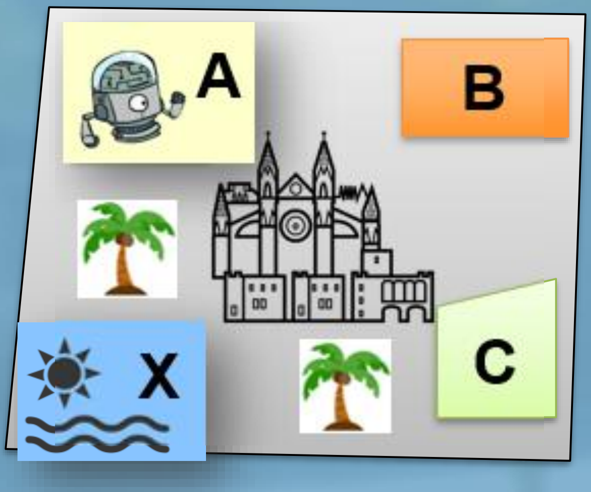
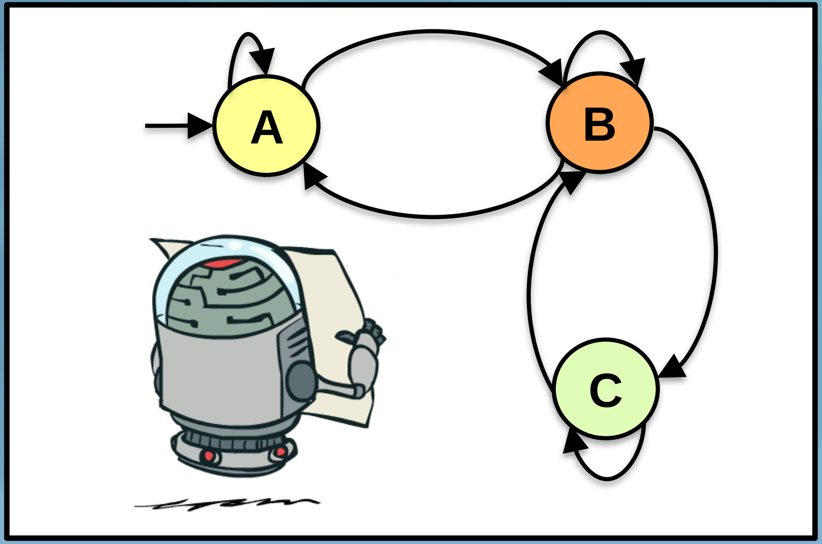
Graz University of Technology
Institute for Applied Information Processing and Communications

