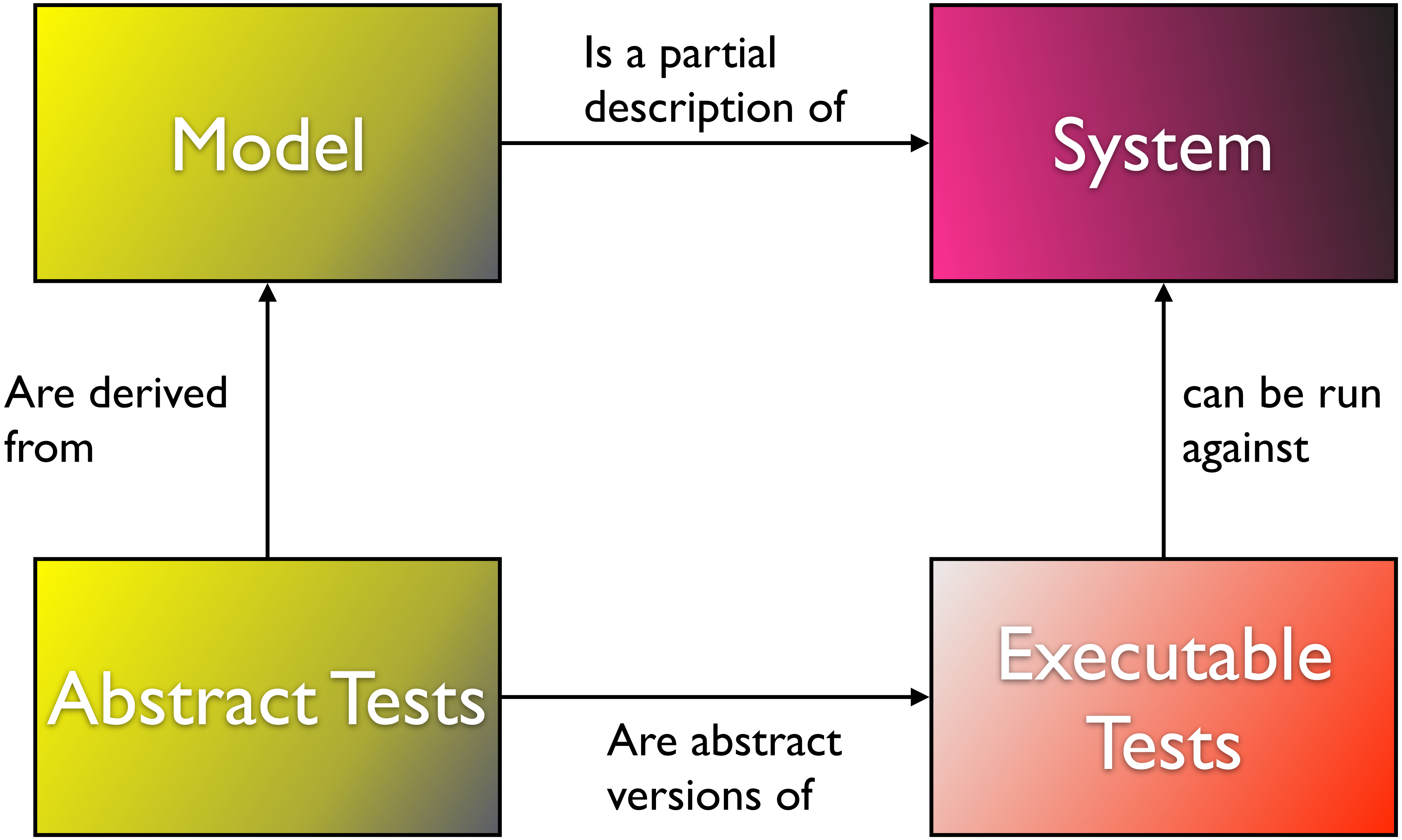
Property Checking of Safety-Critical Systems – Mathematical Foundations and Concrete Algorithms

Wen-ling Huang and Jan Peleska
University of Bremen
{huang,jp}@cs.uni-bremen.de

MBT-Paradigm



Model

Is a partial description of

System

Are derived from

can be run against

Abstract Tests

Are abstract versions of

Executable Tests

Motivation

- Two ways to verify the correctness of a model or of an implementation
  - **Conformance testing** – check whether the implementation conforms to (= „is just as good“) as the reference model
  - **Property testing** – check whether the implementation fulfils a given logical property, for example, a safety requirement

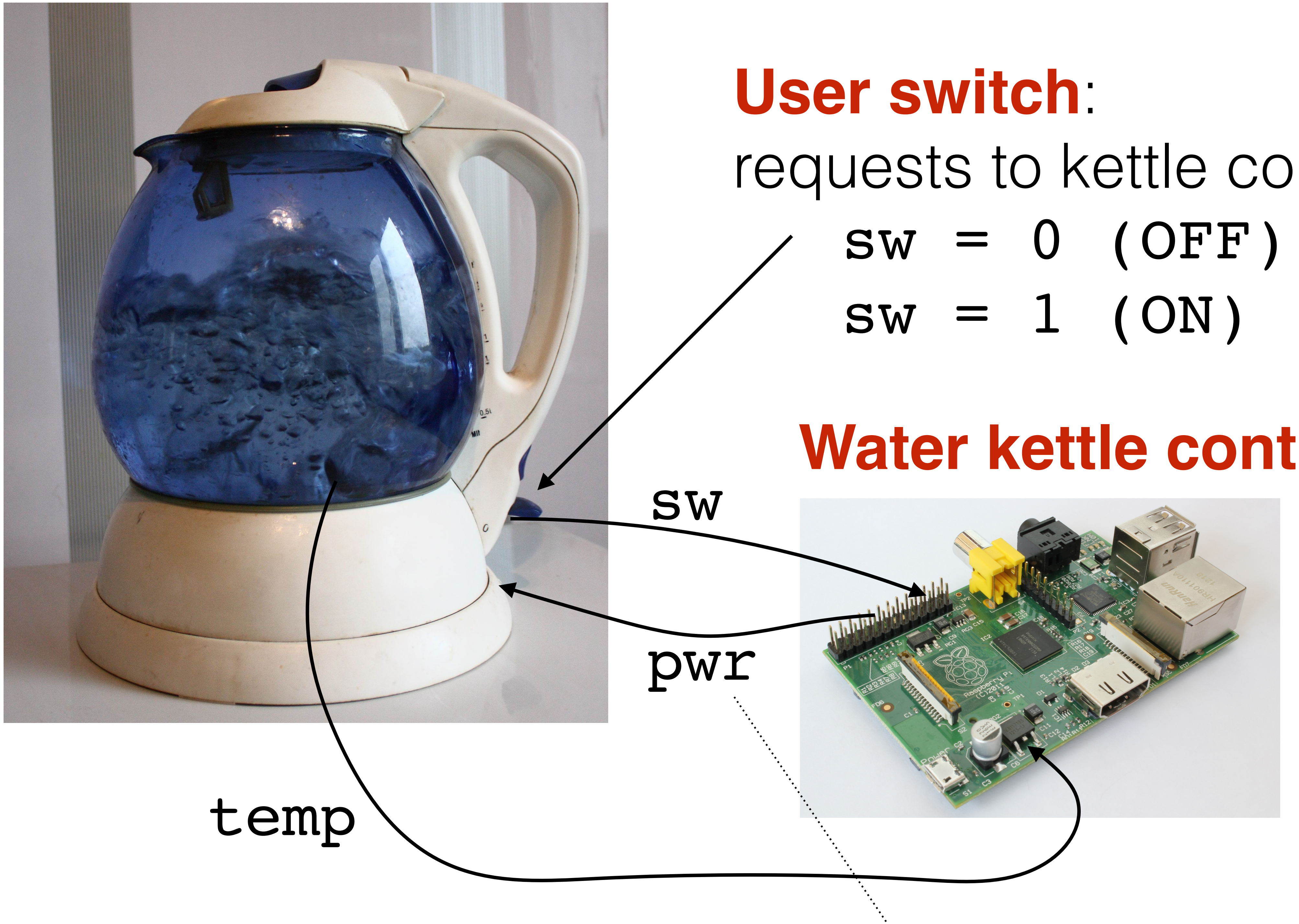
Motivation

- For conformance testing, we already know that the complete testing theories we have introduced before allow to prove equivalence (or reduction) between a model and an implementation
- In this seminar, we show how the same theories can be applied to property testing

