# Online Testing of Real-time Systems Using UPPAAL

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

# Outline

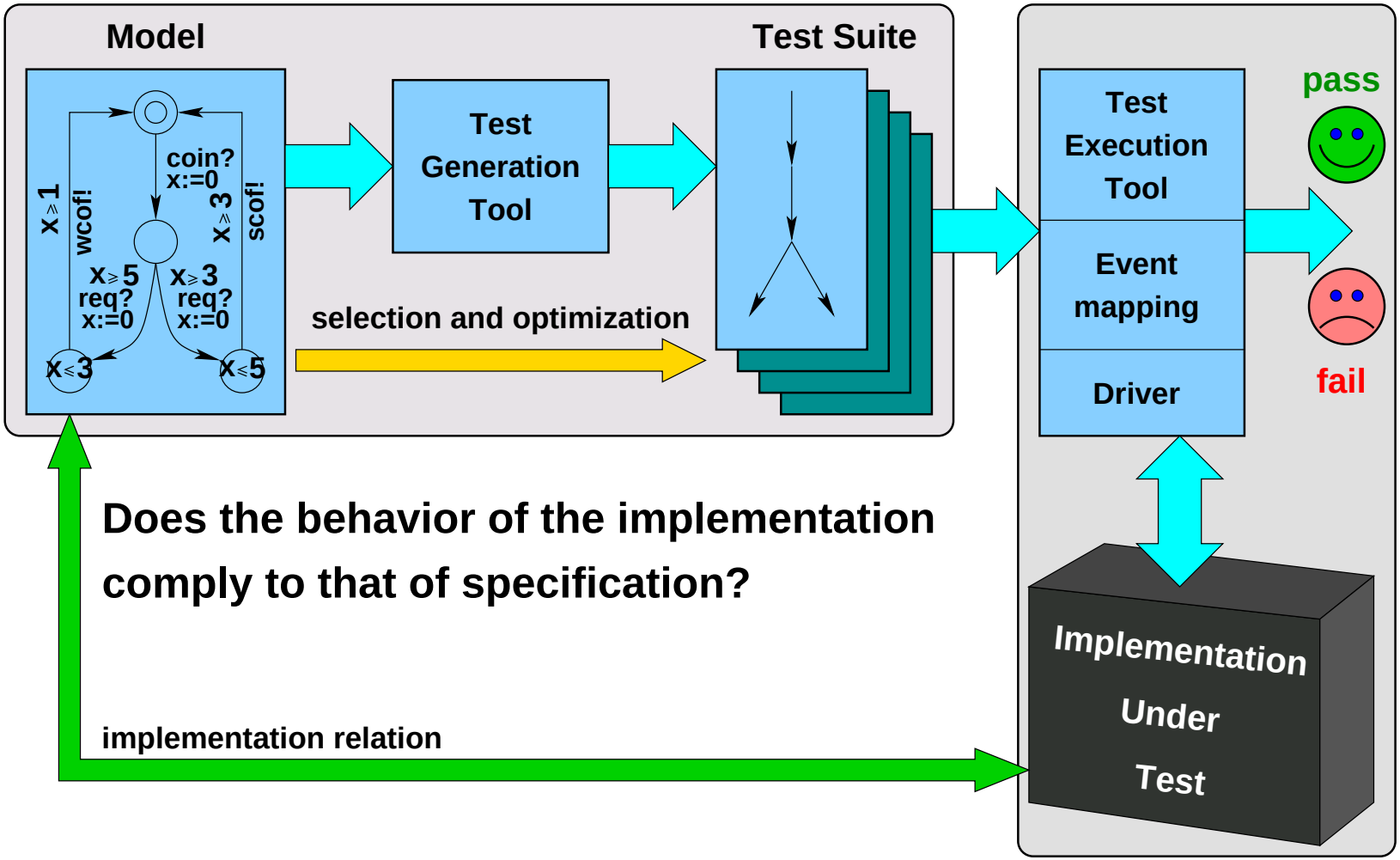
Online testing using UPPAAL: NWPT'03, FATES'04, EMSOFT'05.

- Formal framework of timed conformance testing of black-box:
  - Classical model-based black-box testing.
  - Test setup: from system and specification to testing.
  - Relativized timed input/output conformance relation.
  - Ordering of environments by discriminating power.
- Online Real-time Test Generation
  - Symbolic techniques from UPPAAL.
  - Online testing algorithm animated.
  - Real-time mapping to model and back.
- Evaluation: performance, industrial study, light controller demo.
- Conclusions and future work.

# Motivation for Automated Testing

- What is testing?
  - checking the *quality* (functionality, reliability, . . . ) of an *object*
  - by performing *experiments*
  - in a *controlled* (and systematic) way.
- Testing is the *main* validation *technique* used by industry:
  - 10-20 errors per 1000 lines of code.
  - 30-50% of development time and cost in embedded software.
  - Testing is still ad-hoc, based on heuristics, and error prone.
- "Testing is routine, tedious and boring work" – let *machines* do it.
- But! Testing requires most of *development skills*.
- Verification vs. testing: abstract *models* vs. *real* world.
- Conformance testing is *undecidable*.

# Classical Model-based Testing Framework




- *Model*-based, *black*-box, *conformance* testing.
- *Timed*, *online* (on-the-fly generation and execution in real-time).

# Related Work

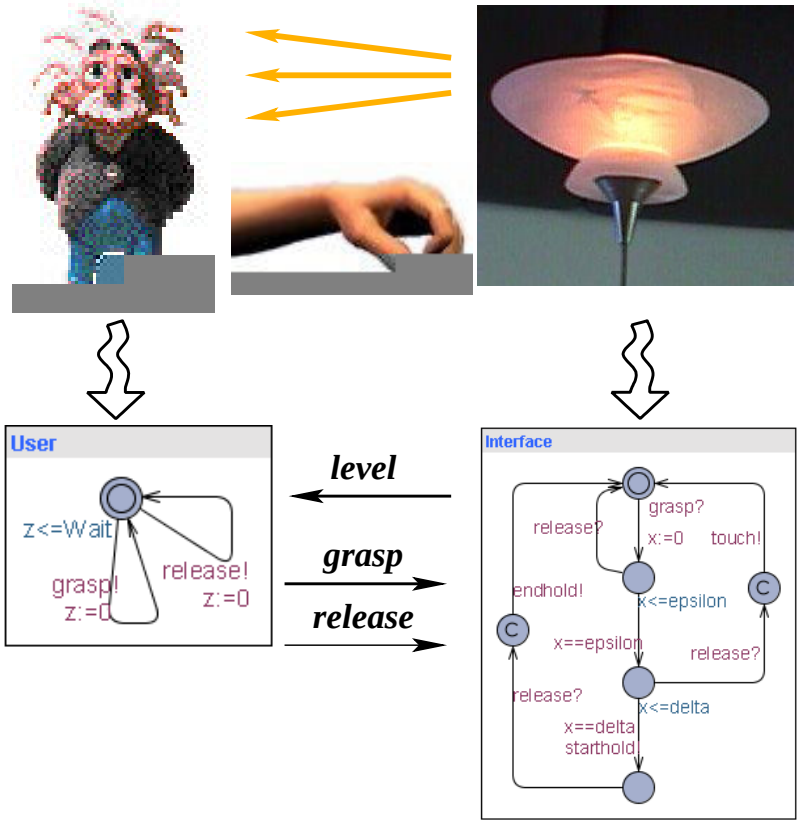
This work is based on the following ideas:

- UPPAAL model-checking algorithms for timed systems (1994).
- Jan Tretmans' testing theory (un-timed, quiescence) (1999).
- TORX testing tool framework (un-timed, w/o environment) (2000).
- Digitization techniques, T.A.Henzinger, Z.Manna, A.Pnueli (1992), J.Ouaknine, J.Worrell (2003).

Other close works:

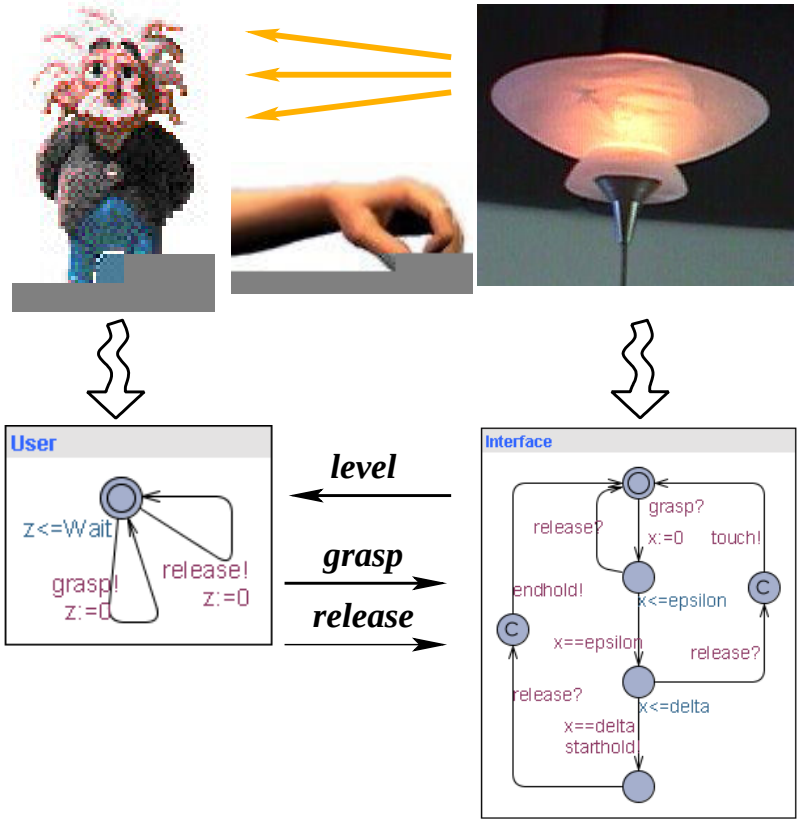
- Time-optimal test cases for RTS, A.Hessel, K.G.Larsen, B.Nielsen, P.Pettersson, A.Skou (2003).
- Black-box conformance testing for RTS, M.Krichen, S.Tripakis (2004).
- Test generation framework for quiescent RTS, L.B.Briones, E.Brinksma (2004).

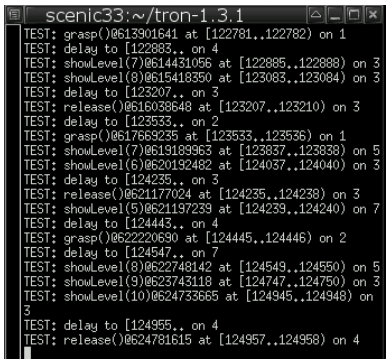
# Modelling and Testing with UPPAAL TRON

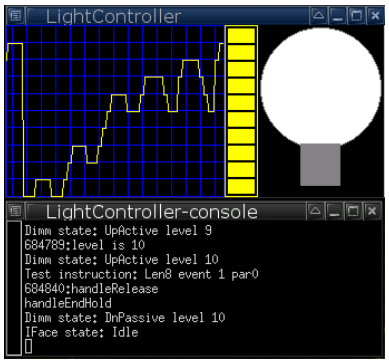



- Modelling a (closed) system:
  - Selected aspects.
  - Use abstraction.
  - Formal notations: UPPAAL TA.
  - Automatically analyze and test.

# Modelling and Testing with UPPAAL TRON

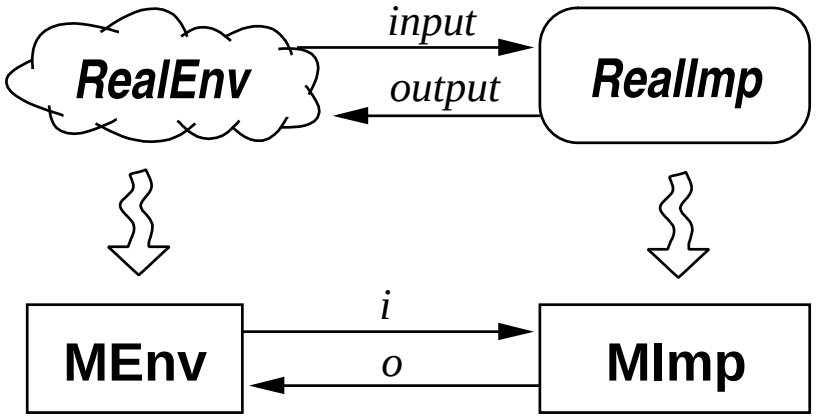


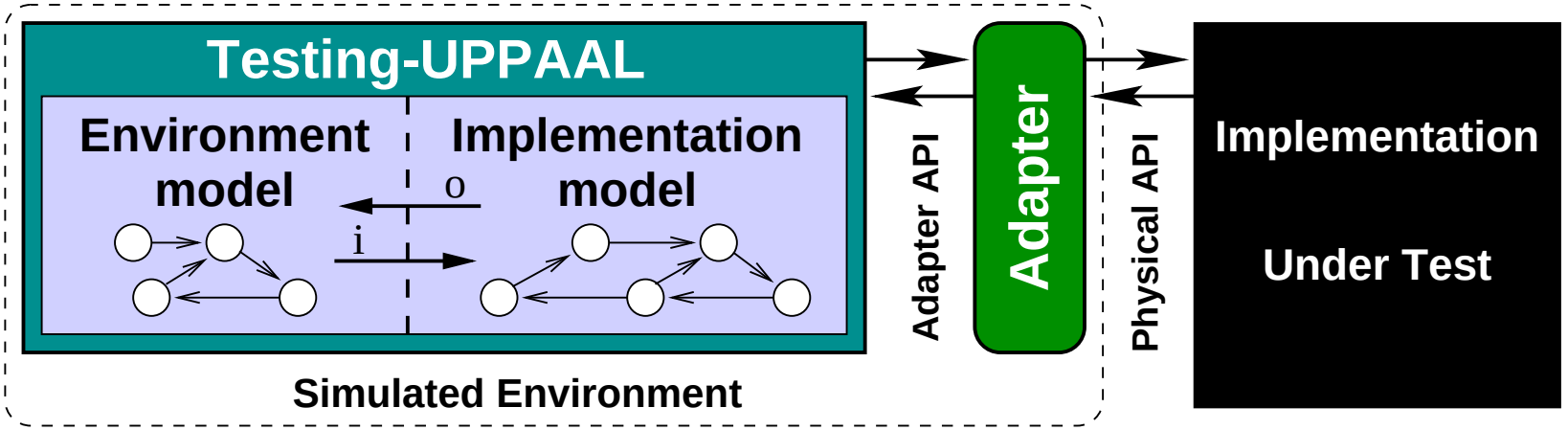

- Modelling a (closed) system:
  - Selected aspects.
  - Use abstraction.
  - Formal notations: UPPAAL TA.
  - Automatically analyze and test.
- Tester acts as environment where:
  - Specification: $Env \| IUT$,
  - $IUT$ model acts as oracle,
  - Load/guiding model is $Env$,
  - Generate only relevant inputs,
  - Modular and flexible.

