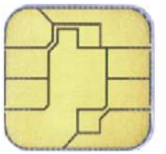
UPPAAL TRON: Testing Real-time systems ONline

Kim G. Larsen, Marius Mikučionis, Brian Nielsen, Arne Skou

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



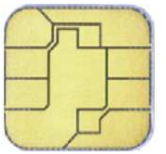
Center for Embedded Software Systems

Basic Research in Computer Science

Aalborg University

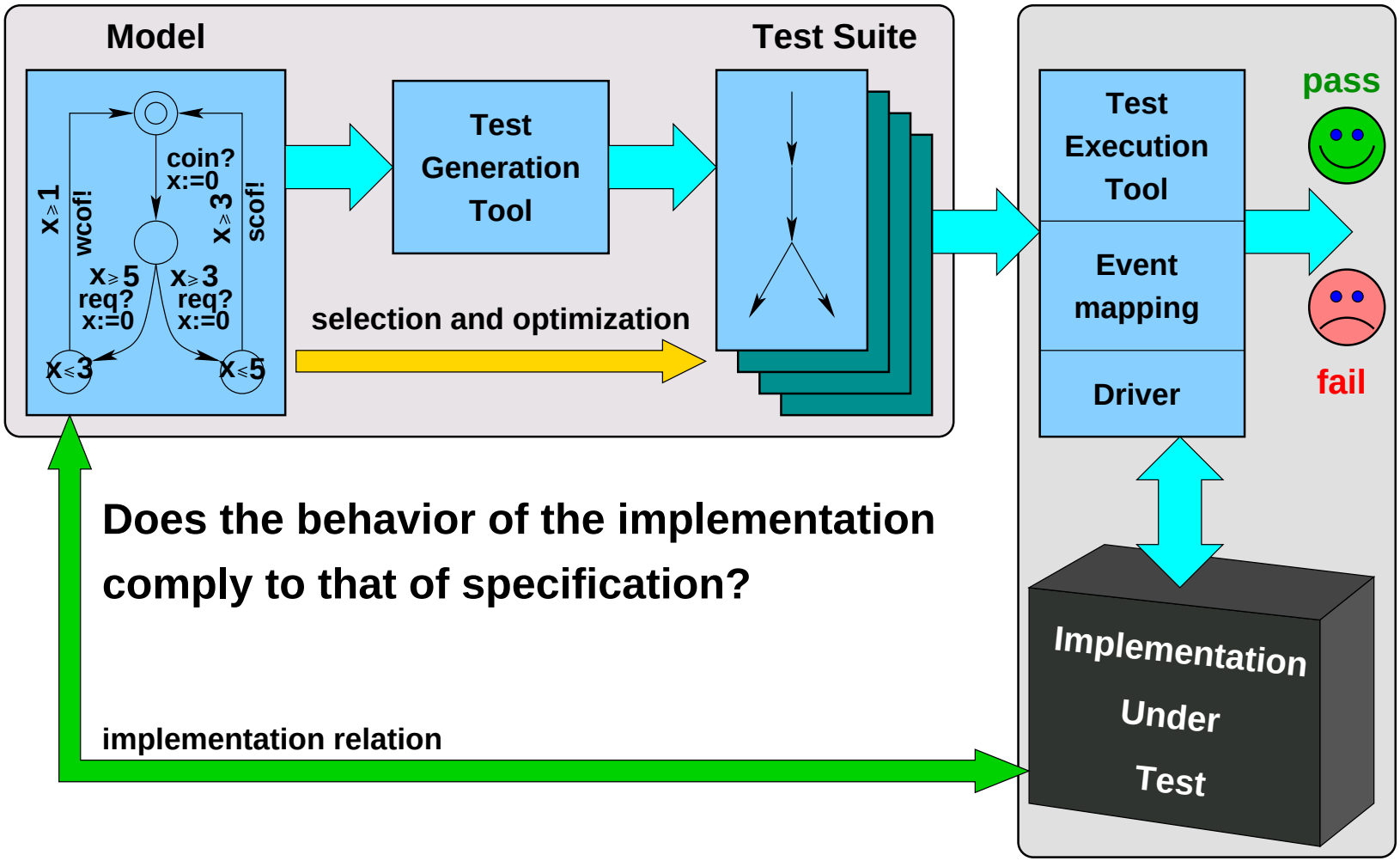
Outline

- Classical model-based testing.
- Online test setup: from specification to testing.
- UPPAAL TRON framework.
- Conformance relations: tioco and rtioco.
- Online test algorithm.
- Conclusions.
- Future work.
- Demo of Java “light controller” online test.

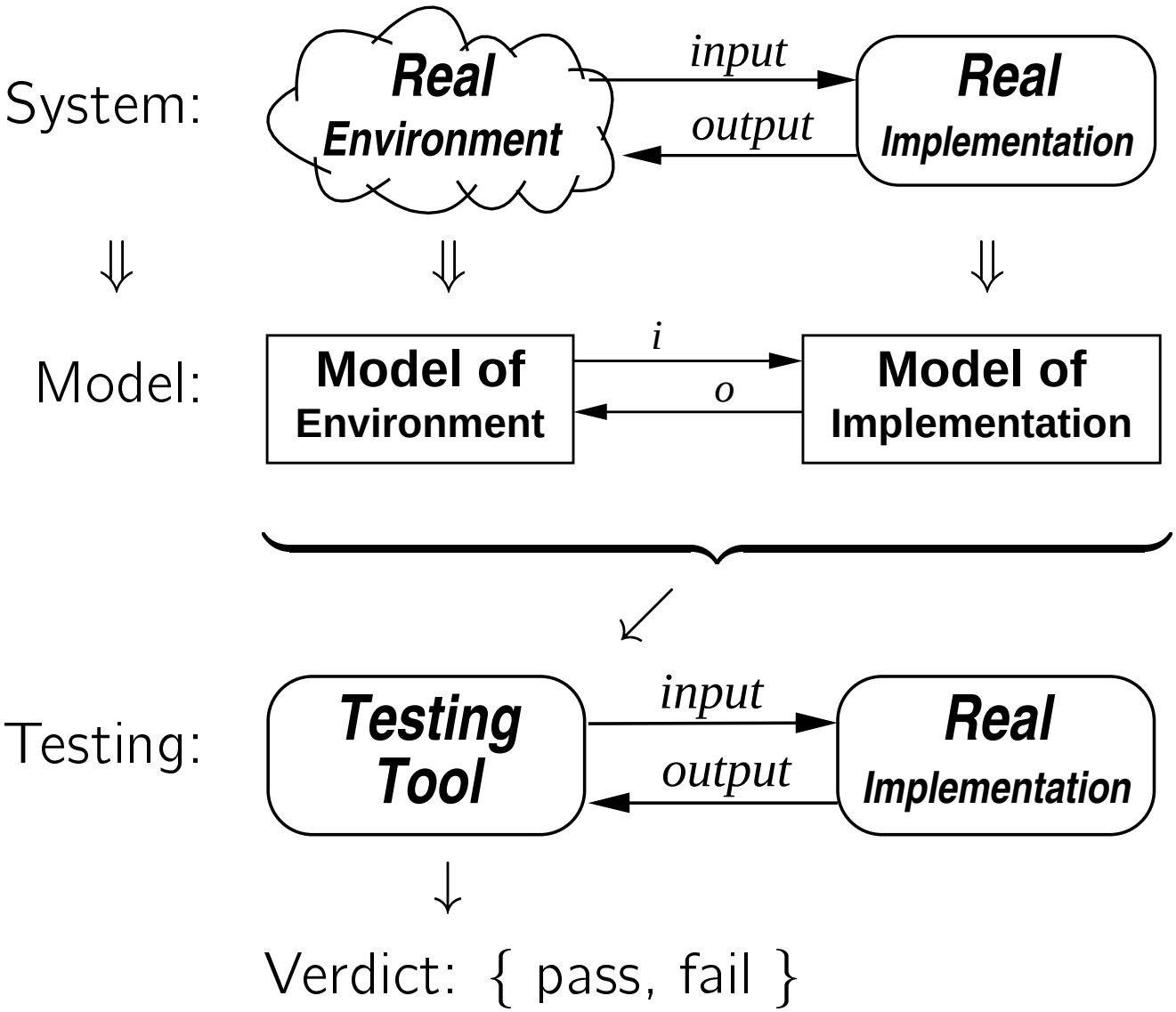


Classical Model-based Testing Framework

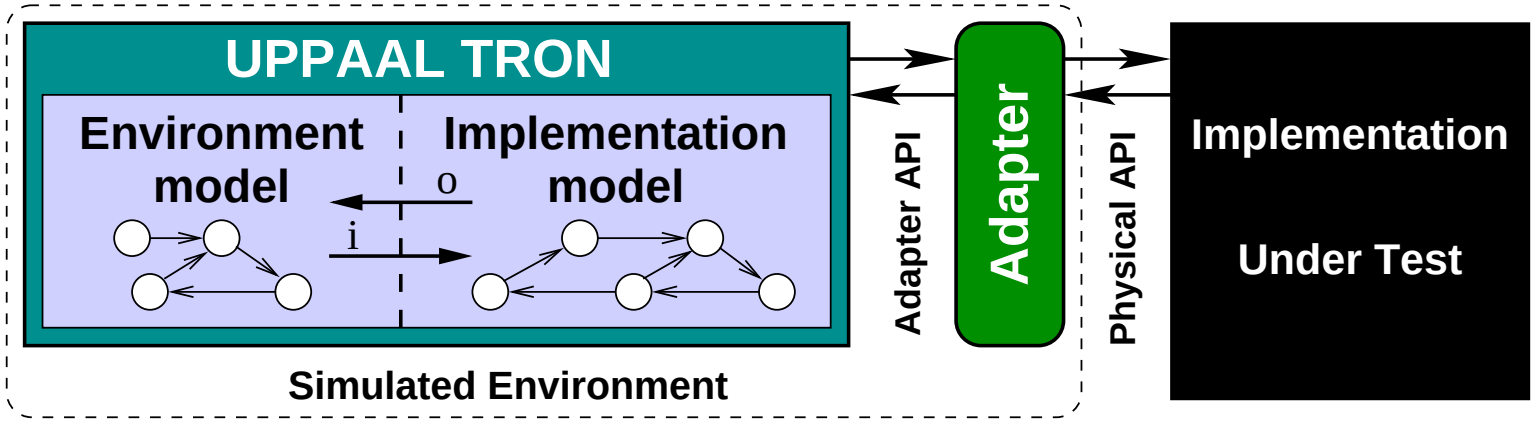
- Black-box, model-based, conformance testing offline.



Online setup: System ⇒ Model ⇒ Testing



UPPAAL TRON Framework



- UPPAAL timed automata specification:
  - Model of (requirements for) implementation,
  - Model of (assumptions about) environment,
  - Contain several concurrent non-deterministic processes.
- Online testing tool:
  - generates test primitives for stimuli (input) with timing,
  - verifies the validity of response (output) with timing,
  - while test primitives are executed online.