Example – Electronic Water Kettle

- A water kettle with integrated controller
- Requirements for the controller (see diagram next slide)

  1. When water boils (**`temp = 100`**), switch power off automatically (**`pwr := 0`**)
  2. When water is not boiling (**`temp < 100`**), switch power on (**`pwr := 1`**) if and only if user switch is in position 1 (**`sw = 1`**)



User switch:
requests to kettle controller

```
sw = 0 (OFF)
sw = 1 (ON)
```

Water kettle controller

Temperature

```
temp < 100 (not boiling)
temp = 100 (boiling)
```

Power control switch

```
pwr = 0 (OFF)
pwr = 1 (ON)
```

Overview

1. How Properties are defined

2. Expressing properties of control systems with **Linear Temporal Logic (LTL)**

3. Safety Properties

4. Maximally nondeterministic state machines representing LTL safety properties

5. Applying complete testing theories for checking LTL safety properties

How Properties are Defined

- **Executions of non-terminating systems** can be described by infinite sequences of sets of atomic propositions
  - **Atomic proposition.** A basic fact that cannot be divided logically into smaller facts
  - The set of atomic propositions that are relevant for the system being analysed is denoted by *AP*
  - In an execution state s , the set of facts from *AP* that hold in *s* is denoted by the subset *L(s)* of *AP*
  - This is interpreted in the sense that also the ***negation*** of all atomic propositions from *(AP – L(s))* holds in *s*

$$(2^{AP})^{\omega}$$

denotes the set of all infinite sequences over sets of atomic propositions

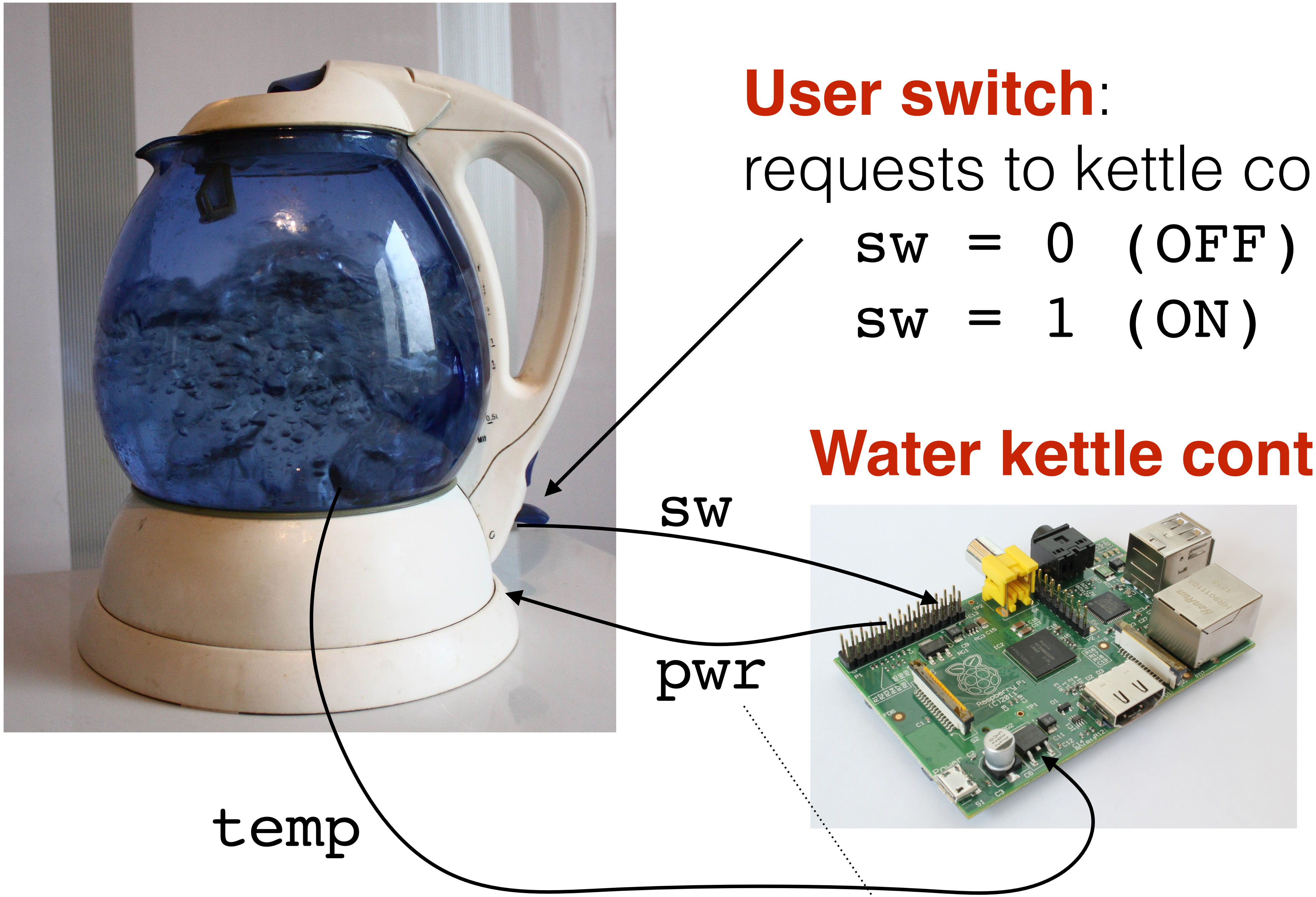
Example – Electronic Water Kettle

- A water kettle with integrated controller
- Requirements for the controller (see diagram next slide)

1. When water boils (**`temp = 100`**), switch power off automatically (**`pwr := 0`**)
2. When water is not boiling (**`temp < 100`**), switch power on (**`pwr := 1`**) if and only if user switch is in position 1 (**`sw = 1`**)

User switch:
requests to kettle controller

```
sw = 0 (OFF)
sw = 1 (ON)
```



Water kettle controller

Temperature

```
temp < 100 (not boiling)
temp = 100 (boiling)
```

Power control switch

```
pwr = 0 (OFF)
pwr = 1 (ON)
```

How Properties are Defined

- Atomic propositions can represent **abstractions of facts** derived from physical observables or from program variables

$AP = \{o, p, h\}$

Atomic proposition	**Meaning**	**Variable Condition**
o	User has switched kettle on	$\texttt{sw} = 1$
p	Controller has switched power on	$\texttt{pwr} = 1$
h	Water is boiling (hot!)	$\texttt{temp} = 100$

How Properties are Defined

- A **(linear time) property P** is just a subset of the set of all infinite sequences over sets of atomic propositions

$$P \subseteq (2^{AP})^{\omega}$$

- A **system fulfils a property P** if and only if every possible system execution is an element of *P*

- We also say in this case that the **system is a model for P**

$$\mathrm{SYSTEM} \models P$$

How Properties are Defined

- In this presentation, we are mainly interested in **safety properties**

$P \subseteq (2^{AP})^{\omega}$ is a safety property if and only if

$$\forall \pi \in (2^{AP})^{\omega} - P : \exists \pi' \text{ prefix of } \pi : P \cap \{\pi'.\pi'' \mid \pi'' \in (2^{AP})^{\omega}\} = \varnothing$$

- Safety violations can always be **detected on a finite prefix** π' of observations

- Safety violations **can never be "undone"**: regardless of how you continue after π', the execution will never become safe again

Example. The electric water kettle controller should never be powered when the water is already boiling – this can be specified by the safety property

$$\begin{aligned} AP &= \{o, p, h\} \\ P &= \{\pi \in (2^{AP})^\omega \mid \forall i \geq 0 : \{p, h\} \not\subseteq \pi(i)\} \end{aligned}$$

A typical execution which is in P would look like

$$\{\} \ldots \{\}.\{o\}.\{o, p\} \ldots \{o, p\}.\{o, h\}.\{o, h\}.\{h\} \ldots \{h\}.\{\} \ldots$$

Linear Temporal Logic

- LTL is introduced to specify linear time properties without having them to enumerate explicitly as sets of infinite sequences of sets of atomic propositions
- LTL introduces additional **operators for expressing causal aspects** of system behaviour („When property 1 becomes true, it is guaranteed that property 2 becomes true ‚a little later'")

Linear Temporal Logic

LTL formulas are constructed from the following elements

- atomic **propositions** from some set AP
- **logical** operators $\wedge, \neg$
- **temporal** operators X (next) and W (weak until)
  - $\mathsf{X}\varphi$ states that φ holds in the next state
  - $\varphi \mathsf{W} \psi$ states that φ holds at least until some state where ψ holds (or forever, if no such state exists)

Linear Temporal Logic

LTL Syntax Rules.

