Online Testing of Real-time Systems

Kim G. Larsen, Marius Mikučionis, Brian Nielsen

{ kgl, marius, bnielsen } @cs.aau.dk.

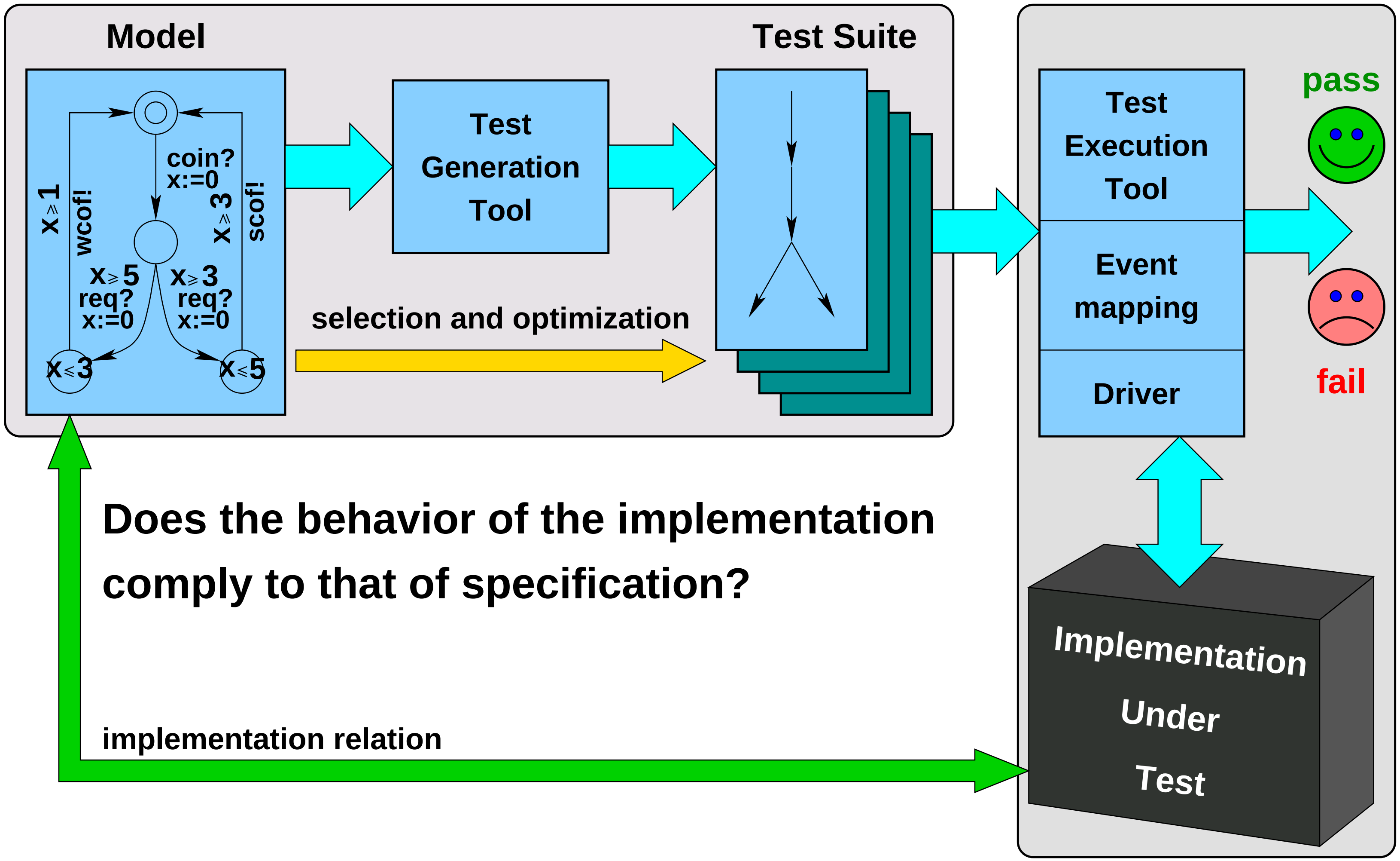
Center for Embedded Software Systems Basic Research in Computer Science Aalborg University

Overview

- Classical model-based testing.
- Motivation and related work for testing.
- Test setup: from specification to testing.
- Relativized timed input/output conformance relation.
- Online testing algorithm.
- Testing online in action.
- Symbolic techniques from UPPAAL.
- T-UPPAAL implementation details.
- Experiments with train-gate: mutants and benchmarks.
- Conclusions.
- Future work.

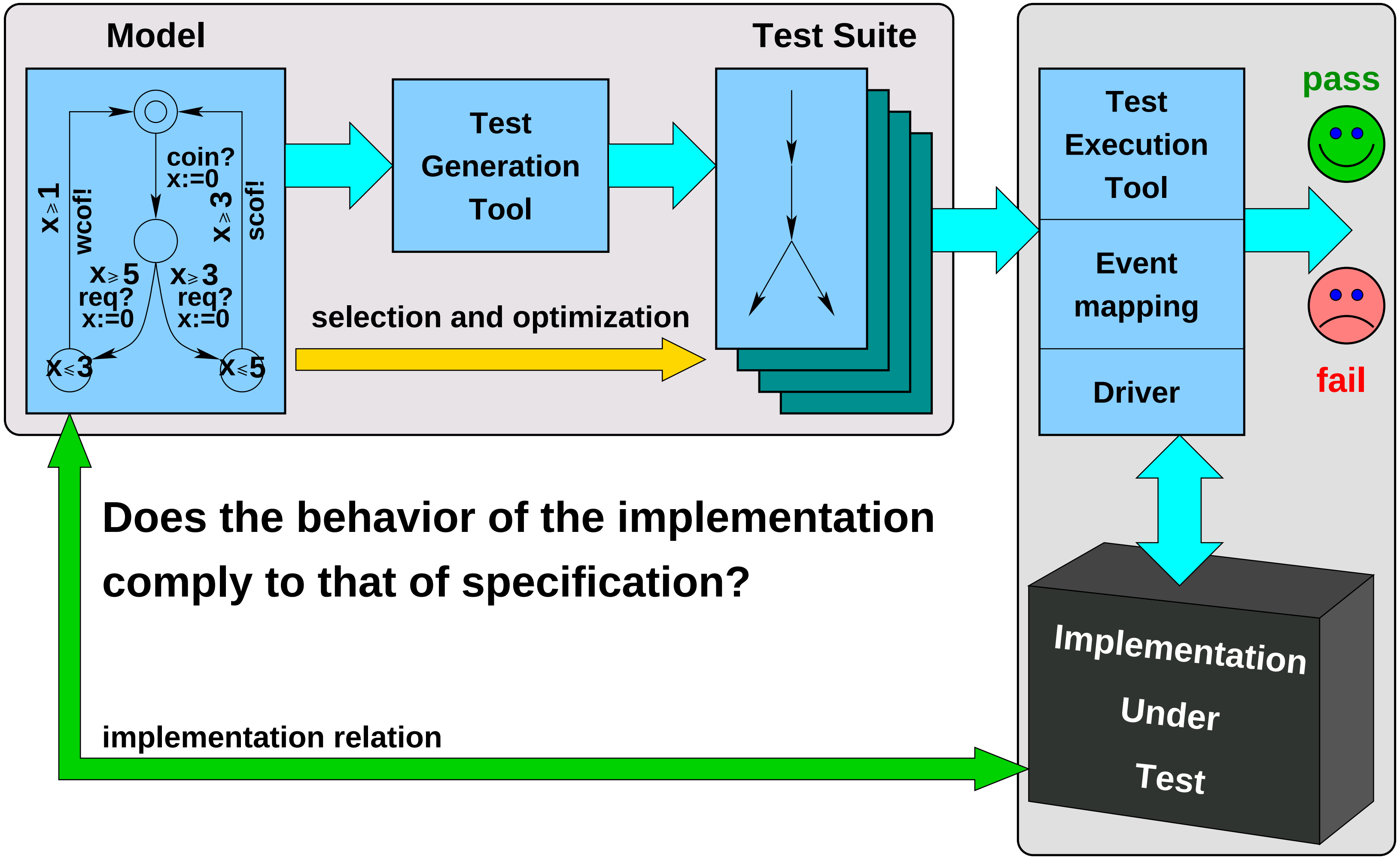
Classical Model-based Testing Framework

- Model-based, black-box, conformance testing offline.



Classical Model-based Testing Framework

- Model-based, black-box, conformance testing offline.



- Timed, online (on-the-fly generation and execution in real-time).

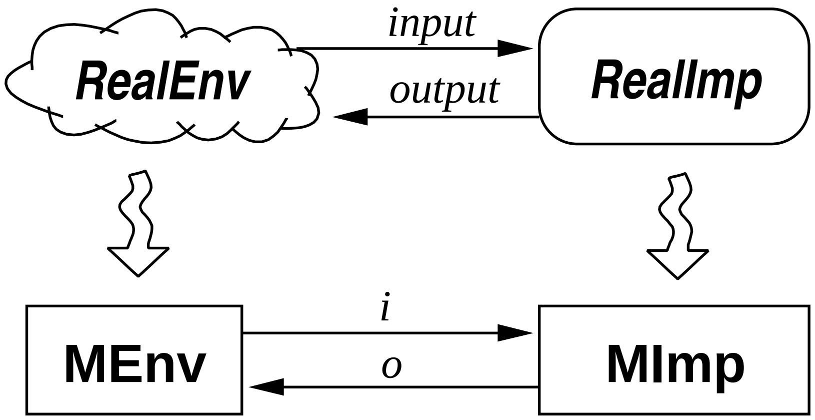
Motivation and Related Work for Testing

- Verification vs. testing: abstract models vs. real world.
- "Testing is routine and boring work": let's automate!
- Testing requires most of development skills.
- Testing is the main validation technique used by industry.
- About $1/3$ of project resources is spent on testing.
- Testing still remains ad-hoc, based on heuristics and error prone.

This work is based on the following ideas:

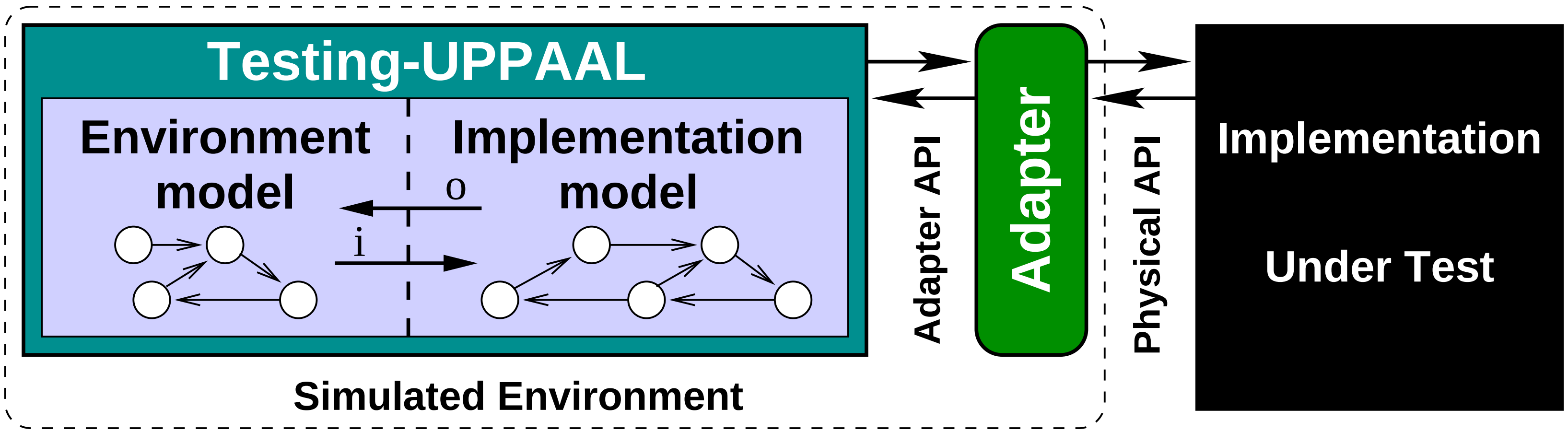
- Jan Tretmans' testing theory (un-timed, with quiescence).
- TORX testing tool framework (un-timed, without environment).
- UPPAAL model-checking algorithms for timed systems.
- Timed systems scheduling (?).

Test Setup: Systems ⇒ Models ⇒ Online Testing



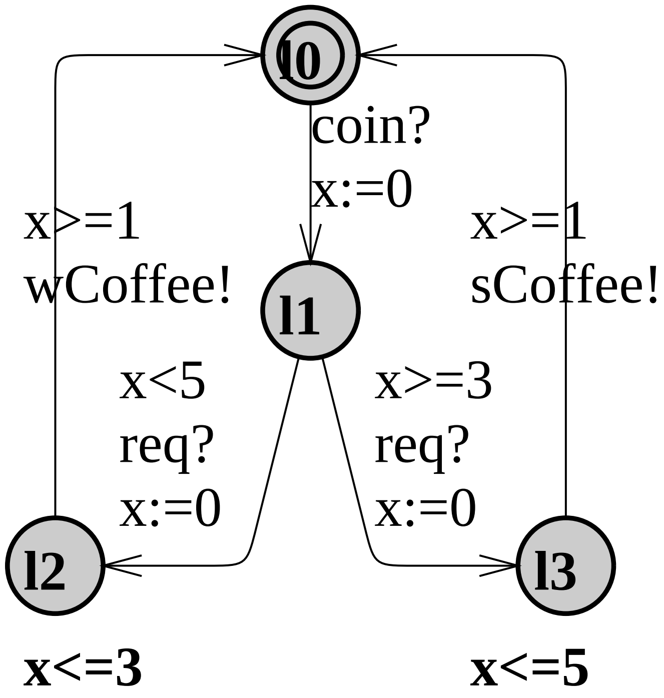
- Imp is (weakly) *input enabled*.
- *Clear* and *explicit* Env assumptions.
- Imp||Env forms a *closed* system.
- *Observable* input/output actions.