Temporal Logic – Part 2

Bettina Könighofer





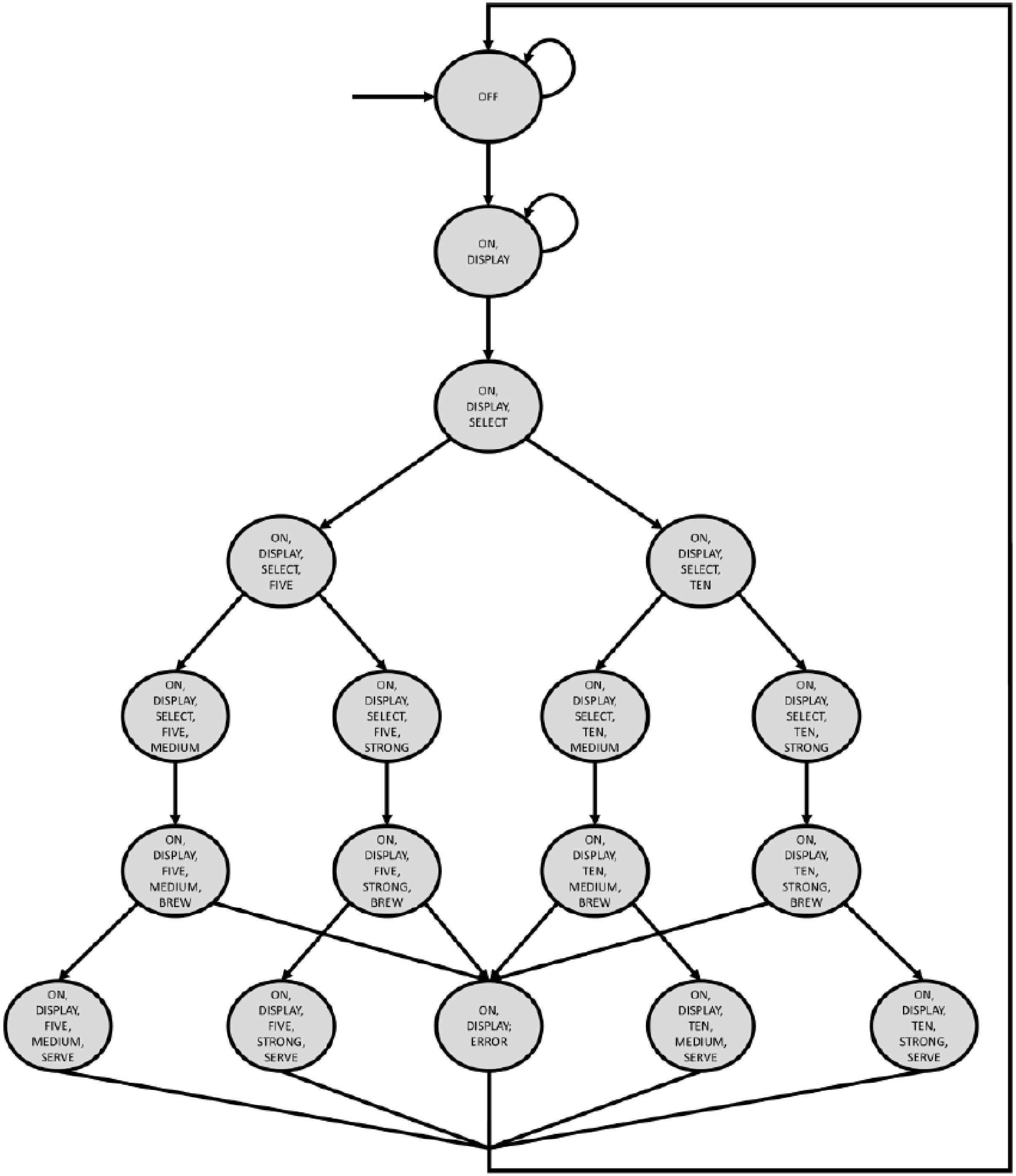
Model Checking SS21

April 29nd 2021

Solutions Homework
The Coffee-Machine-Verification-Problem

Given the following description of a coffee machine.

- The brewer serves **either five or ten cups** of coffee in **either medium or strong** flavour.
- Details:
  - The brewer is normally in the **off state** until it is **switched on**.
  - As long as the coffee machine is switched on, **the display is turned on**.
  - Once the brewer is switched on, the user can **select** the number of cups of coffee and the strength of coffee. The user can either select five or ten cups in either medium or strong flavour.
  - Once the selections have been made, the coffee machine starts the **brewing.**
  - During brewing, if any **error** is detected (say not enough coffee or no milk power), the brewer enters an error state.
  - Alternatively, the brewer is able to finish brewing and can **serve** the coffee
  - Finally, after serving or reaching an error, the coffee machine can be turned off to be eventually turned on again.


OFF
ON, DISPLAY
ON, DISPLAY, SELECT
ON, DISPLAY, SELECT, FIVE
ON, DISPLAY, SELECT, TEN
ON, DISPLAY, SELECT, FIVE, MEDIUM
ON, DISPLAY, SELECT, FIVE, STRONG
ON, DISPLAY, SELECT, TEN, MEDIUM
ON, DISPLAY, SELECT, TEN, STRONG
ON, DISPLAY, FIVE, MEDIUM, BREW
ON, DISPLAY, FIVE, STRONG, BREW
ON, DISPLAY, TEN, MEDIUM, BREW
ON, DISPLAY, TEN, STRONG, BREW
ON, DISPLAY, FIVE, MEDIUM, SERVE
ON, DISPLAY, FIVE, STRONG, SERVE
ON, DISPLAY; ERROR
ON, DISPLAY, TEN, MEDIUM, SERVE
ON, DISPLAY, TEN, STRONG, SERVE

LTL

State formulas:

- **A**f where f is a path formula

Path formulas:

- $p \in AP$
- $\neg f_1$, $f_1 \vee f_2$, $f_1 \wedge f_2$, $\mathbf{X} f_1$, $\mathbf{G} f_1$, $\mathbf{F} f_1$, $f_1 \mathbf{U} f_2$, $f_1 \mathbf{R} f_2$

  where f_1 and f_2 are path formulas

LTL is the set of all state formulas

CTL

CTL is the set of all state formulas, defined below (by means of state formulas only):

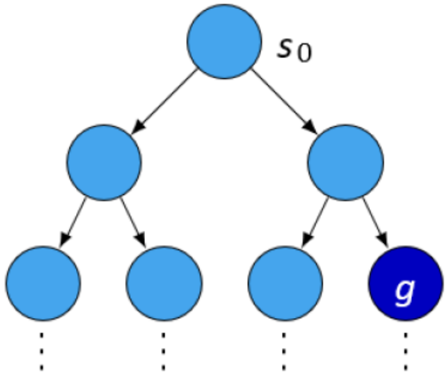
- $p \in AP$
- $\neg g_1$, $g_1 \vee g_2$, $g_1 \wedge g_2$
- **AX** g_1, **AG** g_1, **AF** g_1, **A** $(g_1$ **U** $g_2)$, **A** $(g_1$ **R** $g_2)$
- **EX** g_1, **EG** g_1, **EF** g_1, **E** $(g_1$ **U** $g_2)$, **E** $(g_1$ **R** $g_2)$

where g_1 and g_2 are state formulas

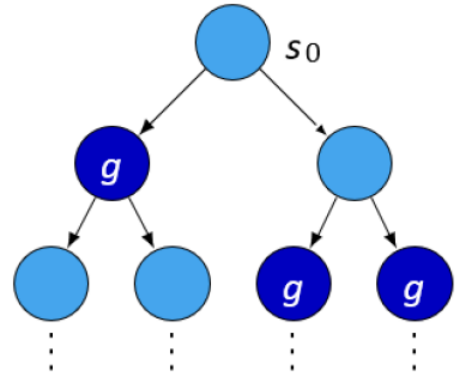
Illustration of CTL Semantics

EFg

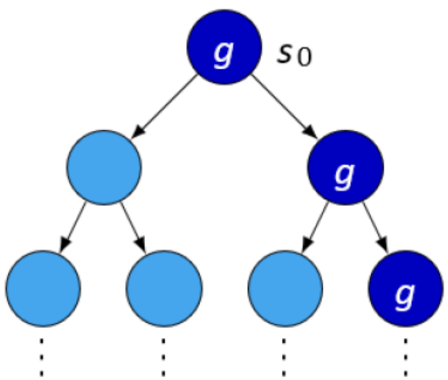
"exists reachable state such that…"