Let $AP = \{a, b, c\}$ be a set of atomic propositions; then

1. The Boolean constants `true`. `false` are valid LTL formulas
2. Every $p \in AP$ is a valid LTL formula
3. If φ is a valid LTL formula, then $\neg\varphi$ is a valid LTL formula
4. If φ and ψ are valid LTL formulas, then $(\varphi \wedge \psi)$ is a valid LTL formula
5. φ is a valid LTL formula, then $\mathsf{X}\varphi$ is a valid LTL formula
6. If φ and ψ are valid LTL formulas, then $(\varphi \mathsf{W} \psi)$ is a valid LTL formula

Linear Temporal Logic

- **Models for LTL formulas**
  - On an abstract level, models are sequences of sets of atomic propositions

$$\pi \in (2^{AP})^\omega \qquad \varphi \in \mathrm{LTL} \qquad \pi \overset{?}{\models} \varphi$$

  - π represents a sequence of observations about an executing system
  - $\pi(i)$ is the set of propositions (= „facts“) that hold in execution state *i*
  - π^i is the path segment of observations starting at $\pi(i)$: $\pi^i = \pi(i).\pi(i+1).\pi(i+2)\ldots$

Linear Temporal Logic

1. $\pi^i \models \texttt{true},\ \forall i \geq 0$

2. $\pi^i \not\models \texttt{false},\ \forall i \geq 0$

3. $\pi^i \models p,\ \text{iff}\ p \in \pi(i)$

4. $\pi^i \models \neg\varphi,\ \text{iff}\ \pi^i \not\models \varphi$

5. $\pi^i \models \varphi \wedge \psi,\ \text{iff}\ (\pi^i \models \varphi)\ \text{and}\ (\pi^i \models \psi)$

6. $\pi^i \models \mathsf{X}\varphi,\ \text{iff}\ \pi^{i+1} \models \varphi$

7. $\pi^i \models \varphi \mathsf{W} \psi,\ \text{iff}\ \big(\forall k \geq i : \pi^k \models \varphi\big) \vee \big(\exists j \geq i : \pi^j \models \psi \wedge \forall i \leq k < j : \pi^k \models \varphi\big)$

Line

πⁱ denotes the path segment starting at π(i):

$\pi^i = \pi(i).\pi(i+1).\pi(i+2). . .$

1. $\pi^i \models \texttt{true},\ \forall i \geq 0$
2. $\pi^i \not\models \texttt{false},\ \forall i \geq 0$
3. $\pi^i \models p$, iff $p \in \pi(i)$
4. $\pi^i \models \neg\varphi$, iff $\pi^i \not\models \varphi$
5. $\pi^i \models \varphi \wedge \psi$, iff $(\pi^i \models \varphi)$ and $(\pi^i \models \psi)$
6. $\pi^i \models \mathsf{X}\varphi$, iff $\pi^{i+1} \models \varphi$
7. $\pi^i \models \varphi \mathsf{W} \psi$, iff $\big(\forall k \geq i : \pi^k \models \varphi\big) \vee \big(\exists j \geq i : \pi^j \models \psi \wedge \forall i \leq k < j : \pi^k \models \varphi\big)$

Linear Temporal Logic

Every path segment fulfils ‚true‘

1. $\pi^i \models \texttt{true},\ \forall i \geq 0$
2. $\pi^i \not\models \texttt{false},\ \forall i \geq 0$
3. $\pi^i \models p,\ \text{iff}\ p \in \pi(i)$
4. $\pi^i \models \neg\varphi,\ \text{iff}\ \pi^i \not\models \varphi$
5. $\pi^i \models \varphi \wedge \psi,\ \text{iff}\ (\pi^i \models \varphi)\ \text{and}\ (\pi^i \models \psi)$
6. $\pi^i \models \mathsf{X}\varphi,\ \text{iff}\ \pi^{i+1} \models \varphi$
7. $\pi^i \models \varphi \mathsf{W} \psi,\ \text{iff}\ \left(\forall k \geq i : \pi^k \models \varphi\right) \vee \left(\exists j \geq i : \pi^j \models \psi \wedge \forall i \leq k < j : \pi^k \models \varphi\right)$

Linear Temporal Logic

1. $\pi^i \models \texttt{true}, \ \forall i \geq 0$
2. $\pi^i \not\models \texttt{false}, \ \forall i \geq 0$
3. $\pi^i \models p, \ \text{iff } p \in \pi(i)$
4. $\pi^i \models \neg\varphi, \ \text{iff } \pi^i \not\models \varphi$
5. $\pi^i \models \varphi \wedge \psi, \ \text{iff } (\pi^i \models \varphi) \text{ and } (\pi^i \models \psi)$
6. $\pi^i \models \mathsf{X}\varphi, \ \text{iff } \pi^{i+1} \models \varphi$
7. $\pi^i \models \varphi \mathsf{W} \psi, \ \text{iff } \left(\forall k \geq i : \pi^k \models \varphi\right) \vee \left(\exists j \geq i : \pi^j \models \psi \wedge \forall i \leq k < j : \pi^k \models \varphi\right)$

No path segment fulfils ‚false‘

Linear Temporal Logic

1. $\pi^i \models \texttt{true},\ \forall i \geq 0$
2. $\pi^i \not\models \texttt{false},\ \forall i \geq 0$
3. $\pi^i \models p$, iff $p \in \pi(i)$
4. $\pi^i \models \neg\varphi$, iff $\pi^i \not\models \varphi$
5. $\pi^i \models \varphi \wedge \psi$, iff $(\pi^i \models \varphi)$ and $(\pi^i \models \psi)$
6. $\pi^i \models \mathsf{X}\varphi$, iff $\pi^{i+1} \models \varphi$
7. $\pi^i \models \varphi \mathsf{W} \psi$, iff $\big(\forall k \geq i : \pi^k \models \varphi\big) \vee \big(\exists j \geq i : \pi^j \models \psi \wedge \forall i \leq k < j : \pi^k \models \varphi\big)$

A path segment fulfils atomic proposition *p* iff *p* is contained in the first element *π(i)* of the segment

Linear Temporal Logic

1. $\pi^i \models \texttt{true}, \ \forall i \geq 0$
2. $\pi^i \not\models \texttt{false}, \ \forall i \geq 0$
3. $\pi^i \models p$, iff $p \in \pi(i)$
4. $\pi^i \models \neg\varphi$, iff $\pi^i \not\models \varphi$
5. $\pi^i \models \varphi \wedge \psi$, iff $(\pi^i \models \varphi)$ and $(\pi^i \models \psi)$
6. $\pi^i \models \mathsf{X}\varphi$, iff $\pi^{i+1} \models \varphi$
7. $\pi^i \models \varphi \mathsf{W} \psi$, iff $\big(\forall k \geq i : \pi^k \models \varphi\big) \vee \big(\exists j \geq i : \pi^j \models \psi \wedge \forall i \leq k < j : \pi^k \models \varphi\big)$

A path segment fulfils the negation of an LTL formula, iff it is ***not*** a model for the formula