# Test Setup: System ⇒ Model ⇒ 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).




- Online generation allows *long* and otherwise *exhaustive* tests.

# 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 $s$: $\{\sigma \in (A \cup \mathbb{R}_{\geq 0})^* \mid s \stackrel{\sigma}{\Rightarrow}\}$
- Timed trace *inclusion* as conf. relation: $\mathsf{TTr}(i) \subseteq \mathsf{TTr}(s)$
- *No illegal* output and *legal* output is observed at *right time*.

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

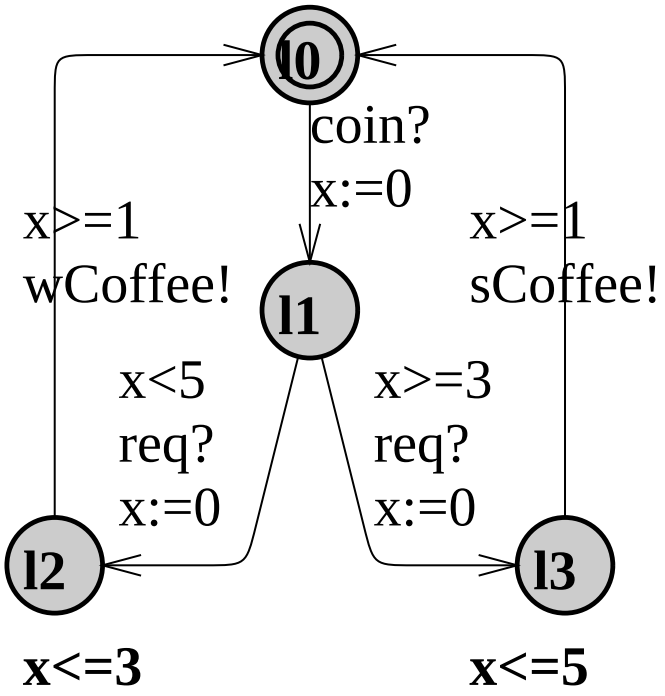
  $s\ \mathsf{rtioco}_e\ t \stackrel{def}{=} \forall \sigma \in \mathsf{TTr}(e).\mathsf{Out}\big((e,s)\ \mathsf{After}\ \sigma\big) \subseteq \mathsf{Out}\big((e,t)\ \mathsf{After}\ \sigma\big)$

$$\boxed{s\ \mathsf{rtioco}_e\ t \iff \mathsf{TTr}(s) \cap \mathsf{TTr}(e) \subseteq \mathsf{TTr}(t) \cap \mathsf{TTr}(e)}$$

- Environment *ordering*. $f$ is more discriminating than $e$:

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

# Test Specification: Timed Automata Networks



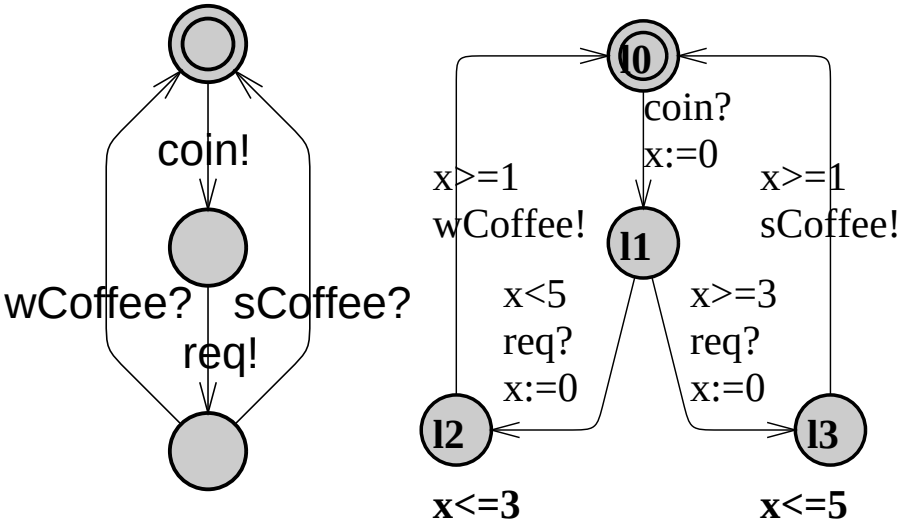

*Timed automaton* over A is $\langle L, l_0, X, D, E, I \rangle$:

- $L$ – 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 $\langle l, g, a, r, l' \rangle \in E$.

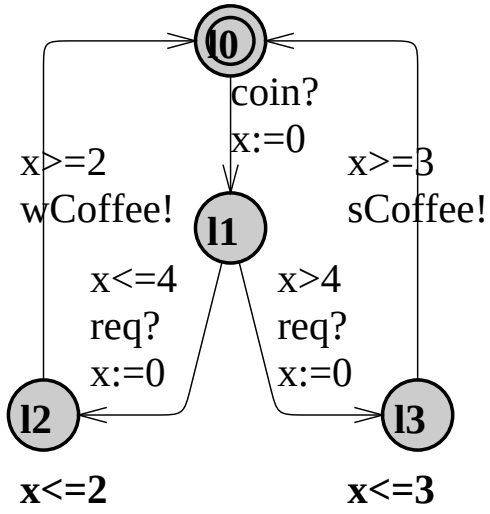
- Has *Labeled Transition System (LTS)* semantics.
- I/O, internal and timing *non-determinism* allow modelling parallelism, abstraction and possible time slacks.
- Test Spec: $\langle\, (\mathcal{E}_1 || \mathcal{E}_2 || \ldots || \mathcal{E}_n) \;||\; (\mathcal{I}_1 || \mathcal{I}_2 || \ldots || \mathcal{I}_n),\;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

# Timed I/O Conformance Relation Example

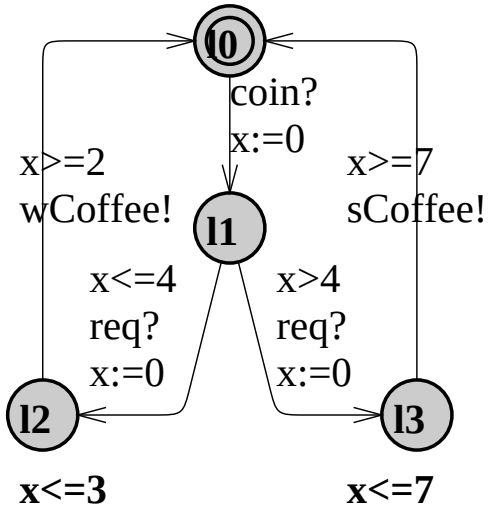
Specification $s$




Implementation $i_1$

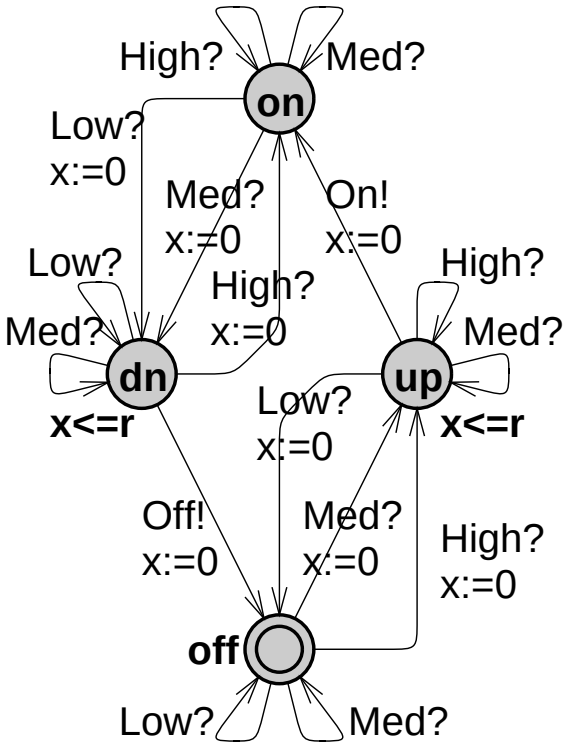



Implementation $i_2$

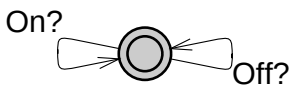



| Trace, $\sigma$ | Out$(s$ After $\sigma)$ | Out$(i_1$ After $\sigma)$ | Out$(i_2$ After $\sigma)$ |
|---|---|---|---|
| $c \cdot 2$ | $\mathbb{R}_{\geq 0}$ | $\mathbb{R}_{\geq 0}$ | $\mathbb{R}_{\geq 0}$ |
| $c \cdot 4 \cdot r \cdot 1$ | $\{wCoffee, sCoffee\} \cup [0,4]$ | $[0,1]$ | $[0,2]$ |
| $c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

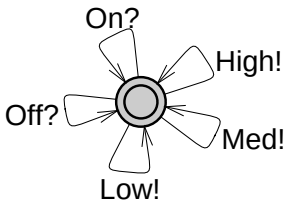
# Discriminating Power of Environments



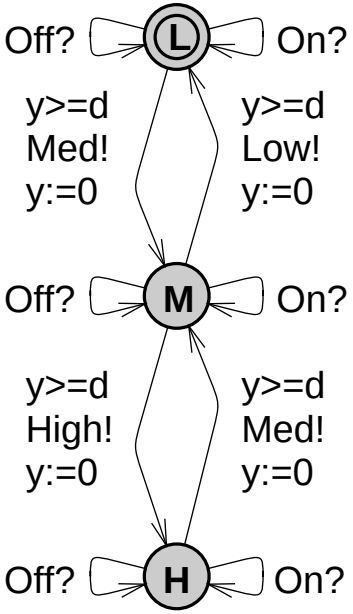

(a) Cooling controller.



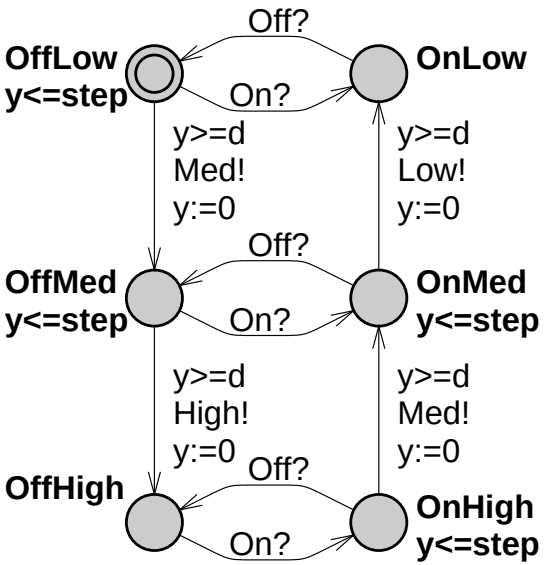

(b) Least




(c) Most




(d) Inertial



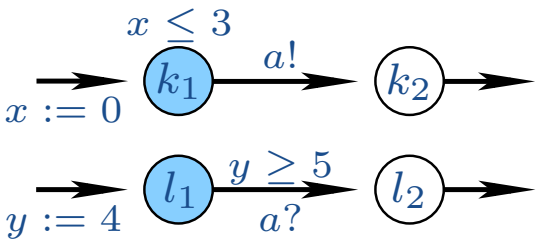

(e) Responsive

Most $\sqsupseteq$ Inertial $\sqsupseteq$ Responsive $\sqsupseteq$ Least

Consider trace: $0 \cdot \mathbf{Med}! \cdot 0 \cdot \mathbf{High}! \cdot 0 \cdot \mathbf{Med}! \cdot 0 \cdot \mathbf{Low}? \cdot r \cdot \ldots$

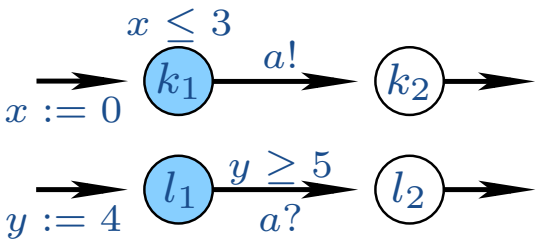
# Symbolic Techniques from UPPAAL

- *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 $z \wedge g \neq \emptyset$,$(z \wedge g)_r \wedge I(\bar{l'}) \neq \emptyset$.