- Testing with general Env is *expensive* and often *unnecessary*.
- *Flexible*: only relevant behavior (Env change, guiding, debug).



- Generation and execution in *RT* allow *long* and *exhaustive* tests.

Test Specification: Timed Automata Networks



Timed automaton over A is $\langle L, l_0, X, D, E, I \rangle$:

- L – set of *locations*,
- $l_0 \in L$ – the *initial* location,
- X – set of real-valued *clocks*,
- D – bounded integer *variables*,
- $I : l \mapsto G(X)$ – location *invariant* mapping,
- $E \subseteq L \times G(X) \times A \times 2^{R(X)} \times L$ is a superset of directed *edges*: $l \xrightarrow{g,a,r} l'$ iff $\langle l, g, a, r, l' \rangle \in E$.

- Has *Labeled Transition System (LTS)* semantics.
- I/O, internal and timing *non-determinism* allow modelling parallelism, abstraction and possible time slacks.
- Test Spec: $\langle\, (\mathcal{E}_1 || \mathcal{E}_2 || \ldots || \mathcal{E}_n) || (\mathcal{I}_1 || \mathcal{I}_2 || \ldots || \mathcal{I}_n),\, A_{in},\, A_{out},\, T \,\rangle$

Relativized Timed Input/Output Conformance

- Idea: extend ioco (J.Tretmans) from TorX with time and env.
- Timed trace e.g.: $\sigma = coin? \cdot 5 \cdot req? \cdot 2 \cdot weakCoffee! \cdot 9 \cdot coin?$
- $\mathsf{TTr}(s)$ – set of *timed traces* from state s: $\{\sigma \in (A \cup \mathbb{R}_{\geq 0})^* \mid s \overset{\sigma}{\Rightarrow}\}$
- Timed trace *inclusion* as conf. relation: $\mathsf{TTr}(i) \subseteq \mathsf{TTr}(s)$
- *No illegal* output and *legal* output is observed at *right time*.

  $\mathsf{Out}\big(P\big) \overset{def}{=} \bigcup\{\alpha \in (A_{out} \cup \mathbb{R}_{\geq 0}) \mid p \in P.\, p \overset{\alpha}{\Rightarrow}\}$
- Relativized Timed Input/Output Conformance:

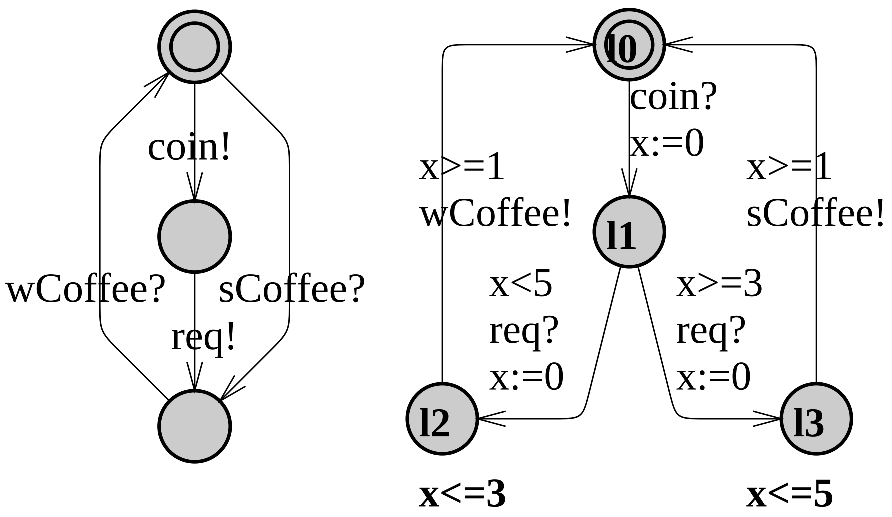
  $s \;\mathsf{rtioco}_e\; t \overset{def}{=} \forall \sigma \in \mathsf{TTr}(e).\mathsf{Out}\big((e,s)\ \mathsf{After}\ \sigma\big) \subseteq \mathsf{Out}\big((e,t)\ \mathsf{After}\ \sigma\big)$

  $$\boxed{s \;\mathsf{rtioco}_e\; t \iff \mathsf{TTr}(s) \cap \mathsf{TTr}(e) \subseteq \mathsf{TTr}(t) \cap \mathsf{TTr}(e)}$$
- Environment *ordering*. f is more discriminating than e:

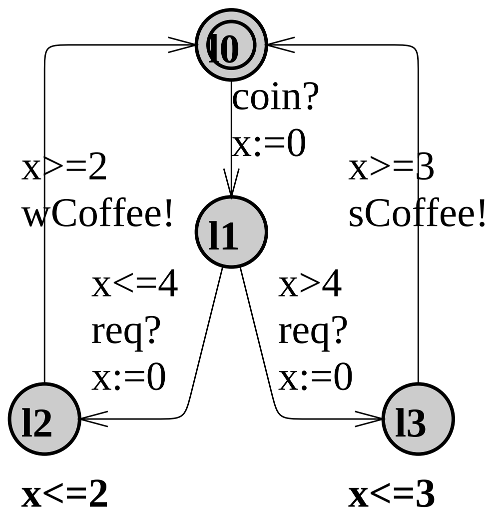
  $$e \sqsubseteq f \overset{def}{=} \mathsf{rtioco}_f \subseteq \mathsf{rtioco}_e$$

Timed I/O Conformance Relation Example

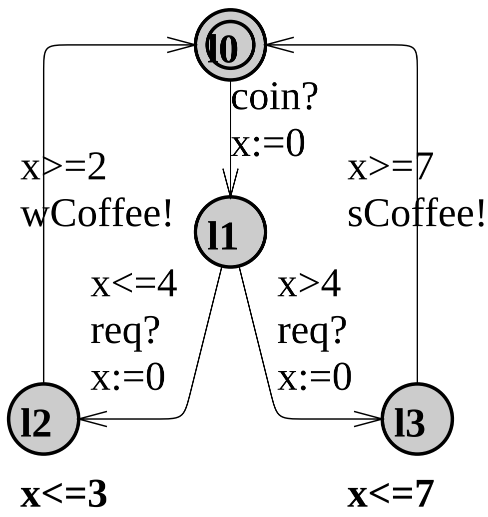
Specification s



Implementation i_1



Implementation i_2



Trace, σ	Out(s After σ)	Out(i_1 After σ)	Out(i_2 After σ)
$c \cdot 2$	$\mathbb{R}_{\geq 0}$	$\mathbb{R}_{\geq 0}$	$\mathbb{R}_{\geq 0}$
$c \cdot 4 \cdot r \cdot 1$	$\{wCoffee, sCoffee\} \cup [0,4]$	$[0,1]$	$[0,2]$
$c \cdot 4 \cdot r \cdot 2$	$\{wCoffee, sCoffee\} \cup [0,3]$	$\{wCoffee, 0\}$	$\{wCoffee\} \cup [0,1]$
$c \cdot 5 \cdot r \cdot 3$	$\{sCoffee\} \cup [0,2]$	$\{sCoffee, 0\}$	$[0,4]$
$c \cdot 5 \cdot r \cdot 5$	$\{sCoffee, 0\}$	$\emptyset$	$[0,2]$

Randomized Test Generation and Execution Online

while $\mathcal{Z} \neq \emptyset \wedge \sharp iterations \leq T$ **do** choose randomly:

1. **if** $\mathsf{EnvOutput}(\mathcal{Z}) \neq \emptyset$ // offer an input

   randomly choose $a \in \mathsf{EnvOutput}(\mathcal{Z})$

   send a to IUT

   $\mathcal{Z} := \mathcal{Z}$ After a

2. randomly choose $\delta \in \mathsf{Delays}(\mathcal{Z})$ // wait for an output

   sleep for δ time units and wake up on output o

   if o occurs at $\delta' \leq \delta$ **then**

   $\mathcal{Z} := \mathcal{Z}$ After δ'

   if $o \notin \mathsf{ImpOutput}(\mathcal{Z})$ **then return fail**

   else $\mathcal{Z} := \mathcal{Z}$ After o

   else $\mathcal{Z} := \mathcal{Z}$ After δ // no output within δ delay

3. $\mathcal{Z} := \{(s_0, e_0)\}$, **reset** IUT //reset and restart

if $\mathcal{Z} = \emptyset$ **then return fail else return pass**

Randomized Test Generation and Execution Online

while $\mathcal{Z} \neq \emptyset \wedge \sharp iterations \leq T$ **do** choose randomly:

1. **if** $\mathsf{EnvOutput}(\mathcal{Z}) \neq \emptyset$ // offer an input

   randomly choose $a \in \mathsf{EnvOutput}(\mathcal{Z})$

   send a to IUT

   $\mathcal{Z} := \mathcal{Z}$ After a

2. randomly choose $\delta \in \mathsf{Delays}(\mathcal{Z})$ // wait for an output

   sleep for δ time units and wake up on output o

   if o occurs at $\delta' \leq \delta$ **then**

   $\mathcal{Z} := \mathcal{Z}$ After δ'

   if $o \notin \mathsf{ImpOutput}(\mathcal{Z})$ **then return fail**

   else $\mathcal{Z} := \mathcal{Z}$ After o

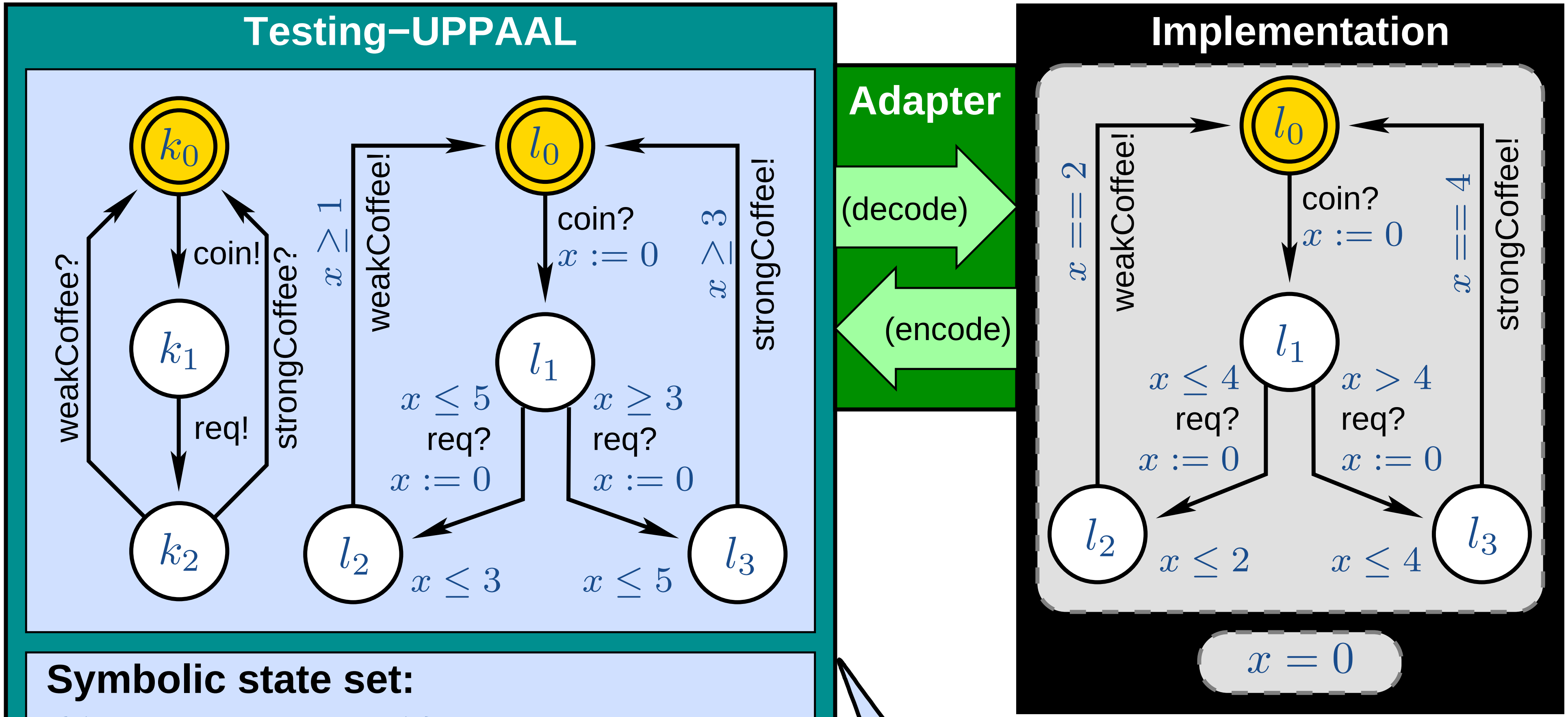
   else $\mathcal{Z} := \mathcal{Z}$ After δ // no output within δ delay

3. $\mathcal{Z} := \{(s_0, e_0)\}$, **reset** IUT //reset and restart

if $\mathcal{Z} = \emptyset$ **then return fail else return pass**

sound and complete in limit

Testing Online in Action



Symbolic state set:

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

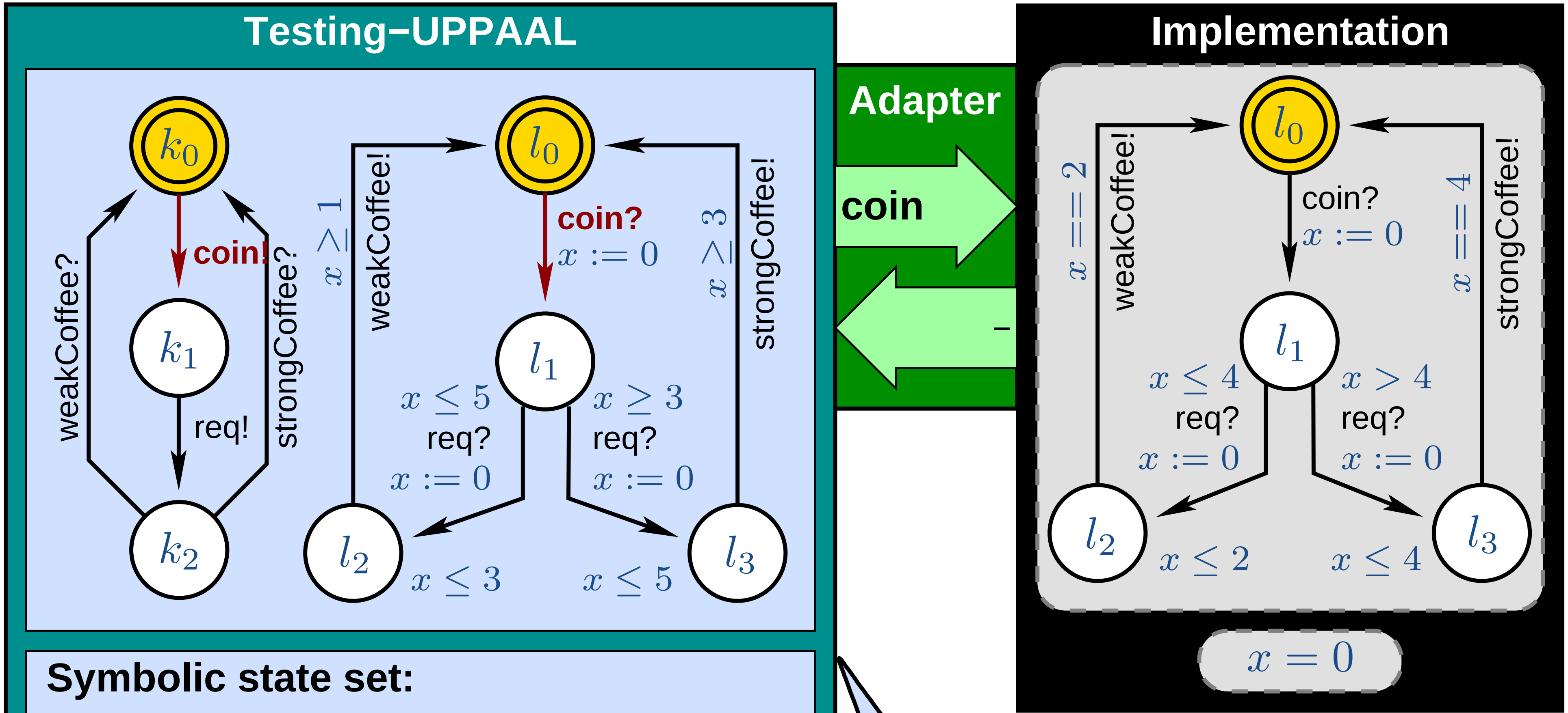
EnvOutput: $\{\text{coin}\}$

EnvInput: $\emptyset$

ImpOutput: $\emptyset$

Wait for output (delay) or offer input?

Testing Online in Action



Symbolic state set:

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

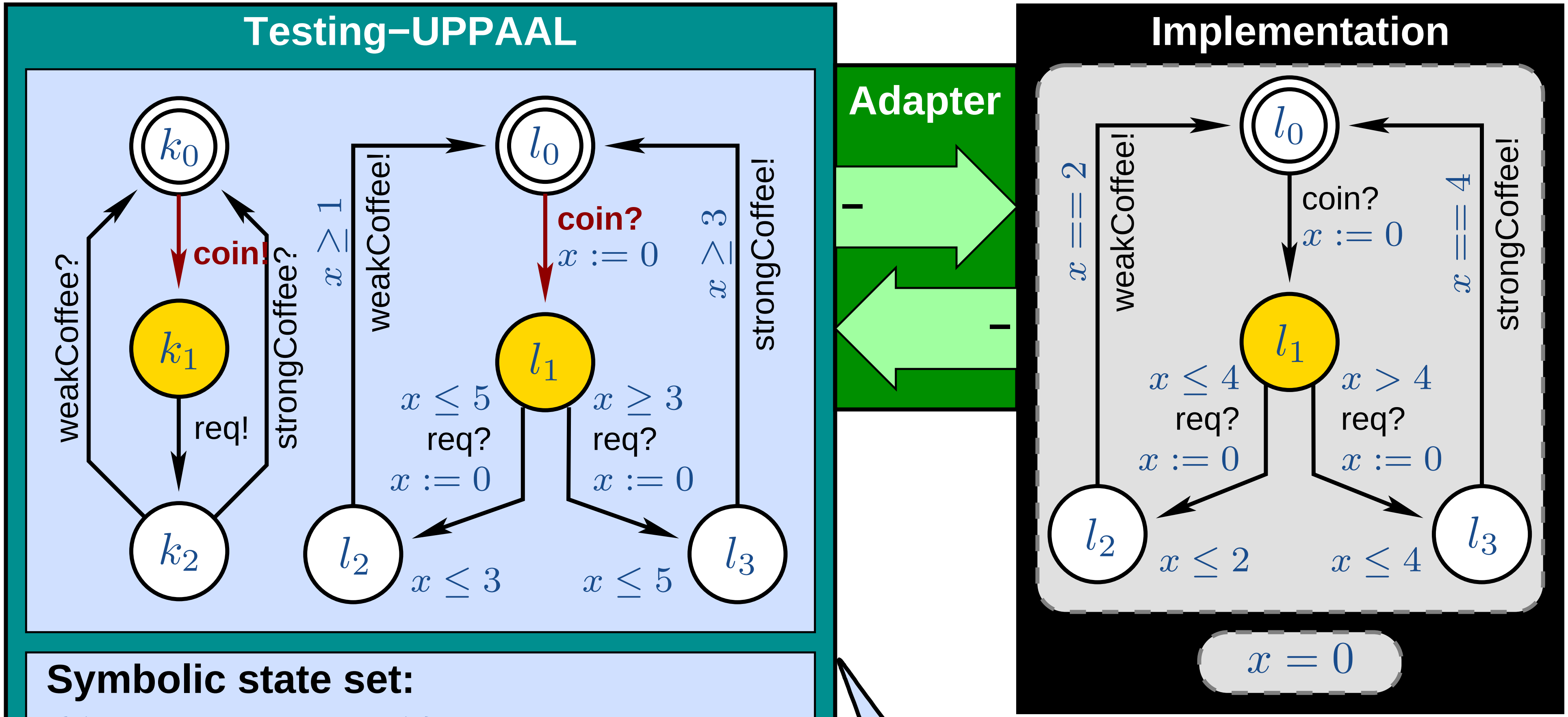
EnvOutput: $\{\text{coin}\}$

EnvInput: $\emptyset$

ImpOutput: $\emptyset$

Let's offer input choose (the only) "coin"

Testing Online in Action



Symbolic state set:

$\{\langle k_1 l_1, 0 \leq x \leq 0 \rangle\}$

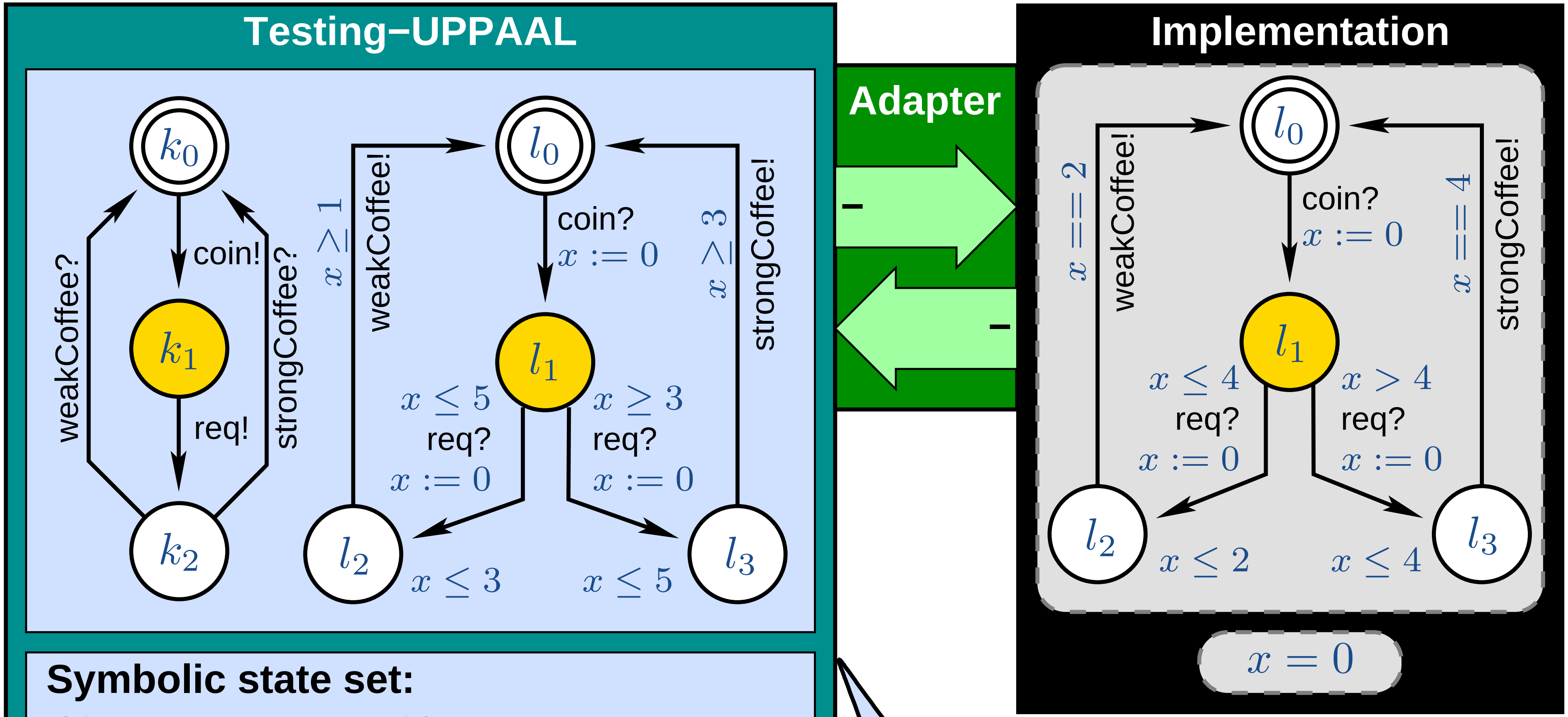
EnvOutput: $\{$req$\}$

EnvInput: $\emptyset$

ImpOutput: $\emptyset$

Update the state set and other variables

Testing Online in Action



Symbolic state set:

$\{\langle k_1 l_1, 0 \le x \le 0\rangle\}$

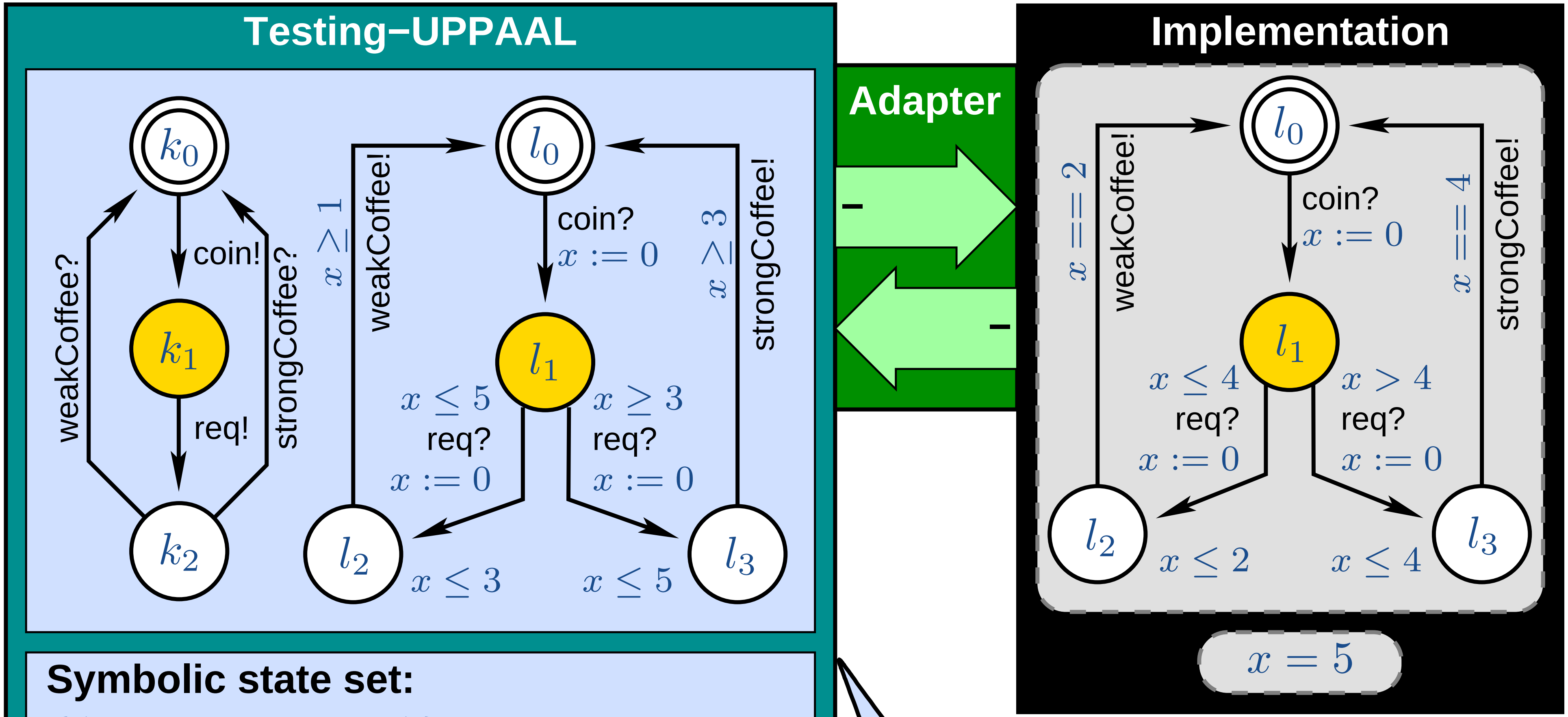
EnvOutput: $\{$req$\}$

EnvInput: $\emptyset$

ImpOutput: $\emptyset$

Wait or offer input?
Let's wait for 5 units

Testing Online in Action



Symbolic state set:

$\{\langle k_1 l_1, 5 \leq x \leq 5\rangle\}$

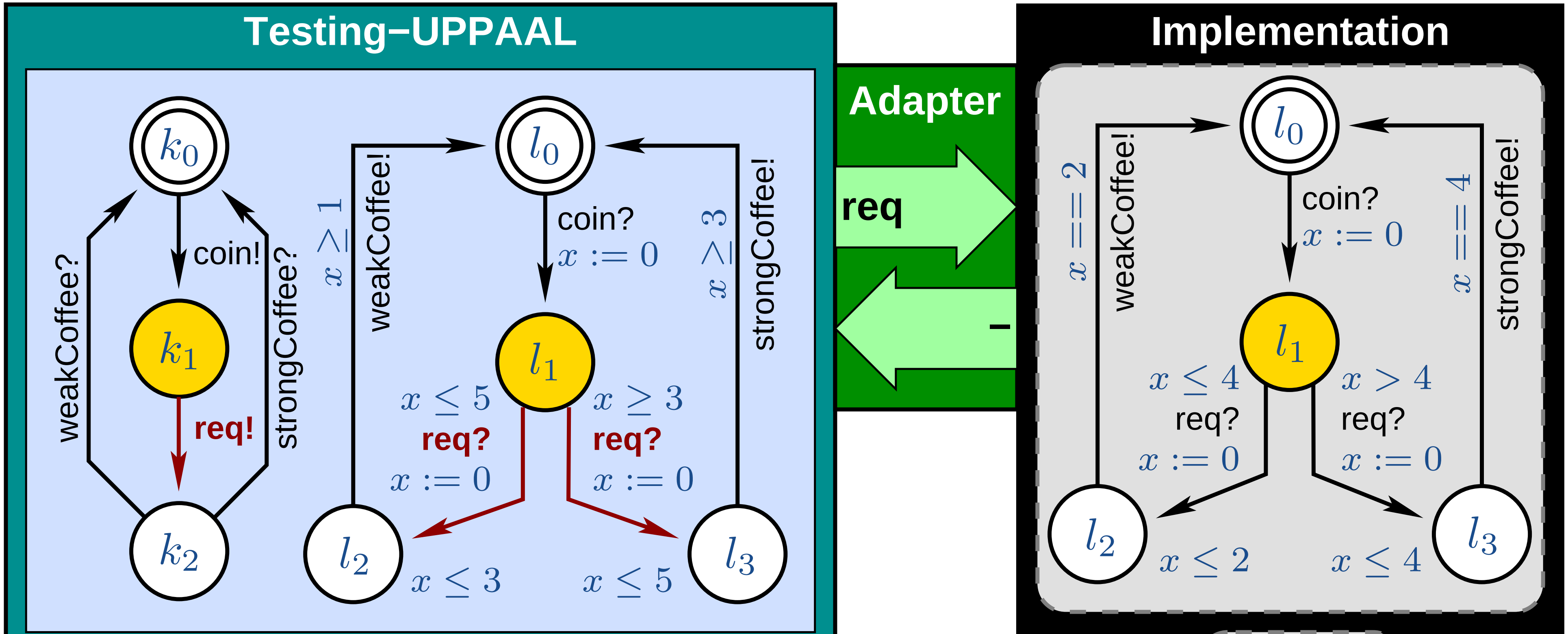
EnvOutput: $\{$req$\}$

EnvInput: $\emptyset$

ImpOutput: $\emptyset$

..no output so far: update the state set..

Testing Online in Action



Symbolic state set:

$\{\langle k_1 l_1, 5 \le x \le 5\rangle\}$

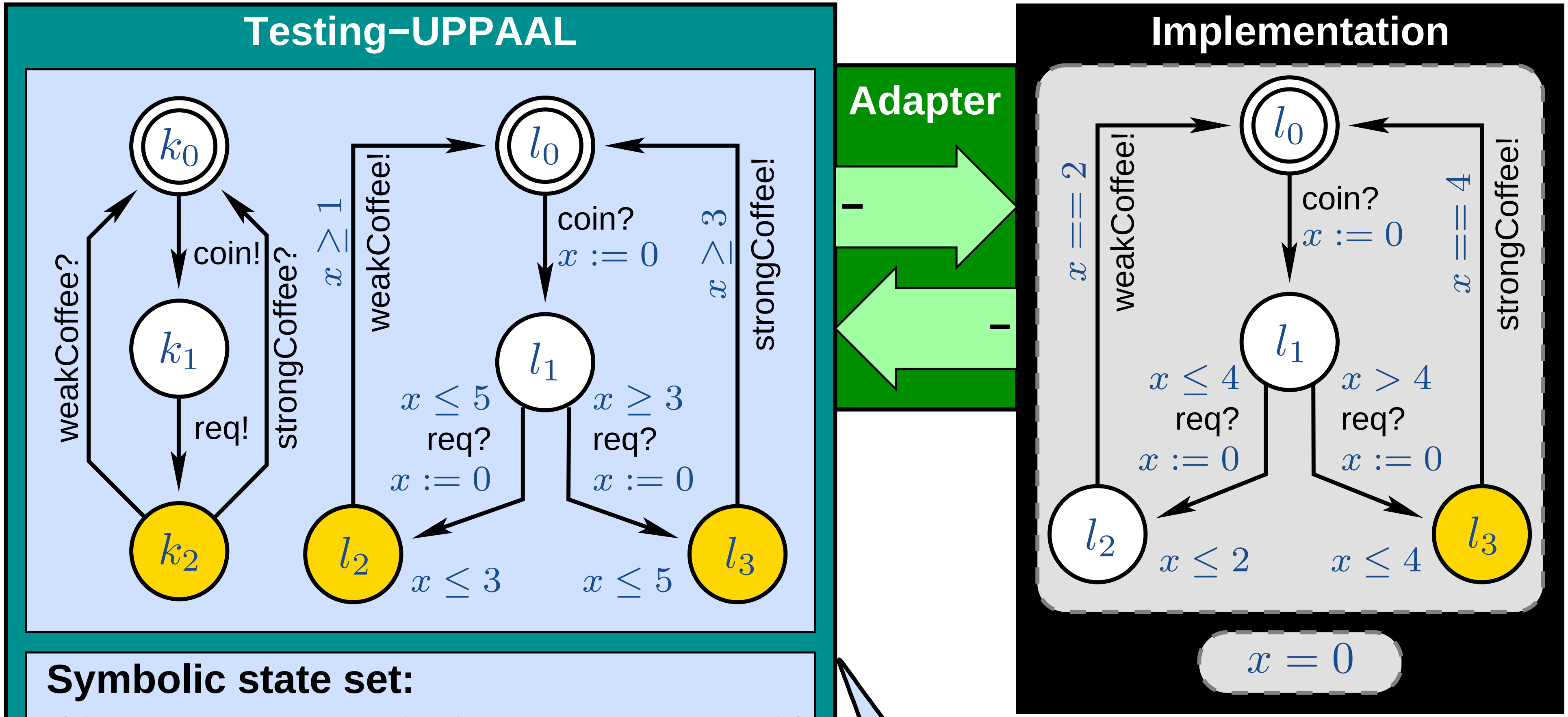
EnvOutput: $\{$req$\}$

EnvInput: $\emptyset$

ImpOutput: $\emptyset$

Wait or offer input?
let's offer "req"

Testing Online in Action



Symbolic state set:

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

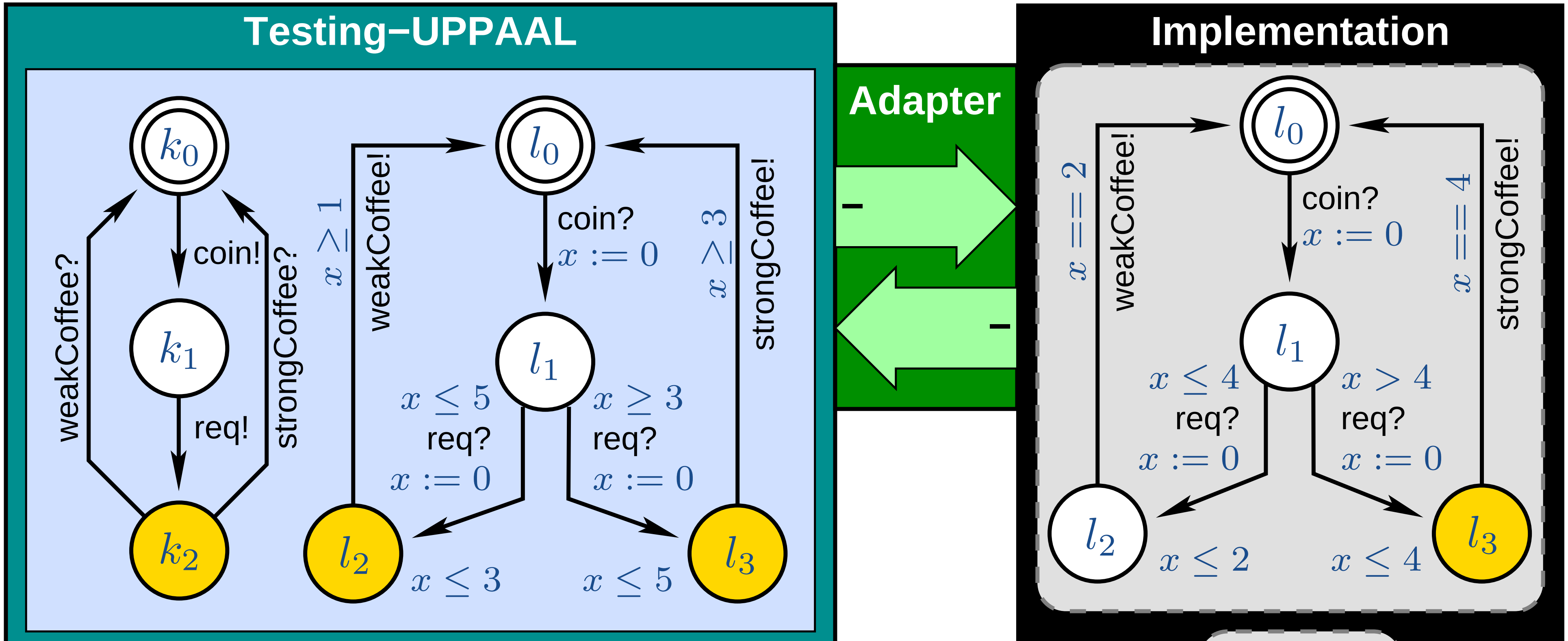
EnvOutput: $\emptyset$

EnvInput: {weakCoffee, strongCoffee}

ImpOutput: {weakCoffee, strongCoffee}

Update the state set and other variables

Testing Online in Action



Symbolic state set:

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

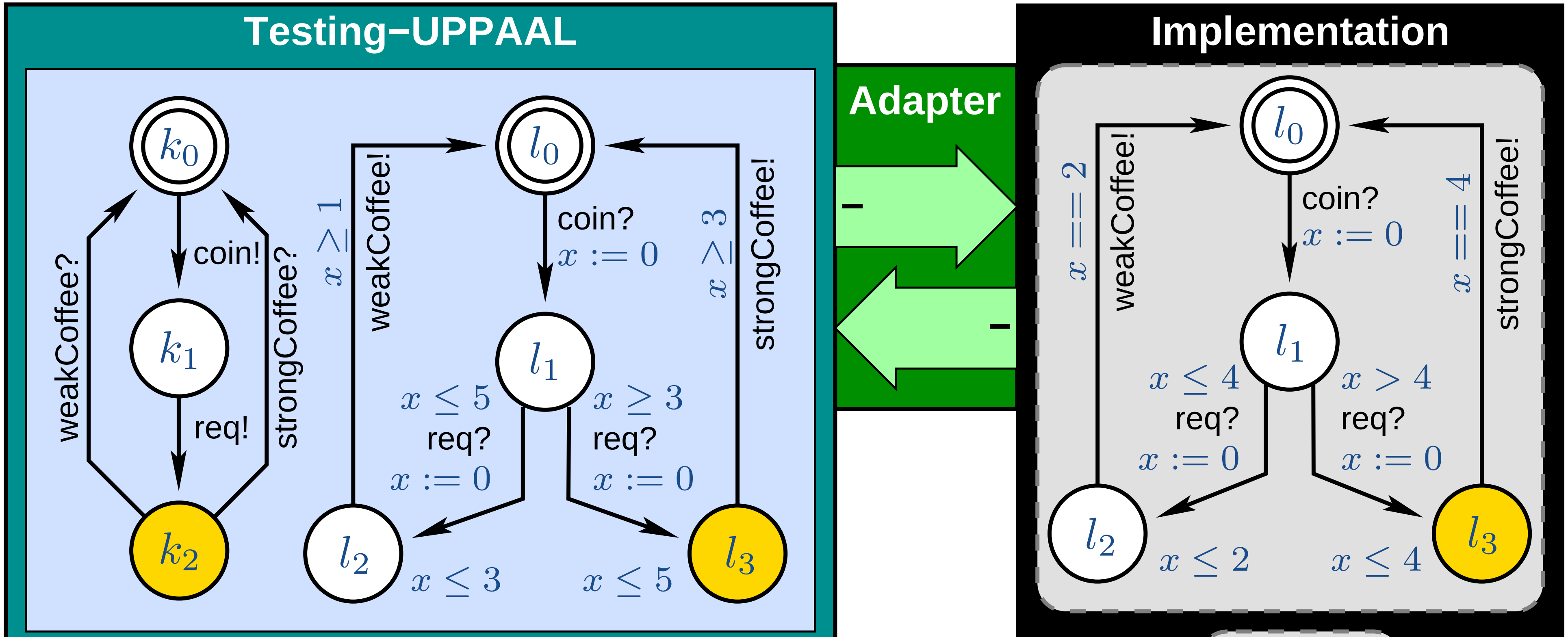
EnvOutput: $\emptyset$

EnvInput: {weakCoffee, strongCoffee}

ImpOutput: {weakCoffee, strongCoffee}

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

Testing Online in Action



Symbolic state set:

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

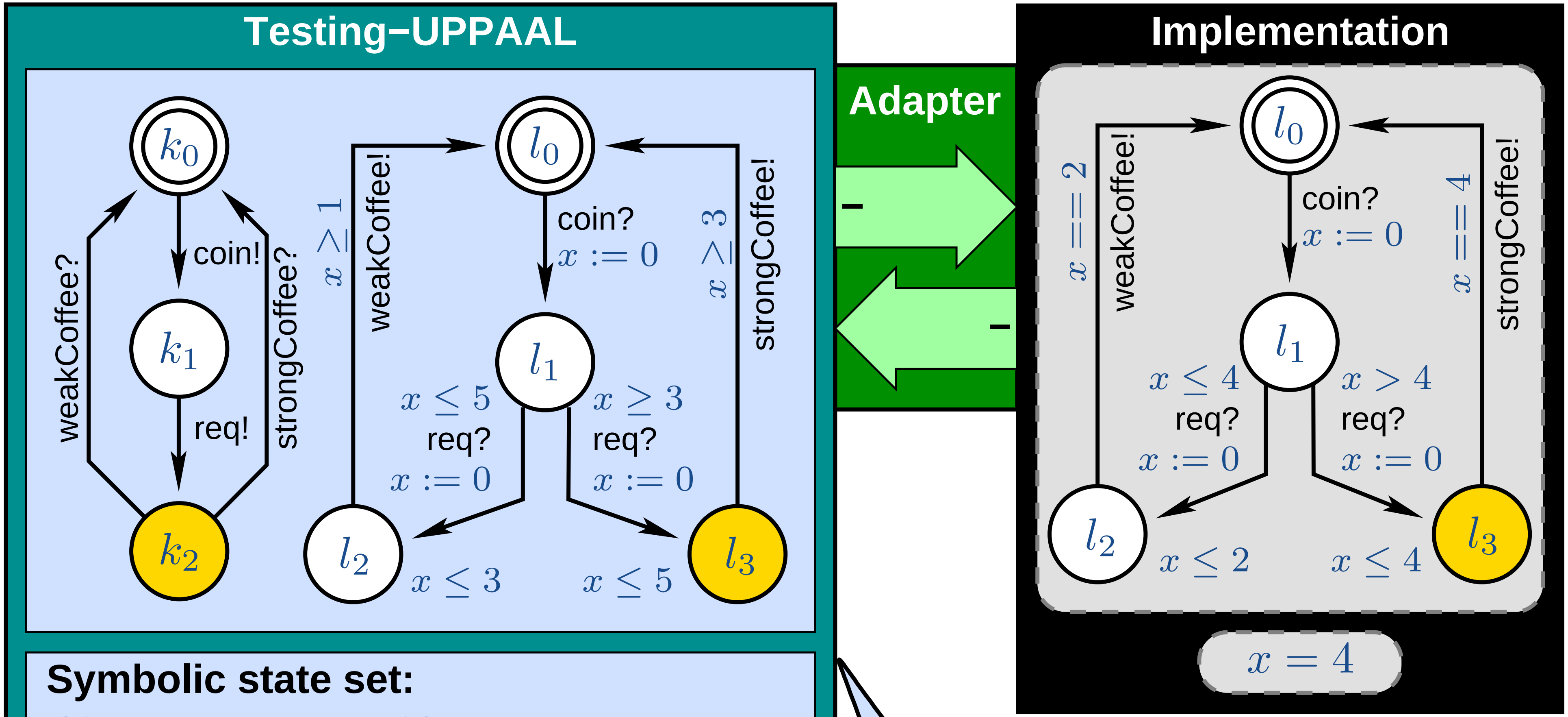
EnvOutput: $\emptyset$

EnvInput: {strongCoffee}

ImpOutput: {strongCoffee}

..no output so far: update the state set..

Testing Online in Action



Symbolic state set:

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

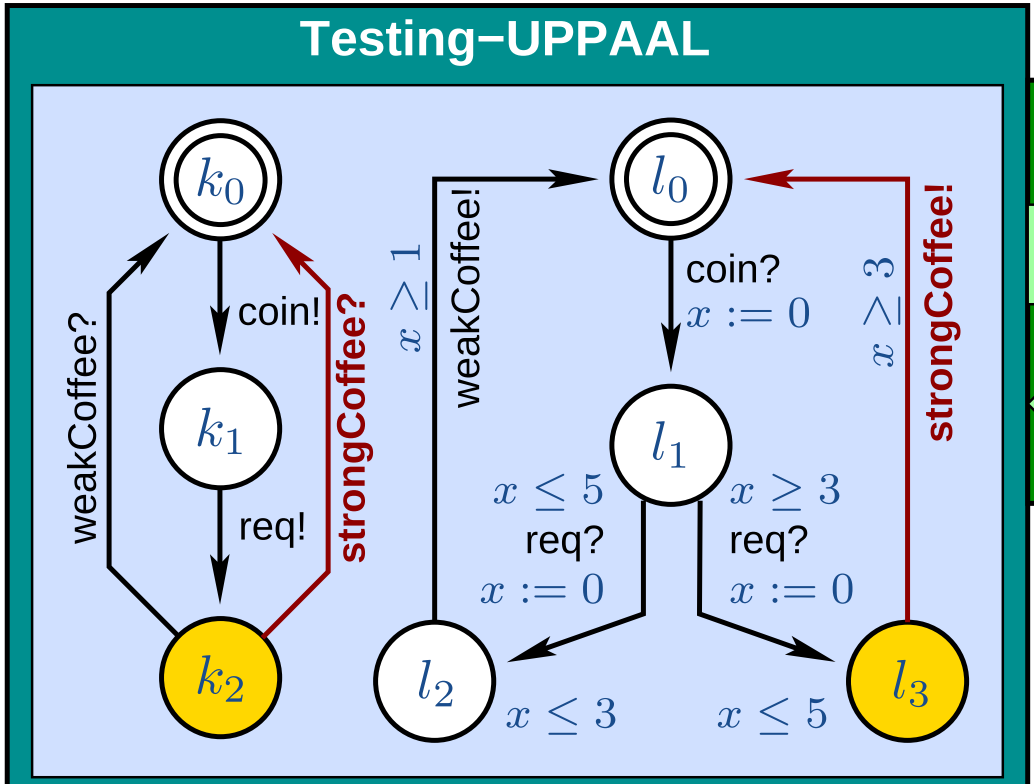
EnvOutput: $\emptyset$

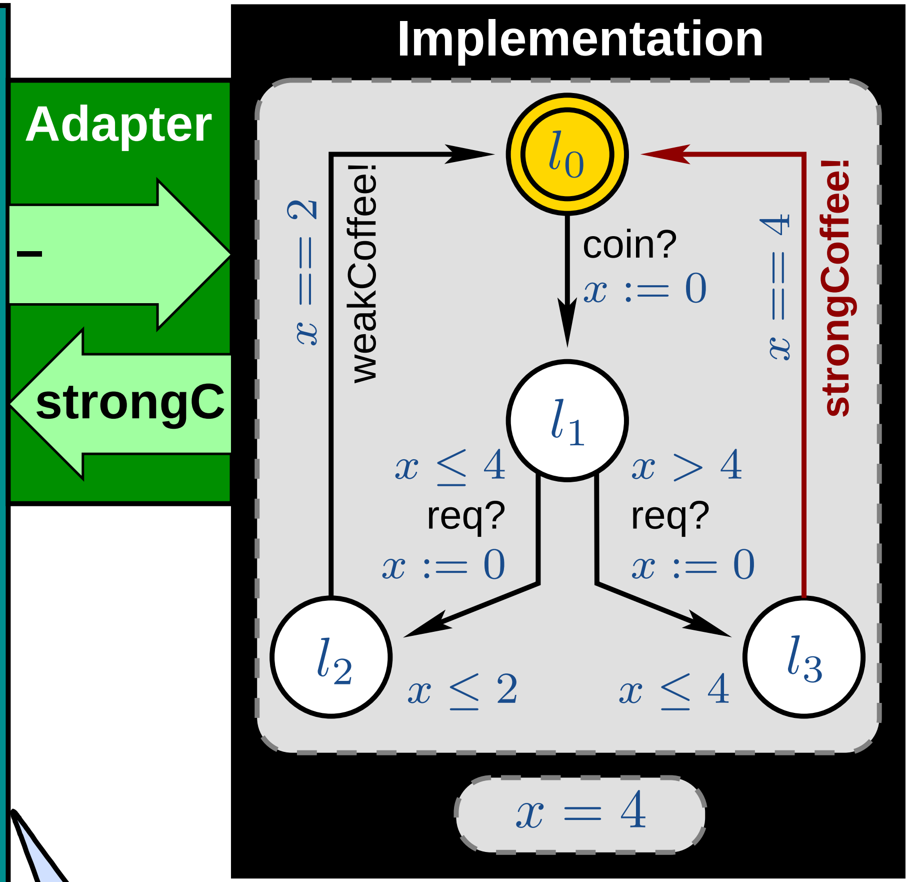
EnvInput: {strongCoffee}

ImpOutput: {strongCoffee}

Wait or offer input?
Let's wait for 2 units

Testing Online in Action





Symbolic state set:

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

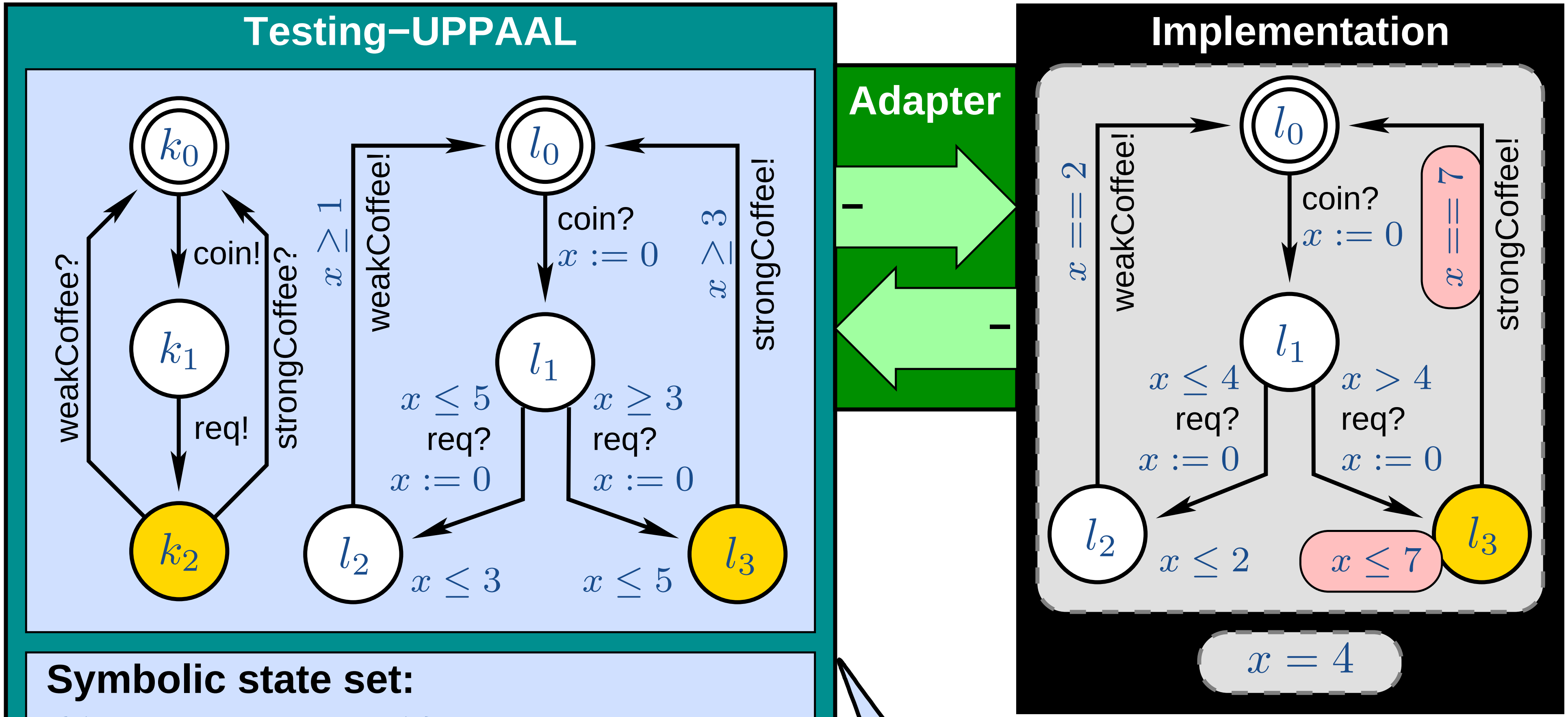
EnvOutput: $\emptyset$

EnvInput: $\{$strongCoffee$\}$

ImpOutput: $\{$strongCoffee$\}$

got output after 0 delay: update the state set

Testing Online in Action



Symbolic state set:

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

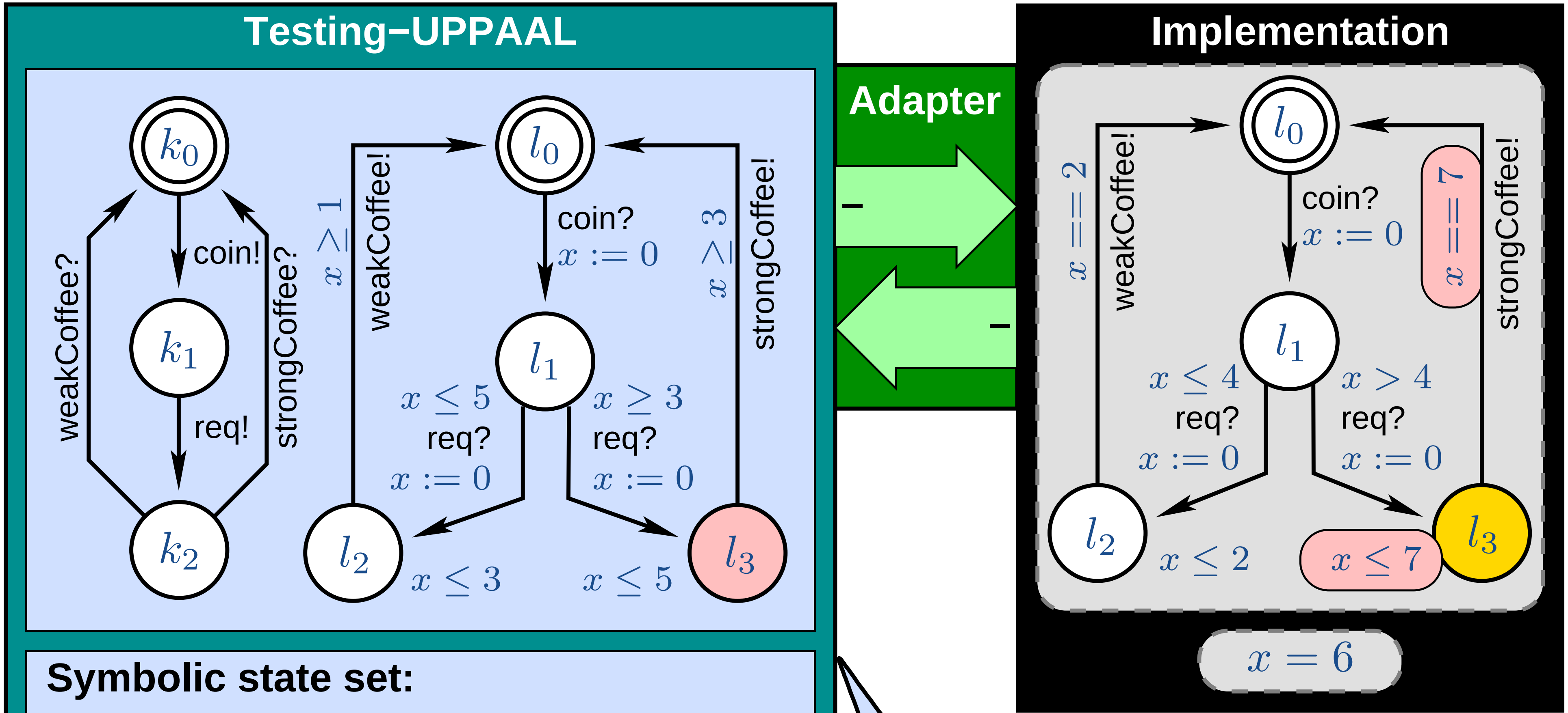
EnvOutput: $\emptyset$

EnvInput: {strongCoffee}

ImpOutput: {strongCoffee}

(what if there is a bug?)
Let's wait for 2 units

Testing Online in Action



Symbolic state set:

$\emptyset$

EnvOutput: $\emptyset$

EnvInput: $\emptyset$

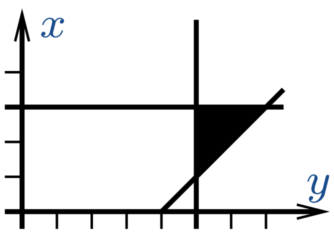
ImpOutput: $\emptyset$

..no output so far: update the state set.. (!)

Symbolic Techniques in UPPAAL

- *Zone* is a conjunction of clock constraints of the form:
  $\{x_i - x_j \prec c_{ij}\} \cup \{a_i \prec x_i\} \cup \{x_j \prec b_j\}$ where $\prec \in \{<, \leq\}$

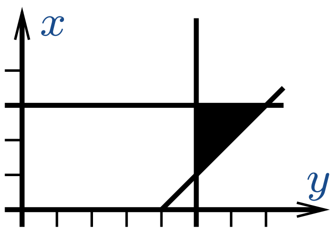
$$z = [(y - x \leq 4) \wedge (y \geq 5) \wedge (x \leq 3)]$$



Symbolic Techniques in UPPAAL

- *Zone* is a conjunction of clock constraints of the form: $\{x_i - x_j \prec c_{ij}\} \cup \{a_i \prec x_i\} \cup \{x_j \prec b_j\}$ where $\prec \in \{<, \leq\}$
- *Difference bound matrix* – compact representation.

$$z = [(y - x \leq 4) \wedge (y \geq 5) \wedge (x \leq 3)]$$

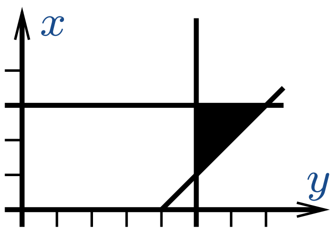


	0	y	x
0	–	-5	0
y	∞	–	4
x	3	∞	–

Symbolic Techniques in UPPAAL

- *Zone* is a conjunction of clock constraints of the form: $\{x_i - x_j \prec c_{ij}\} \cup \{a_i \prec x_i\} \cup \{x_j \prec b_j\}$ where $\prec \in \{<, \leq\}$
- *Difference bound matrix* – compact representation.
- Symbolic state set $\mathcal{Z} = \{\langle \bar{l}_1, z_1 \rangle, \ldots, \langle \bar{l}_n, z_n \rangle\}$

$$z = [(y - x \leq 4) \wedge (y \geq 5) \wedge (x \leq 3)]$$

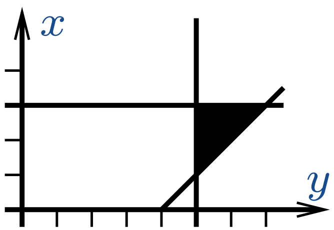


	0	y	x
0	–	-5	0
y	∞	–	4
x	3	∞	–

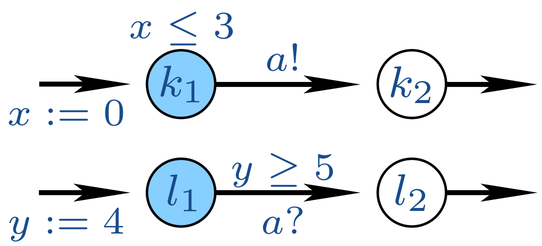
Symbolic Techniques in UPPAAL

- *Zone* is a conjunction of clock constraints of the form: $\{x_i - x_j \prec c_{ij}\} \cup \{a_i \prec x_i\} \cup \{x_j \prec b_j\}$ where $\prec \in \{<, \leq\}$
- *Difference bound matrix* – compact representation.
- Symbolic state set $\mathcal{Z} = \{\langle \bar{l}_1, z_1 \rangle, \ldots, \langle \bar{l}_n, z_n \rangle\}$
- *Action* transition: $\langle \bar{l}, z \rangle \xrightarrow{a} \langle \bar{l}', (z \wedge g)_r \wedge I(\bar{l}') \rangle$ **iff**:
  $l \xrightarrow{g,a,r} l'$ is a-action transition and $z \wedge g \neq \emptyset$,$(z \wedge g)_r \wedge I(\bar{l}') \neq \emptyset$.

$$z = [(y - x \leq 4) \wedge (y \geq 5) \wedge (x \leq 3)]$$



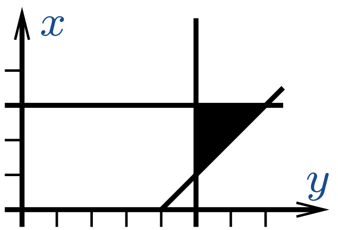
	0	y	x
0	–	-5	0
y	∞	–	4
x	3	∞	–



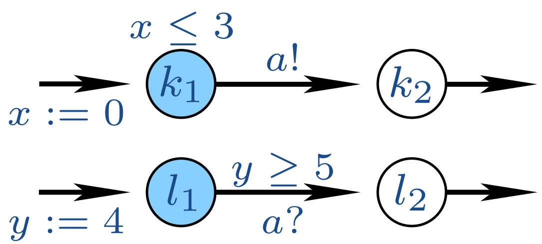
Symbolic Techniques in UPPAAL

- *Zone* is a conjunction of clock constraints of the form: $\{x_i - x_j \prec c_{ij}\} \cup \{a_i \prec x_i\} \cup \{x_j \prec b_j\}$ where $\prec \in \{<, \leq\}$
- *Difference bound matrix* – compact representation.
- Symbolic state set $\mathcal{Z} = \{\langle \bar{l}_1, z_1 \rangle, \ldots, \langle \bar{l}_n, z_n \rangle\}$
- *Action* transition: $\langle \bar{l}, z \rangle \xrightarrow{a} \langle \bar{l}', (z \wedge g)_r \wedge I(\bar{l}') \rangle$ **iff**: $l \xrightarrow{g,a,r} l'$ is a-action transition and $z \wedge g \neq \emptyset$,$(z \wedge g)_r \wedge I(\bar{l}') \neq \emptyset$.
- *Delay* transition: $\langle \bar{l}, z \rangle \xrightarrow{\delta} \langle \bar{l}, z^{+\delta} \wedge I(\bar{l}) \rangle$ **iff** $z^{+\delta} \wedge I(\bar{l}) \neq \emptyset$.

$$z = [(y - x \leq 4) \wedge (y \geq 5) \wedge (x \leq 3)]$$



	0	y	x
0	–	-5	0
y	∞	–	4
x	3	∞	–

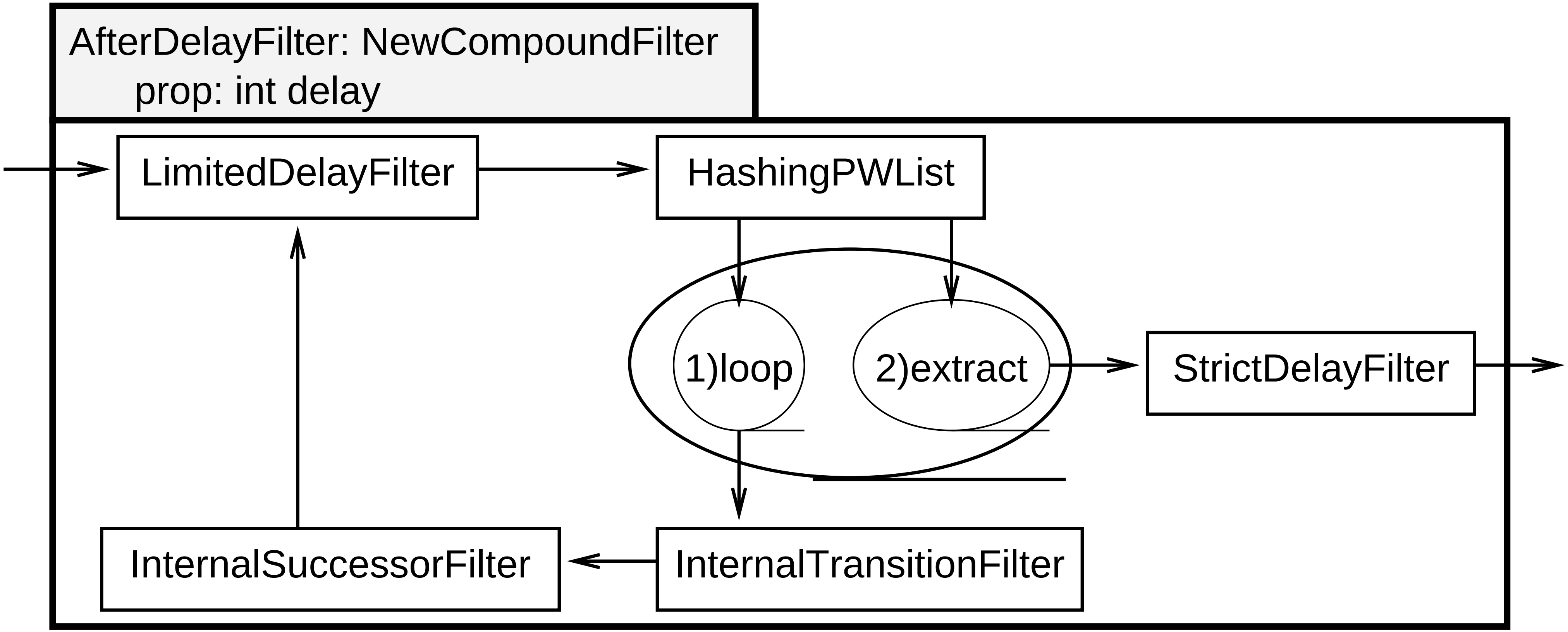


T-UPPAAL Implementation Details

- Reachability algorithms for *afterDelay* and *afterAction*

$$\text{Closure}_{\delta_\tau}(\mathcal{Z}, d) = \bigcup_{0 \le \delta \le d} \left\{ \langle \bar{\ell}', z' \rangle \mid \langle \bar{\ell}, z \rangle \in \mathcal{Z},\ \langle \bar{\ell}, z \rangle \xRightarrow{\delta} \langle \bar{\ell}', z' \rangle \right\}$$

$$\mathcal{Z} \text{ After } d = \left\{ \langle \bar{\ell}, z' \rangle \mid \langle \bar{l}, z \rangle \in \text{Closure}_{\delta_\tau}(\mathcal{Z}, d),\ z' = \left(z \wedge (t == d) \right)_{t:=0} \right\}$$