Linear Temporal Logic

1. $\pi^i \models \texttt{true}, \ \forall i \geq 0$
2. $\pi^i \not\models \texttt{false}, \ \forall i \geq 0$
3. $\pi^i \models p$, iff $p \in \pi(i)$
4. $\pi^i \models \neg\varphi$, iff $\pi^i \not\models \varphi$
5. $\pi^i \models \varphi \wedge \psi$, iff $(\pi^i \models \varphi)$ and $(\pi^i \models \psi)$
6. $\pi^i \models \mathsf{X}\varphi$, iff $\pi^{i+1} \models \varphi$
7. $\pi^i \models \varphi \mathsf{W} \psi$, iff $\big(\forall k \geq i : \pi^k \models \varphi\big) \vee \big(\exists j \geq i : \pi^j \models \psi \wedge \forall i \leq k < j : \pi^k \models \varphi\big)$

A path segment fulfils (*phi* ∧ *psi*) if the segment is a model for both *phi* and *psi*

Linear Temporal Logic

1. $\pi^i \models \texttt{true},\ \forall i \geq 0$
2. $\pi^i \not\models \texttt{false},\ \forall i \geq 0$
3. $\pi^i \models p$, iff $p \in \pi(i)$
4. $\pi^i \models \neg\varphi$, iff $\pi^i \not\models \varphi$
5. $\pi^i \models \varphi \wedge \psi$, iff $(\pi^i \models \varphi)$ and $(\pi^i \models \psi)$
6. $\pi^i \models \mathsf{X}\varphi$, iff $\pi^{i+1} \models \varphi$
7. $\pi^i \models \varphi \mathsf{W} \psi$, iff $\left(\forall k \geq i : \pi^k \models \varphi\right) \vee \left(\exists j \geq i : \pi^j \models \psi \wedge \forall i \leq k < j : \pi^k \models \varphi\right)$

A path segment fulfils *X phi* if the segment starting with *π(i+1)* fulfils *phi*

Linear Temporal Logic

1. $\pi^i \models \texttt{true},\ \forall i \geq 0$
2. $\pi^i \not\models \texttt{false},\ \forall i \geq 0$
3. $\pi^i \models p,$ iff p
4. $\pi^i \models \neg\varphi,$ iff
5. $\pi^i \models \varphi \wedge \psi,$
6. $\pi^i \models \mathsf{X}\varphi,$ iff π $\models \varphi$
7. $\pi^i \models \varphi \mathsf{W} \psi,$ iff $(\forall k \geq i : \pi^k \models \varphi) \vee (\exists j \geq i : \pi^j \models \psi \wedge \forall i \leq k < j : \pi^k \models \varphi)$

A path segment fulfils (*ϕ W ψ)* if it

- fulfils *ϕ* in ***every*** segment *π^i^* , *π^i+1^*, *π^i+2^*, . . .
- or has a sub-segment *π^k^* fulfilling *ψ* and fulfils *ϕ* until *π^k^* is reached

Example.

$$\begin{aligned} AP &= \{a, b, c\} \\ \pi &= \pi(0).\pi(1).\pi(2).\pi(3)\ldots \\ &= \{a\}.\{b\}.\{a, b\}.\{c\}\ldots \end{aligned}$$

Prove that $\pi \models \big(\neg(\neg a \wedge \neg b)\big)\mathsf{W}c \qquad (*)$

Proof.

1. $\pi^3 \models c$ because $c \in \pi(3)$
2. Rules 3,4,5 imply that $\pi(0) = \{a\} \models \neg(\neg a \wedge \neg b)$
3. Rules 3,4,5 imply that $\pi(1) = \{b\} \models \neg(\neg a \wedge \neg b)$
4. Rules 3,4,5 imply that $\pi(2) = \{a, b\} \models \neg(\neg a \wedge \neg b)$
5. Rule 7 implies (*)

LTL Formulas and Properties

Since the models of LTL formulas are executions, we can associate each LTL formula in a natural way with a property: the set of all executions that are models for the formula

$$\text{Property} \quad P(\varphi) = \{\pi \in (2^{AP})^{\omega} \mid \pi \models \varphi\}$$

In [3] it has been proven that

Theorem. Every safety property *P* can be expressed by an **LTL safety formula** (defined on the next slide).

LTL Formulas and Properties

The safety properties originally defined by

$P \subseteq (2^{AP})^{\omega}$ is a safety property if and only if

$$\forall \pi \in (2^{AP})^{\omega} - P : \exists \pi' \text{ prefix of } \pi : P \cap \{\pi'.\pi'' \mid \pi'' \in (2^{AP})^{\omega}\} = \varnothing$$

can be characterised by **safety LTL** formulas defined as

1. Every propositional LTL formula (i.e. a formula without operators X, W) is a safety formula
2. If φ, ψ are safety formulas, then so are $\varphi \wedge \psi$, $\varphi \vee \psi$, $\mathsf{X}\varphi$, and $\varphi \; \mathsf{W} \; \psi$

Linear Temporal Logic

Additional “more intuitive” operators are introduced by syntactic equivalence

$\varphi \vee \psi$	$\equiv$	$\neg(\neg\varphi \wedge \neg\psi)$	Execution π fulfils φ or ψ
$\varphi \Rightarrow \psi$	$\equiv$	$\neg(\varphi \wedge \neg\psi)$	On execution π, φ implies ψ
$\mathsf{G}\varphi$	$\equiv$	$(\varphi\ \mathsf{W}\ \mathtt{false})$	φ holds in **every** state of the execution π
$\mathsf{F}\varphi$	$\equiv$	$(\neg\mathsf{G}\neg\varphi)$	**Finally** φ holds in some state of the execution π
$(\varphi\ \mathsf{U}\ \psi)$	$\equiv$	$(\varphi\ \mathsf{W}\ \psi) \wedge \mathsf{F}\psi$	**Finally** ψ holds in a state of π, and until then, φ holds

Linear Temporal Logic

Example. Consider again the safety property introduced for the electric water kettle

$$\begin{aligned} AP &= \{o, p, h\} \\ P &= \{\pi \in \left(2^{AP}\right)^{\omega} \mid \forall i \geq 0 : \{p, h\} \not\subseteq \pi(i)\} \end{aligned}$$

This can be expressed in LTL by the safety formula expressing an **invariant** (a proposition that holds in every state on a given execution path)

$$\varphi_P \equiv \mathsf{G}\big(\neg(p \wedge h)\big)$$

Controlled Atomic Propositions

- As in the water kettle example shown above, atomic propositions may be related to observables (e.g. program variables) whose value can be set by the control system under consideration. We call these atomic propositions **controlled.**

  - **Example.** The proposition p (`pwr = 1`) can be controlled by setting the `pwr`-output to 1 or 0. Propositions o (`sw = 1`) and h (`temp = 100`) are uncontrolled, because the controller has to accept the inputs `sw` and `tmp` generated by the users and the temperature sensor

Controlled Atomic Propositions

- Let *AP* be a set of atomic propositions over mixed input and output variables

- Suppose that the output variables have finite ranges (inputs may be infinite)

  Lemma. AP can be transformed into an equivalent set AP’ where all atomic propositions refer either to input variables or to output variables only

Controlled Atomic Propositions

Example. Consider an alternative set of atomic propositions for the water kettle

$AP' = \{o, p, z\}$

Atomic proposition	**Meaning**	**Variable Condition**
o	User has switched kettle on	$\texttt{sw} = 1$
p	Controller has switched power on	$\texttt{pwr} = 1$
z	Power switch state fulfils relation to temperature	$\texttt{temp} \leq 100 - \texttt{pwr}$

$AP' = \{o, p, z\}$

Atomic proposition	**Meaning**	**Variable Condition**
o	User has switched kettle on	$\texttt{sw} = 1$
p	Controller has switched power on	$\texttt{pwr} = 1$
z	Power switch state fulfils relation to temperature	$\texttt{temp} \leq 100 - \texttt{pwr}$

Now apply transformation formula for separation of input and output variables

$$\begin{aligned} z &\equiv \bigvee_{e \in \{0,1\}} \left(z[e/\mathrm{pwr}] \wedge \mathrm{pwr} = e \right) \\ &\equiv (\mathrm{temp} \leq 100 \wedge \mathrm{pwr} = 0) \vee (\mathrm{temp} \leq 99 \wedge \mathrm{pwr} = 1) \end{aligned}$$