# Symbolic Techniques from UPPAAL

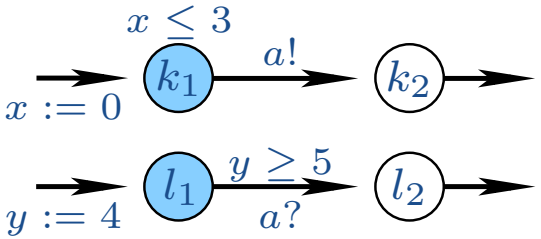
- *Action* transition: $\langle \bar{l}, z \rangle \xrightarrow{a} \langle \bar{l'}, (z \wedge g)_r \wedge I(\bar{l'}) \rangle$ **iff**:
  $l \xrightarrow{g,a,r} l'$ is $a$-action transition and $z \wedge g \neq \emptyset$,$(z \wedge g)_r \wedge I(\bar{l'}) \neq \emptyset$.
- *Delay* transition: $\langle \bar{l}, z \rangle \xrightarrow{\delta} \langle \bar{l}, z^{+\delta} \wedge I(\bar{l}) \rangle$ **iff** $z^{+\delta} \wedge I(\bar{l}) \neq \emptyset$.



# Symbolic Techniques from UPPAAL

- *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** $z^{+\delta} \wedge I(\bar{l}) \neq \emptyset$.
- *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 $\prec \in \{<, \leq\}$

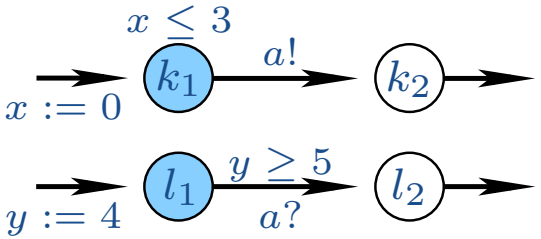
$$z = [(y - x \leq 4) \wedge (y \geq 5) \wedge (x \leq 3)]$$



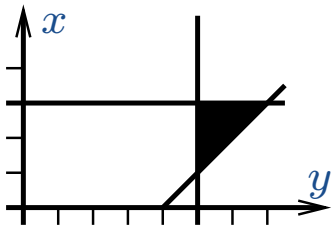
# Symbolic Techniques from UPPAAL

- *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** $z^{+\delta} \wedge I(\bar{l}) \neq \emptyset$.
- *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.

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



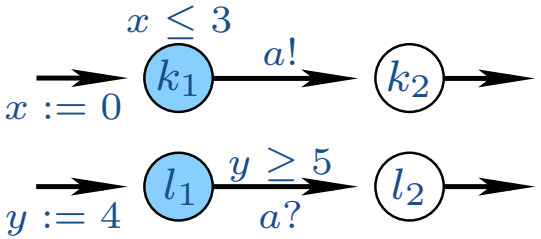

| | 0 | $y$ | $x$ |
|---|---|---|---|
| 0 | – | $-5$ | 0 |
| $y$ | $\infty$ | – | 4 |
| $x$ | 3 | $\infty$ | – |



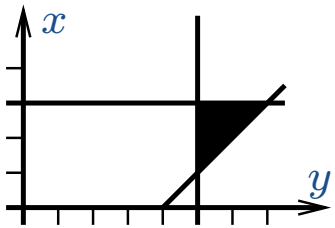
# Symbolic Techniques from UPPAAL

- *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** $z^{+\delta} \wedge I(\bar{l}) \neq \emptyset$.
- *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 |
| $y$ | $\infty$ | – | 4 |
| $x$ | 3 | $\infty$ | – |



# 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 $\delta$ time units and wake up on output $o$
   **if** $o$ occurs at $\delta' \leq \delta$ **then**
      $\mathcal{Z} := \mathcal{Z}$ After $\delta'$
      **if** $o \notin \mathsf{ImpOutput}(\mathcal{Z})$ **then return fail**
      **else** $\mathcal{Z} := \mathcal{Z}$ After $o$
   **else** $\mathcal{Z} := \mathcal{Z}$ After $\delta$ // no output within $\delta$ 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

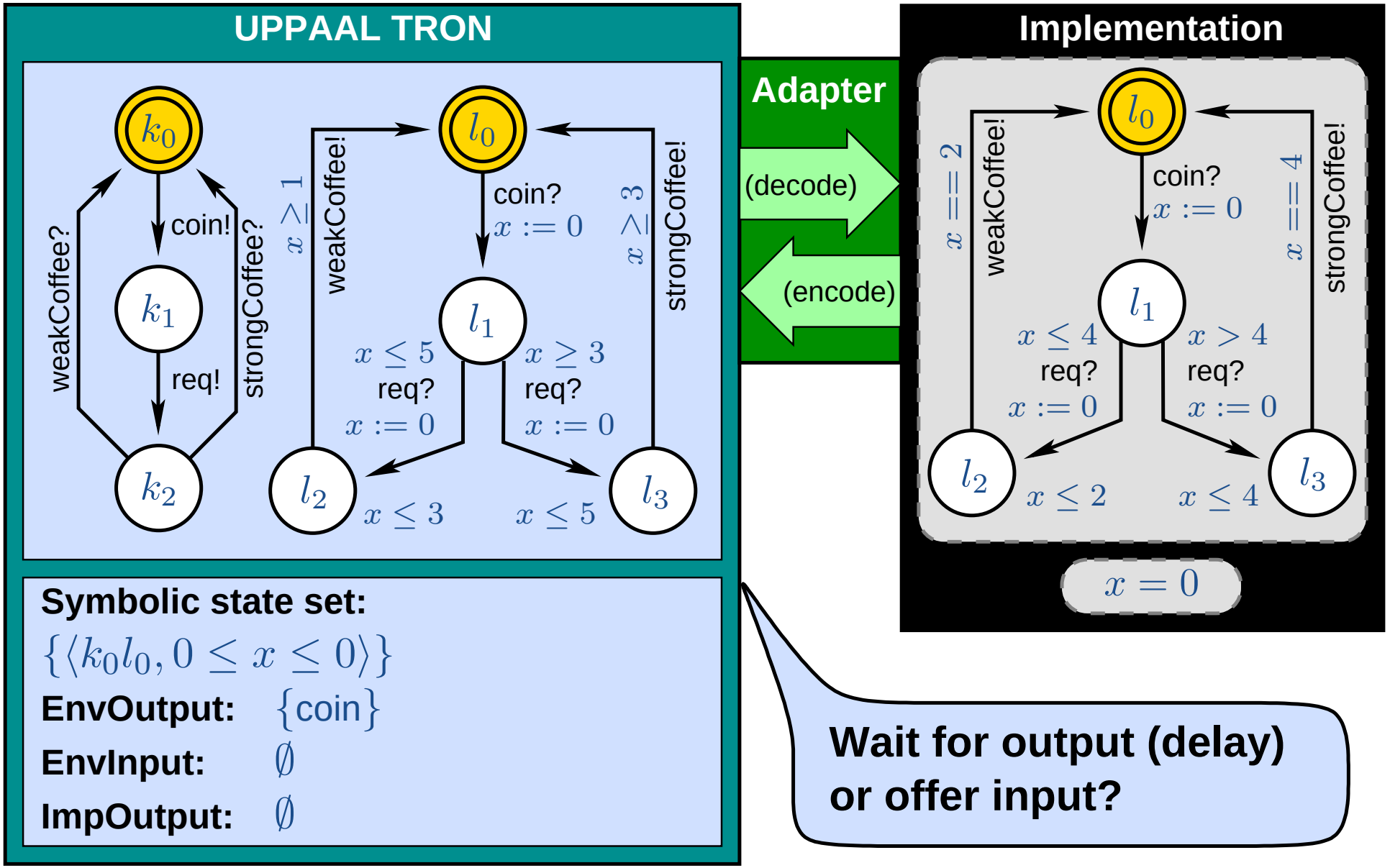
**while** $\mathcal{Z} \neq \emptyset \wedge \sharp iterations \leq T$ **do** choose randomly:

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 $\delta$ time units and wake up on output $o$
   **if** $o$ occurs at $\delta' \leq \delta$ **then**
      $\mathcal{Z} := \mathcal{Z}$ After $\delta'$
      **if** $o \notin \mathsf{ImpOutput}(\mathcal{Z})$ **then return fail**
      **else** $\mathcal{Z} := \mathcal{Z}$ After $o$
   **else** $\mathcal{Z} := \mathcal{Z}$ After $\delta$ // no output within $\delta$ delay
3. $\mathcal{Z} := \{(s_0, e_0)\}$, **reset** IUT //reset and restart

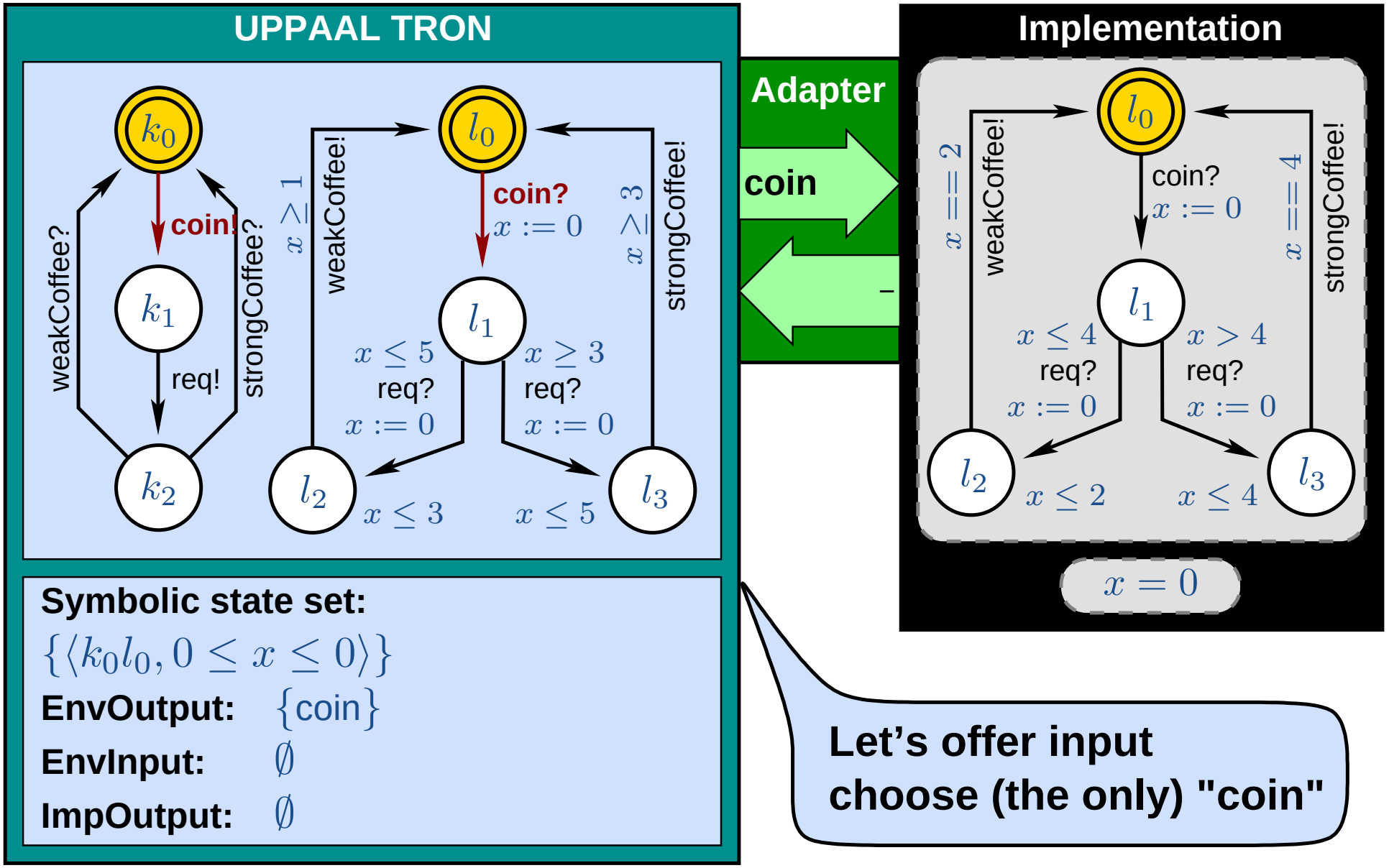
**if** $\mathcal{Z} = \emptyset$ **then return fail else return pass**