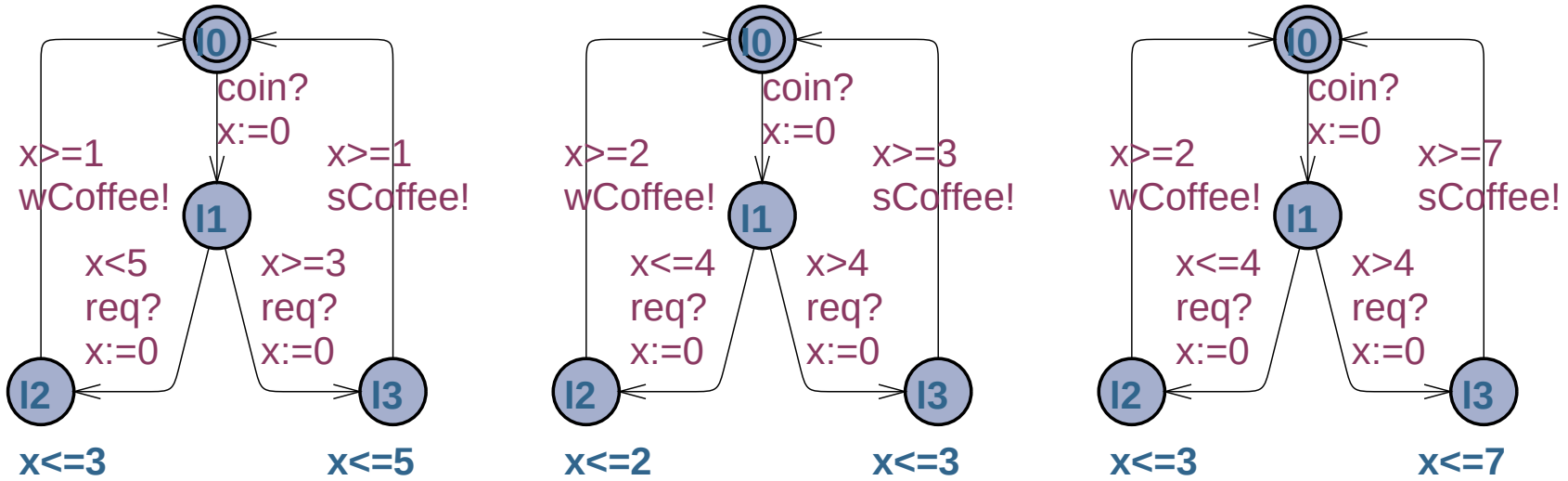
Timed I/O Conformance Relation

- Defines the correctness criteria by analyzing observable traces.
- Intuitively:
  - timed trace e.g.: $\sigma = coin? \cdot 5 \cdot req? \cdot 2 \cdot weakCoffee! \cdot 9 \cdot coin?$
  - observed output should be matched and allowed in specification,
  - timing of observed output should be allowed in specification,
  - any observation outside the specification is a fault.
- Formally:
  - *Timed traces* from state s: $\mathsf{TTr}(s) \stackrel{def}{=} \{\sigma \in (A \cup \mathbb{R}_{\geq 0})^* \mid s \stackrel{\sigma}{\Rightarrow}\}$

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

$$m \text{ tioco } s \stackrel{def}{=} \forall \sigma \in \mathsf{TTr}(s).\mathsf{Out}(m \text{ After } \sigma) \subseteq \mathsf{Out}(s \text{ After } \sigma)$$

Timed I/O Conformance Example



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

Relativized Timed I/O Conformance

- Testing in *relation* to specific assumptions about *environment*.
- Relativized Timed Input/Output Conformance:

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

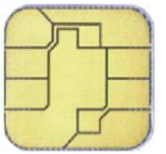
- Under certain conditions, rtioco is the same as *trace inclusion*:

$$m \text{ rtioco}_e \ s \iff \mathsf{TTr}(m) \cap \mathsf{TTr}(e) \subseteq \mathsf{TTr}(s) \cap \mathsf{TTr}(e)$$

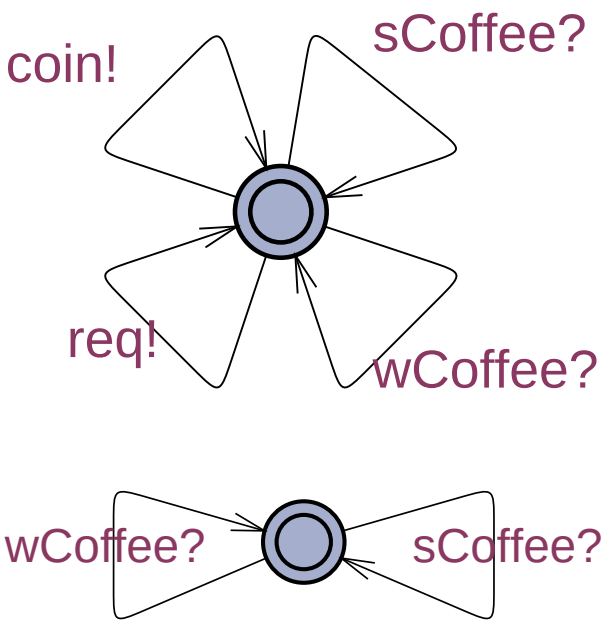
- *Ordering*: g is stronger than f:

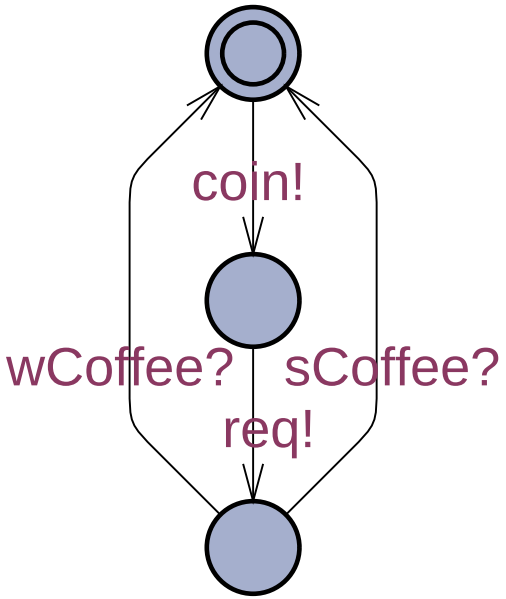
$$g \sqsubseteq f \stackrel{def}{=} \mathsf{TTr}(g) \subseteq \mathsf{TTr}(f)$$

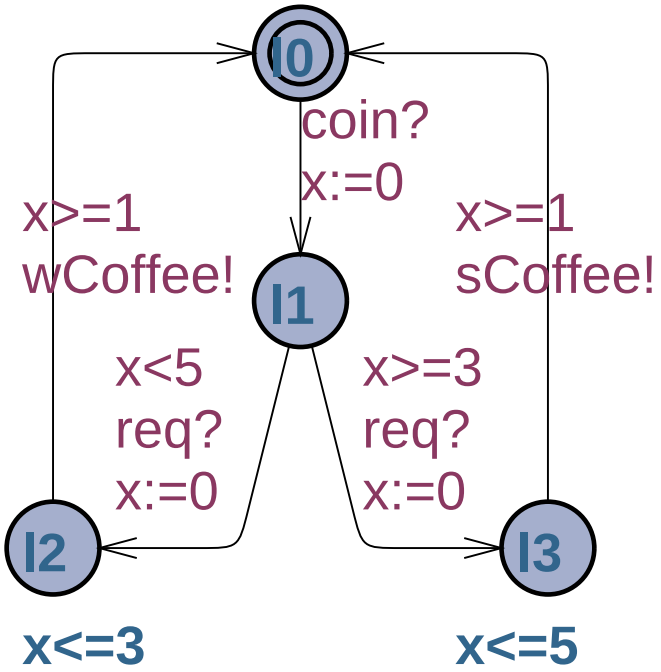
- Allows to *reuse testing effort*, assuming m rtioco$_e$ s:
  - Stronger environment e': if $e' \sqsubseteq e$ then m rtioco$_{e'}$ s.
  - Weaker specification s': if $s \sqsubseteq s'$ then m rtioco$_e$ s'.

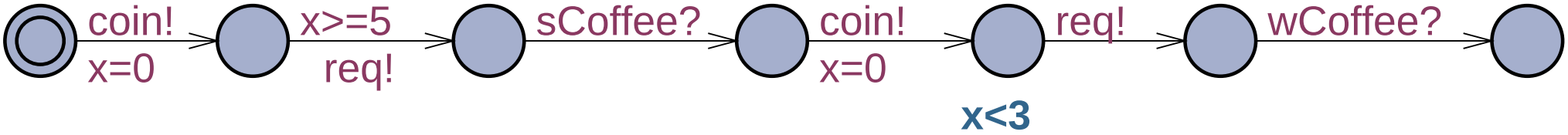


Relativized Conformance Example

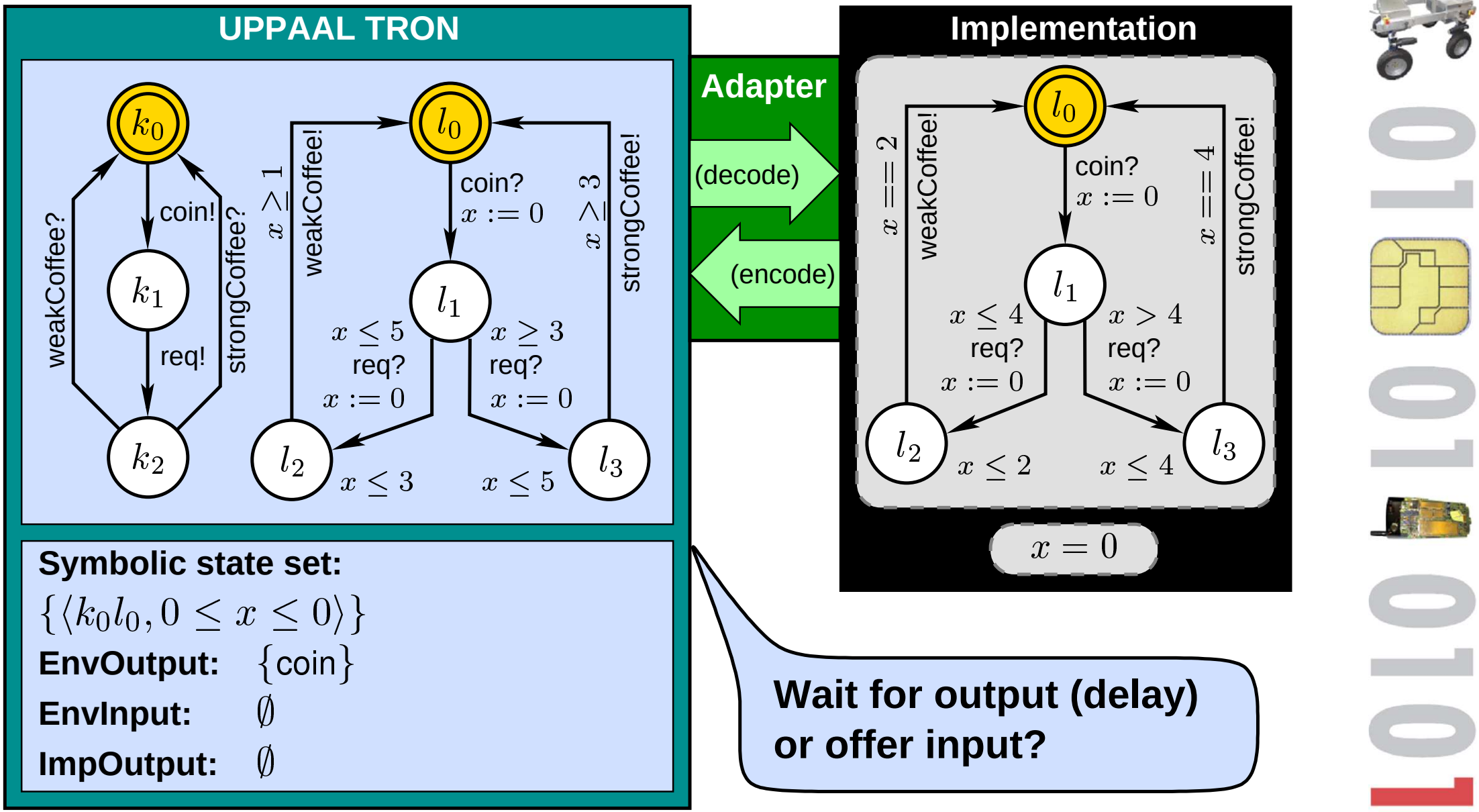








Online Test in Action



Symbolic state set:

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

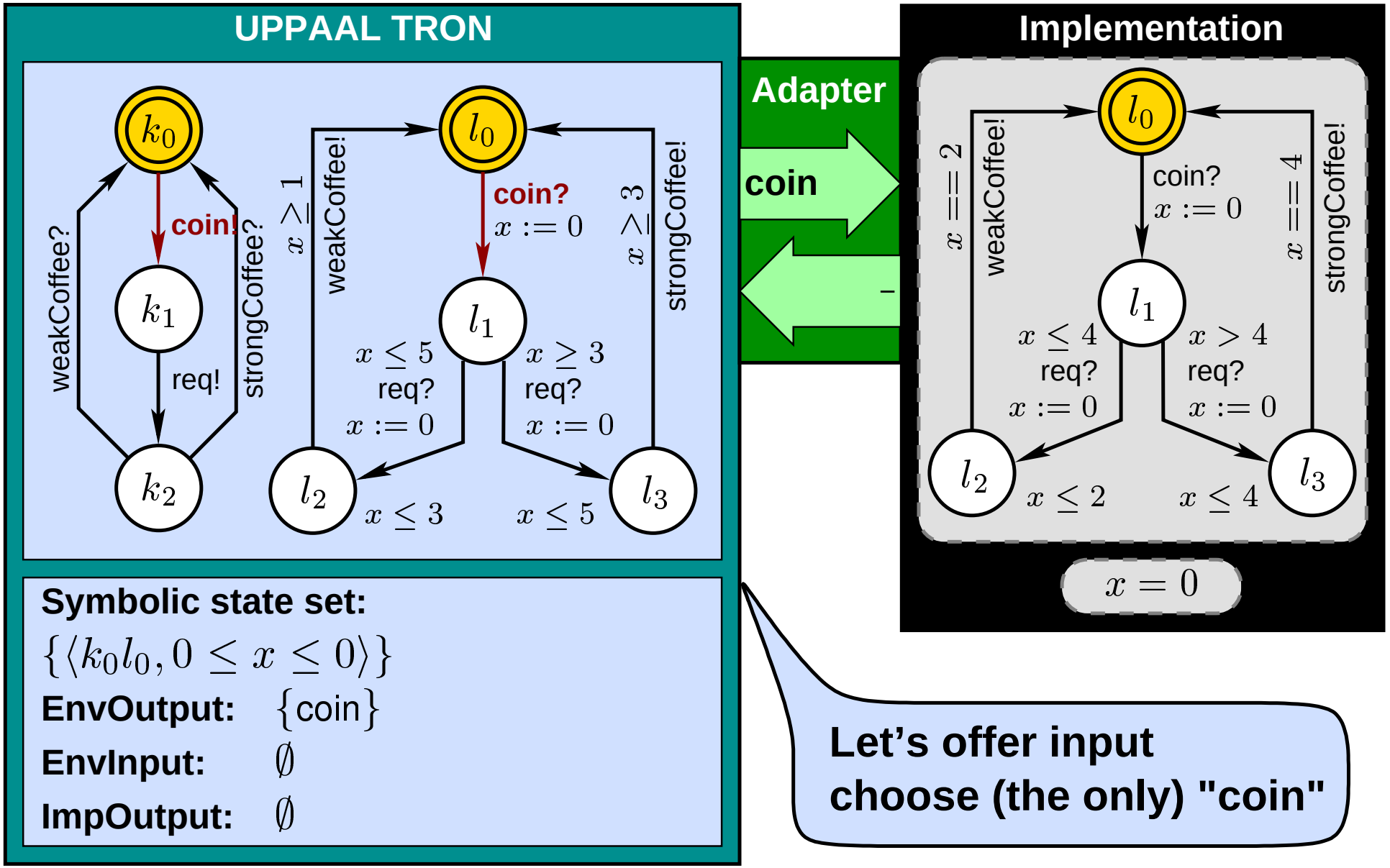
EnvOutput: $\{\text{coin}\}$

EnvInput: $\emptyset$

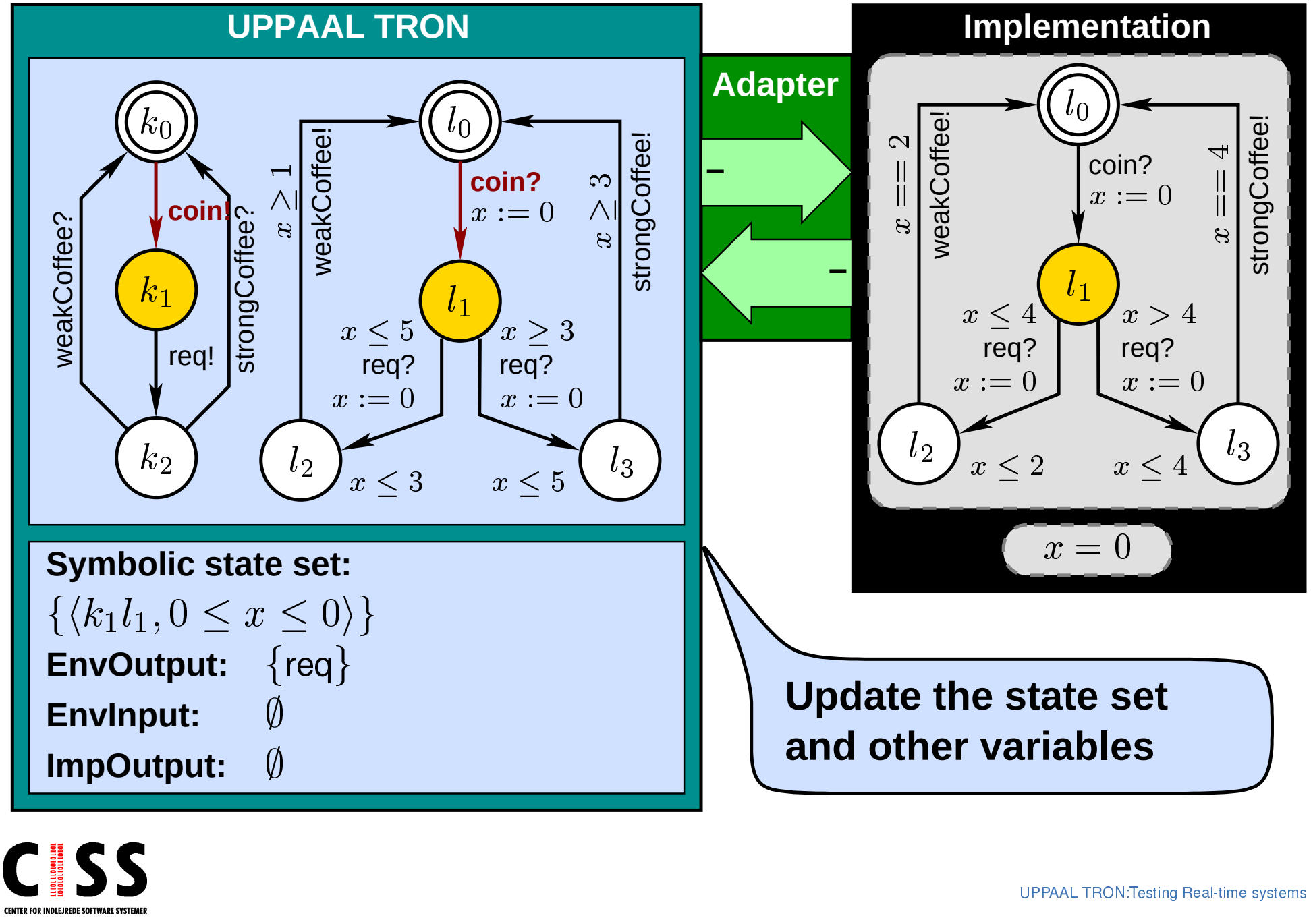
ImpOutput: $\emptyset$

Wait for output (delay) or offer input?