This can be equivalently expressed in the original atomic propositions from AP as

$$z \equiv \neg p \vee \neg h$$

$AP' = \{o, p, z\}$

Atomic proposition	Meaning	Variable Condition
o	User has switched kettle on	$\texttt{sw} = 1$
p	Controller has switched power on	$\texttt{pwr} = 1$
z	Power switch state fulfils relation to temperature	$\texttt{temp} \leq 100 - \texttt{pwr}$

Exchange every occurrence of `pwr` in z by value *e*

Now apply transformation formula for separation of input and output variables

$$z \equiv \bigvee_{e \in \{0,1\}} (z[e/\mathrm{pwr}] \wedge \mathrm{pwr} = e)$$

$$\equiv (\mathrm{temp} \leq 100 \wedge \mathrm{pwr} = 0) \vee (\mathrm{temp} \leq 99 \wedge \mathrm{pwr} = 1)$$

This can be equivalently expressed in the original atomic propositions from AP as

$$z \equiv \neg p \vee \neg h$$

Safety Formulas and Finite State Machines

From the theory of generalised nondeterministic Buchi Automata [1] we can derive

Theorem. Every LTL safety formula can be expressed by a maximally nondeterministic FSM which fulfils this formula

This FSM is constructed according to the following specifications

1. Separate the atomic propositions into $AP = AP_I \cup AP_O$, such that the $p \in AP_I$ refer to input variables only, and the $p' \in AP_O$ to output variables only.

2. Define $\mathrm{FSM}(\varphi) = (Q, q_0, \Sigma_I, \Sigma_O, h)$ with

   (a) $\Sigma_I = 2^{AP_I}$

   (b) $\Sigma_O = 2^{AP_O}$

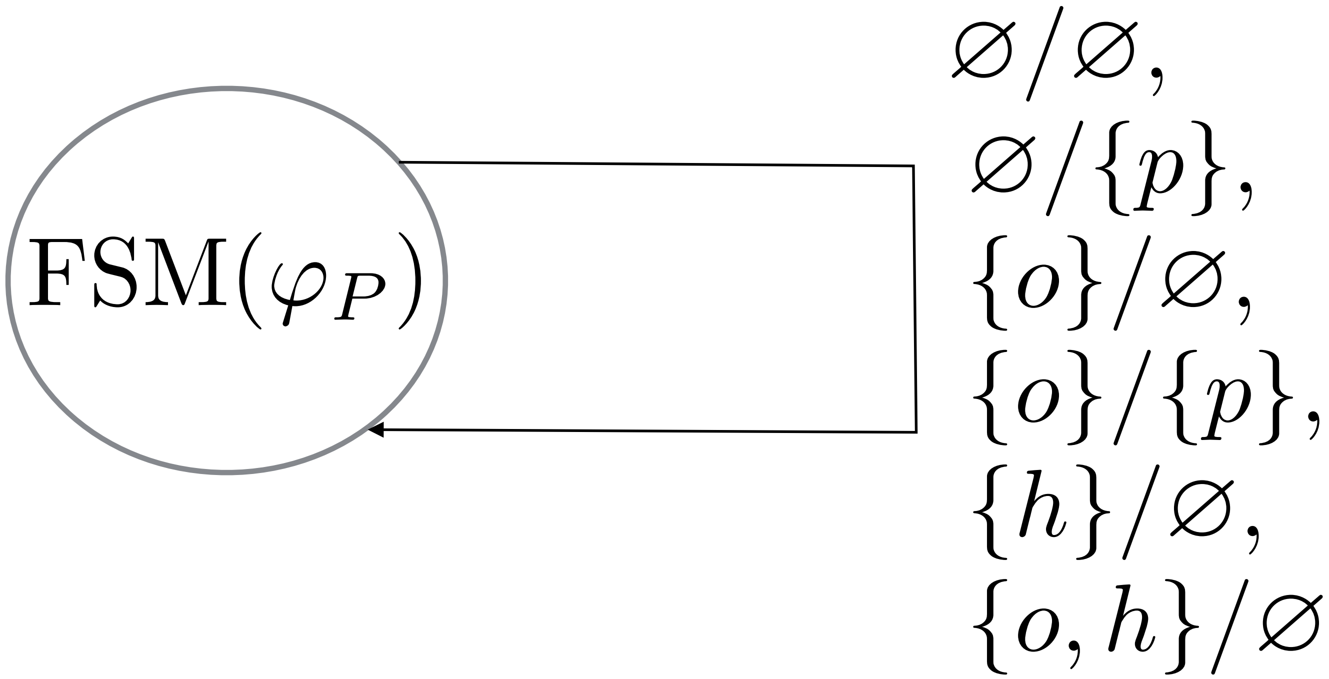
   (c) $L(\mathrm{FSM}(\varphi)) \models \varphi$

   (d) $\forall P \subseteq (2^{AP})^{\omega} : P \models \varphi \Rightarrow P \subseteq L(\mathrm{FSM}(\varphi))$

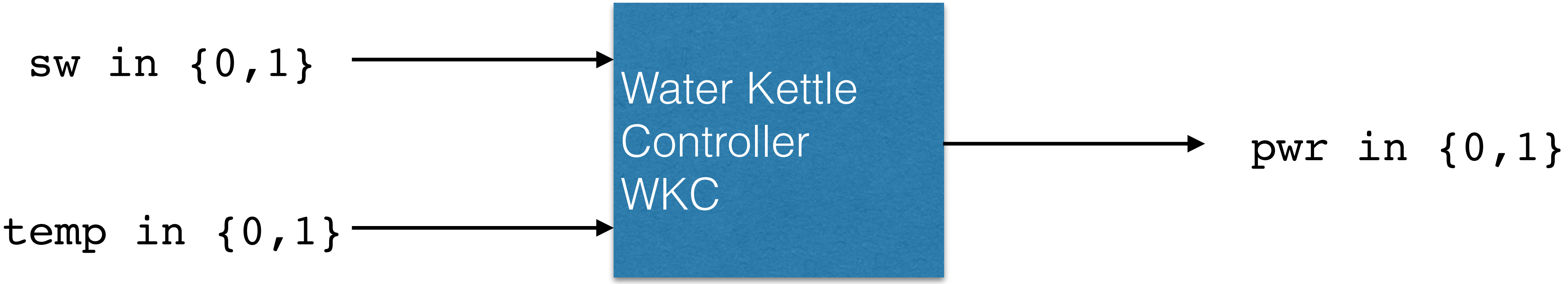
Example. The most nondeterministic FSM fulfilling

$$\varphi_P \equiv \mathsf{G}\big(\neg(p \wedge h)\big)$$

only has a single state, because it is a simple state invariant



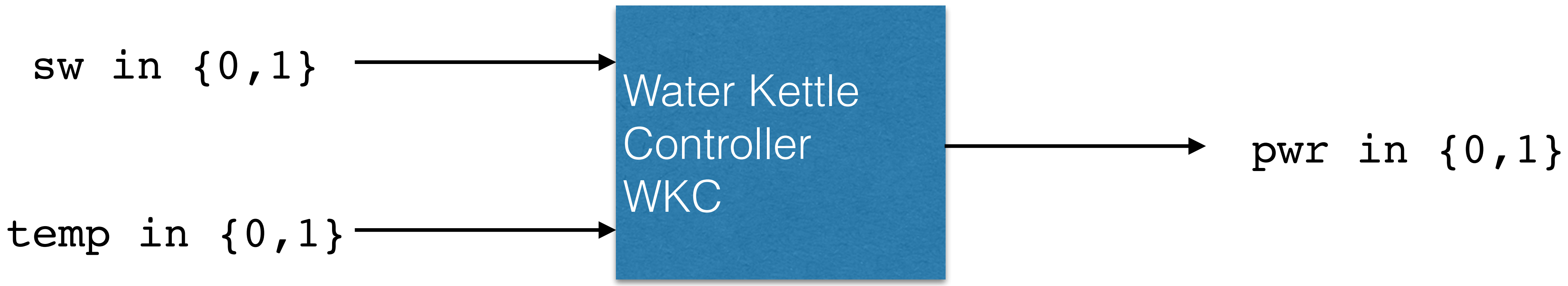
Property checking for an implementation of the water kettle controller