**sound and complete in limit**

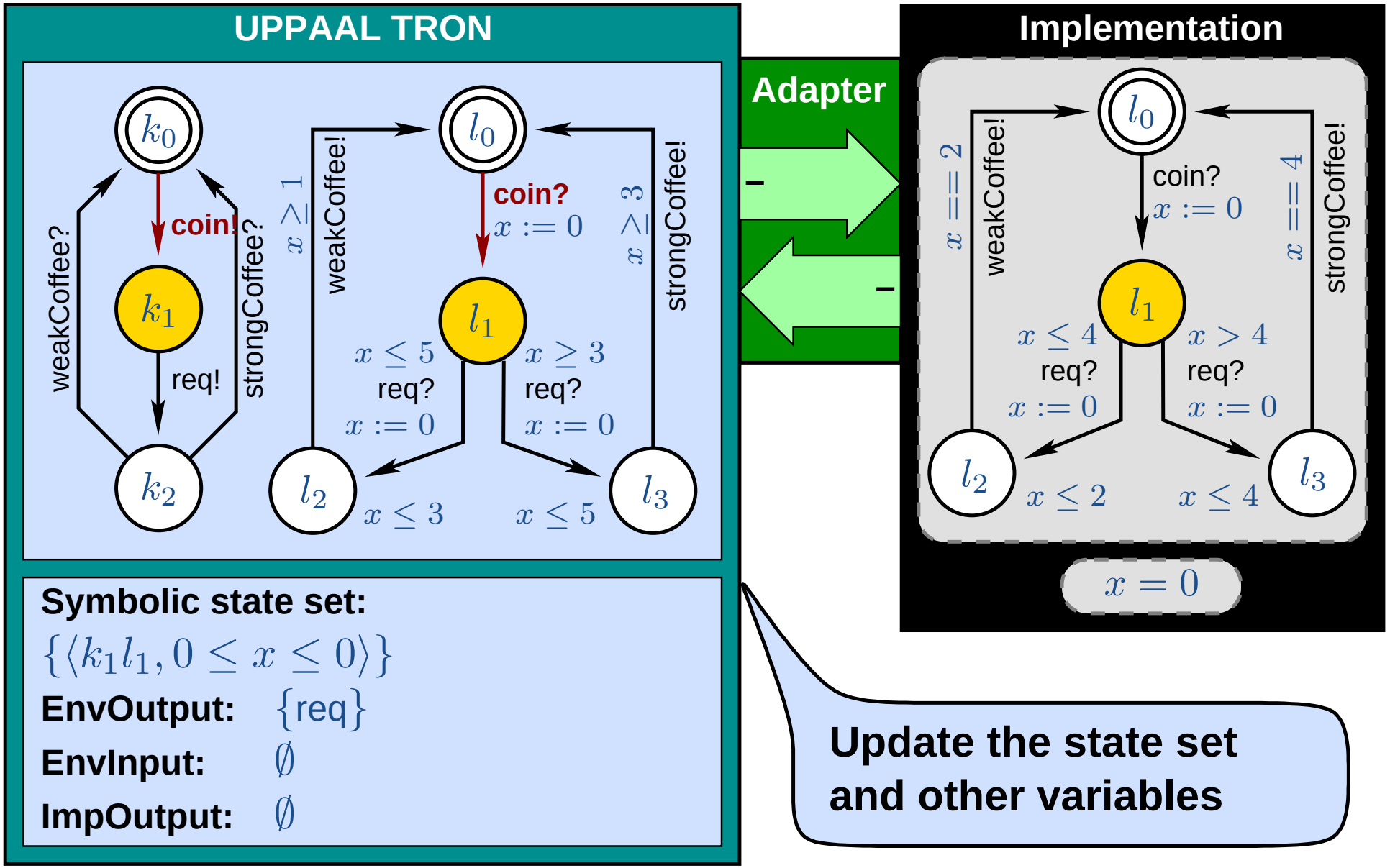
# Testing Online in Action



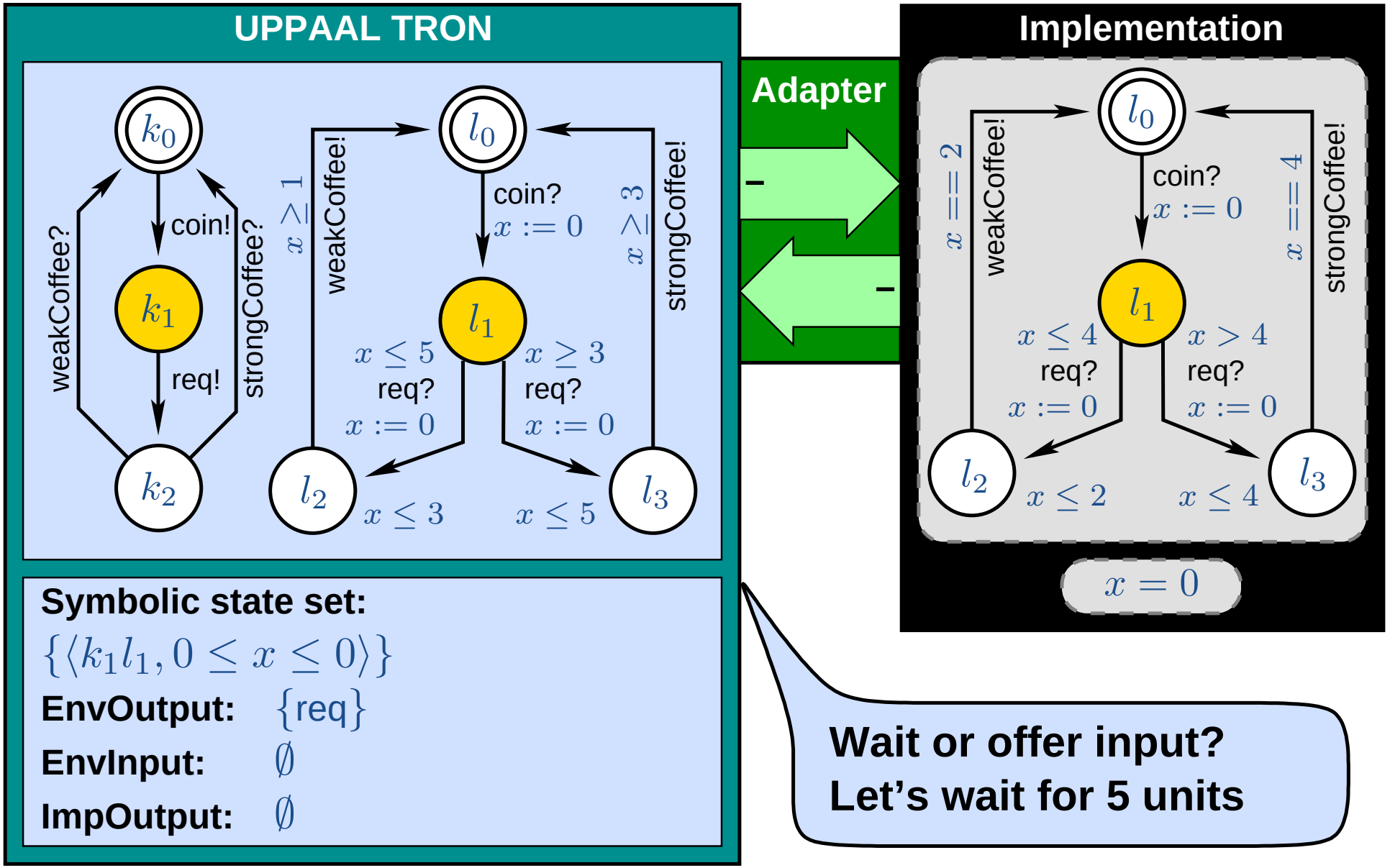
# Testing Online in Action



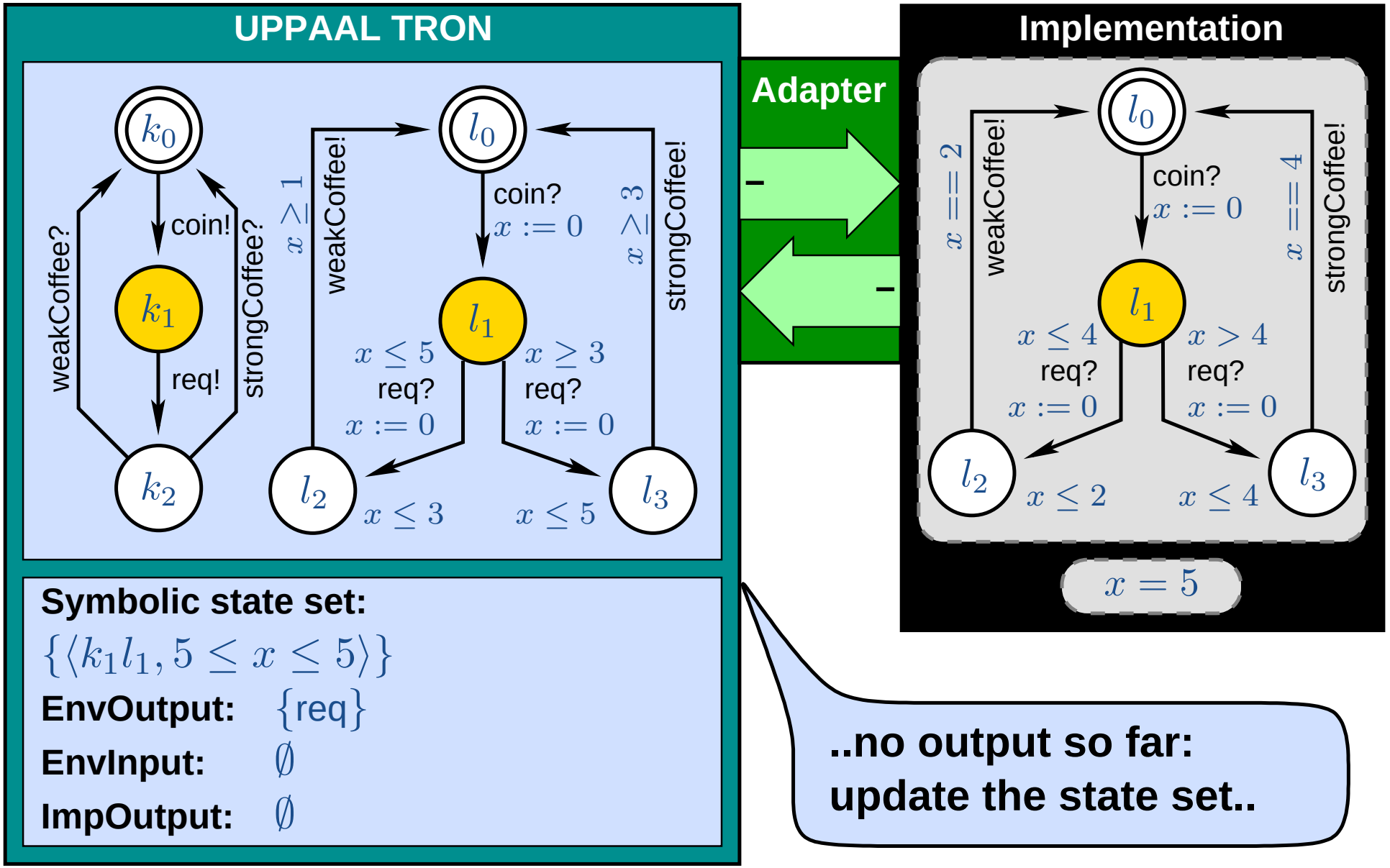
# Testing Online in Action



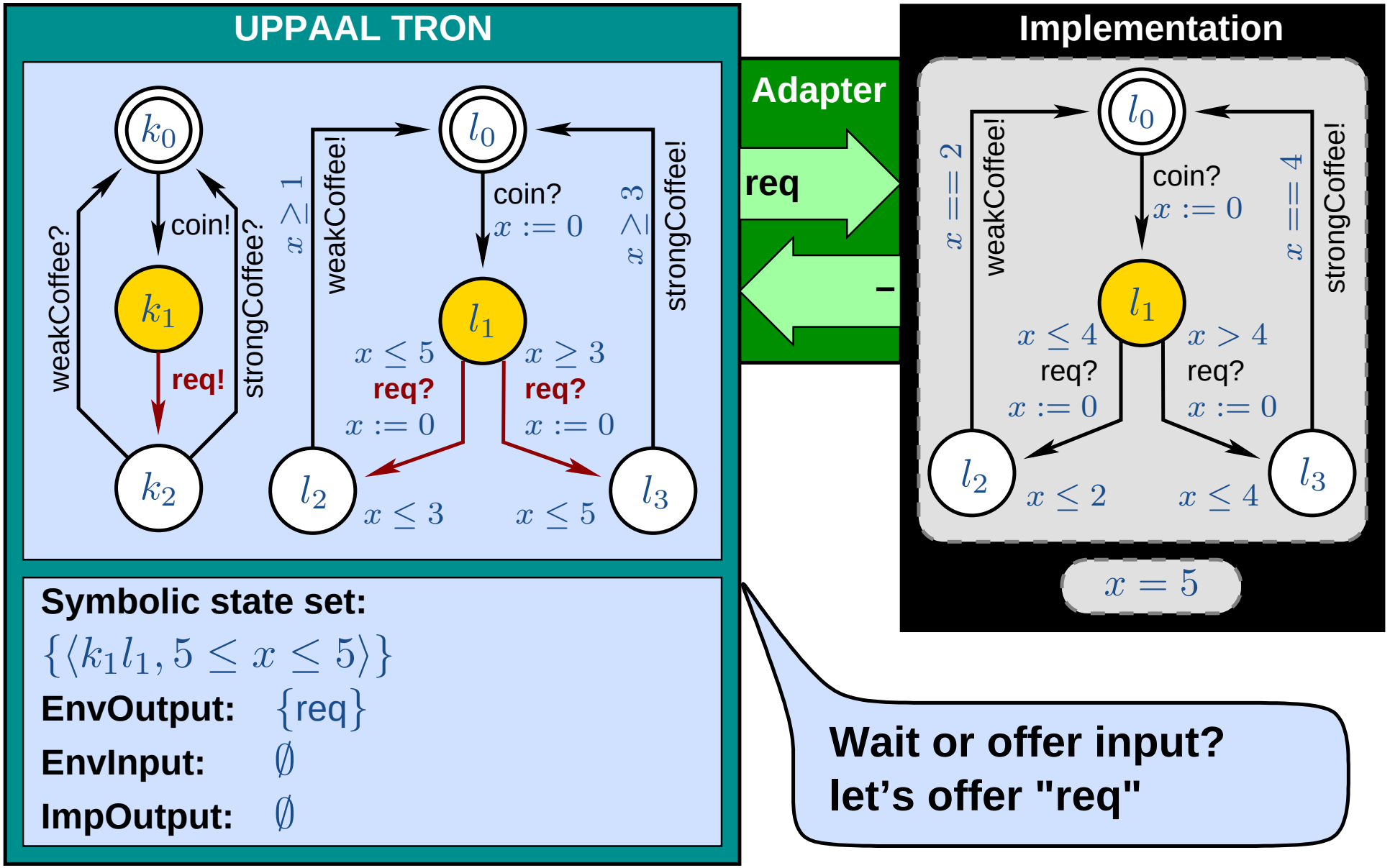
# Testing Online in Action



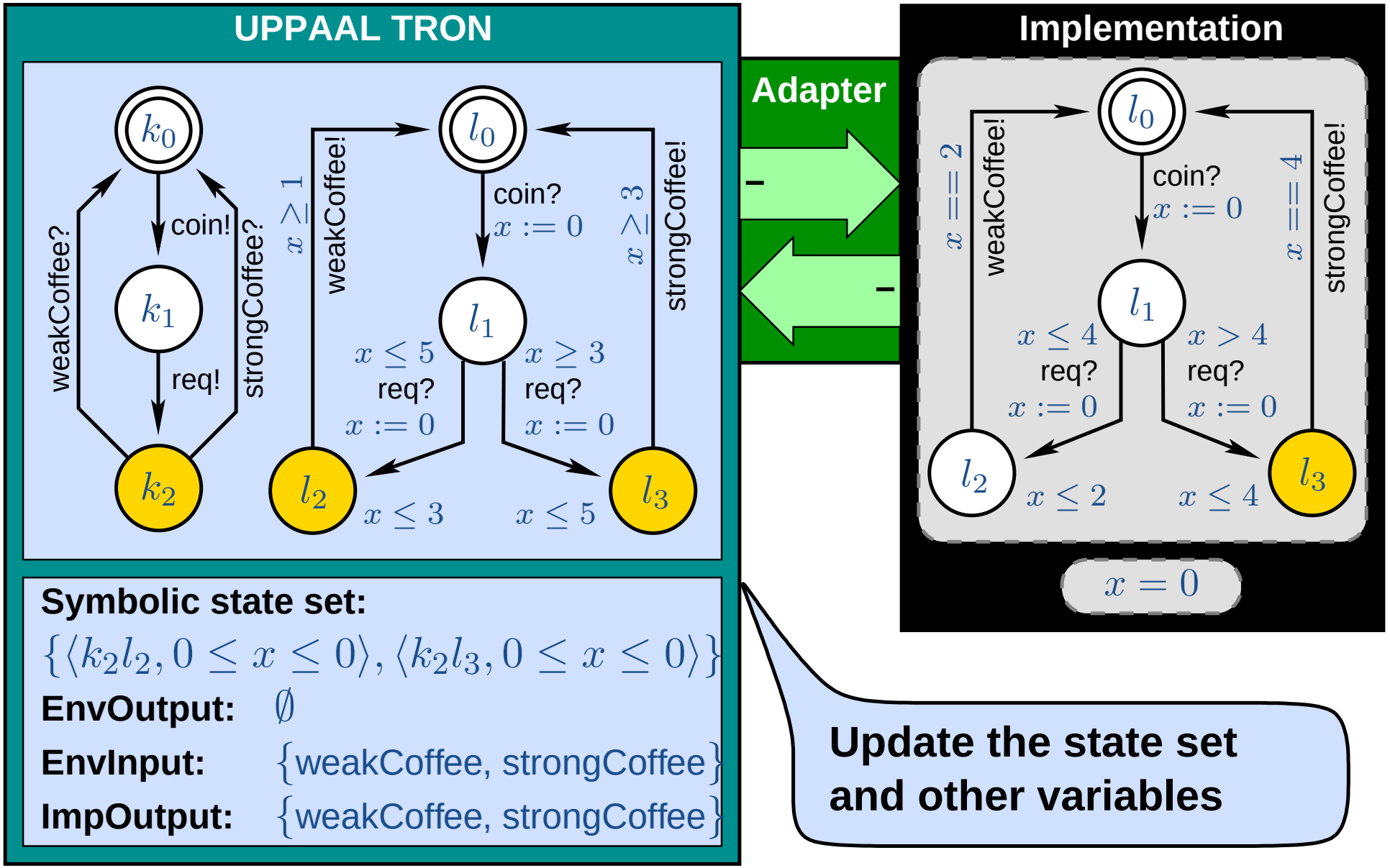
# Testing Online in Action



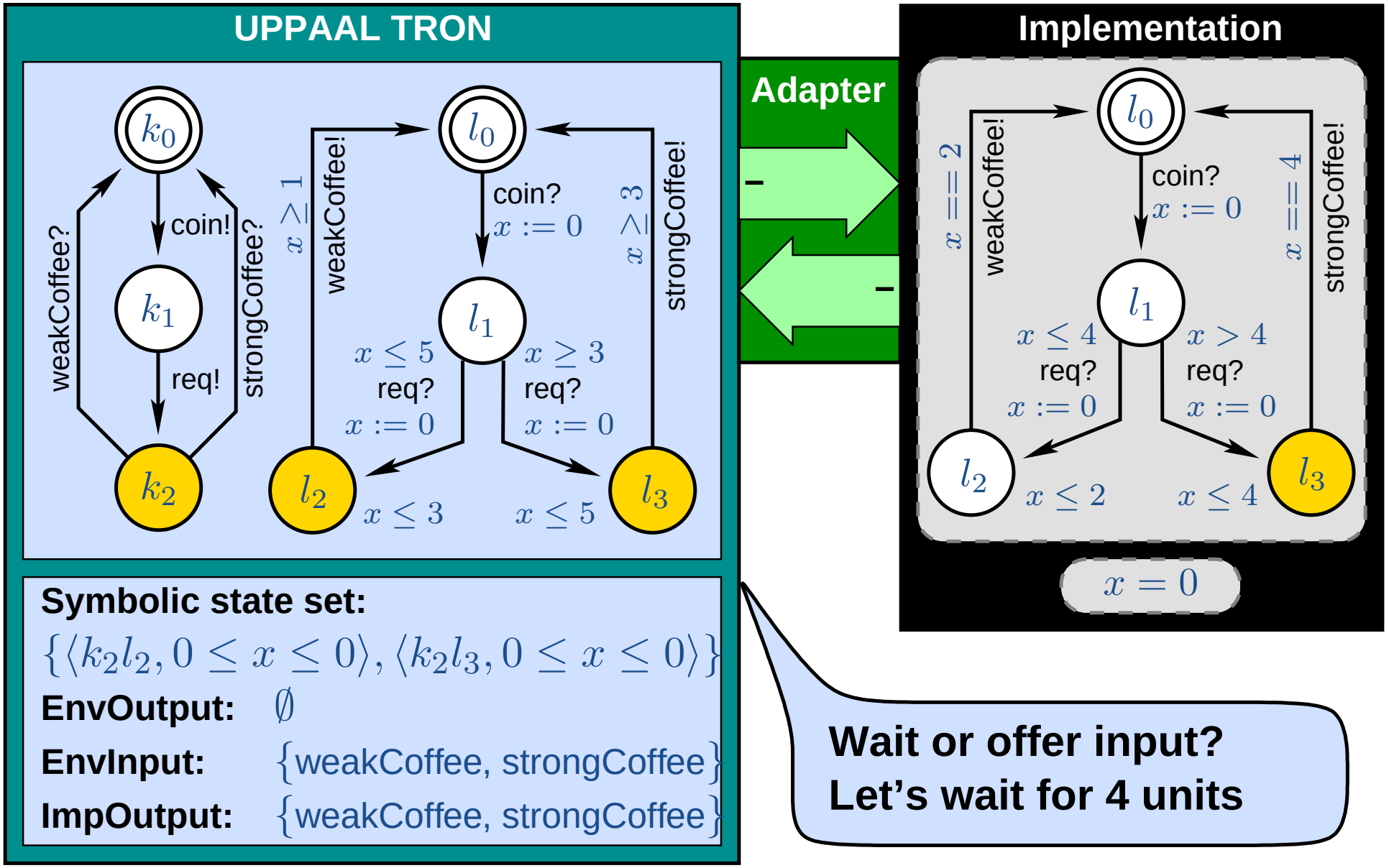
# Testing Online in Action



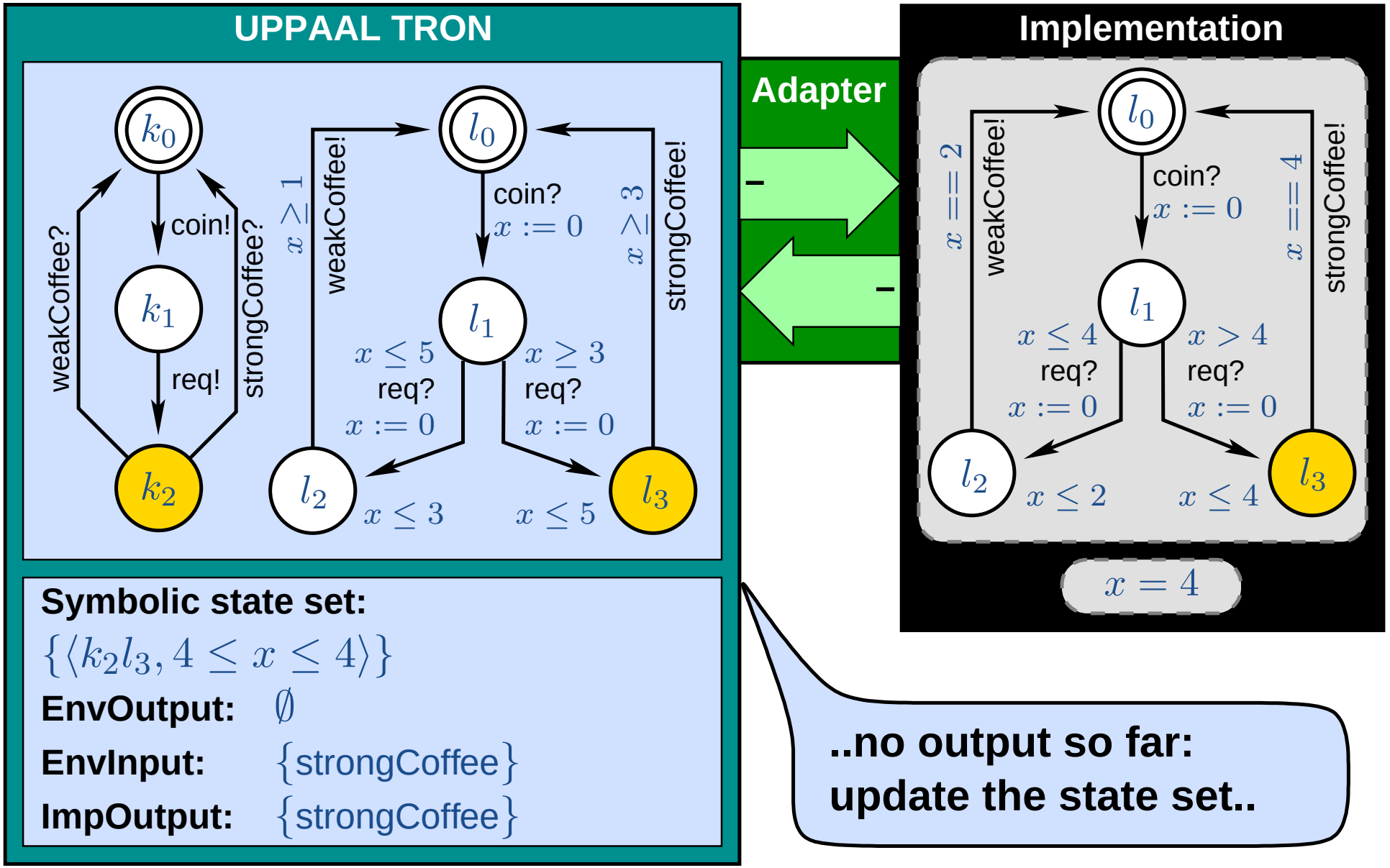
# Testing Online in Action



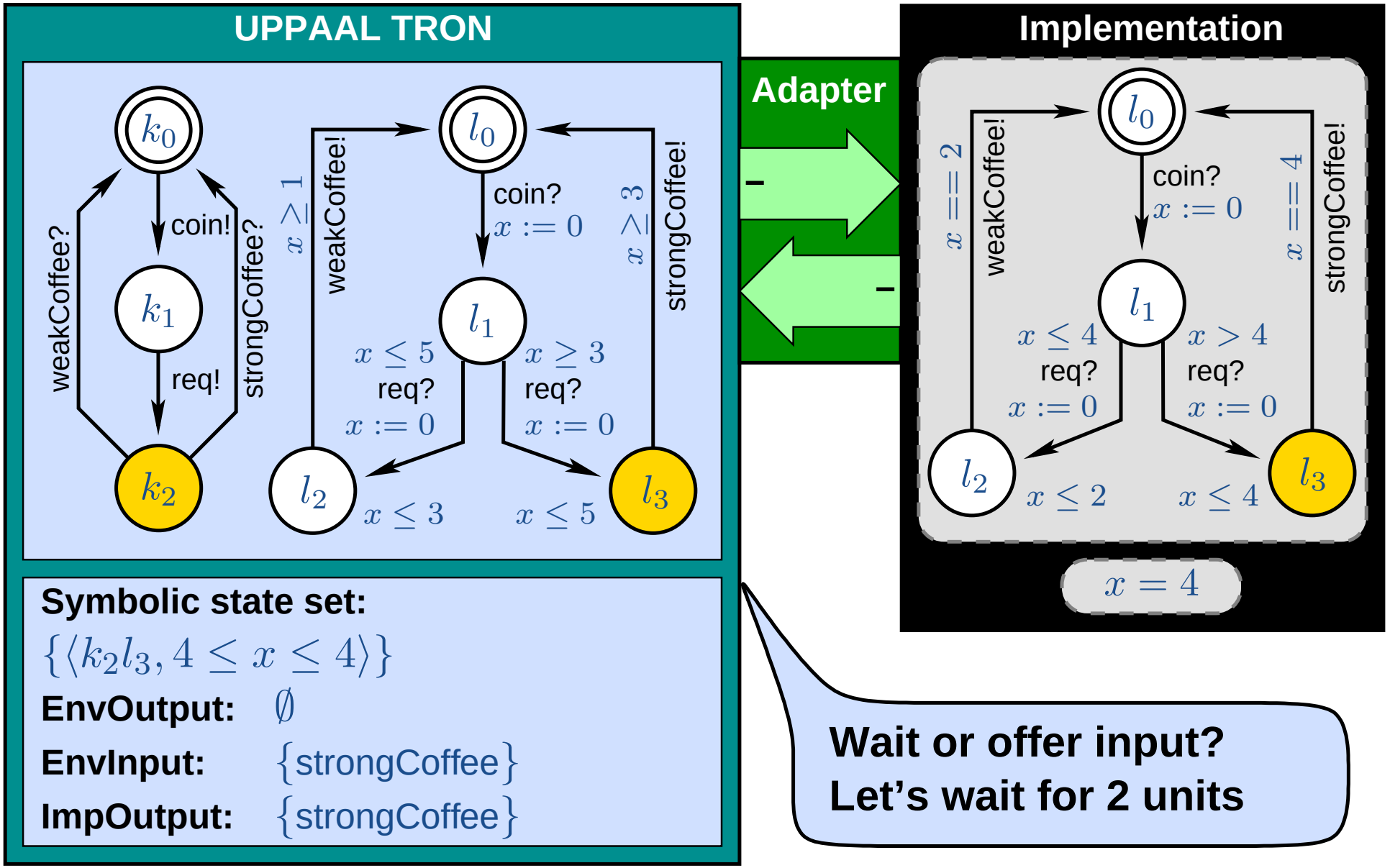