AFg

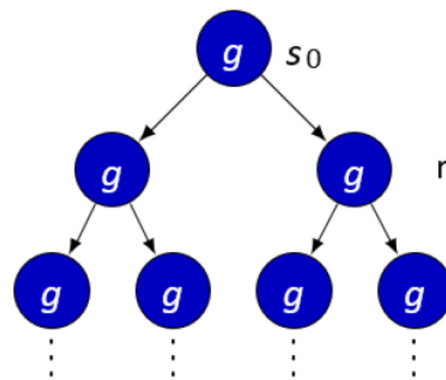


EGg



AGg

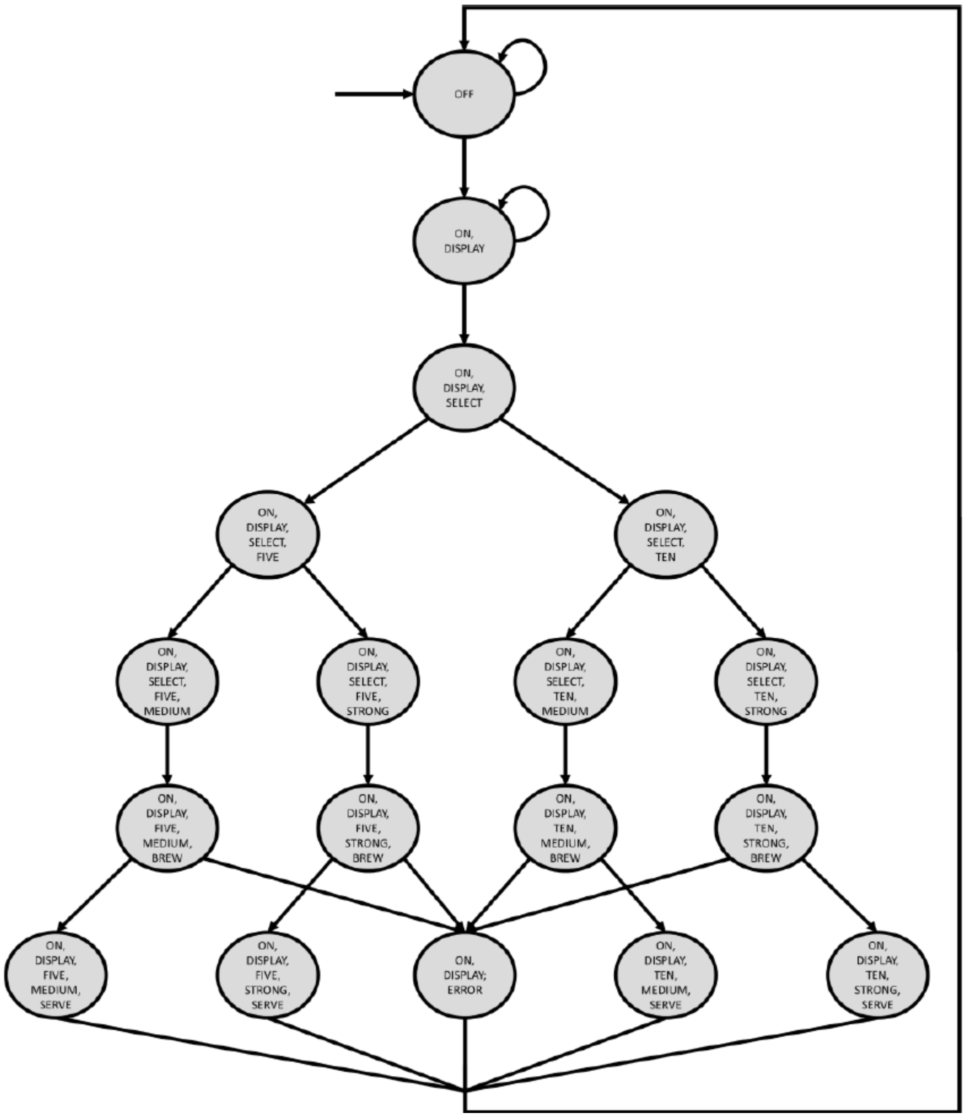
"all reachable states…"



Properties in LTL/CTL/CTL*

1 **From any state one can possibly eventually select ten cups of coffee and once selected, ten cups will always be served (or an error encountered) in the future.**

$$f_{1,CTL} = (AG\ EF\ coffee\_10) \wedge (AG(coffee\_10 \rightarrow AF(serve\_10\_medium \vee serve\_10\_strong \vee error)))$$