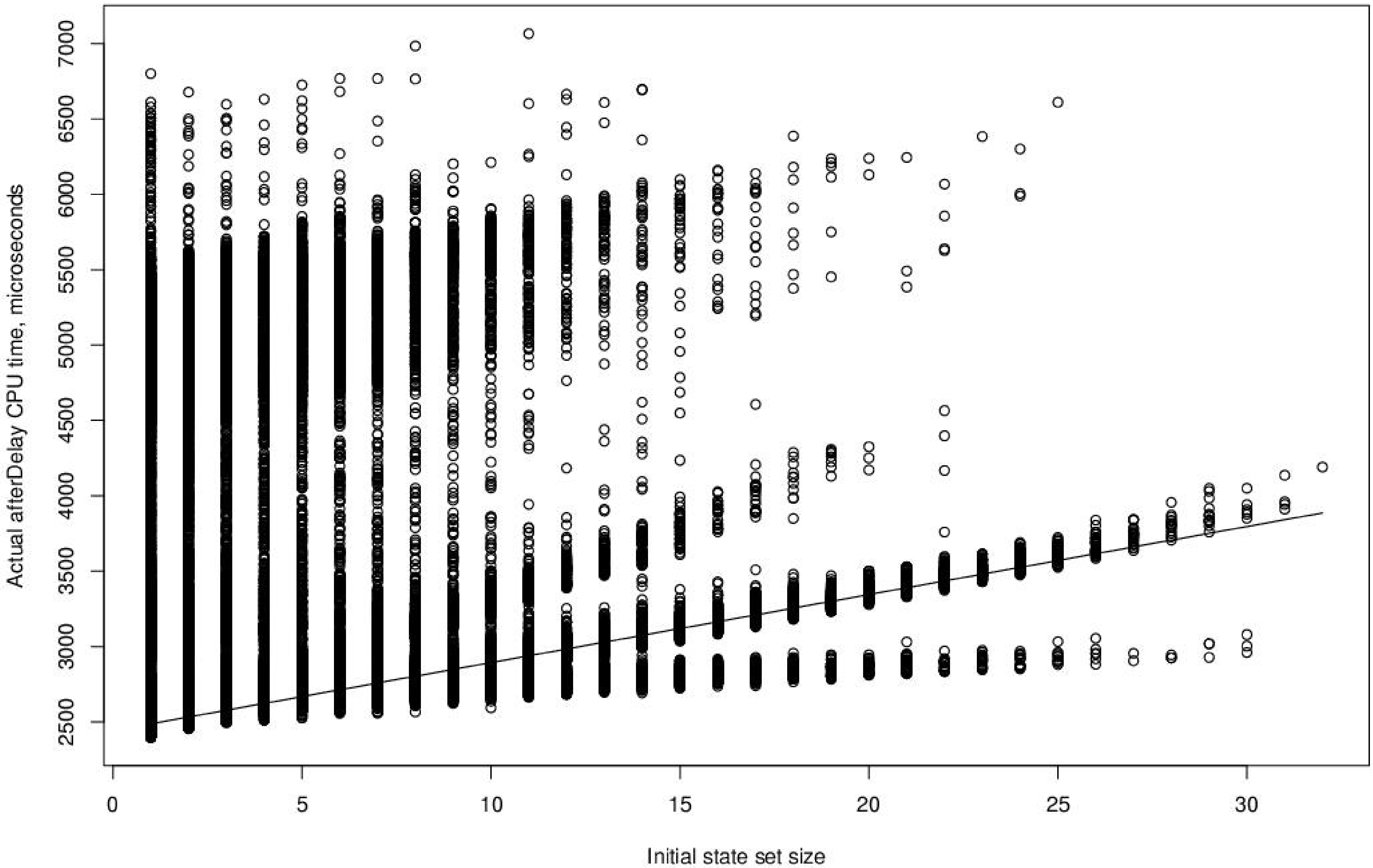


Mutant Experiment: Error Detection Capability

- Specification: train-gate example of 9 timed automata.
- Implementation: 4 threads with a shared queue in C++.
- 7 mutants: M1-M6 with seeded error, M0 correct.

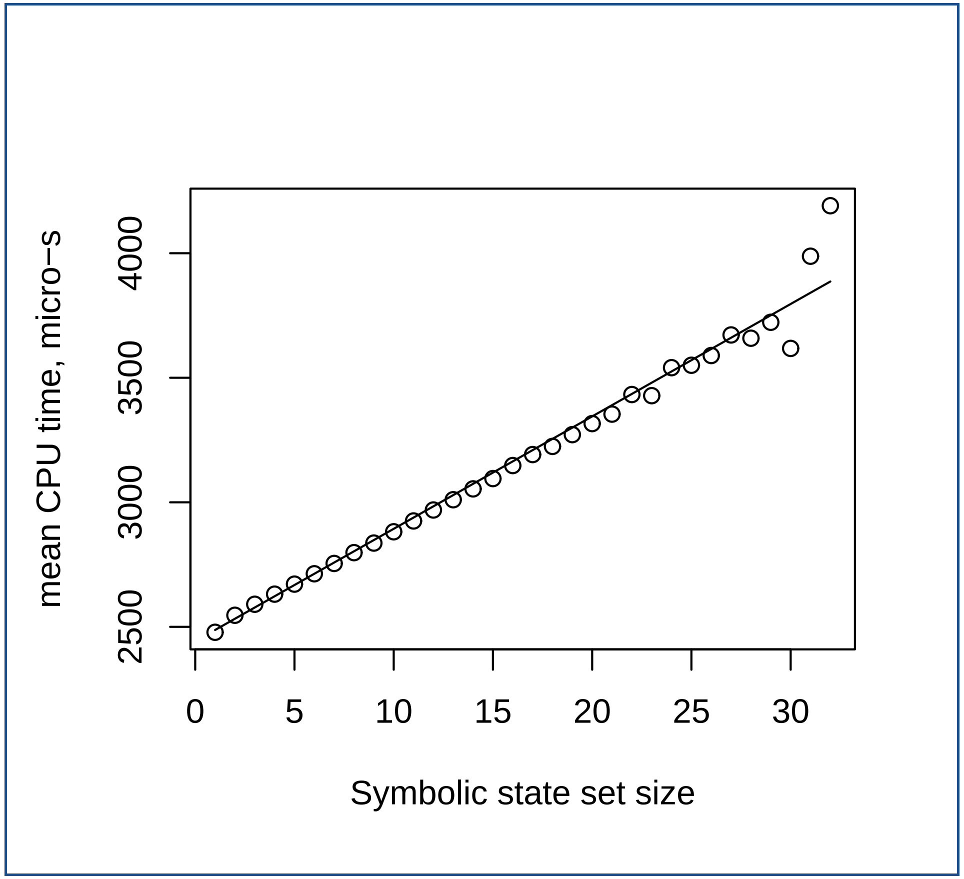
Mu-tant	Number of input actions			Duration, *mtu*		
	Min	Avg	Max	Min	Avg	Max
M1	2	4.8	16	0	68.8	318
M2	2	4.6	13	1	66.4	389
M3	2	4.7	14	0	66.4	398
M4	6	8.5	18	28	165.0	532
M5	4	5.6	12	14	89.8	364
M6	2	14.1	92	0	299.6	2077
M0	3565	3751.4	3966	10^5	10^5	10^5

Benchmark Data: Computing Performance (instances)

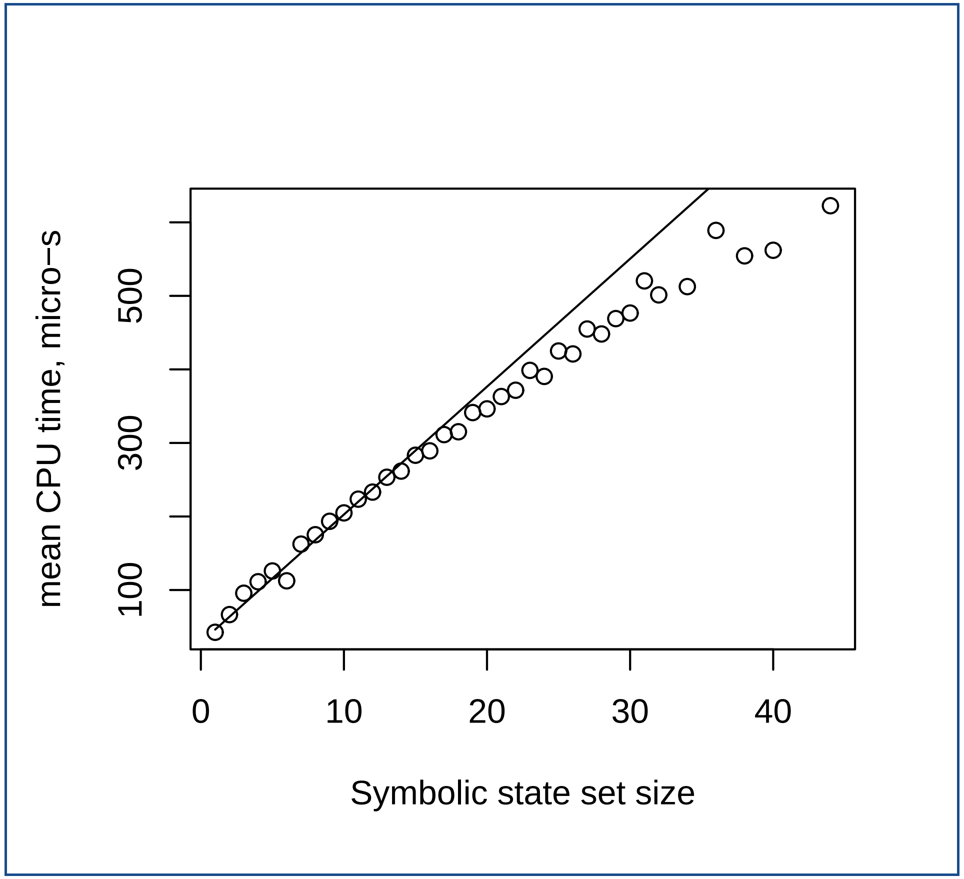


Benchmark Data: Computing Performance (means)

after delay



after action



Benchmark Data: Summary

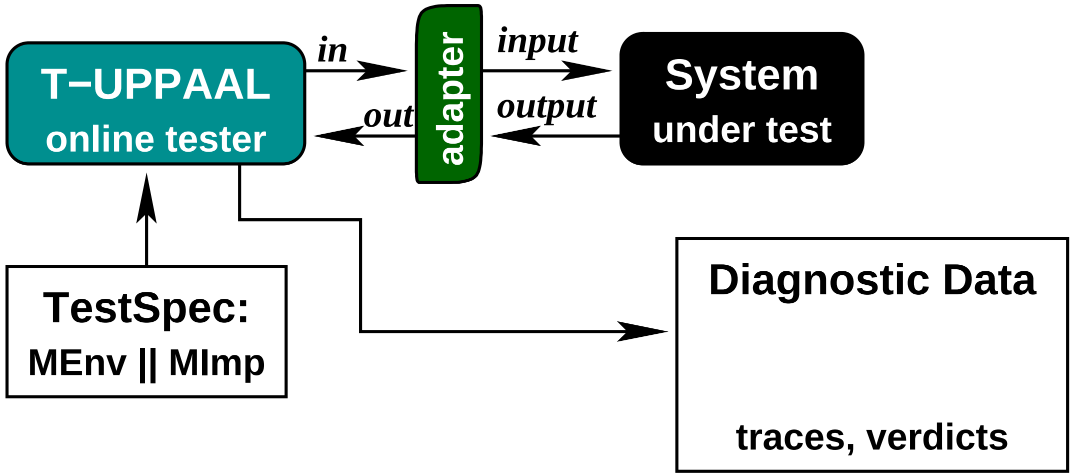
- Executed on Sun SPARC, 8x900MHz, 32GB RAM, Sun Solaris 9.