# Testing Online in Action



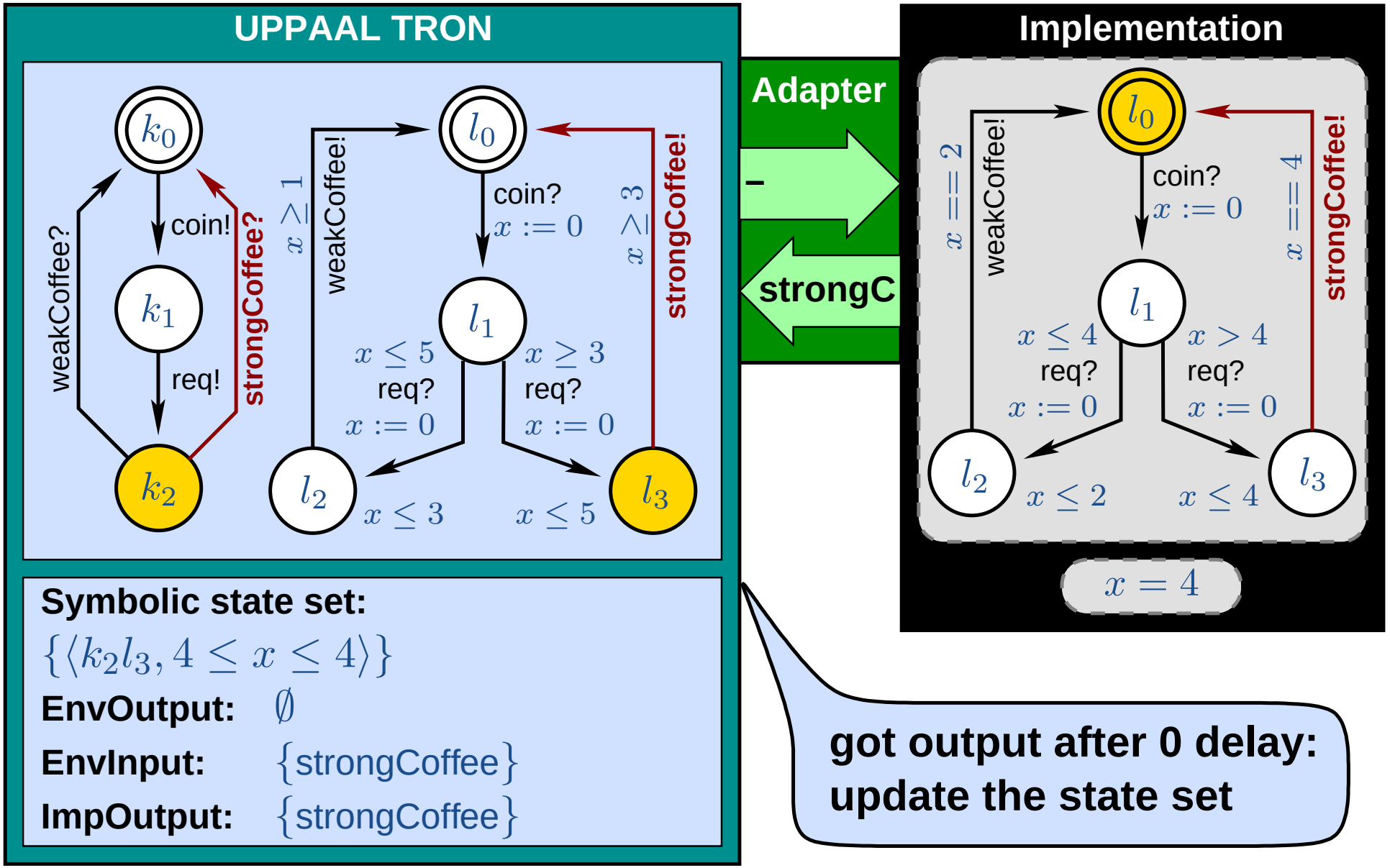
# Testing Online in Action



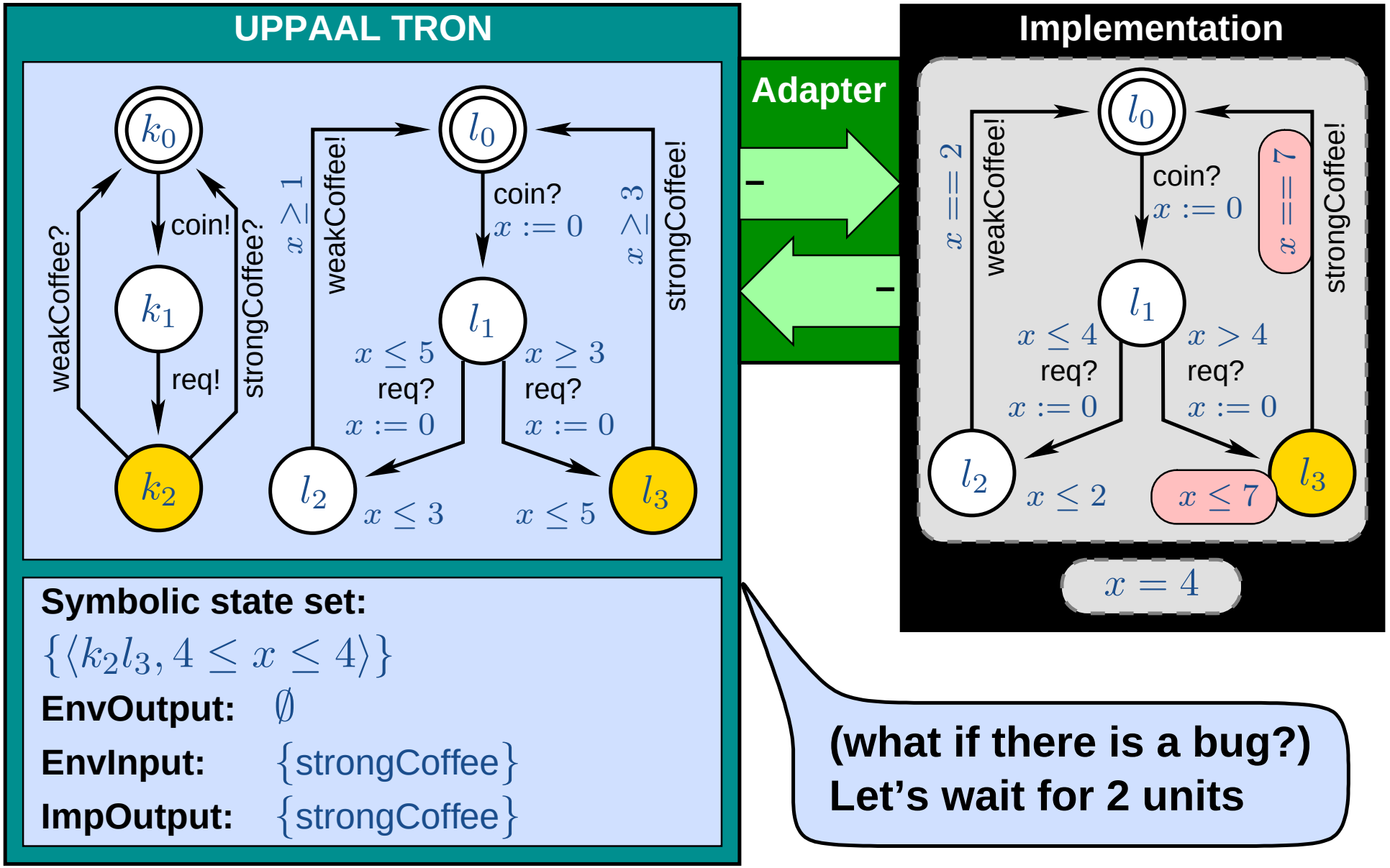
# Testing Online in Action



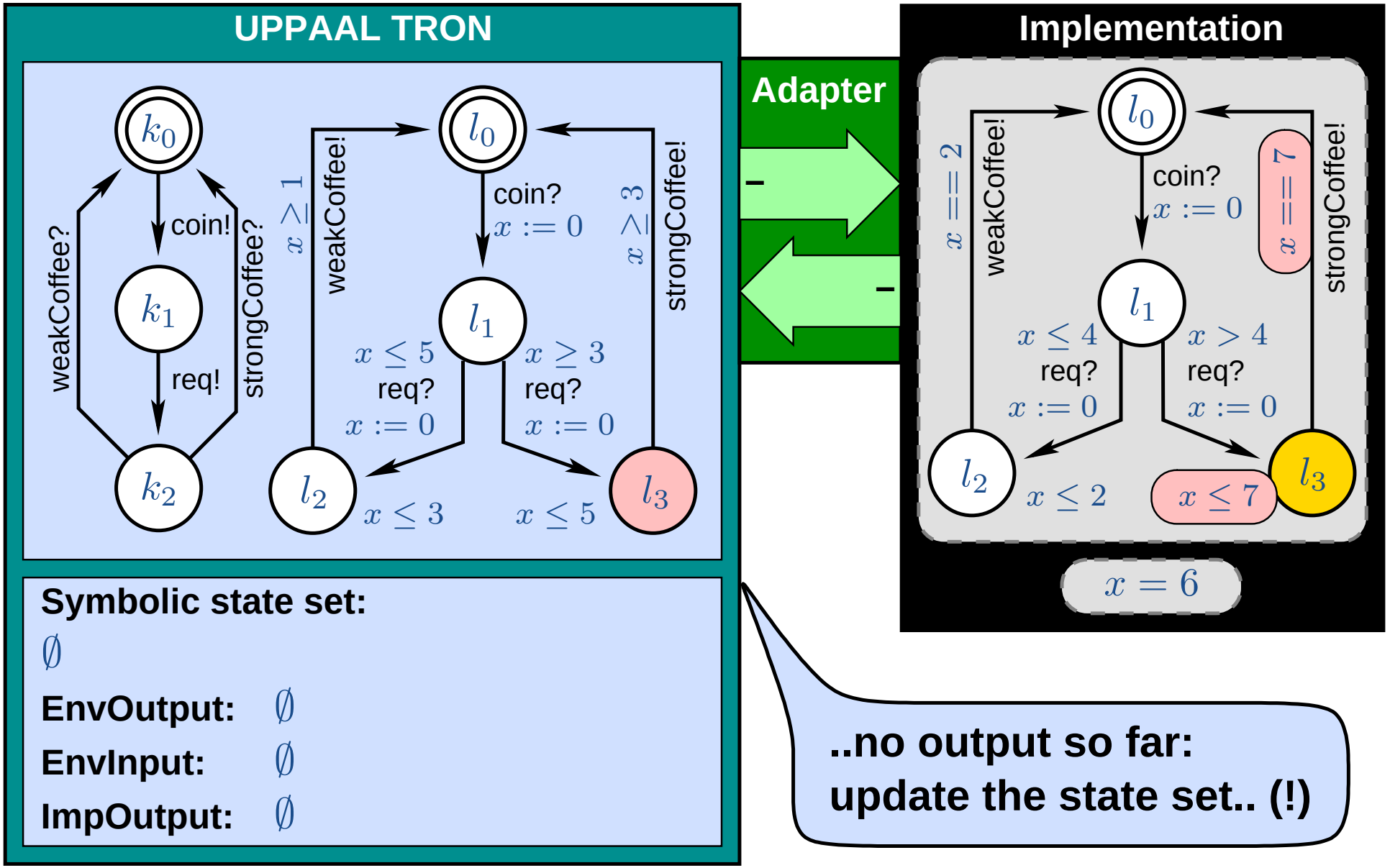
# Testing Online in Action



# Testing Online in Action



# Testing Online in Action



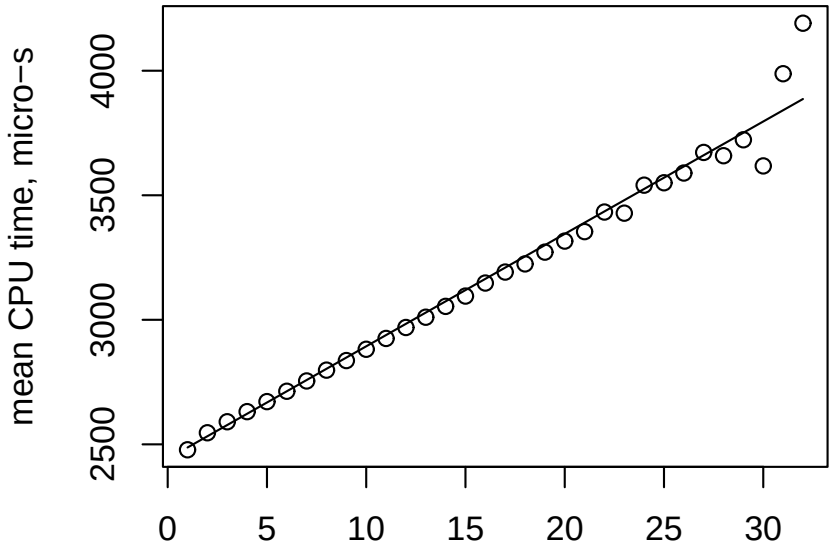
# Error Detection Capability: Mutant Experiment

- 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.

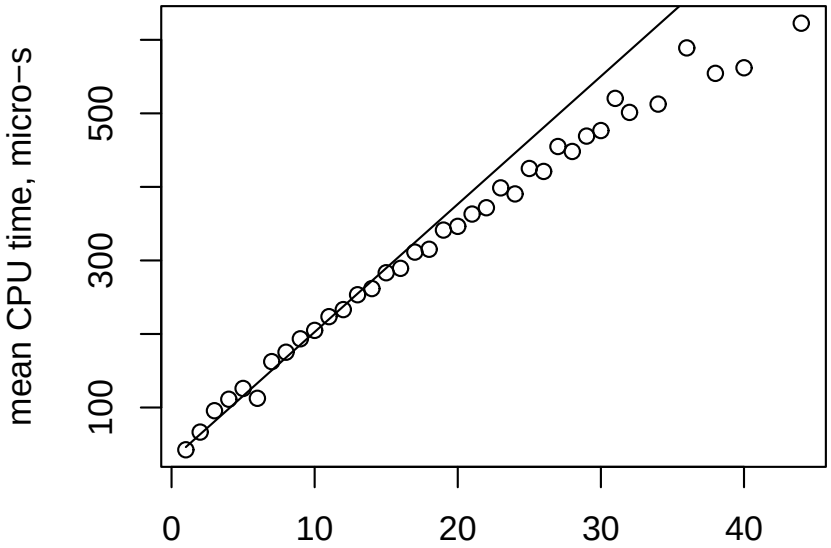
| Mu-tant | Number of input actions | | | Duration, *mtu* | | |
|---|---|---|---|---|---|---|
| | Min | Avg | Max | Min | Avg | Max |