Yes

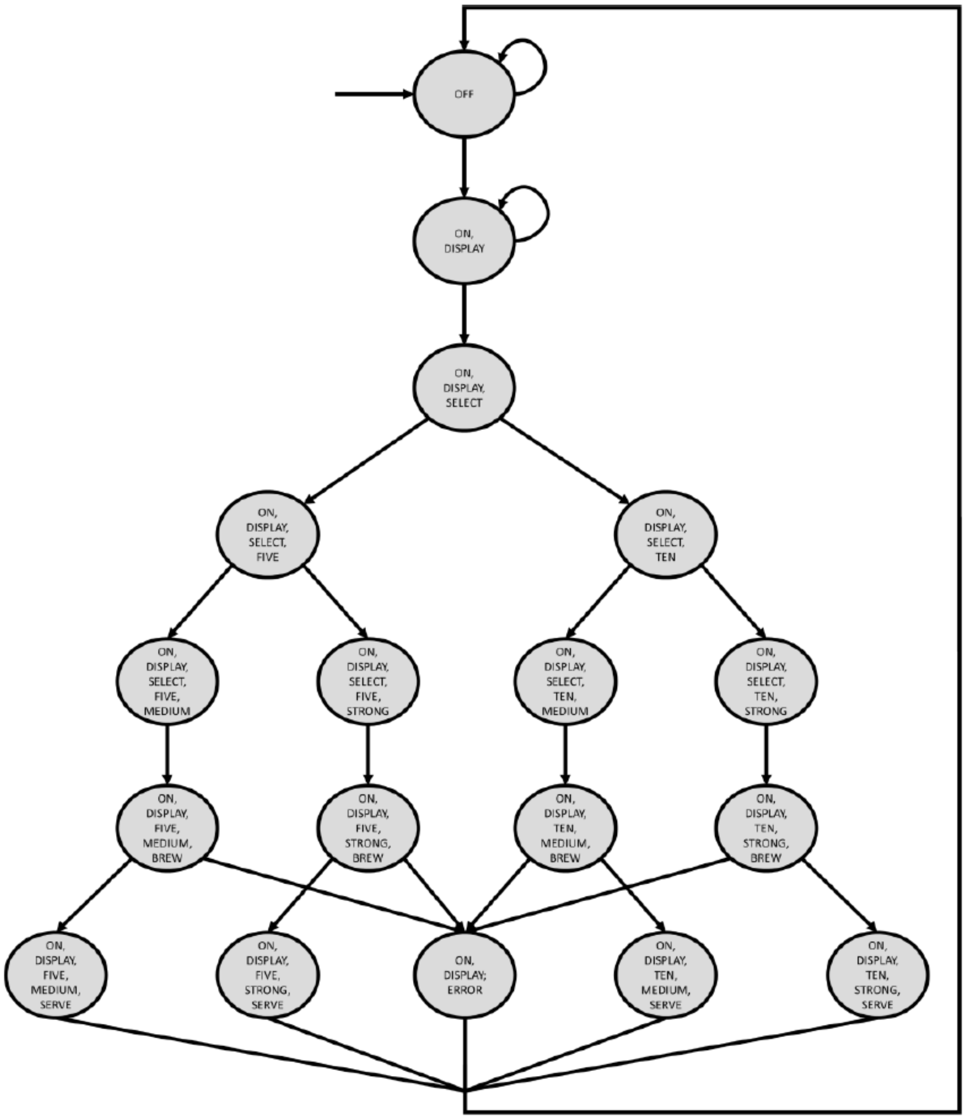
1 From any state one can possibly eventually select ten cups of coffee and once selected, ten cups will always be served (or an error encountered) in the future.

$$f_{1,CTL} = (AG\ EF\ coffee\_10) \wedge (AG(coffee\_10 \rightarrow AF(serve\_10\_medium \vee serve\_10\_strong \vee error)))$$

Properties in LTL/CTL/CTL*

2 It is possible to eventually reach an error for any selection of coffee strength or number of cups made.

$$f_{2,CTL} = AG((coffee\_10 \lor coffee\_5 \lor str\_medium \lor str\_strong) \rightarrow EF\ error)$$