Mu-tant	Number of states in $\mathcal{Z}$				CPU execution time, μs			
	After (delay)		After (action)		After (delay)		After (action)	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
M1	2.3	18	2.7	28	1113	3128	141	787
M2	2.3	22	2.8	30	1118	3311	147	791
M3	2.2	22	2.7	30	1112	3392	141	834
M4	**2.8**	24	**3.1**	**48**	1113	3469	125	936
M5	**2.8**	24	**3.3**	**48**	1131	3222	146	919
M6	2.7	27	2.9	36	1098	3531	110	861
M0	**2.7**	31	2.9	**46**	**1085**	3591	**101**	950

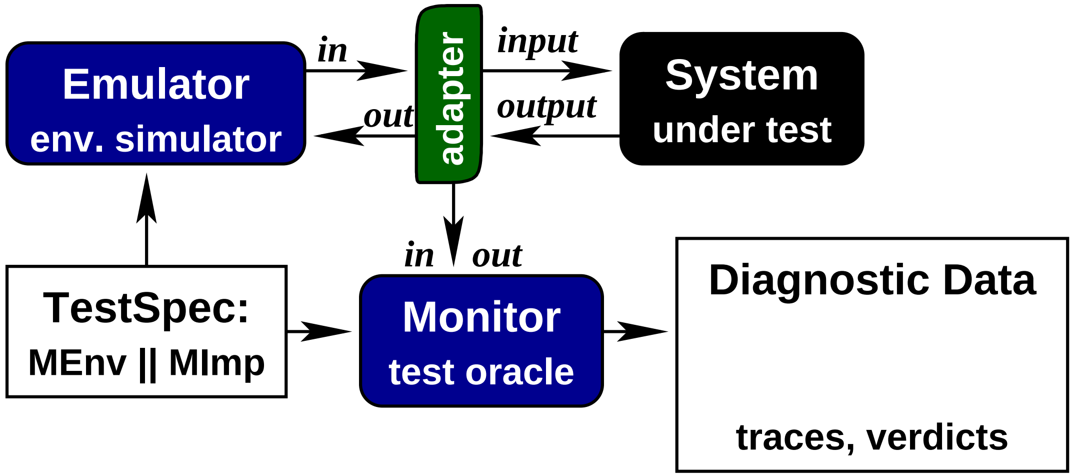
Conclusions

- Online real-time testing theoretically *sound* and *complete*.
- Relativized conformance allows to *minimize cost* of testing.
- Environment assumptions should be *known* and *explicit*.
- *Implemented* in T-UPPAAL using *efficient* algorithms from UPPAAL.
- Encouraging *error detection* capability and *performance*.
- T-UPPAAL allows *abstract* and *non-deterministic* specifications.
- *Extreme* non-determinism may degrade performance.

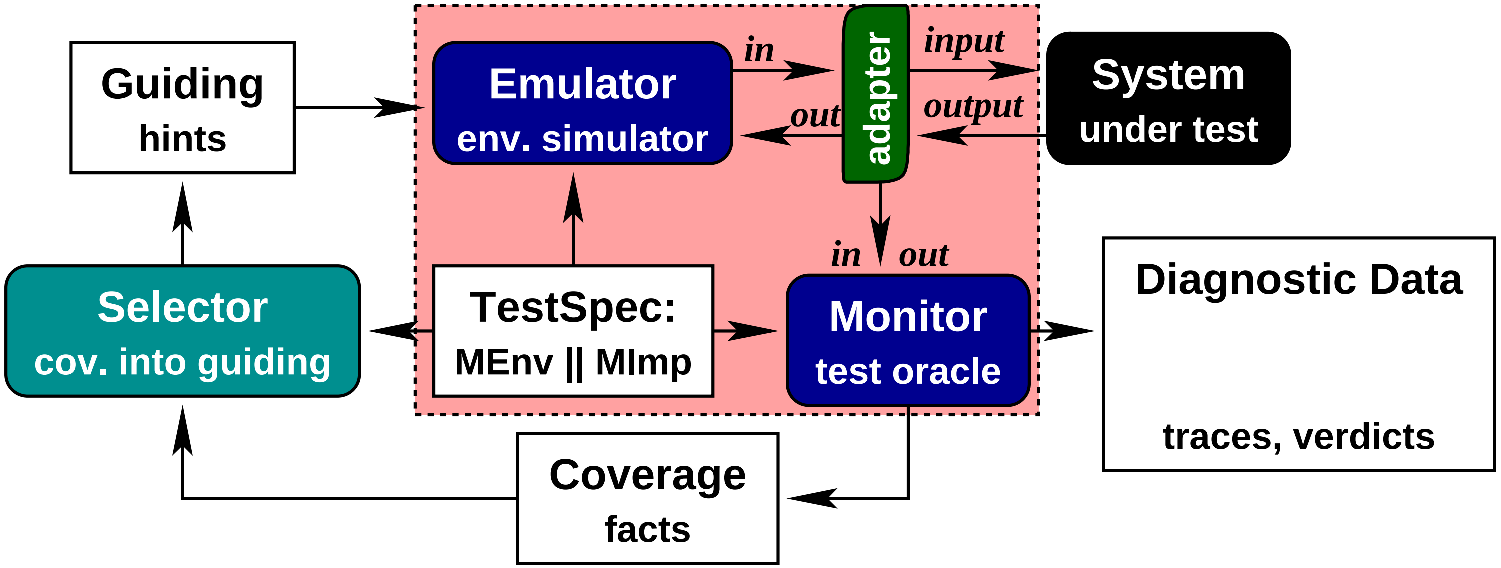
Summary and Future Work



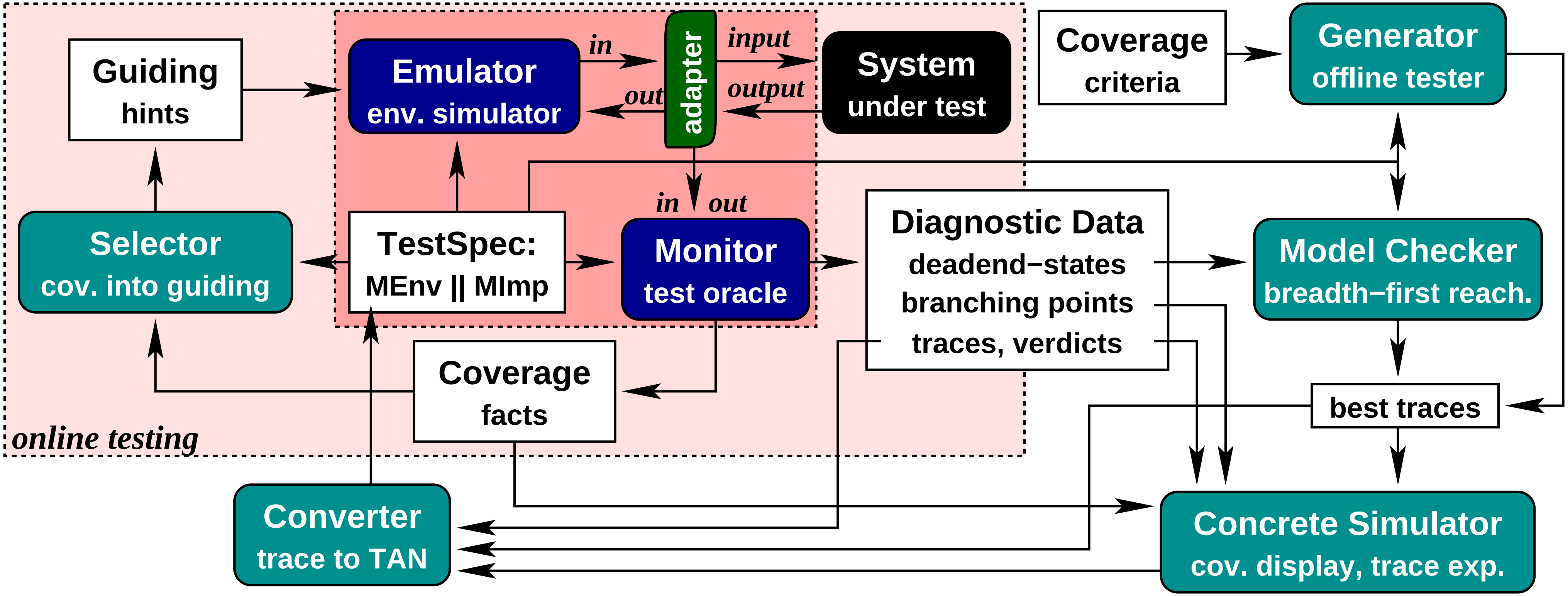
Summary and Future Work



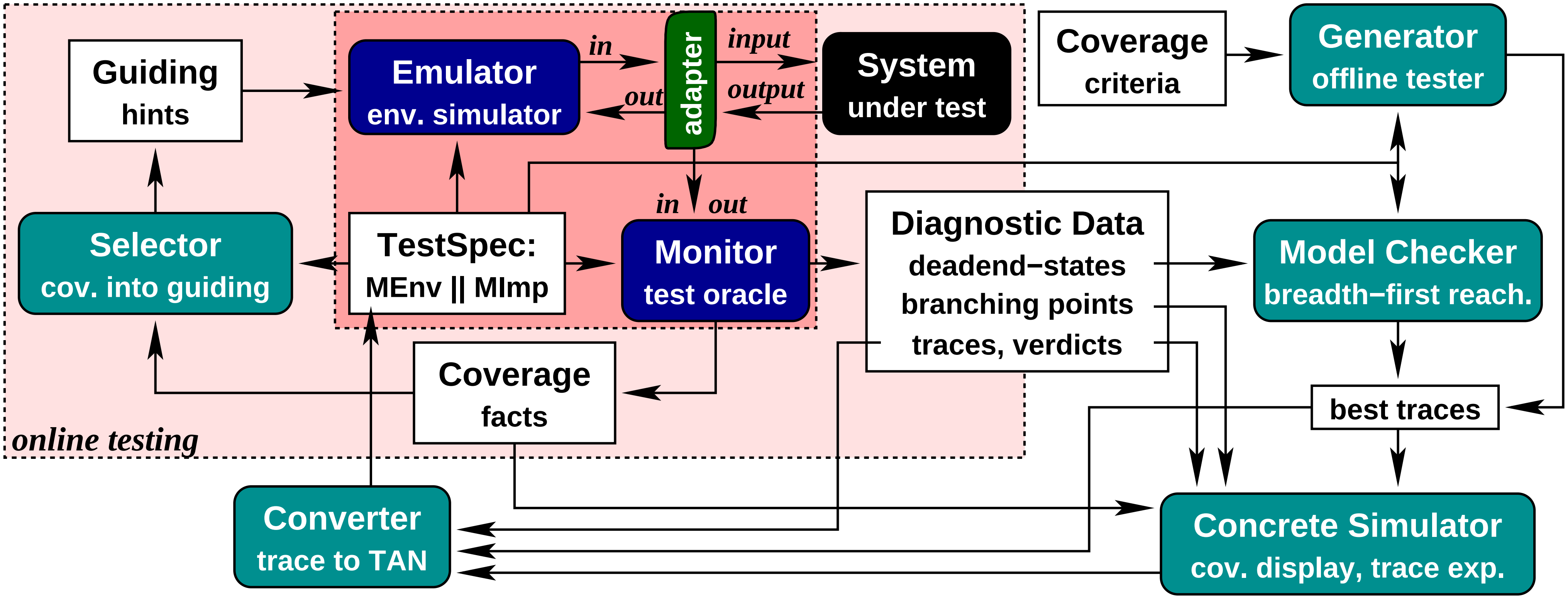
Summary and Future Work



Summary and Future Work

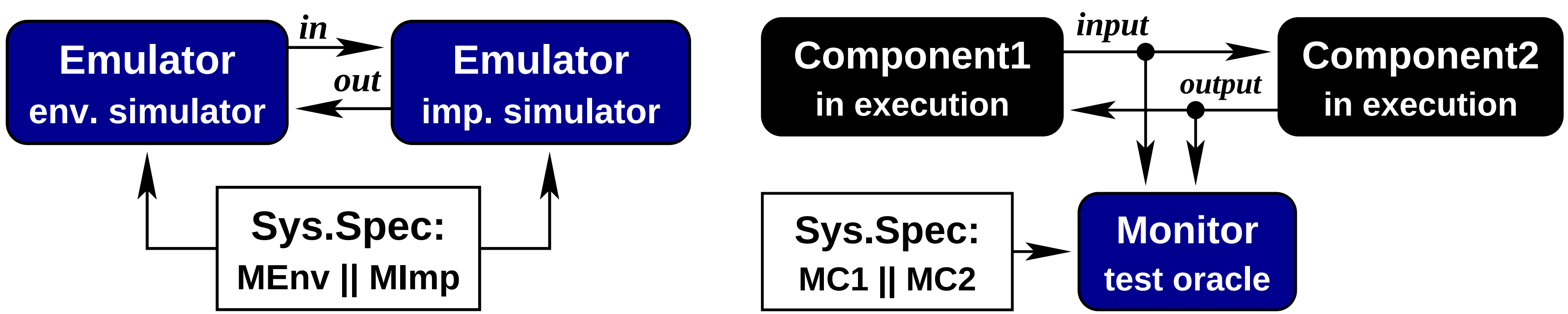


Summary and Future Work



Prototyping via simulation:

Execution monitoring:



Future Work

- Clock synchronization, event observation uncertainty.
- Value passing mechanisms in specification and adapters.
- New UPPAAL features (broadcast, committed, U-Code).
- Termination of testing (specify property expressions?).
- Relativized conformance in practice: specialized applications of generalized controllers, test-case guiding, debugging.
- T-UPPAAL in monitoring, testing via simulation and monitoring.
- Coverage estimation, use coverage in guiding.
- Relativized conformance in interface compatibility: unit testing.
- Diagnostic information display; model learning during experiment.
- Industrial case studies.

Danfoss Case Study: EKC – Refrigeration Controller