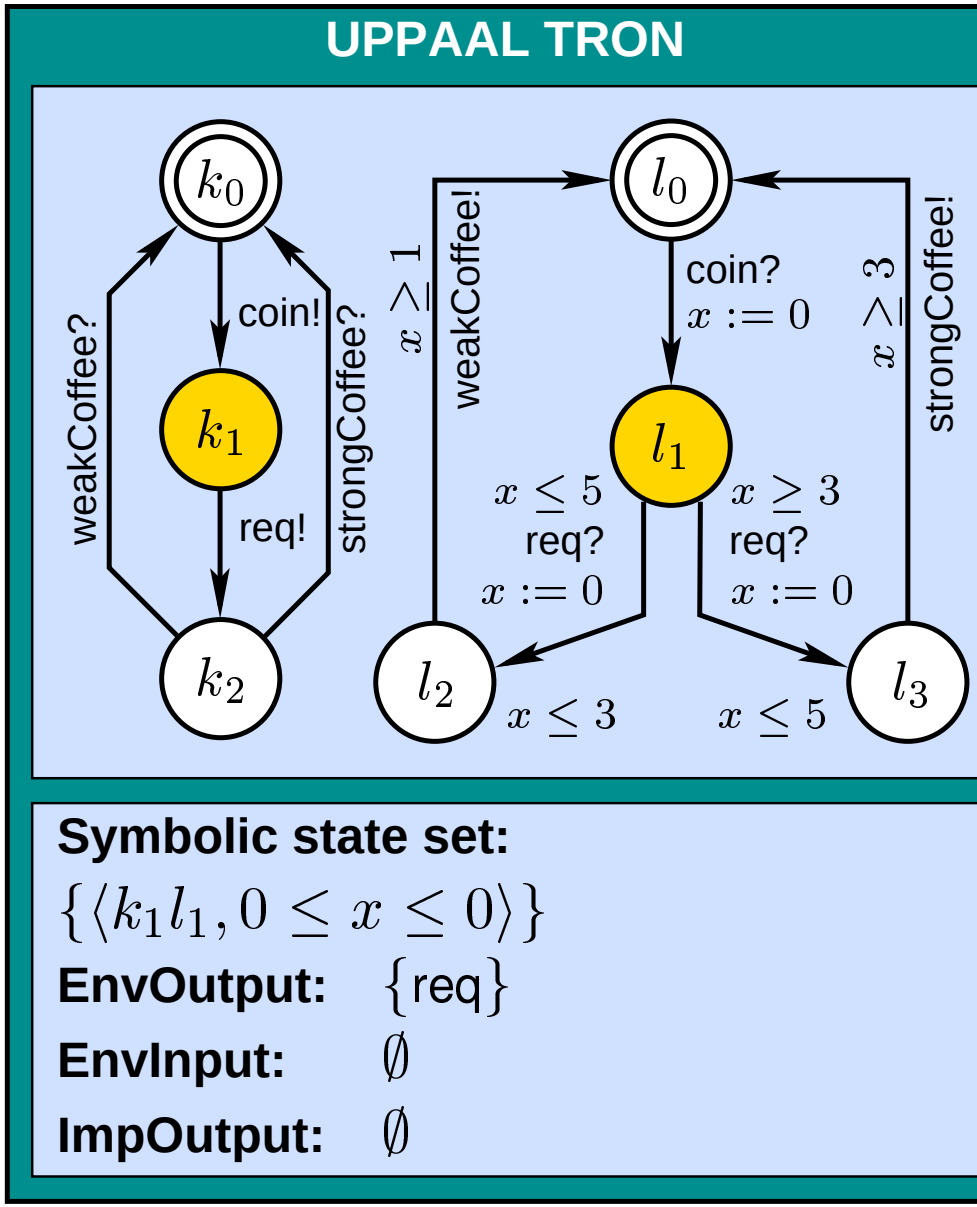
Online Test in Action

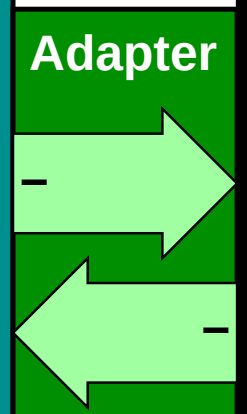


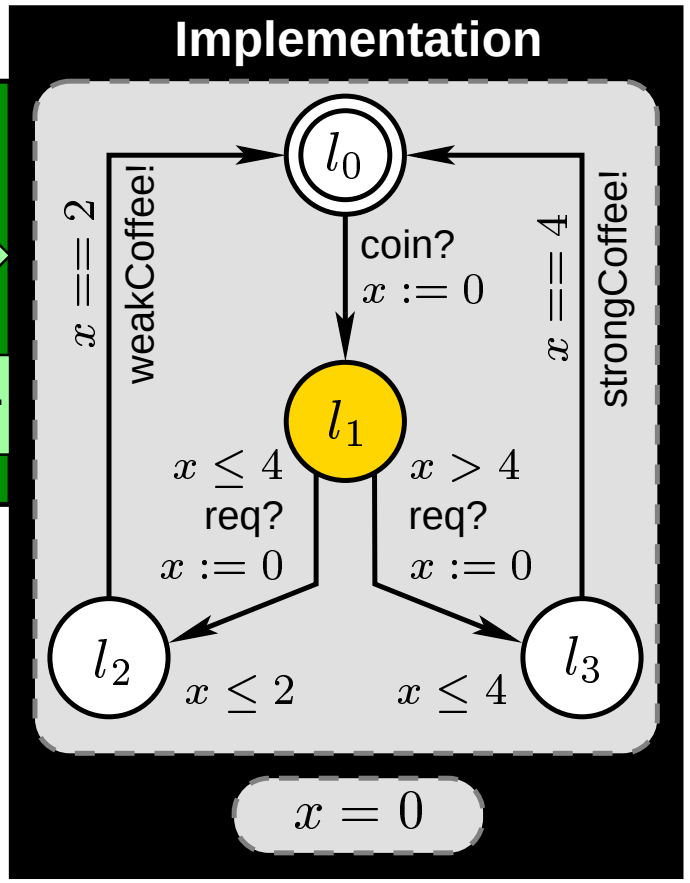
Online Test in Action



Online Test in Action

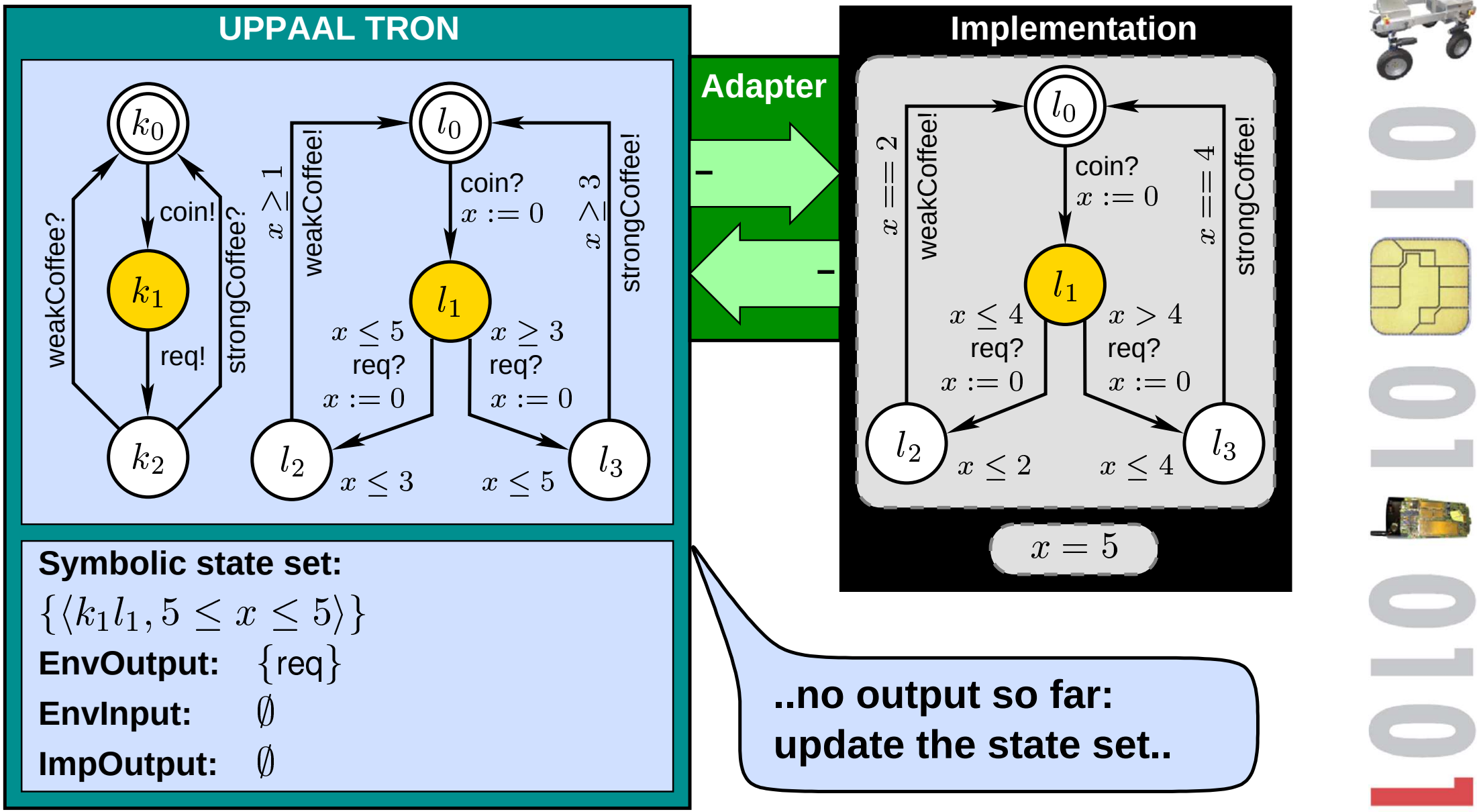




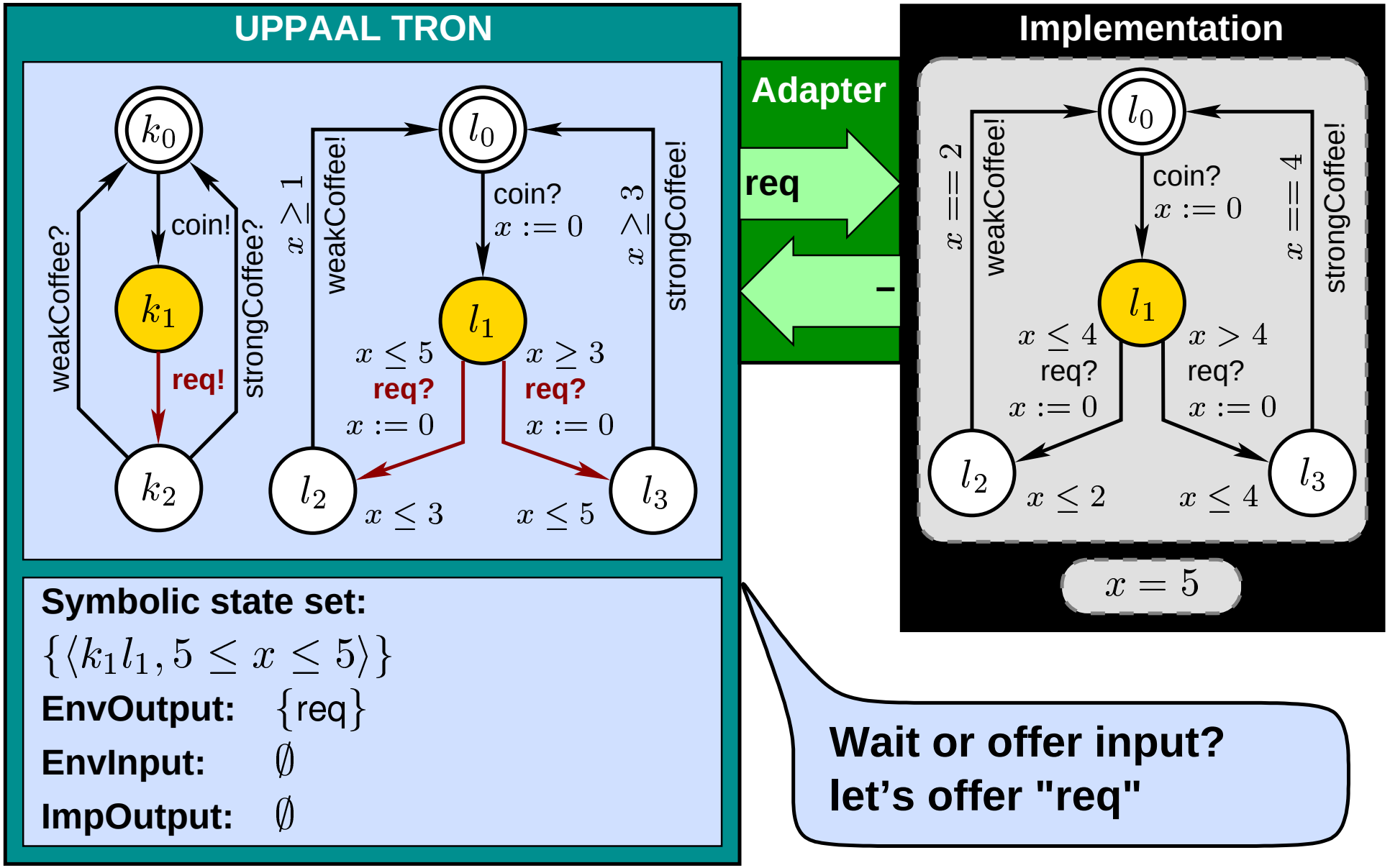


Wait or offer input?
Let's wait for 5 units

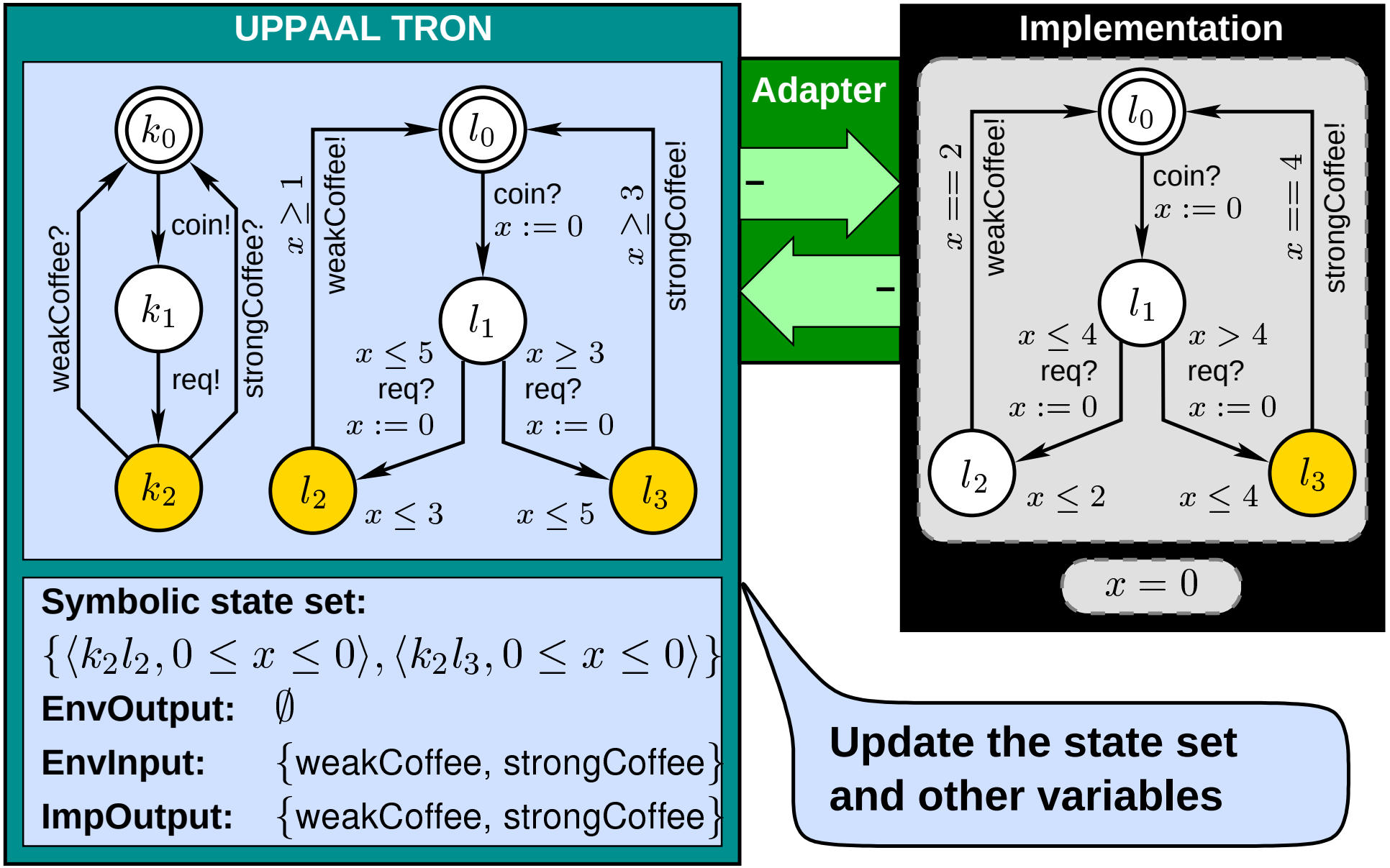
Online Test in Action



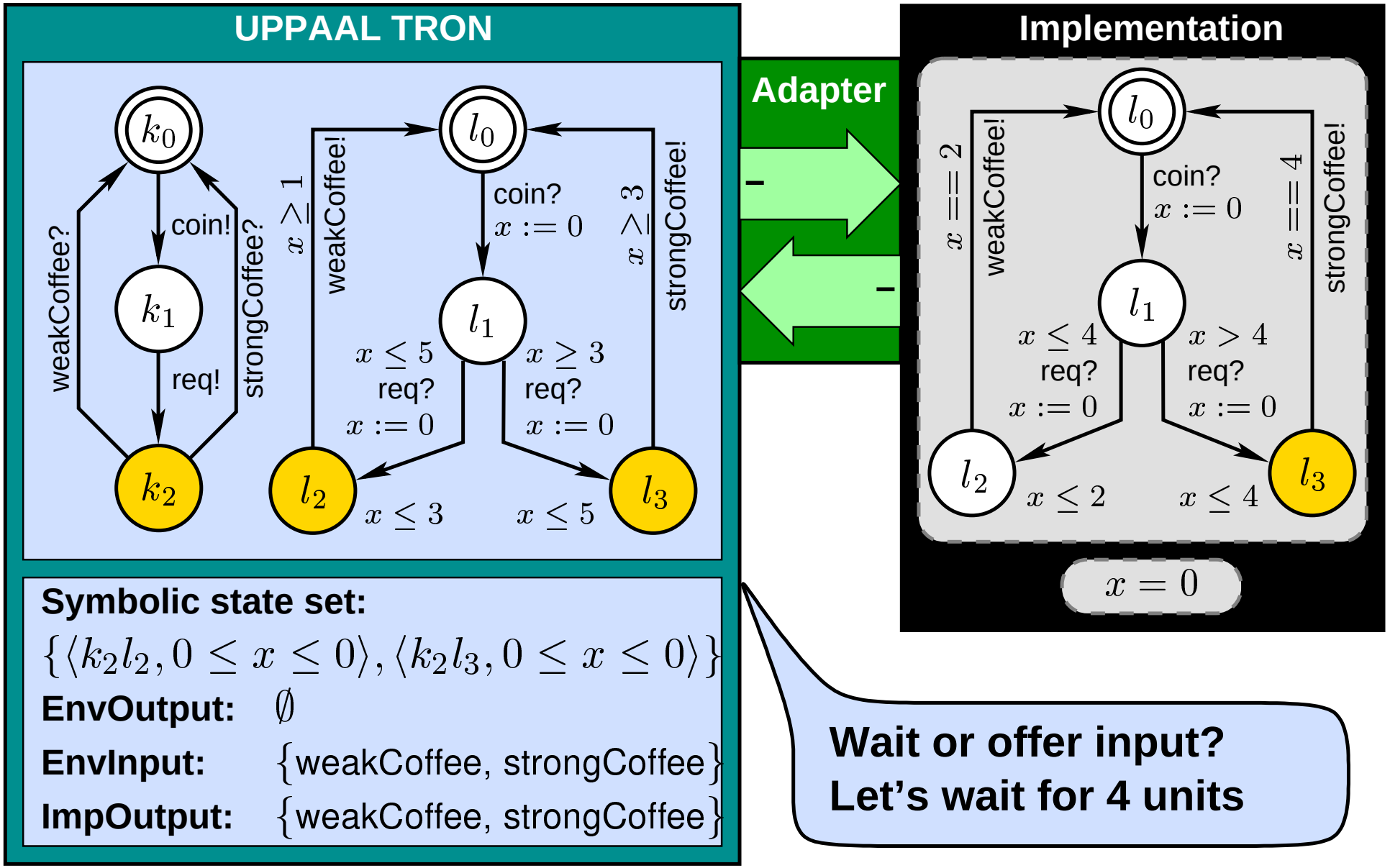
Online Test in Action



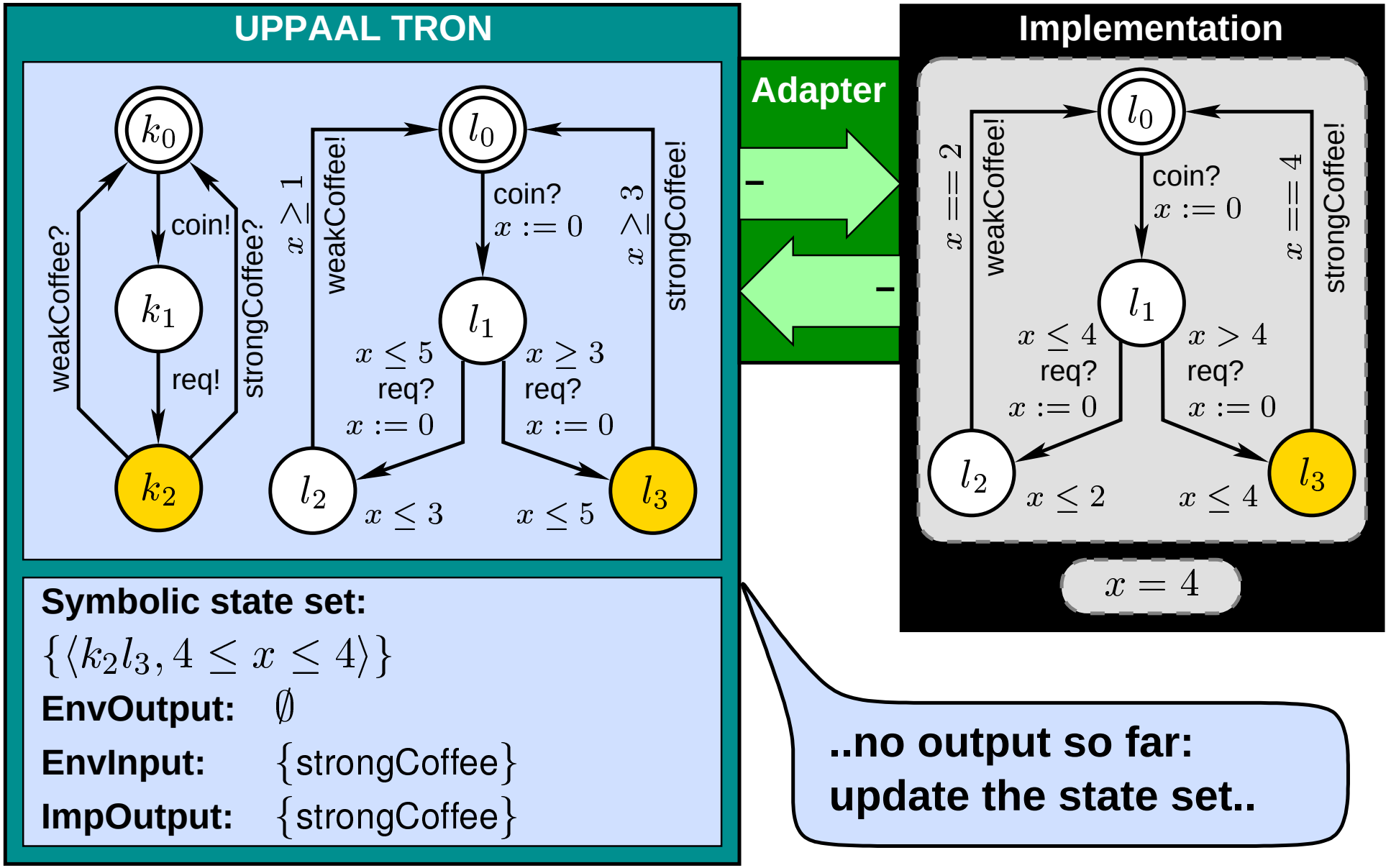
Online Test in Action



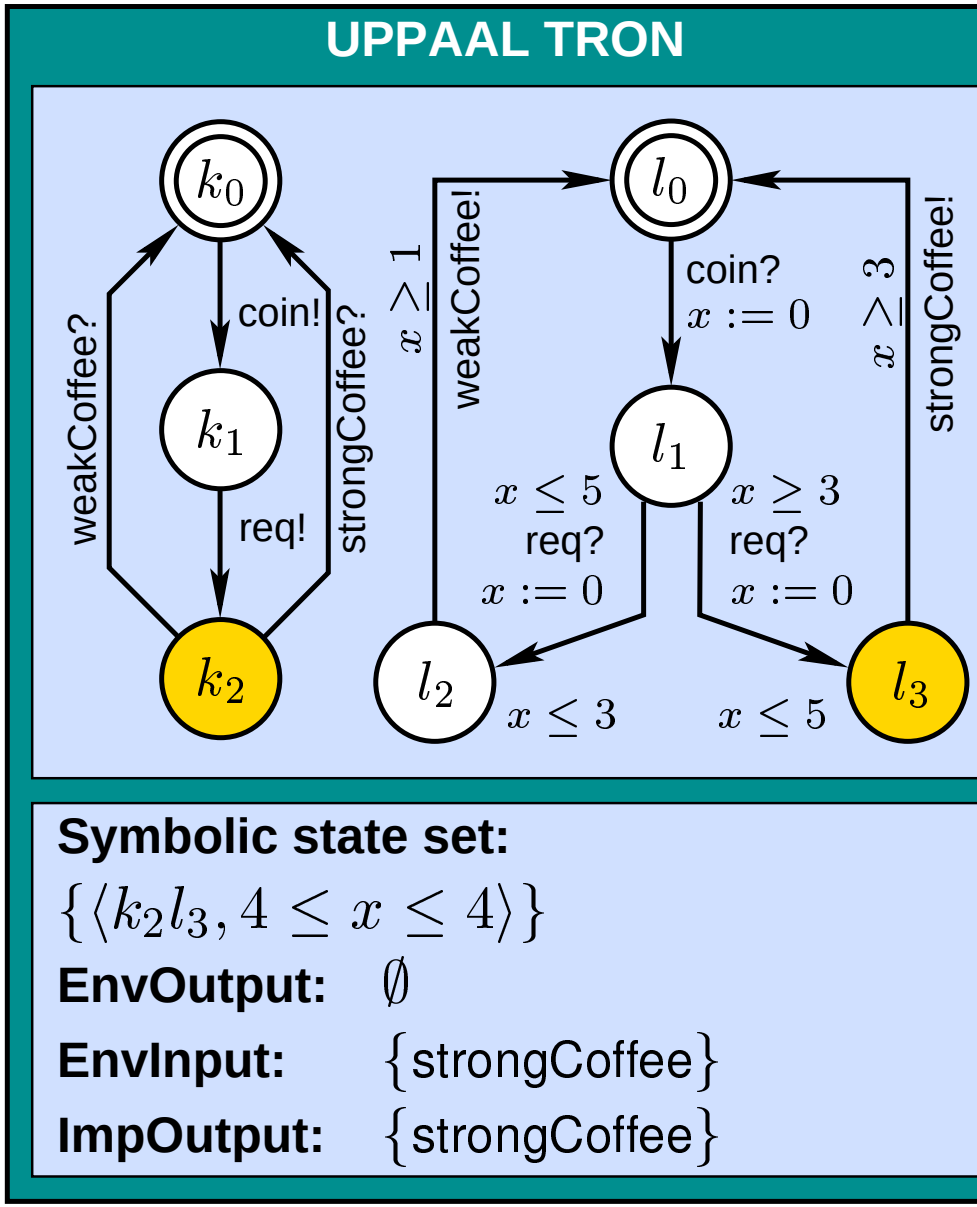
Online Test in Action

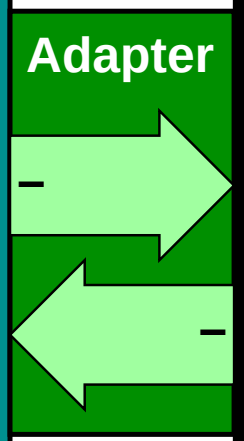