Yes

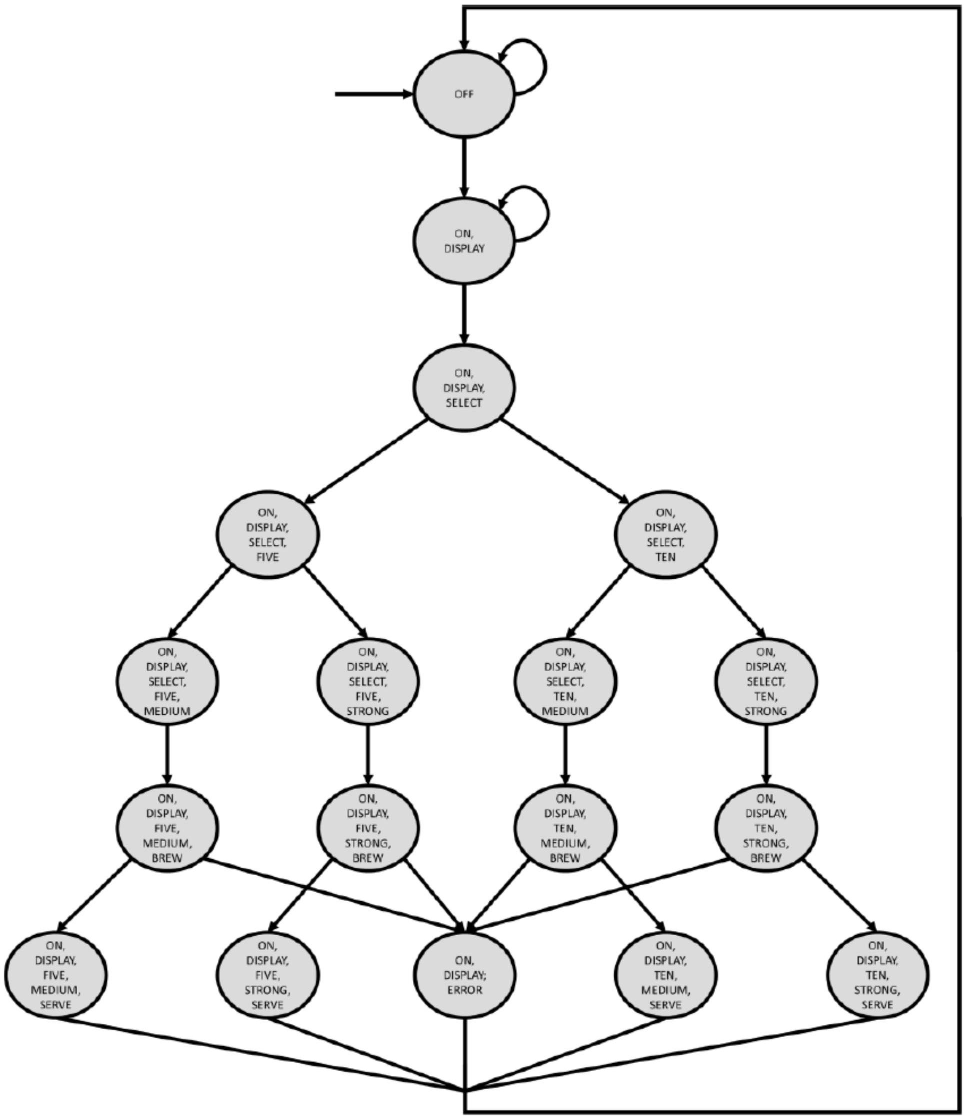
2 It is possible to eventually reach an error for any selection of coffee strength or number of cups made.

$$f_{2,CTL} = AG((coffee\_10 \vee coffee\_5 \vee str\_medium \vee str\_strong) \rightarrow EF\ error)$$

Properties in LTL/CTL/CTL*

3 For any execution, the coffee machine will forever be turned off from some point on.

$$f_{3,CTL} = AG\,EF\,(AG\,\neg on)$$



No

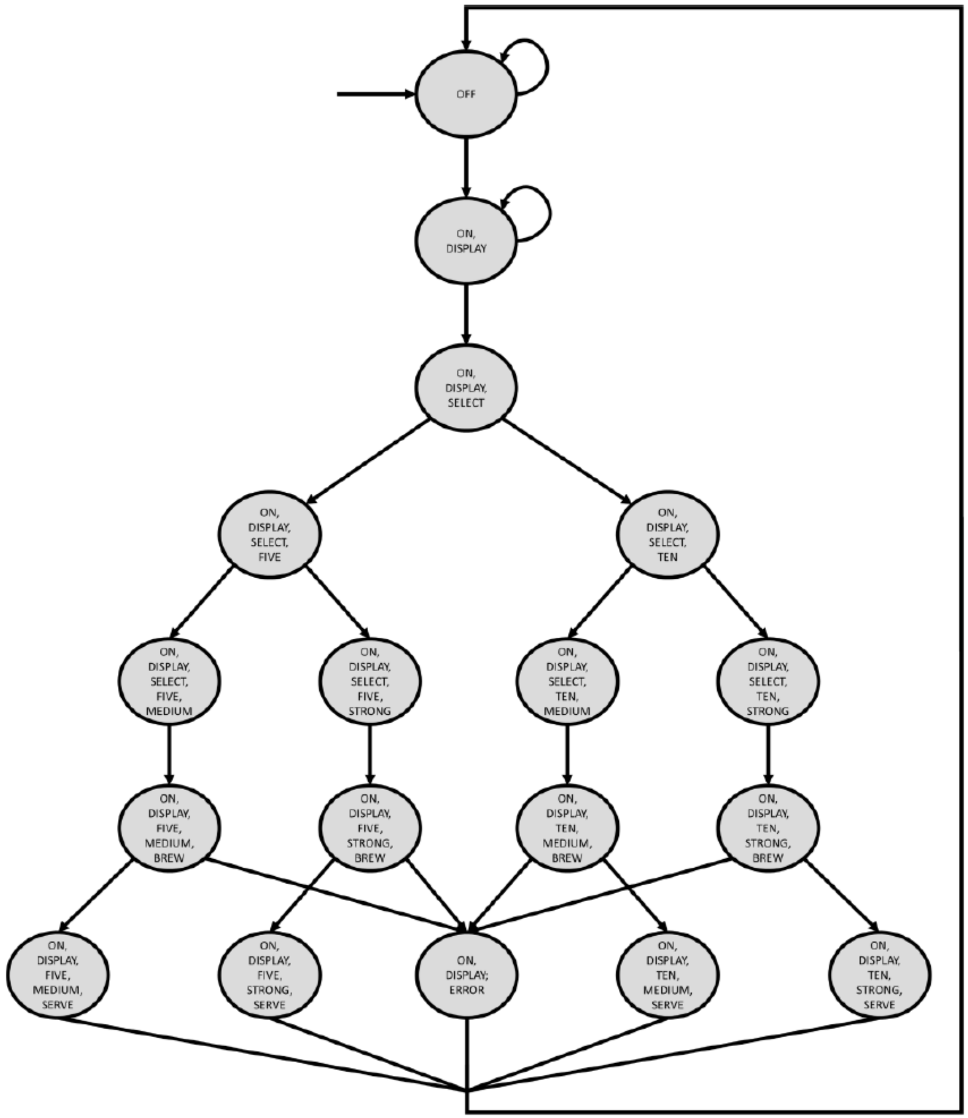
3 **For any execution, the coffee machine will forever be turned off from some point on.**