We assume that WKC is a member of the following fault domain for reactive I/O transition systems

$$
\begin{aligned}
\mathrm{WKC} &\in \mathcal{D}(m = 3, \mathcal{I}) \\
\mathcal{I} &= \{X_1, X_2, X_3, X_4\} \\
X_1 &= \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 0 \wedge \mathrm{temp} < 100\} = \{(\mathrm{sw}, \mathrm{temp}) \mid \neg o \wedge \neg h\} \\
X_2 &= \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 0 \wedge \mathrm{temp} = 100\} = \{(\mathrm{sw}, \mathrm{temp}) \mid \neg o \wedge h\} \\
X_3 &= \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 1 \wedge \mathrm{temp} < 100\} = \{(\mathrm{sw}, \mathrm{temp}) \mid o \wedge \neg h\} \\
X_4 &= \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 1 \wedge \mathrm{temp} = 100\} = \{(\mathrm{sw}, \mathrm{temp}) \mid o \wedge h\}
\end{aligned}
$$

Property checking for an implementation of the water kettle controller



We assume that WKC is a r

fault domain for reactive I/C

Recall:
o is equivalent to *(sw = 1)*
h is equivalent to *(temp = 100)*

$$
\begin{aligned}
\mathrm{WKC} &\in \mathcal{D}(m = 3, \mathcal{I}) \\
\mathcal{I} &= \{X_1, X_2, X_3, X_4\} \\
X_1 &= \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 0 \wedge \mathrm{temp} < 100\} = \{(\mathrm{sw}, \mathrm{temp}) \mid \neg o \wedge \neg h\} \\
X_2 &= \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 0 \wedge \mathrm{temp} = 100\} = \{(\mathrm{sw}, \mathrm{temp}) \mid \neg o \wedge h\} \\
X_3 &= \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 1 \wedge \mathrm{temp} < 100\} = \{(\mathrm{sw}, \mathrm{temp}) \mid o \wedge \neg h\} \\
X_4 &= \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 1 \wedge \mathrm{temp} = 100\} = \{(\mathrm{sw}, \mathrm{temp}) \mid o \wedge h\}
\end{aligned}
$$

Property checking for an implementation of the water kettle controller

From the previous lecture about input equivalence class testing we know that the WKC can be abstracted to a finite state machine *T(WKC)* with inputs $X_1,\ldots,X_4$ and outputs pwr := 0 and pwr := 1

By changing the notation for the input and output alphabets, we can alternatively write the labels as

$$\begin{aligned}
&\xrightarrow{\varnothing/\varnothing} \text{ for } \xrightarrow{X_1/\text{pwr:=}0} &\qquad& \xrightarrow{\varnothing/\{p\}} \text{ for } \xrightarrow{X_1/\text{pwr:=}1}\\
&\xrightarrow{\{h\}/\varnothing} \text{ for } \xrightarrow{X_2/\text{pwr:=}0} &\qquad& \xrightarrow{\{h\}/\{p\}} \text{ for } \xrightarrow{X_2/\text{pwr:=}1}\\
&\xrightarrow{\{o\}/\varnothing} \text{ for } \xrightarrow{X_3/\text{pwr:=}0} &\qquad& \xrightarrow{\{o\}/\{p\}} \text{ for } \xrightarrow{X_3/\text{pwr:=}1}\\
&\xrightarrow{\{o,h\}/\varnothing} \text{ for } \xrightarrow{X_4/\text{pwr:=}0} &\qquad& \xrightarrow{\{o,h\}/\{p\}} \text{ for } \xrightarrow{X_4/\text{pwr:=}1}
\end{aligned}$$

Theorem. Let $\mathcal{S}'$ be a reactive I/O transition system with input equivalence classes $X_1, \ldots, X_n$ and output values $y \in \{e_1, \ldots, e_k\}$. Suppose that the input classes have defining propositions $\psi_1, \ldots, \psi_n$.
Let φ be a safety LTL formula with propositions from the set

$$\{\psi_1, \ldots, \psi_n, y = e_1, \ldots, y = e_k\}$$

Then

$$\mathcal{S}' \models \varphi \Leftrightarrow T(\mathcal{S}') \models \varphi$$

$$\begin{tikzcd}
T(Sig_1) \arrow[r, "\models"] & \text{SafetyLTL} \arrow[d, "id"] \\
Sig_1 \arrow[u, "T"] \arrow[r, "\models"] & \text{SafetyLTL}
\end{tikzcd}$$

Proof sketch.

1. Suppose that $\mathcal{S}' \models \varphi$.

2. Then, for every execution $\pi = (x_1/y_1).(x_2/y_2)\cdots \in L(\mathcal{S}')$, $\pi \models \varphi$ holds.

3. Every $\pi = (x_1/y_1).(x_2/y_2)\dots$ is abstracted to a sequence $[\pi] = (X_1/y_1).(X_2/y_2)\dots$, such that $x_i \in X_i$ for all $i = 1, 2, \dots$

4. We can write this equivalently as sequences of formulas $[\pi] = (\psi_1/y = y_1).(\psi_2/y = y_2)\dots$, where ψ_i is the defining formula of input equivalence class X_i

5. By construction of T, the FSM $T(\mathcal{S}')$ also executes $[\pi] = (\psi_1/y = y_1).(\psi_2/y = y_2)\dots$

6. Since φ is a formula with propositions in $\{\psi_1, \dots, \psi_n, y = e_1, \dots, y = e_k\}$, this formula can be evaluated on $[\pi]$ and holds there, if and only if it holds on $\pi = (x_1/y_1).(x_2/y_2)\dots$

7. Since $\mathcal{S}$ and $T(\mathcal{S}')$ perform exactly the same $[\pi]$, this concludes the proof.

Safety Formulas and Refinement of the Input Equivalence Class Partitions

- What do we do if we wish to check whether the system under test satisfies a safety formula whose propositions do not come from the input classes?
  - Just refine the input equivalence class partition, so that the resulting new propositions allow to define the formula
- What do we do if the formula contains atomic propositions referring to output variables, but not with equality?
  - Just re-define the formula with propositions using conjunctions and disjunctions of atoms involving equality only

Example. Suppose we wish to test whether the WKC fulfils formula

$$\mathsf{G}\big(\mathrm{temp} < 50 \wedge \mathrm{on} \implies \mathrm{pwr} > 0\big)$$

This can be equivalently transformed to

$$\mathsf{G}\big(\mathrm{temp} < 50 \wedge \mathrm{on} \implies \mathrm{pwr} = 1\big)$$

and we can refine the input partitioning to

$$\begin{array}{rcl} X_{11} & = & \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 0 \wedge \mathrm{temp} < 50\} \\ X_{12} & = & \{(\mathrm{sw}, \mathrm{temp}) \mid \mathrm{sw} = 0 \wedge 50 \leq \mathrm{temp} < 100\} \\ \ldots & & \end{array}$$

- Now we can consider the nondeterministic $\mathrm{FSM}(\varphi_P)$ developed for representing the safety formula as the **reference model** for the FSM abstraction *FSM(WKC)* of the water kettle controller

- New: Since $\mathrm{FSM}(\varphi_P)$ is the maximally nondeterministic FSM satisfying the safety formula, it performs actions that are not wanted for the real implementation

- Solution: we do not test for equivalence, but we test for **reduction**, applying again a complete test suite for that purpose