| **M1** | 2 | 4.8 | 16 | 0 | 68.8 | 318 |
| **M2** | 2 | 4.6 | 13 | 1 | 66.4 | 389 |
| **M3** | 2 | 4.7 | 14 | 0 | 66.4 | 398 |
| **M4** | 6 | 8.5 | 18 | 28 | 165.0 |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

# Computing Performance (means)

## after delay




## after action

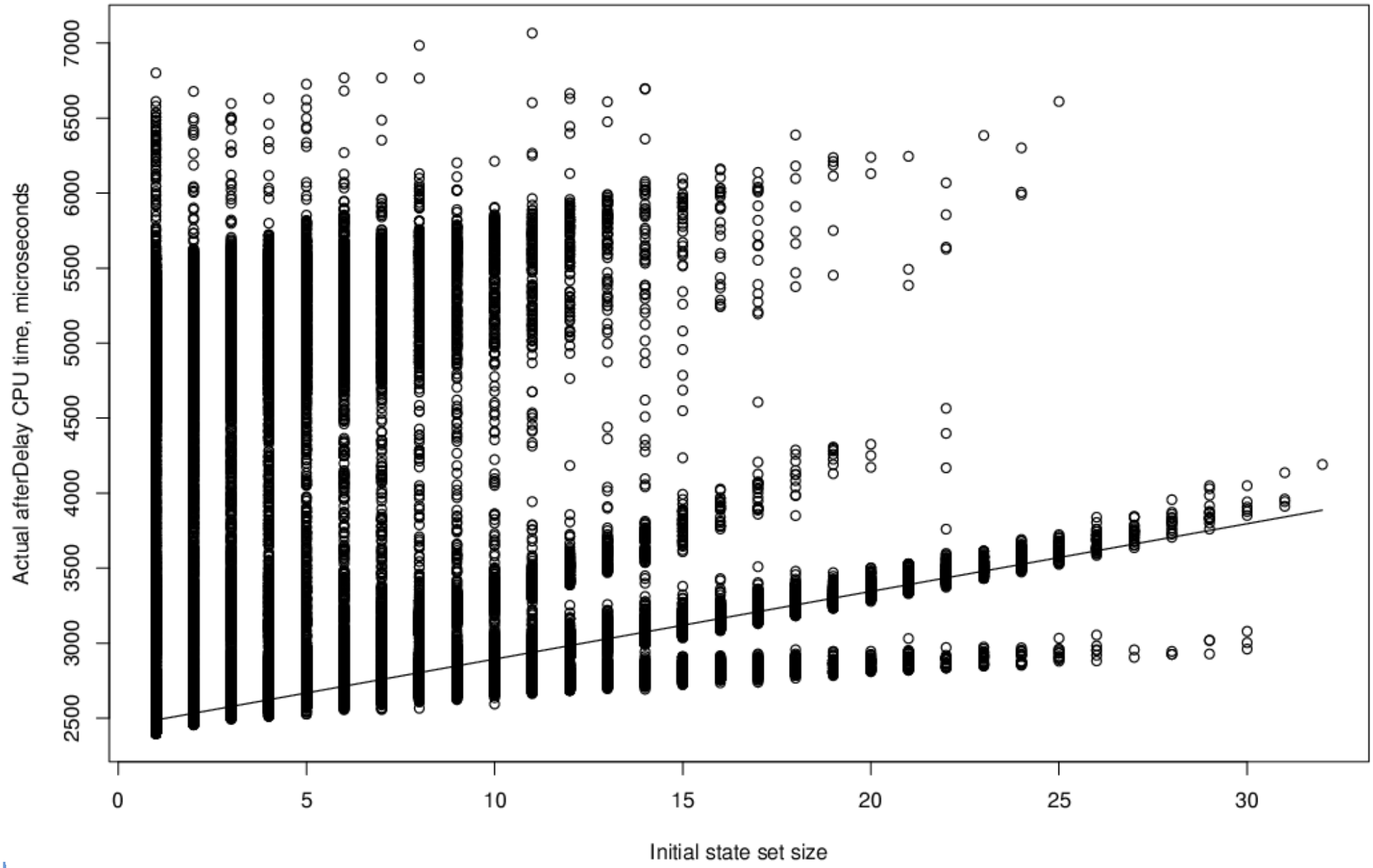


# Benchmark Data: Summary

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

| Mu-tant | Number of states in $\mathcal{Z}$ | | | | CPU execution time, $\mu s$ | | | |
|---|---|---|---|---|---|---|---|---|
| | 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 |