$$f_{3,CTL} = AG\, EF\, (AG\, \neg on)$$

Properties in LTL/CTL/CTL*

4 **Before the coffee brewer gets turned off forever which will always happen eventually, there was eventually coffee.**

$$f_{4,LTL} = A((F\,(serve\_10\_medium \vee serve\_10\_strong \vee serve\_5\_medium \vee serve\_5\_strong)U(G\,\neg on)))$$



No

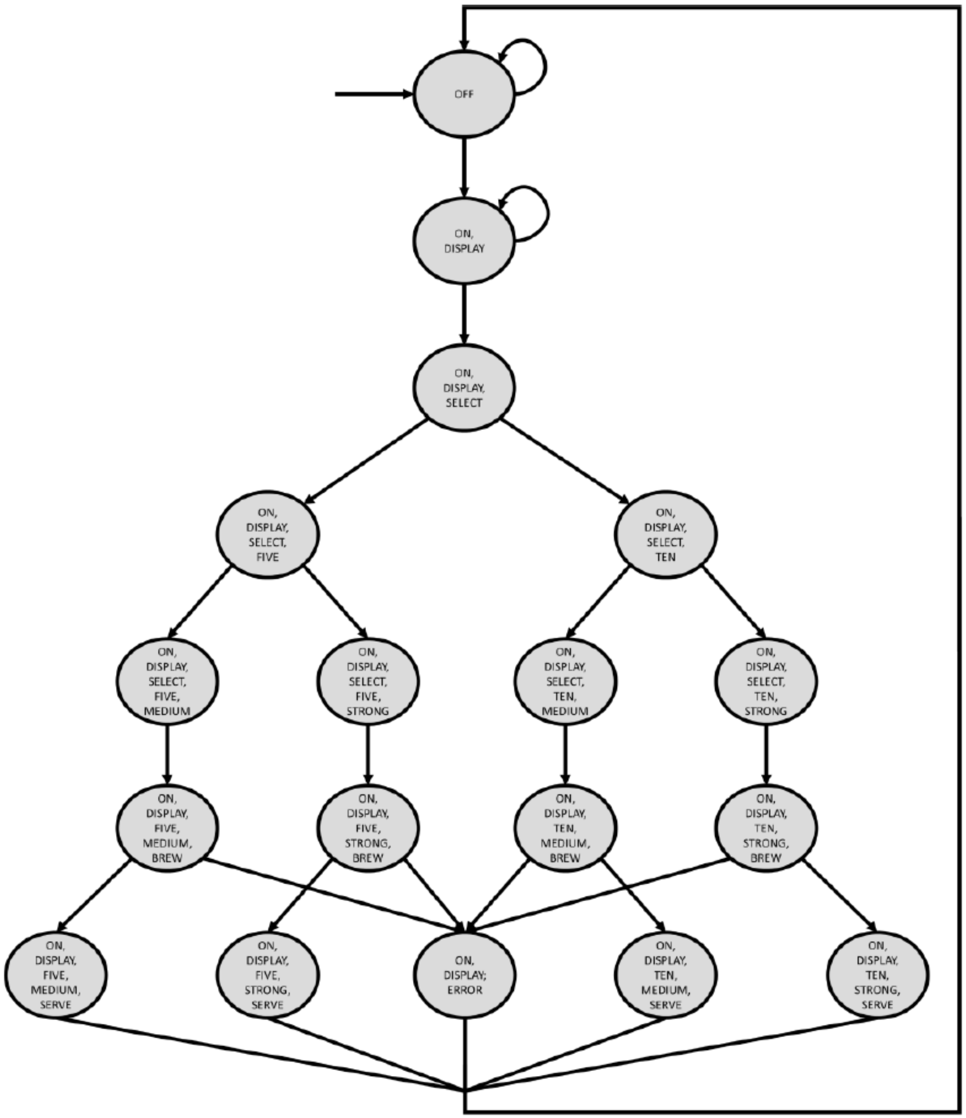
4 **Before the coffee brewer gets turned off forever which will always happen eventually, there was eventually coffee.**

$$f_{4,LTL} = A((F\,(serve\_10\_medium \lor serve\_10\_strong \lor serve\_5\_medium \lor serve\_5\_strong)\,U(G\,\neg on)))$$