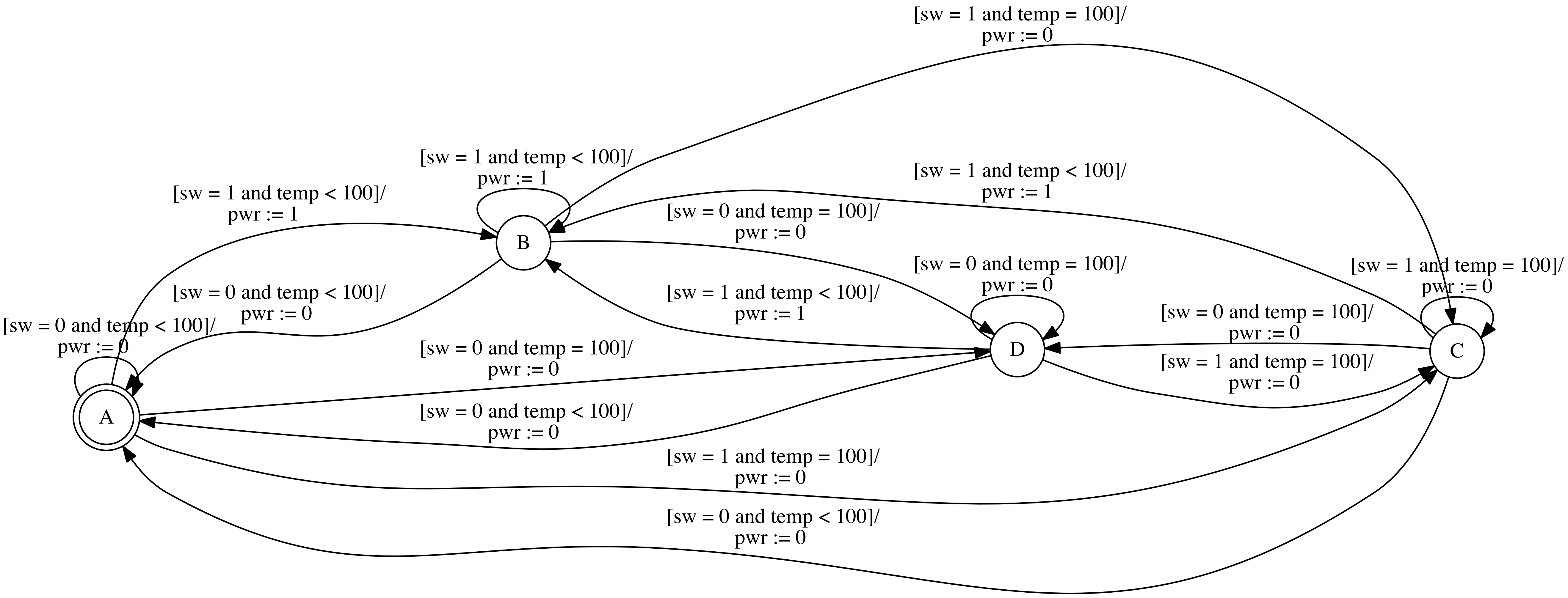
- If the tests are all passed by the SUT, this means that its language is a sub-language of $\mathrm{FSM}(\varphi_P)$, and therefore the *FSM(WKC)* fulfils the safety formula

- Since the safety formula is expressed in *o, h, p*, and these atomic propositions are directly represented in the input equivalence class partitioning and the finite outputs of the WKC, every WKC-execution can be abstracted to an execution in $(2^{\{o,h,p\}})^\omega$

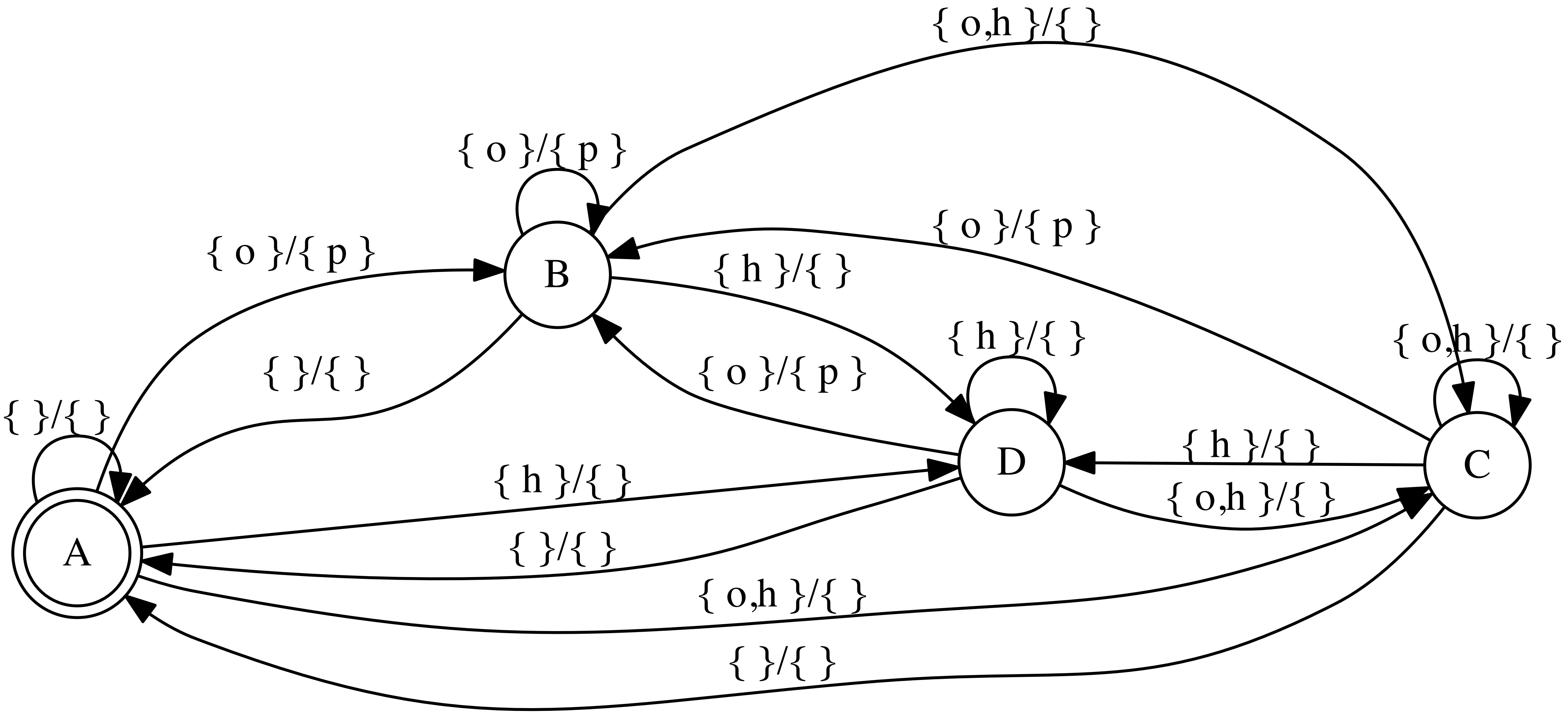
- By construction of *T*, *WKC* and *FSM(WKC)* execute exactly the same sequences in $(2^{\{o,h,p\}})^\omega$

- Therefore, since FSM(WKC) only performs executions fulfilling the formula φ_P, the same holds for WKC

WKC implementing the electronic water kettle controller


[sw = 1 and temp = 100]/ pwr := 0
[sw = 1 and temp < 100]/ pwr := 1
[sw = 1 and temp < 100]/ pwr := 1
[sw = 1 and temp < 100]/ pwr := 1
[sw = 0 and temp = 100]/ pwr := 0
B
[sw = 0 and temp = 100]/ pwr := 0
[sw = 1 and temp = 100]/ pwr := 0
[sw = 0 and temp < 100]/ pwr := 0
[sw = 1 and temp < 100]/ pwr := 1
[sw = 0 and temp < 100]/ pwr := 0
[sw = 0 and temp = 100]/ pwr := 0
D
C
[sw = 0 and temp = 100]/ pwr := 0
[sw = 1 and temp = 100]/ pwr := 0
A
[sw = 0 and temp < 100]/ pwr := 0
[sw = 1 and temp = 100]/ pwr := 0
[sw = 0 and temp < 100]/ pwr := 0

FSM abstraction T(WKC)



Testing T(WKC) against $\mathrm{FSM}(\varphi_P)$ for reduction

- A simple complete test suite for checking whether a deterministic implementation is a reduction of a nondeterministic FSM reference model is as follows
  - **Test cases** use all input sequences of length *nm*, where
    - *n* is the number of states in the reference model
    - *m* is the maximal number of states in the implementation
    - A test case executed against the implementation is PASS, if and only if the outputs observed for the input sequence are **one possible output sequence** according to the reference model