# Performance Unpredictable: instances



# Danfoss Case Study: EKC – Refrigeration Controller

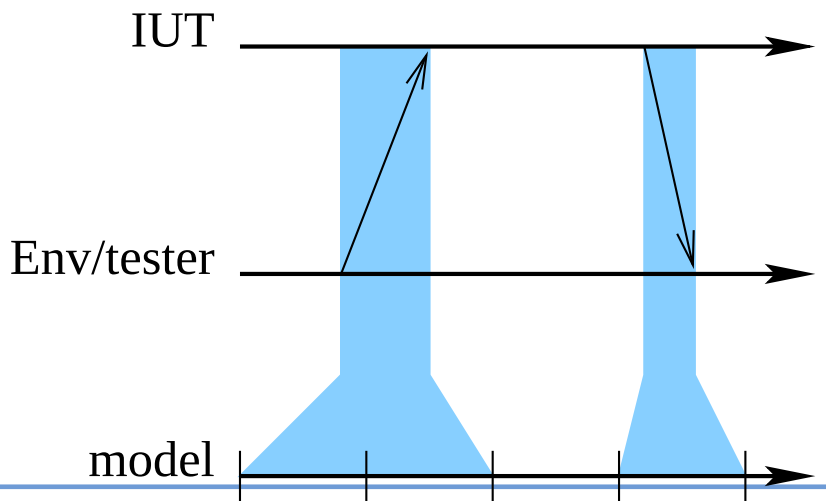
# Time Mapping to the Model and Back

- Reachability algorithms for *afterDelay* and *afterAction*:

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

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

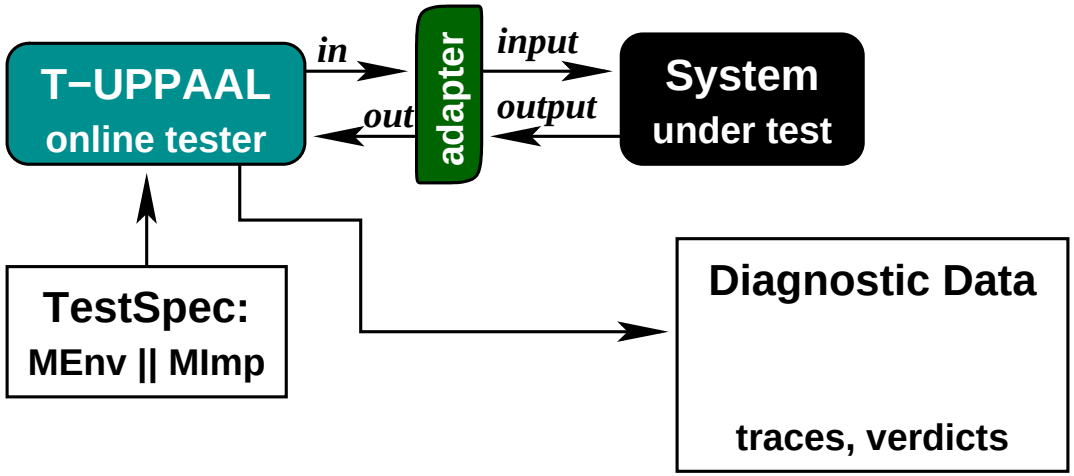
- Everything above works well in controlled-time.
- But in real world, communication doesn't happen instantaneously.
- Clocks at Env/tester and IUT may drift.
- Models of queues and drifts contain non-determinism.