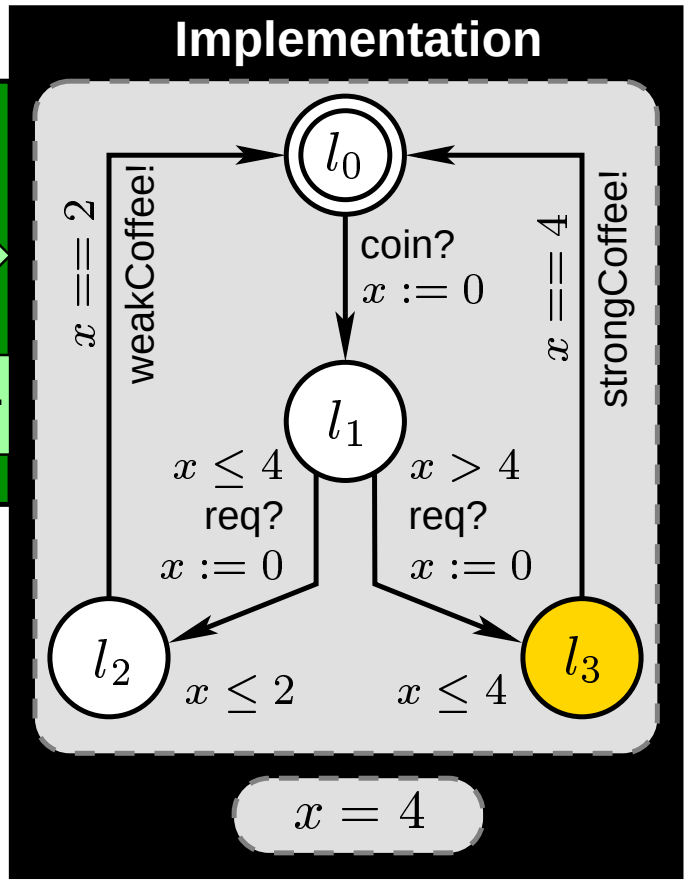
Online Test in Action



Online Test in Action

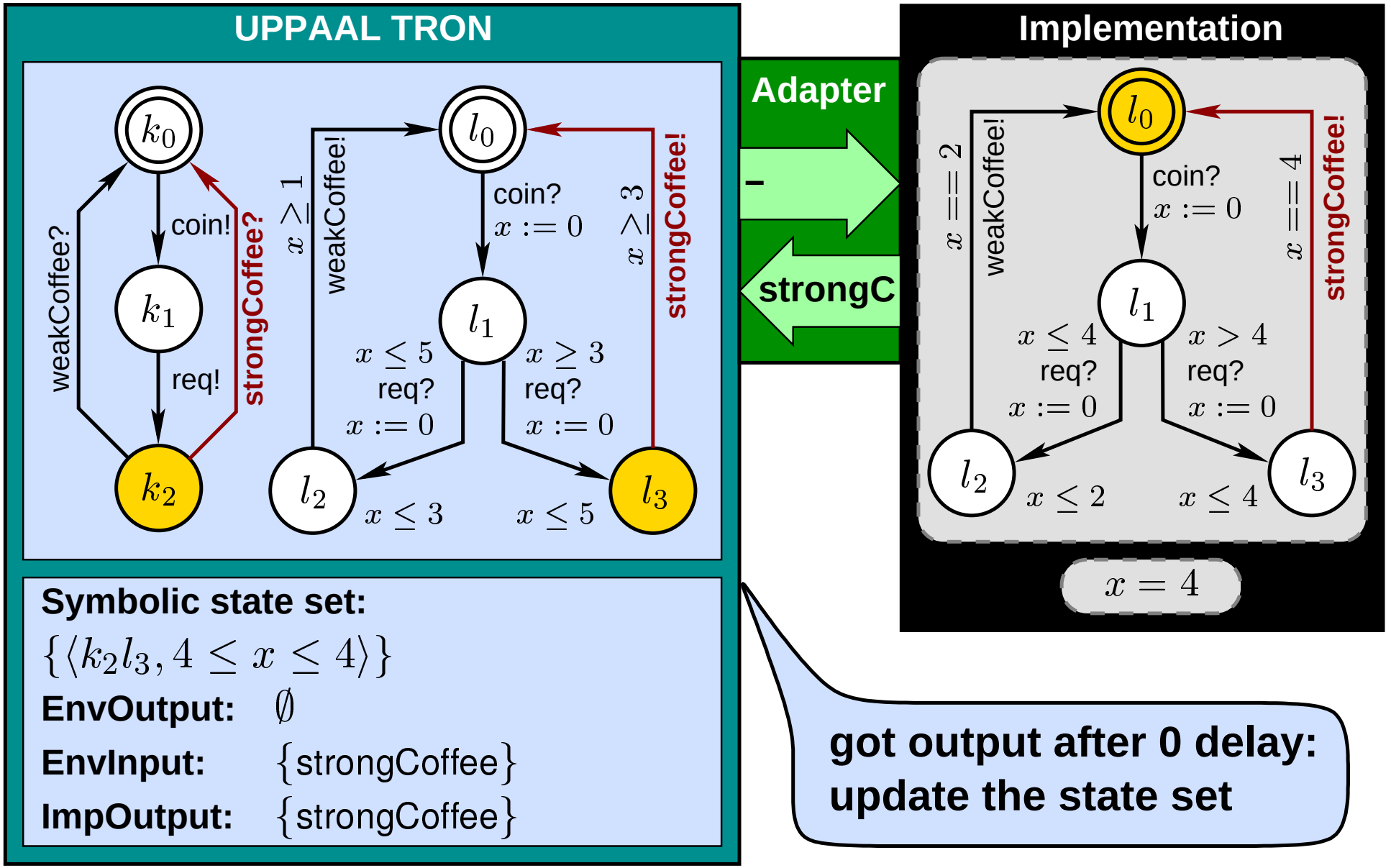




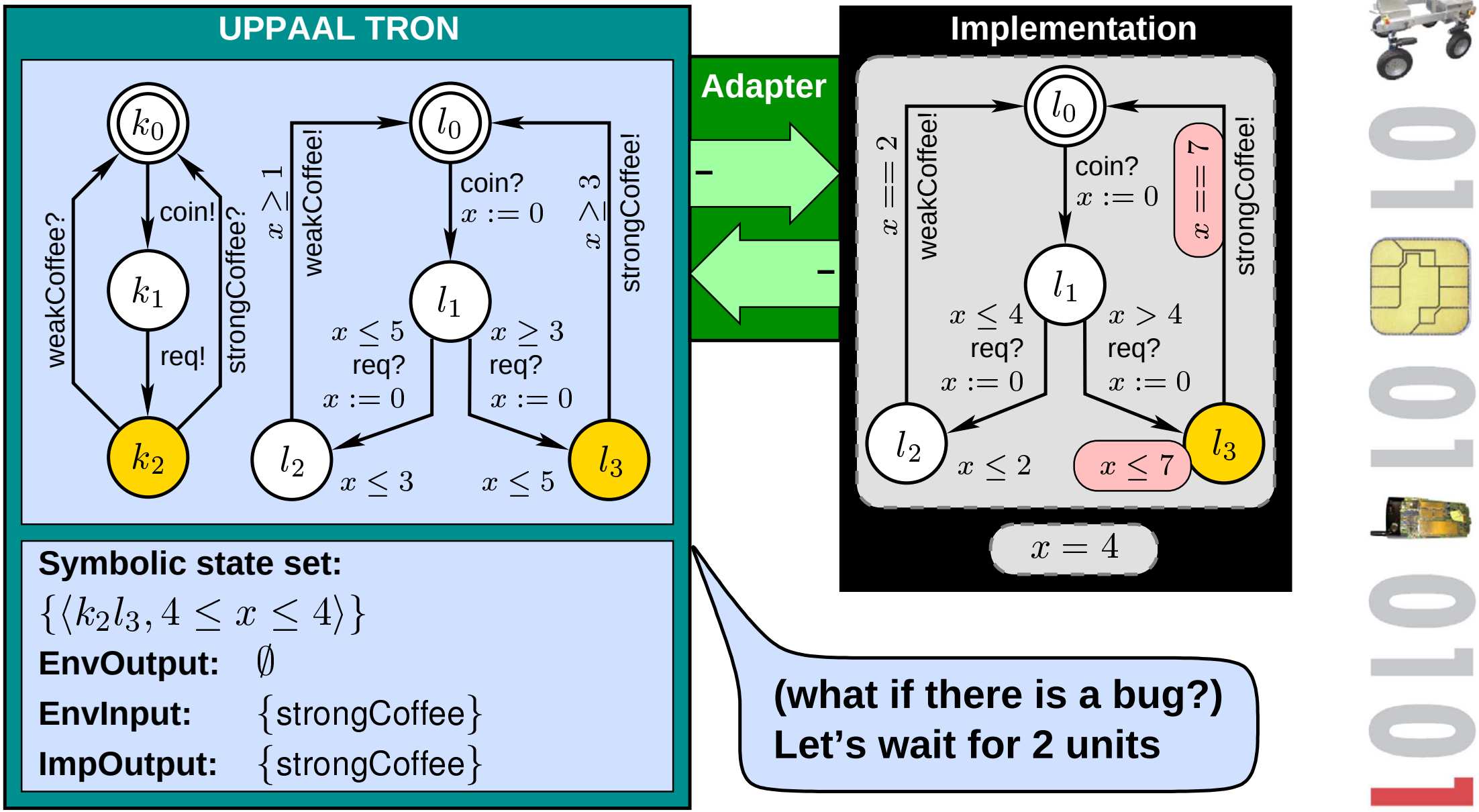


Wait or offer input?
Let's wait for 2 units

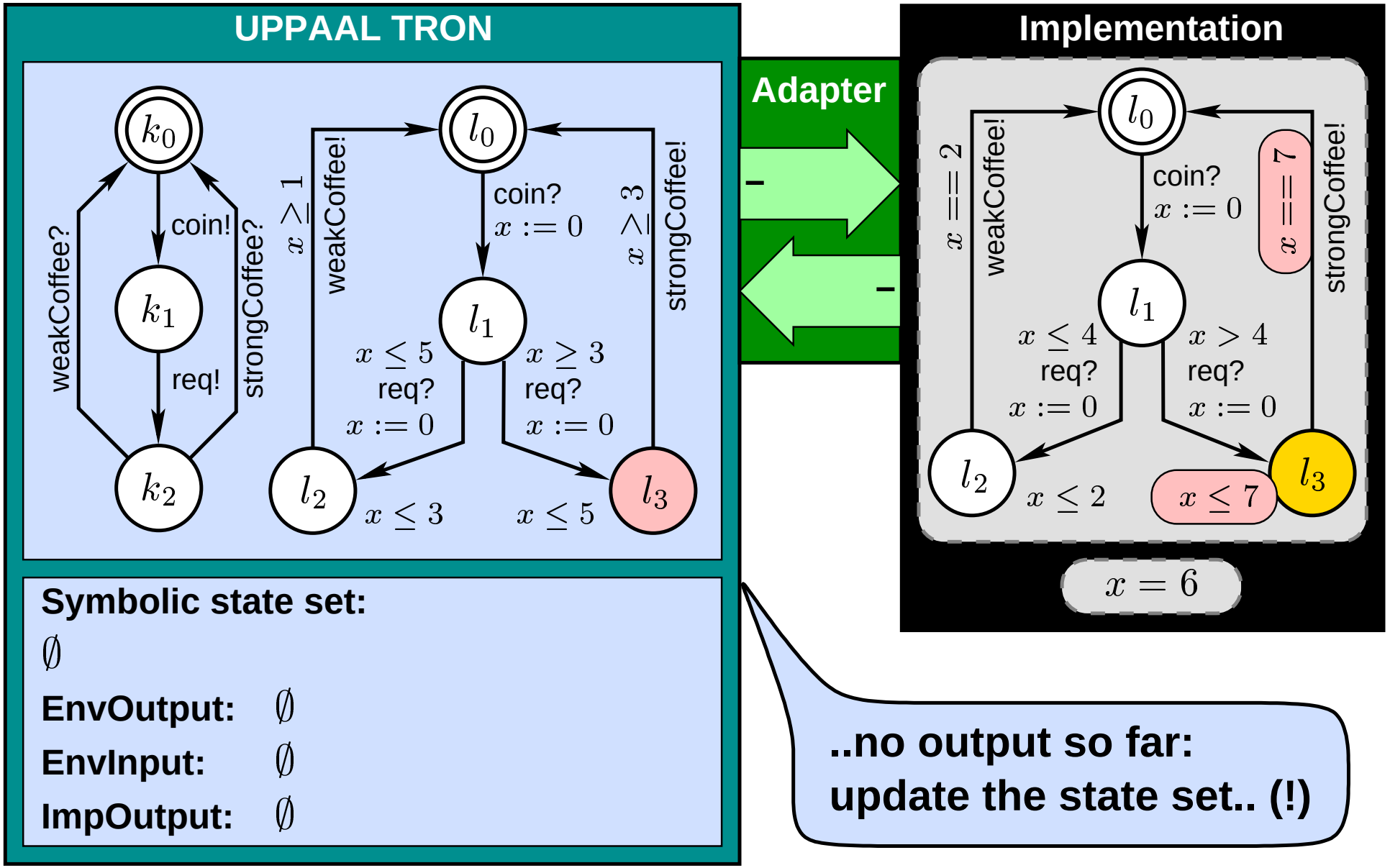
Online Test in Action



Online Test in Action



Online Test in Action

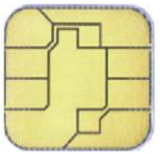


Online Test Algorithm

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

1. **if** EnvOutput$(\mathcal{Z}) \neq \emptyset$ // offer an input
   randomly choose $a \in$ EnvOutput$(\mathcal{Z})$
   send a to IUT
   $\mathcal{Z} := \mathcal{Z}$ After a
