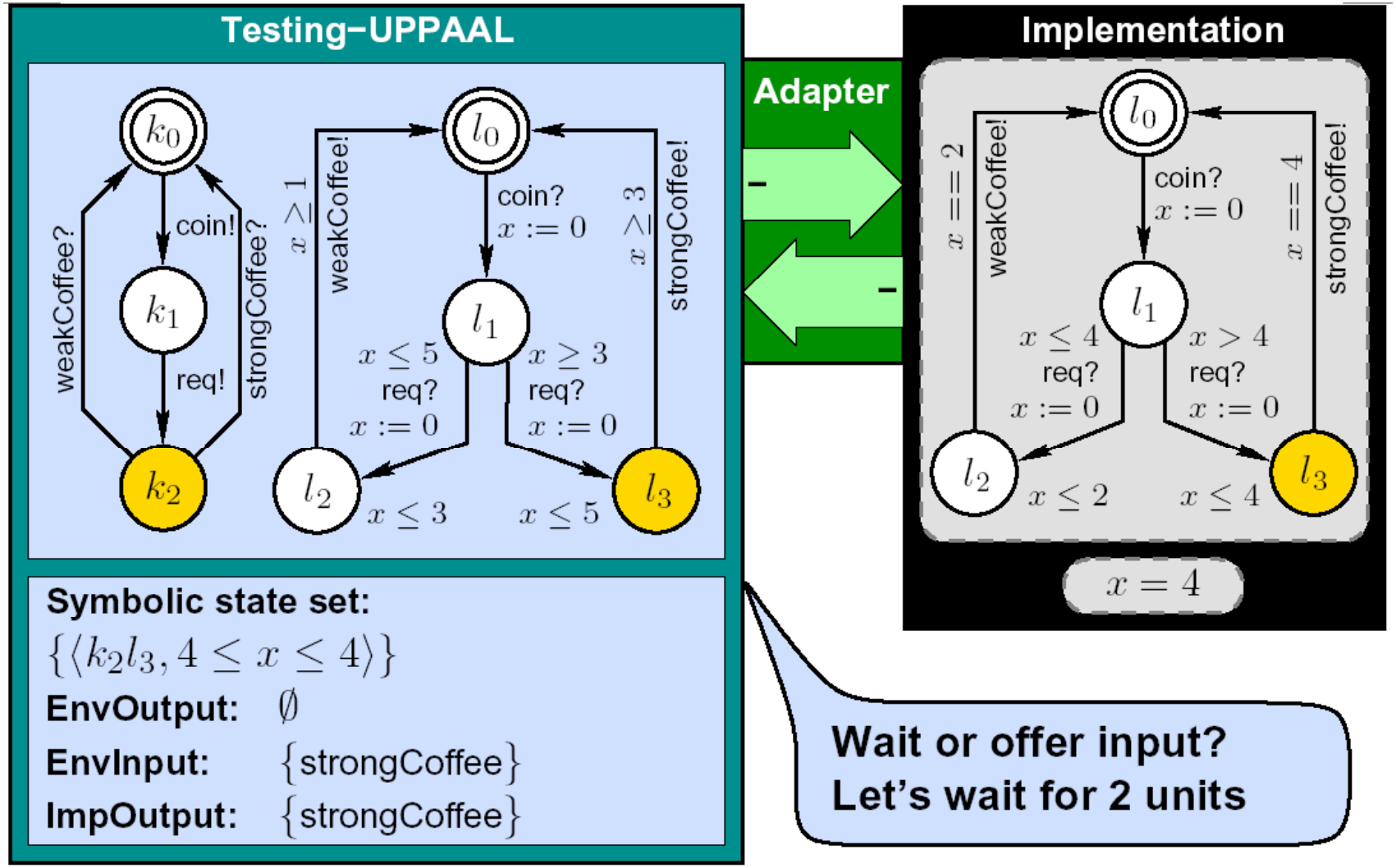
Online Testing