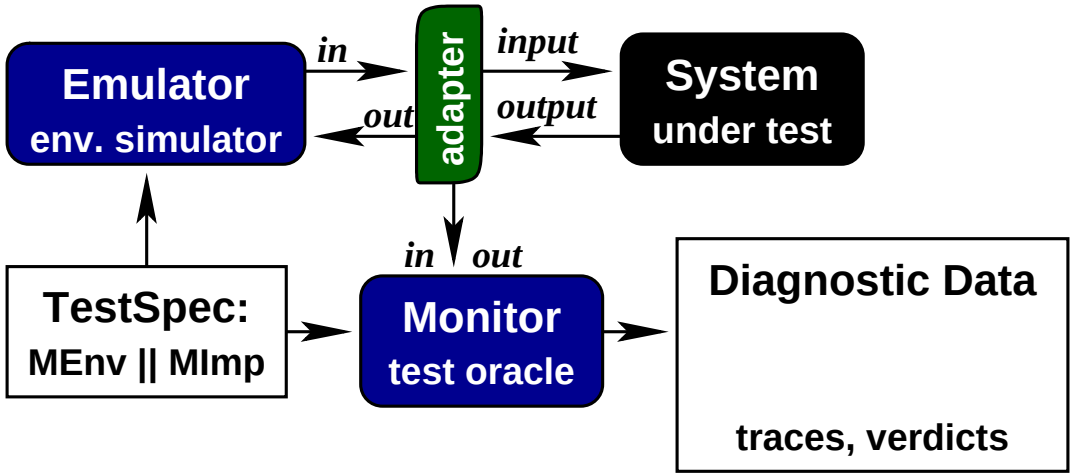
# Concluding Remarks

- Online real-time testing theoretically *sound* and *complete in limit*.
- Environment assumptions should be *known* and *explicit*.
- Relativized conformance allows to *minimize cost* of testing.
- *Implemented* in TRON using *efficient* algorithms from UPPAAL.
- Encouraging *error detection* capability and *performance*.
- TRON allows *abstract* and *non-deterministic* specifications.
- *Extreme* non-determinism may degrade performance.
- Testable environments are limited by CPU and comm. latency.
- $IUT$ models just need to be deadlock free and input-enabled.

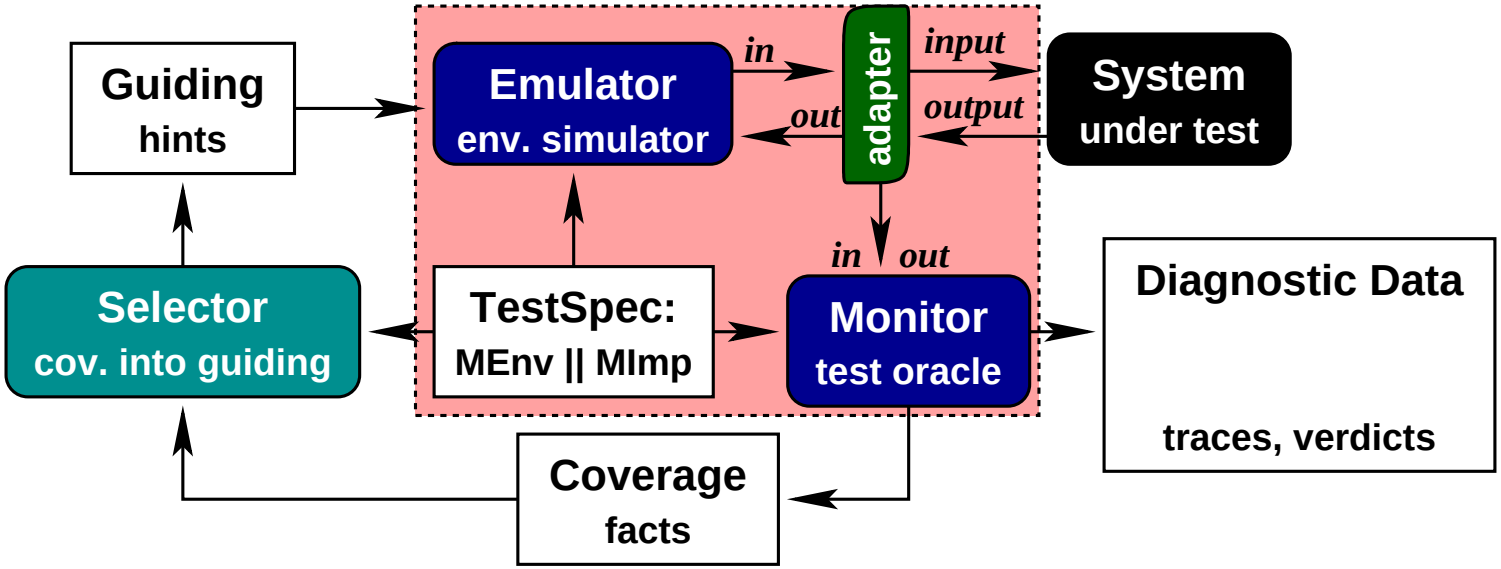
# Summary and Future Work



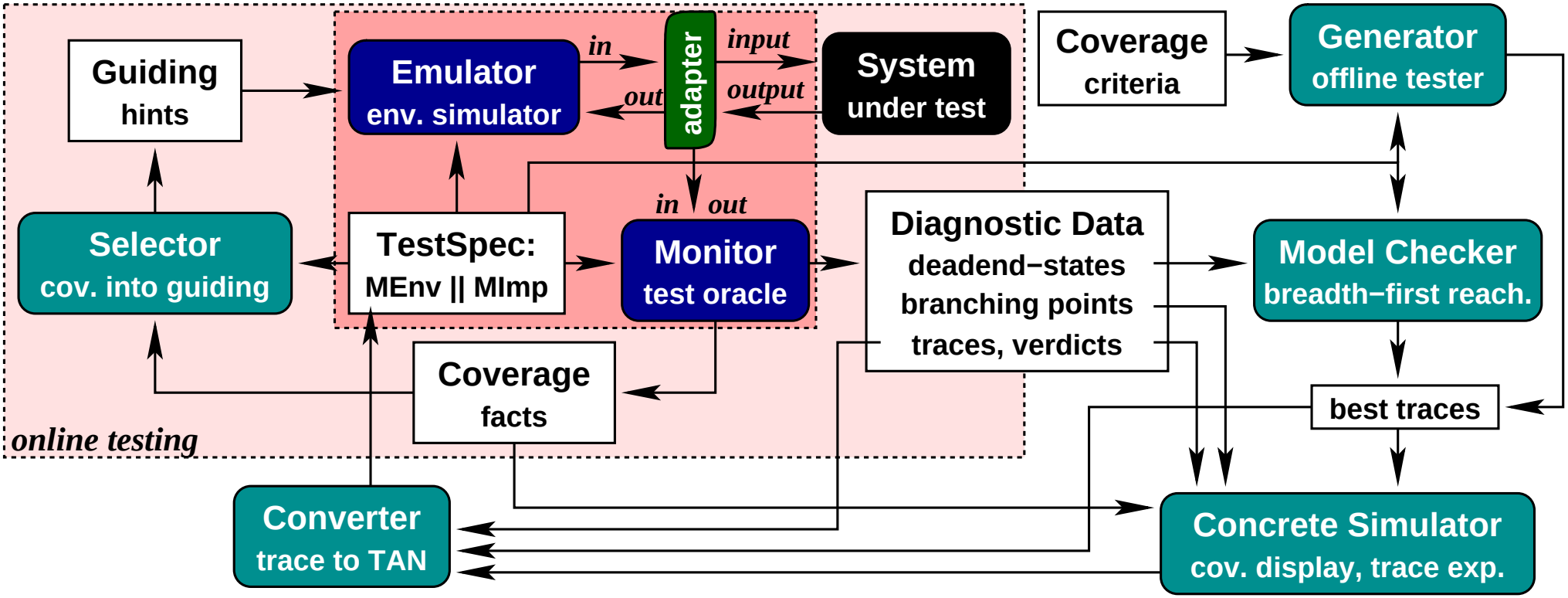
# Summary and Future Work



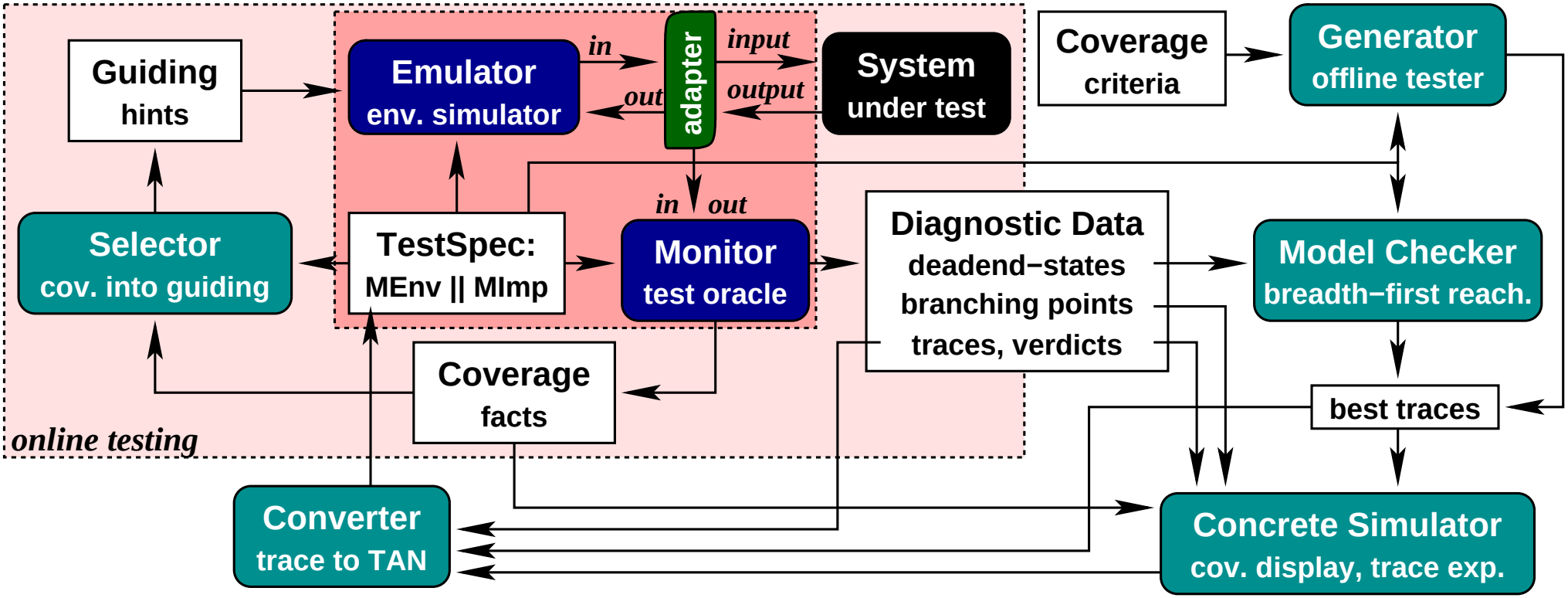
# Summary and Future Work



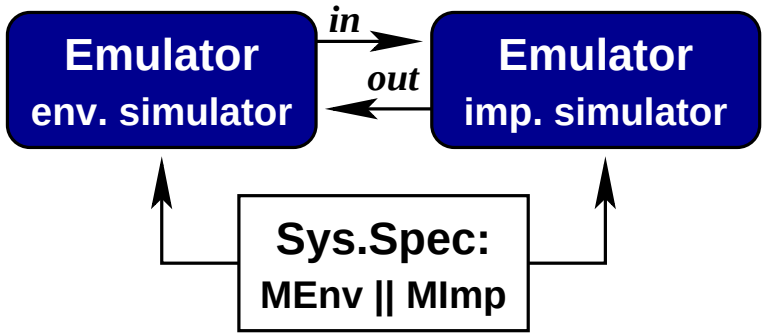
# Summary and Future Work



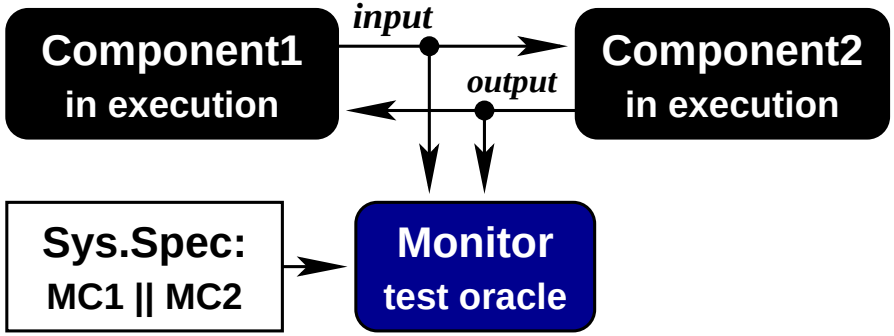
# Summary and Future Work




## Prototyping via simulation:




## Execution monitoring:



# Future Work

- Research tasks:
  - Clock synchronization, latency, jitter.
  - Coverage estimation, use coverage in guiding.
  - Diagnostics, fault location.
  - Model learning during experiment.
  - Relativized conformance in interface compatibility: unit testing.
- Engineering tasks:
  - New UPPAAL features (broadcast, committed, U-Code).
  - Termination of testing (specify property expressions?).
  - TRON in monitoring, testing via simulation and monitoring.
  - Relativized conformance in *practice*: specialized applications of generalized controllers, test-case guiding, debugging.
  - Industrial case studies.

# Download...

UPPAAL TRON is available for research and non-commercial use at:

http://www.cs.aau.dk/∼marius/tron

Thank You!