2. randomly choose $\delta \in$ 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 $o \notin$ 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**

Online Test Algorithm

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

1. **if** EnvOutput$(\mathcal{Z}) \neq \emptyset$ // offer an input
   randomly choose $a \in$ EnvOutput$(\mathcal{Z})$
   send a to IUT
   $\mathcal{Z} := \mathcal{Z}$ After a
2. randomly choose $\delta \in$ 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 $o \notin$ 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

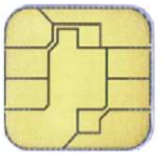
if $\mathcal{Z} = \emptyset$ **then return fail else return pass**

sound and complete in limit

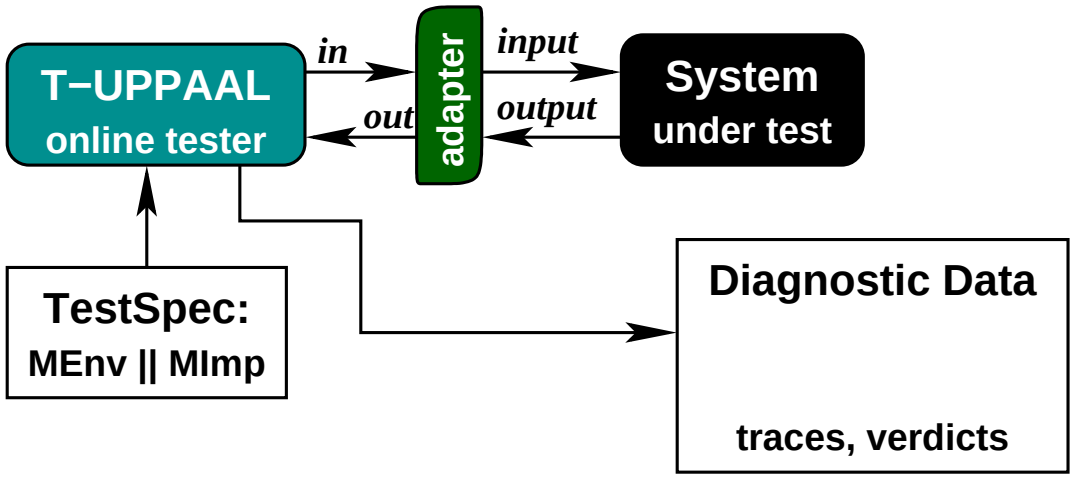



Conclusions

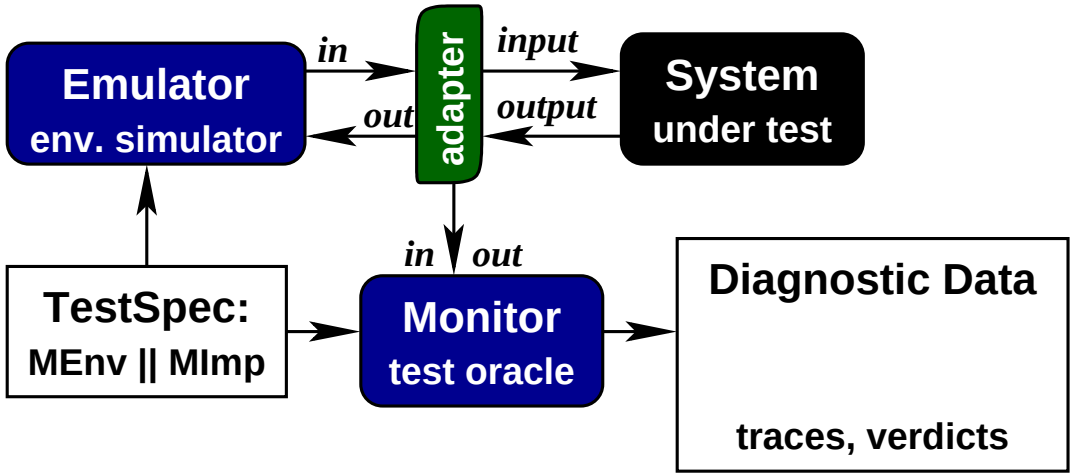
- Online real-time testing theoretically *sound* and *complete*.
- Relativized conformance provides:
  - Explicit environment assumptions
  - Realism and guiding
  - Separation of concerns
  - Modularity
  - Theoretical properties
- Implementation:
  - Allows *abstract* and *non-deterministic* specifications
  - Shows encouraging *error detection* capability and *performance*
  - Using UPPAAL model-checker constructs, online testing setup can also be applied to other model-checker.