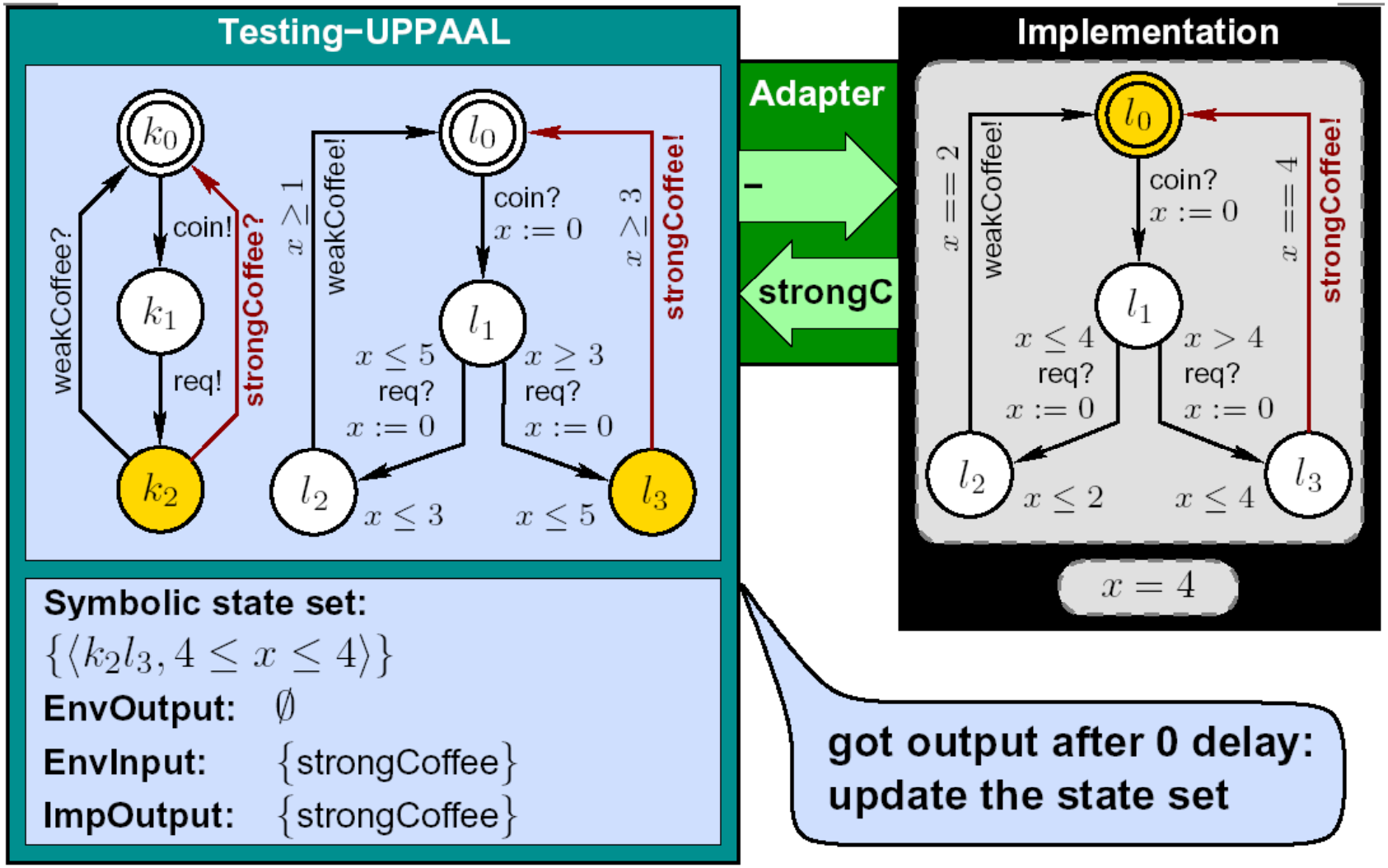
Online Testing



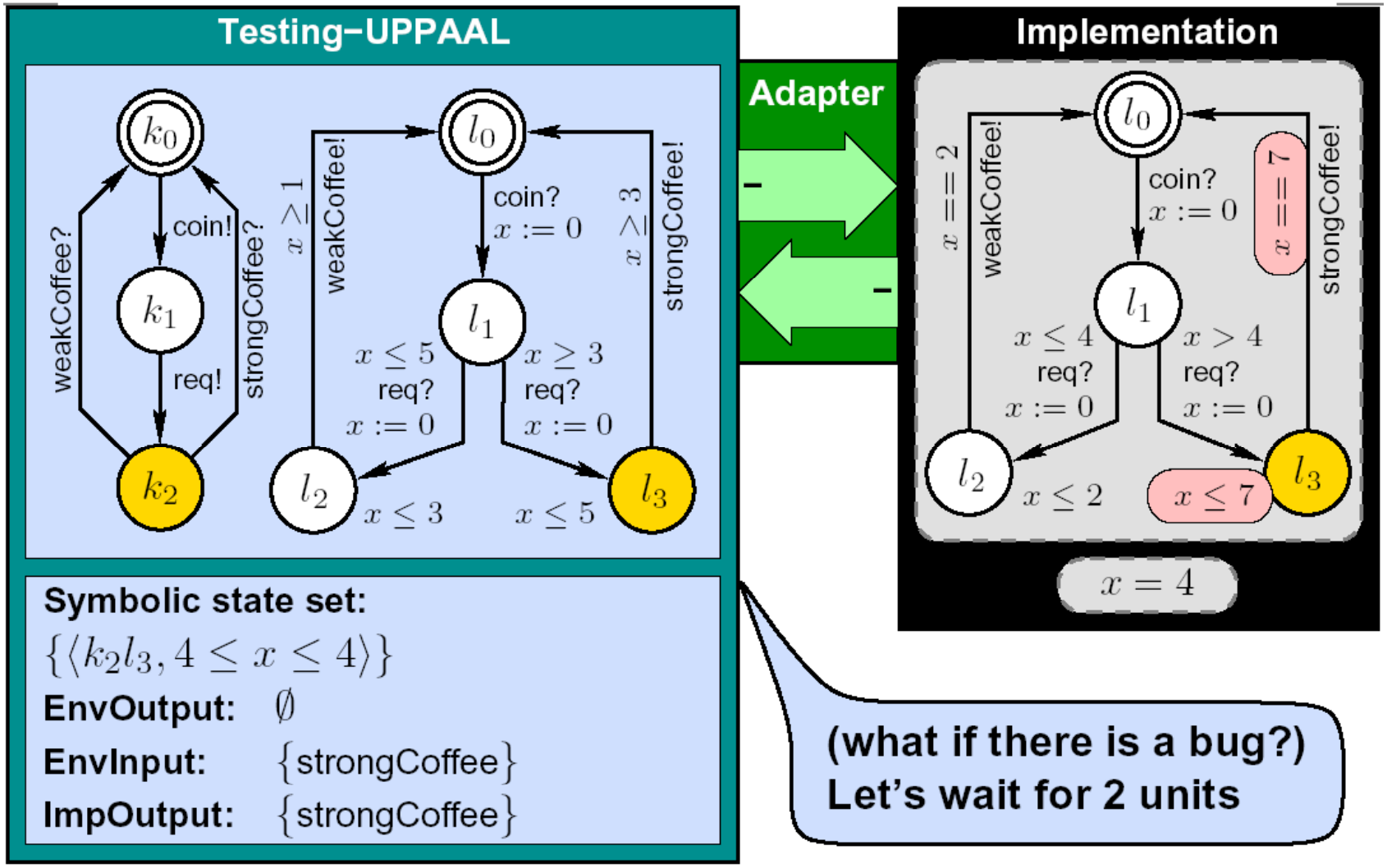