Properties in LTL/CTL/CTL*

5 Always, once the brewing is done, the display lighting is eventually turned off.

$$f_{5,LTL} = A(brew \rightarrow F\neg display\_on)$$



Yes

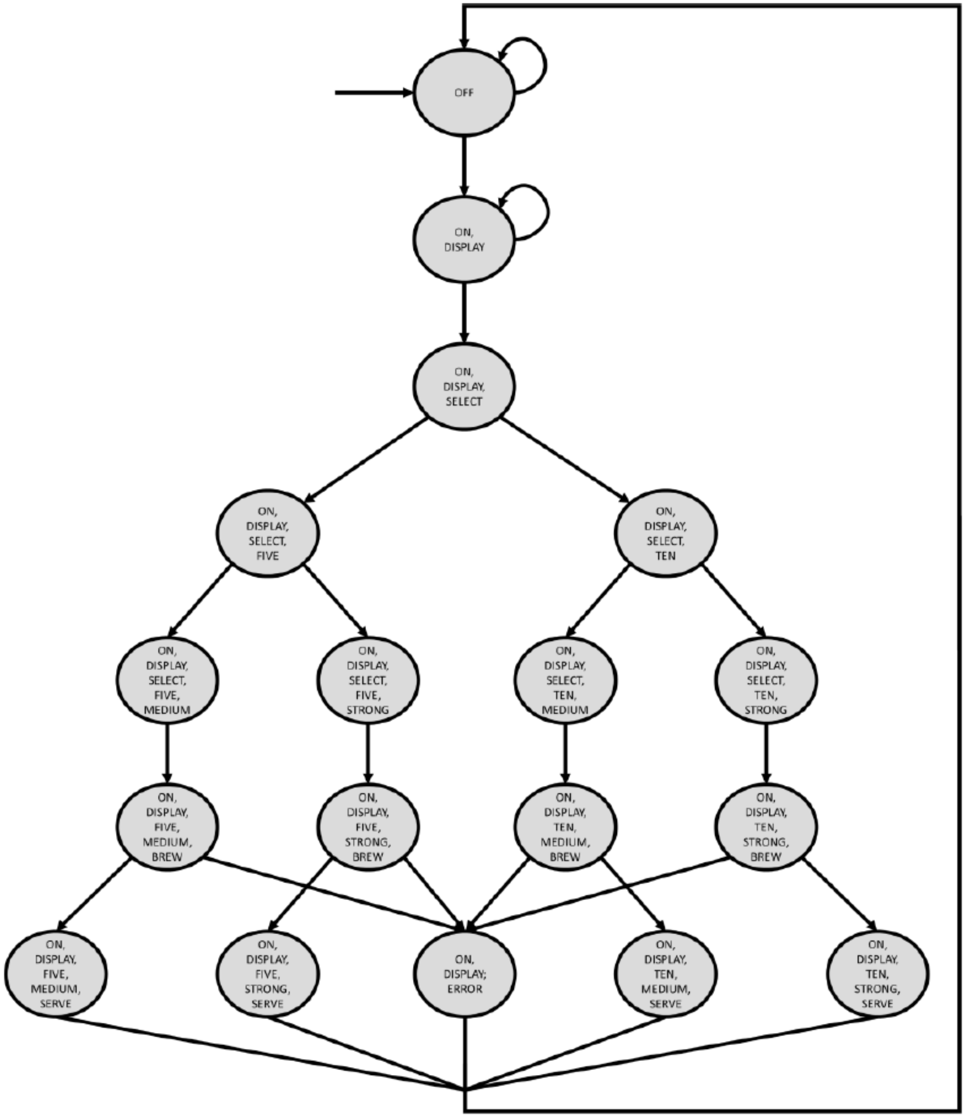
5 Always, once the brewing is done, the display lighting is eventually turned off.

$$f_{5,LTL} = A(brew \rightarrow F \neg display\_on)$$

Properties in LTL/CTL/CTL*

6 It can be the case that we reach an error, but get eventually 10 cups of coffee nevertheless.

$$f_{6,CTL*} = E\,F\,(\neg(F\ error \rightarrow A\,G\,\neg(serve\_10\_medium \vee serve\_10\_strong)))$$