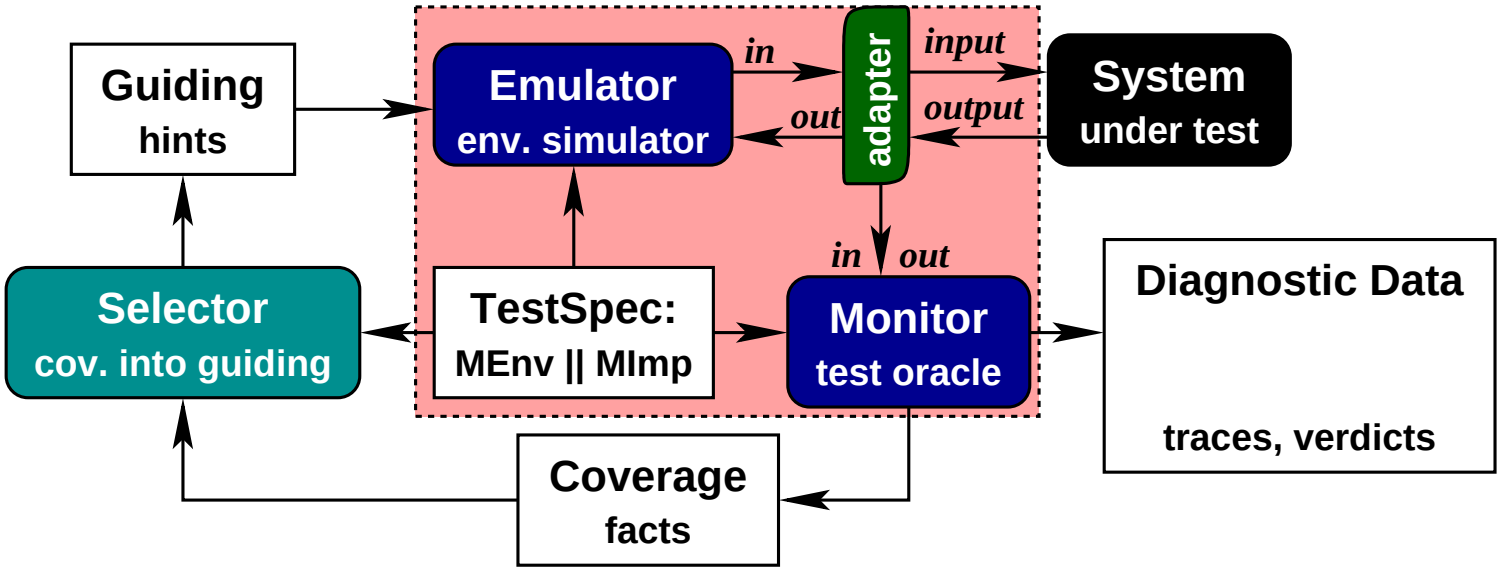
Summary and Future Work



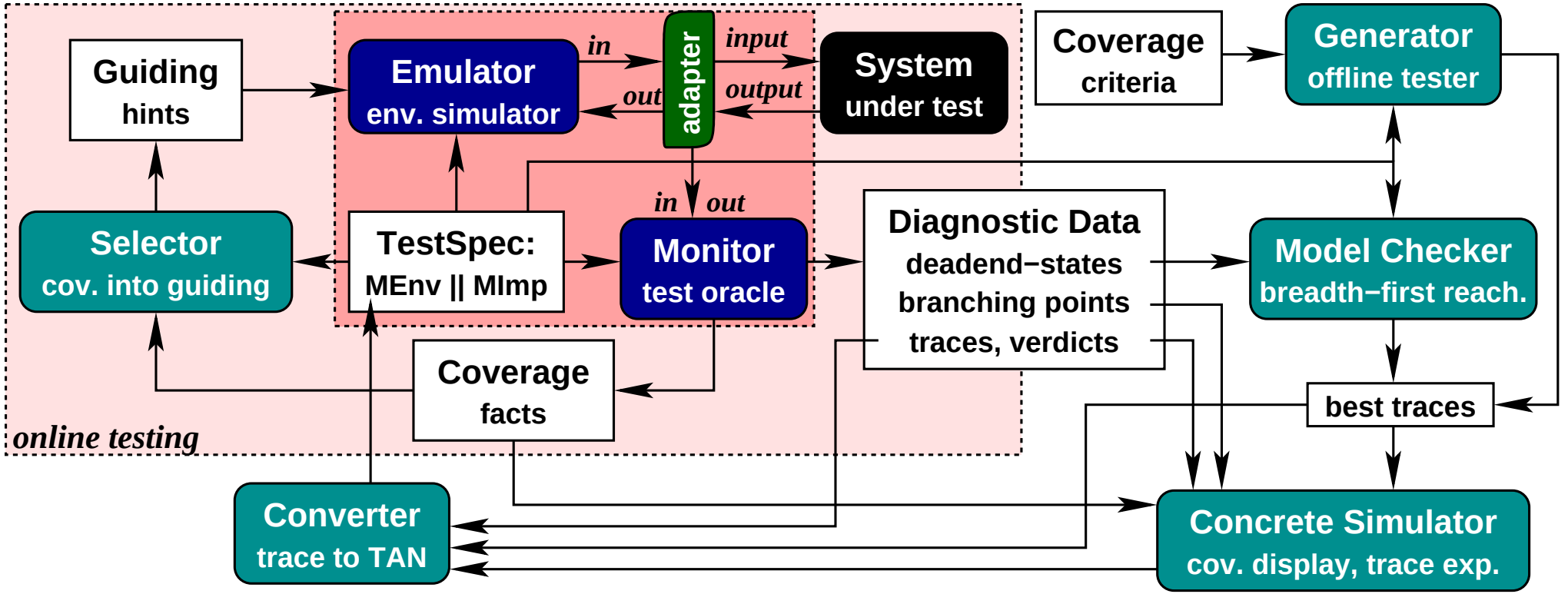
Summary and Future Work



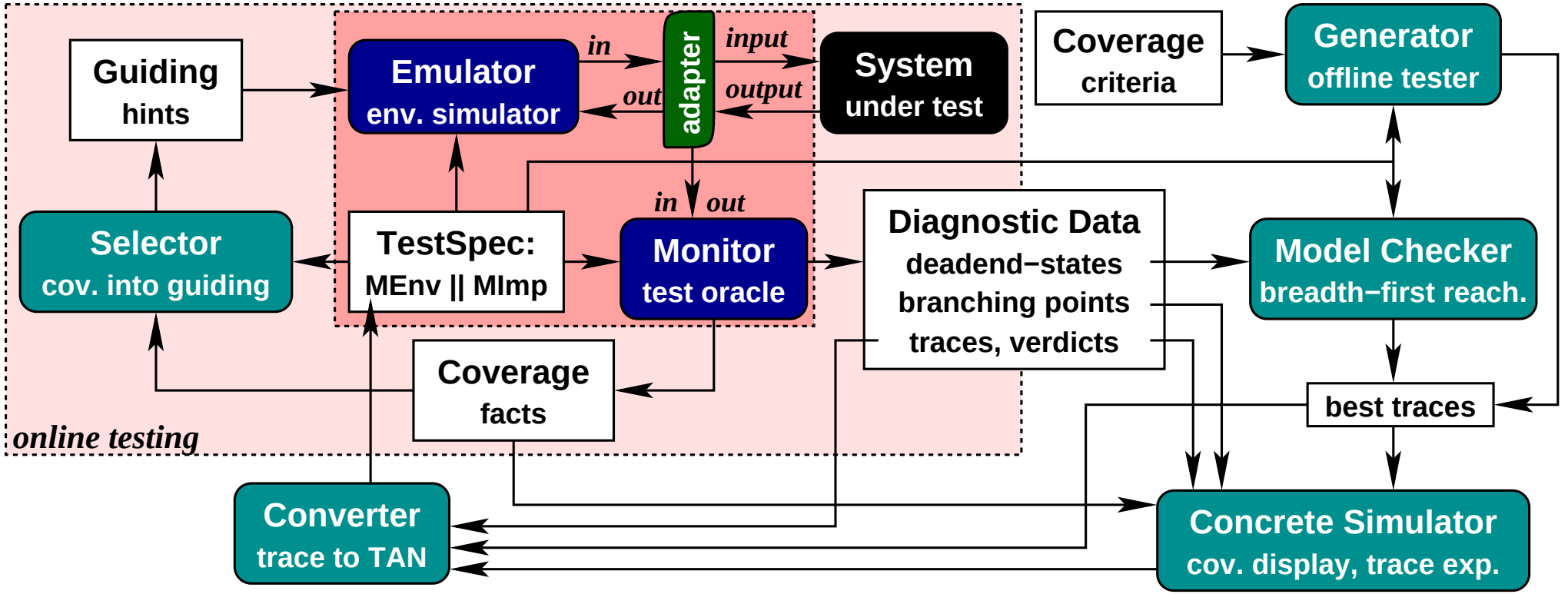
Summary and Future Work



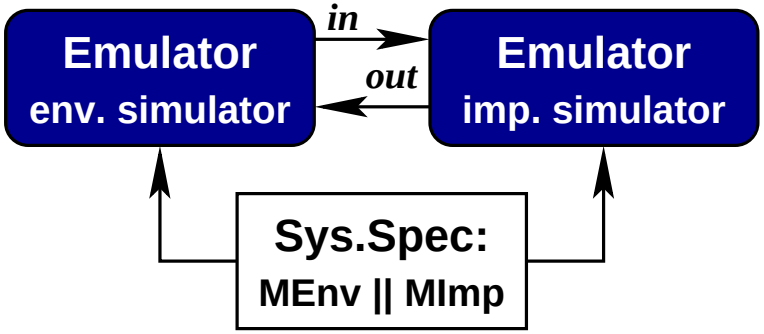
Summary and Future Work



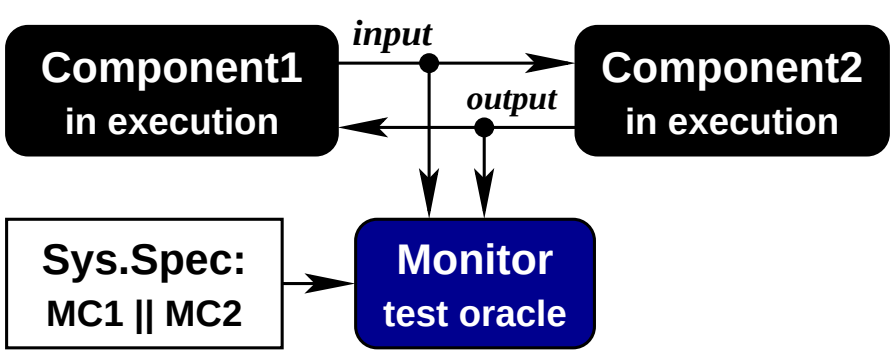
Summary and Future Work



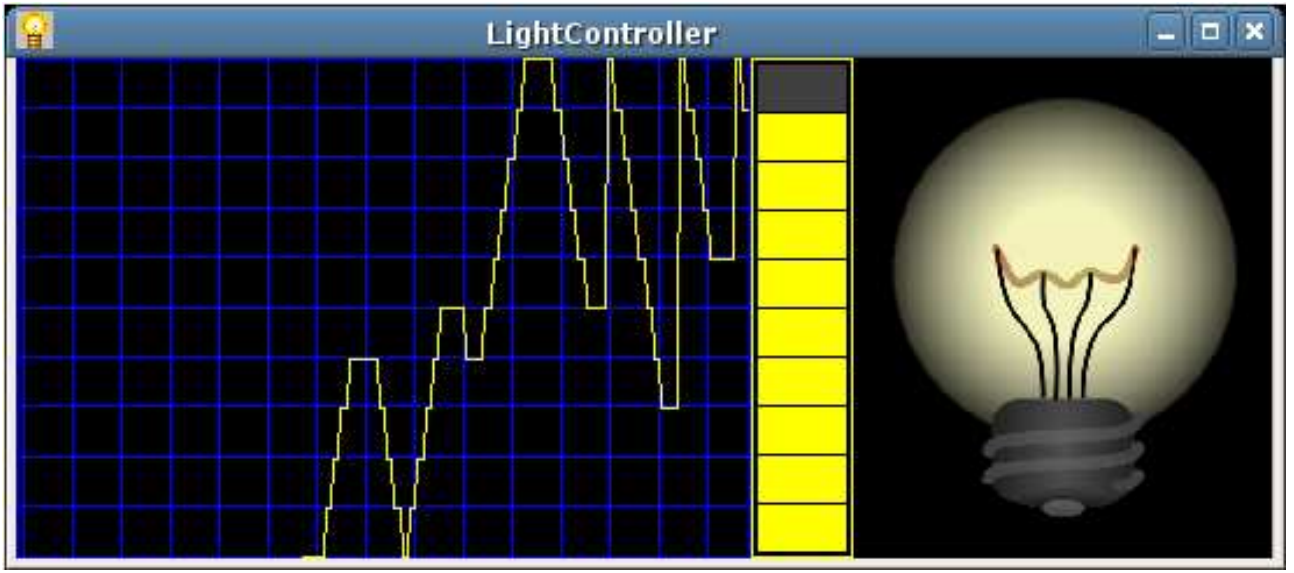
Prototyping via simulation:



Execution monitoring:



Demo of Java "Light Controller"



Supported Platforms

Compiled for the following operating systems:

- POSIX: Linux on Intel and Sun Solaris on Sparc
- Windows 2000/XP (and probably will work on others)

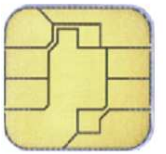
Currently available adapters:

- C dynamically linked library
- Text via standard input-output streams
- TCP/IP client-server sockets:
  C/C++ and Java implementations

Danfoss Refrigeration Controller

Beyond timed automata...

- From EKC manual: controller calculates the current temperature about every second via: $T_n = 80\% \cdot T_{n-1} + 20\% \cdot T_{sensor}$
- New manual: sensor values are calibrated during deployment to increase the precision from $0.5°$ to $0.1°$ using new Pt sensors.
- Display temperature can be roughly described by:
  $\frac{dT}{dt} = 0.2 \cdot (T_s - T) \qquad \Rightarrow \qquad T(t) = T_0 + \alpha_{T_s} \cdot e^{-\beta_{T_s} \cdot t}$
- The display temperature update interval varies and sampling is performed on a soft-real-time system $\Rightarrow$ difficult to model accurately (in timed automata with integers).
- PHAVer is more natural in modeling the dynamic behavior:
  $T_{sensor} > T: \quad 0.15 \cdot (T_{sensor} - T) \leq T' \leq 0.6 \cdot (T_{sensor} - T)$
  $T_{sensor} < T: \quad 0.6 \cdot (T_{sensor} - T) \leq T' \leq 0.15 \cdot (T_{sensor} - T)$

Monitoring EKC as HS using PHAVer