Applying the test theory for the WKC implementation

1. $n = 1$ (number of states in $\mathrm{FSM}(\varphi_P)$)
2. Assume $m = 4$ (number of states in $T(\mathrm{WKC})$)
3. Need all input sequences of length 4 to construct the test cases
4. 4 possible inputs in each step: $\varnothing, \{o\}, \{h\}, \{o, h\}$
5. This results in $4^4 = 256$ test cases

Applying the test theory for the WKC implementation – Test case example

1. Input sequence $\{o\}.\{o\}.\{h\}.\{o\}$

2. Admissible outputs according to FSM(φ_P):

   (a) $\{\}.\{\}.\{\}.\{\}$

   (b) $\{\}.\{\}.\{\}.\{p\}$

   (c) $\{\}.\{p\}.\{\}.\{\}$

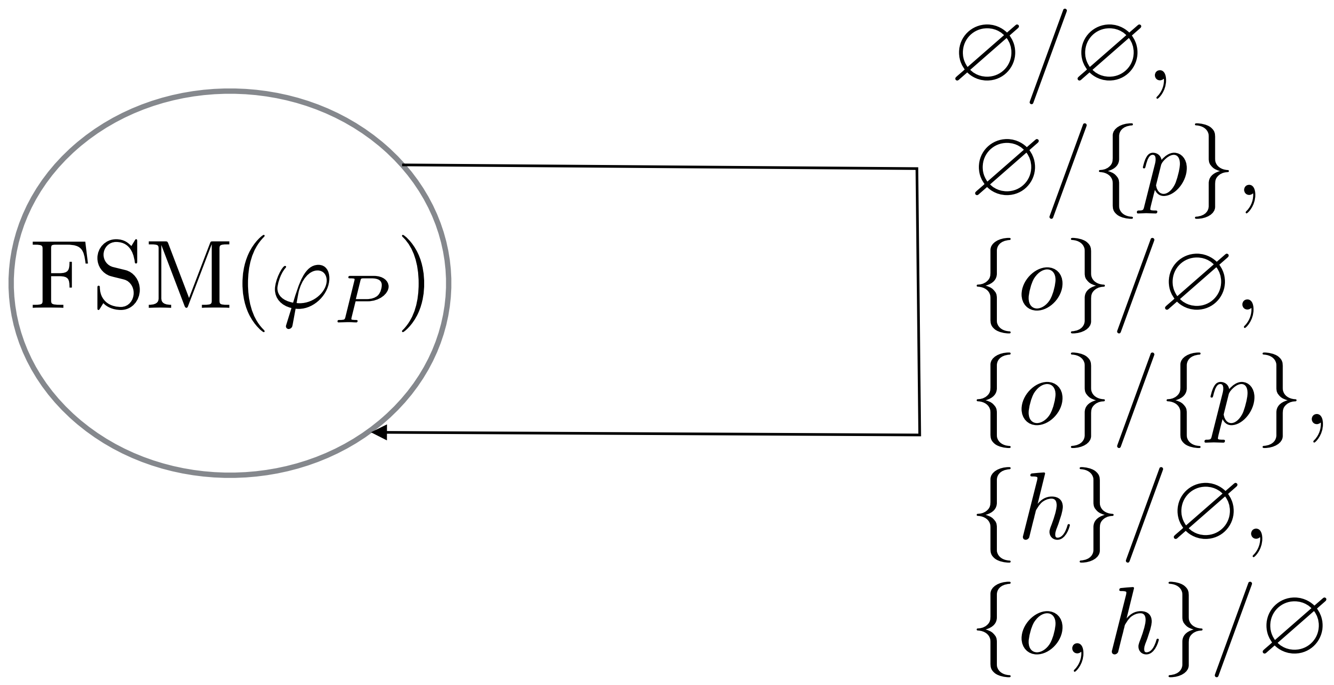
   (d) $\{\}.\{p\}.\{\}.\{p\}$

   (e) $\{p\}.\{\}.\{\}.\{\}$

   (f) $\{p\}.\{\}.\{\}.\{p\}$

   (g) $\{p\}.\{p\}.\{\}.\{\}$

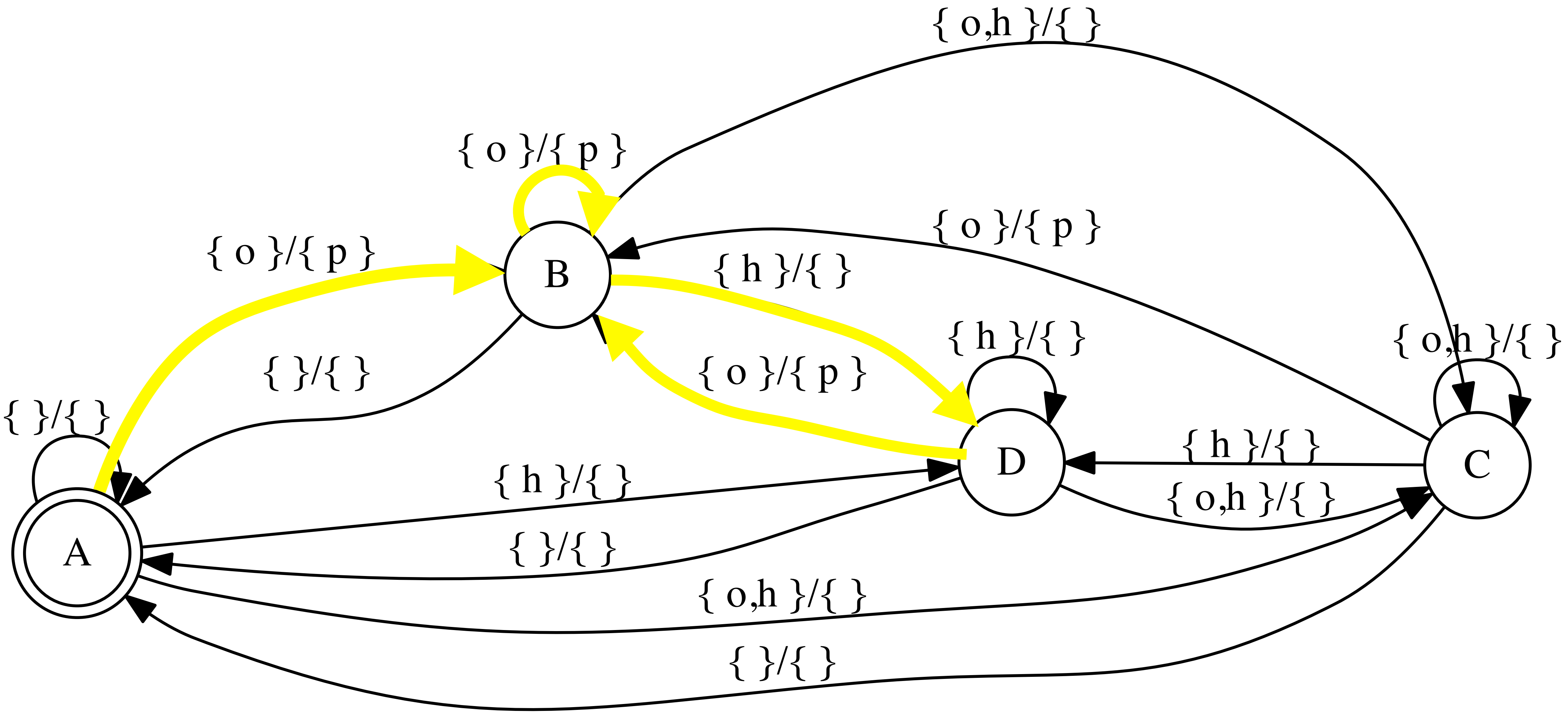
   (h) $\{p\}.\{p\}.\{\}.\{p\}$



3. T(WKC) produces outputs $\{p\}.\{p\}.\{\}.\{p\}$

4. Test case is PASSed

FSM abstraction T(WKC)



Further Reading

1. Christel Baier and Joost-Pieter Katoen. Principles of Model Checking. The MIT Press, Cambridge, Massachusetts, 2008.

2. Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. Model Checking. The MIT Press, Cambridge, Massachusetts, 1999.

3. A. Prasad Sistla. Safety, liveness and fairness in temporal logic. Formal Aspects of Computing, 6(5):495–511, September 1994.