Online Testing



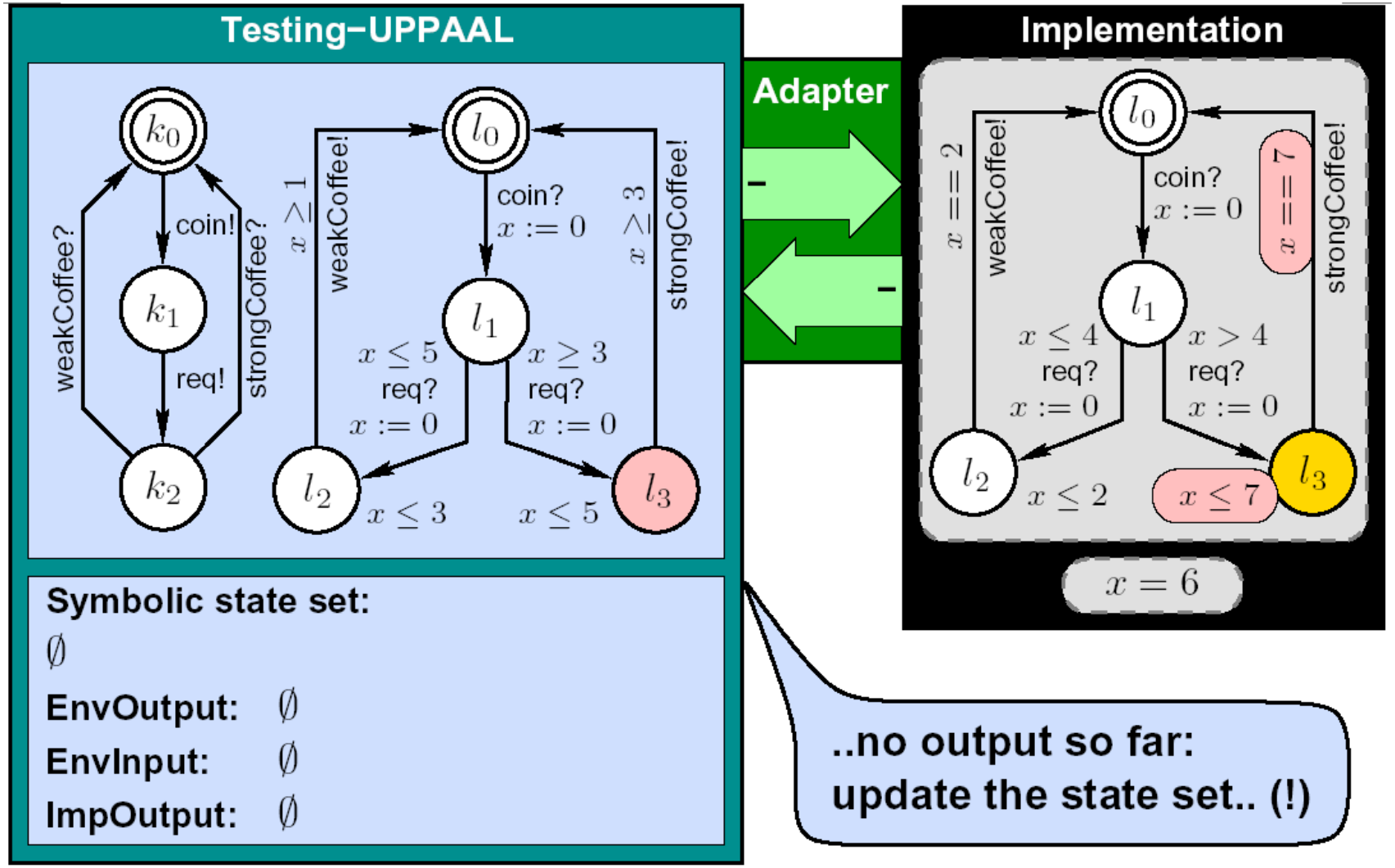
Online Testing



Online Testing



Online Testing



Industrial Application:

Danfoss 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

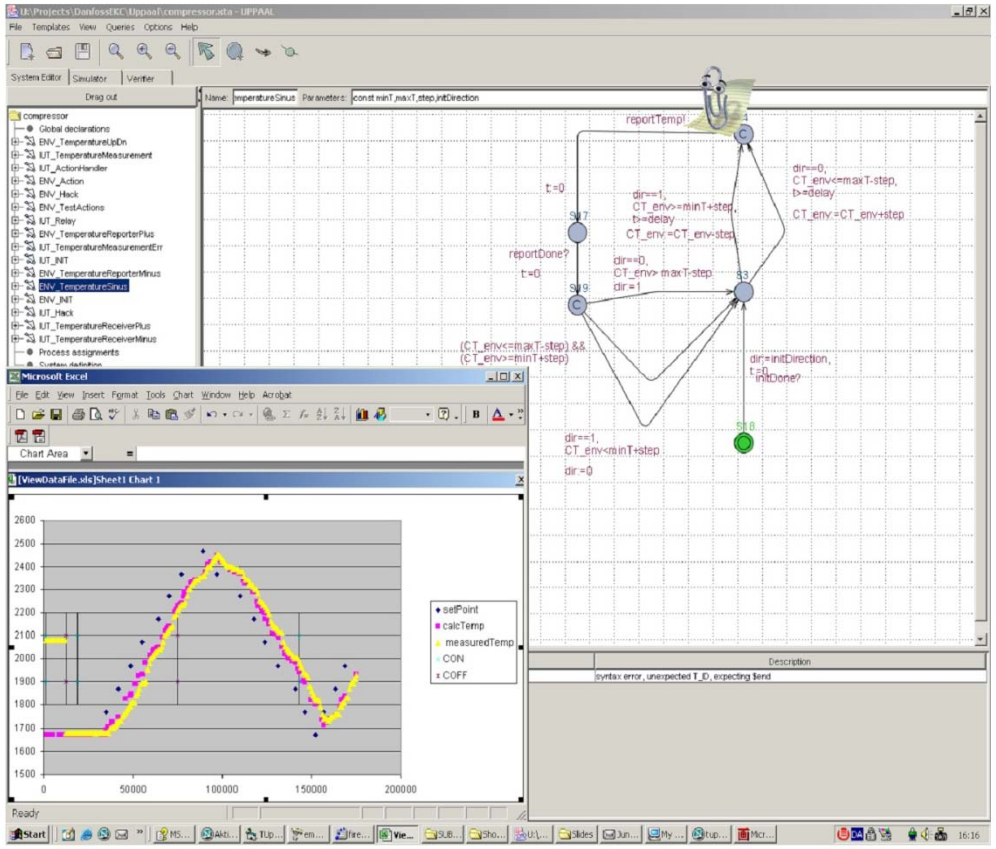
Industrial Cooling Plants

Industrial Application:

Danfoss Electronic Cooling Controller

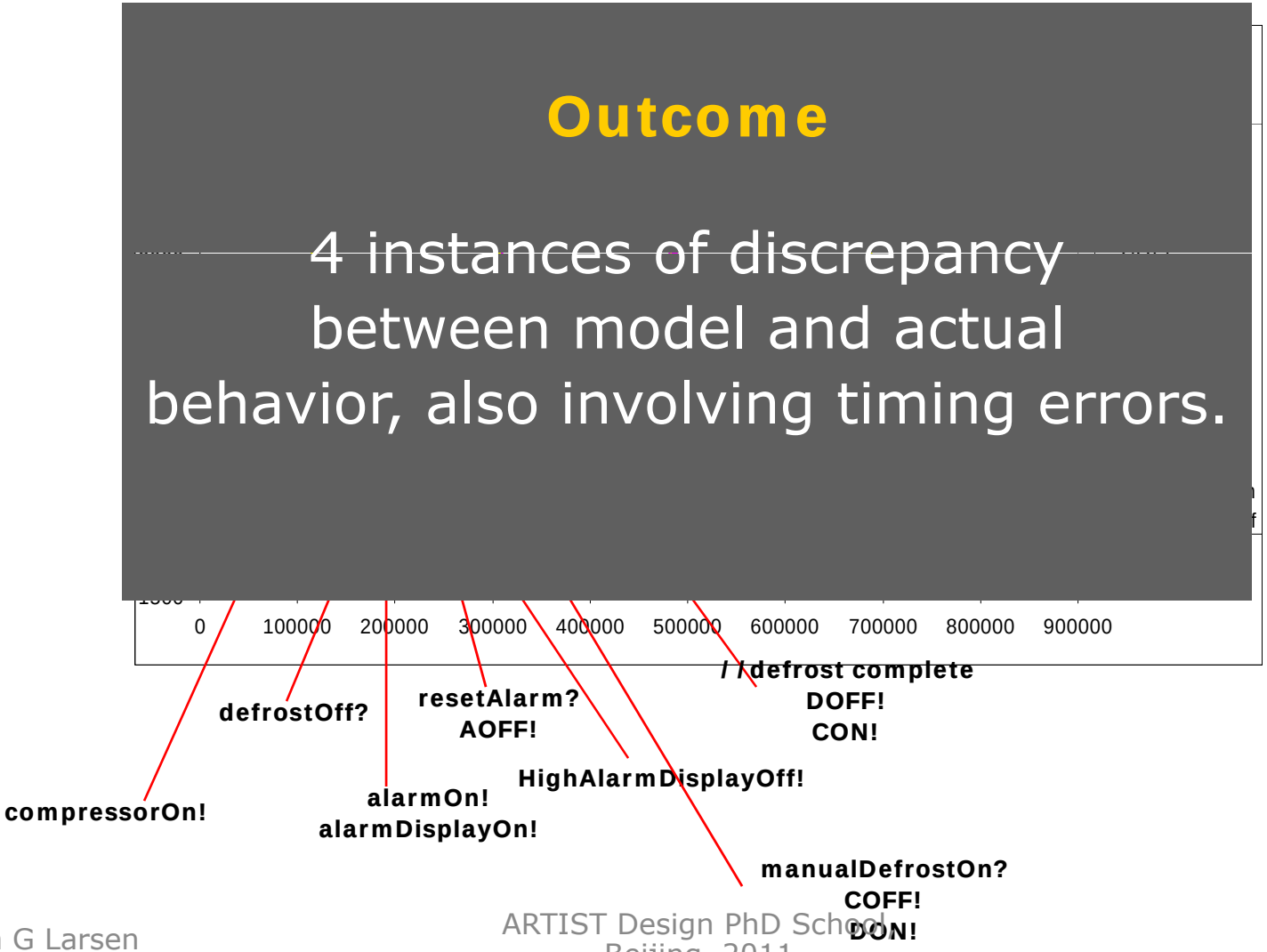
Sensor Input

- air temperature sensor



- Optional real-time clock or LON network module

Example Test Run



Model-based Testing of Real Time Systems

Conclusions

Advantages of MBT

- Engineer focus on **what** to test at a high level of abstraction
- Avoids cost of making scripts
  - As much test code as production code
  - Maintenance nightmare
- Heard of, but is still considered an **advanced technique** by industry
- Industry is very motivated, MB A&T will give
  - **10% cost reduction**
  - **20% quality improvement**

Verification & Testing

Verification

- Abstract models
- Exhaustive "proof"
- Limited size and expressivity

Testing

- Checks the actual implementation
- Only few executions checked
- But is the most direct method

How to effectively ***combine*** the different verification and testing techniques?

Conclusions

- Testing real-time systems is theoretically and practically challenging
- Promising techniques and tools
- 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
  - Many open research issues

Research Problems

- Testing Theory
- Timed games with partial observability
- Hybrid extensions
- Other Quantitative Properties
- Probabilistic Extensions, Performance testing
- Efficient data structures and algorithms for state set computation
- Diagnosis & Debugging
- Guiding and Coverage Measurement
- Real-Time execution of TRON
- Adaptor Abstraction, IUT clock synchronization
- Further Industrial Cases

Related Work

- Formal Testing Frameworks
  - [Brinksma, Tretmans]
- Real-Time Implementation Relations
  - [Khoumsi'03, Briones'04, Krichen'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]