Yes

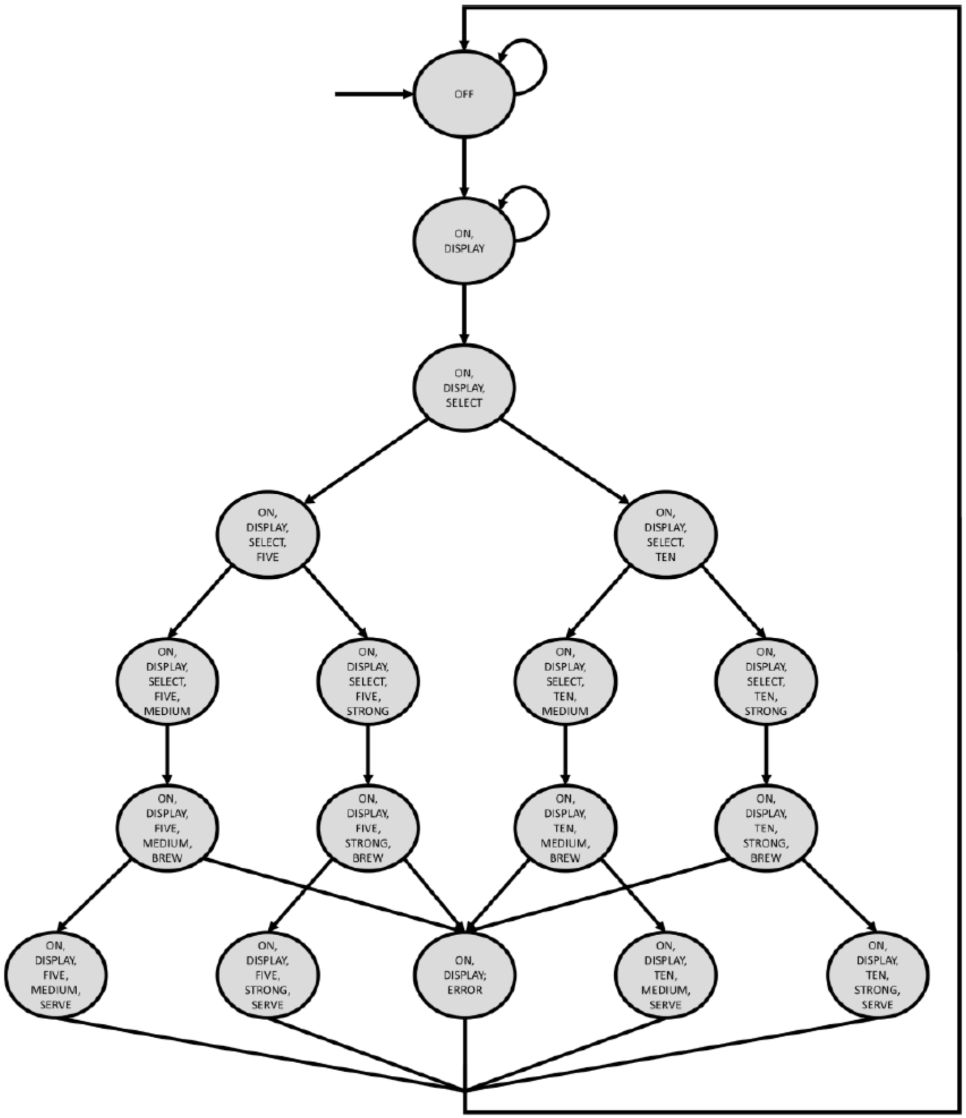
6 **It can be the case that we reach an error, but get eventually 10 cups of coffee nevertheless.**

$$f_{6,CTL*} = E\,F\,(\neg(F\ error \rightarrow A\,G\,\neg(serve\_10\_medium \vee serve\_10\_strong)))$$

Properties in LTL/CTL/CTL*

7 All reachable states can result in 10 cups of coffee eventually.

$$f_{7,CTL} = EG\ AF\ (EF\ (serve\_10\_strong \lor serve\_10\_medium))$$

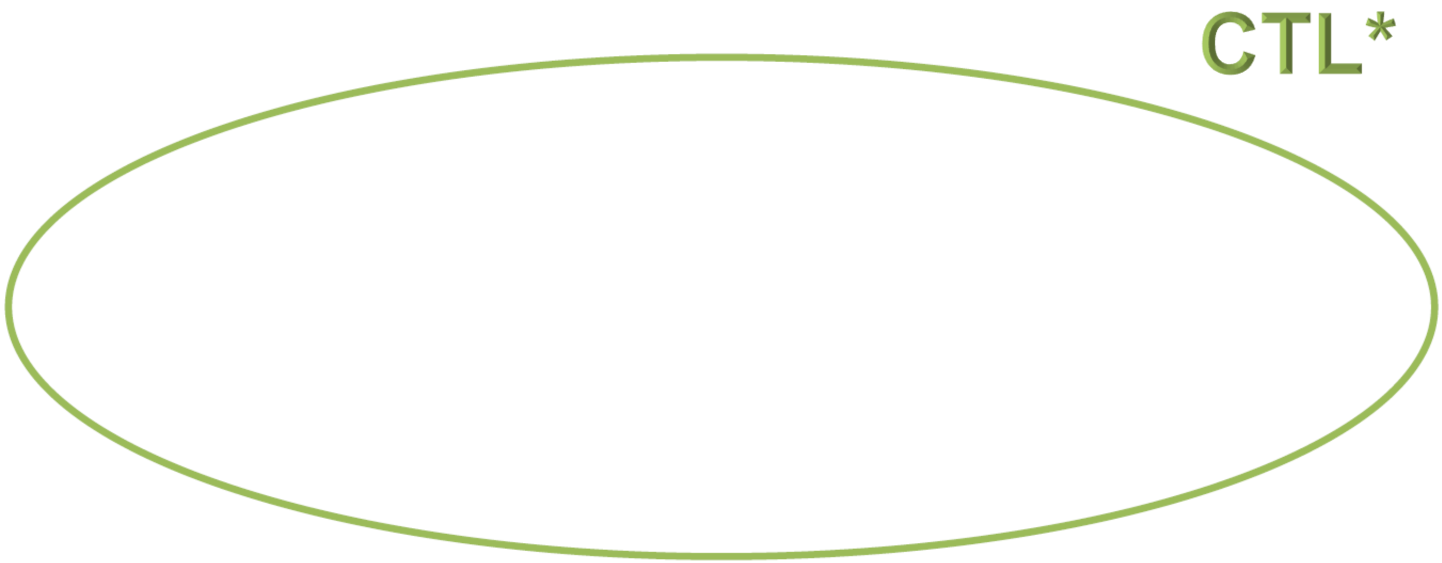


Yes

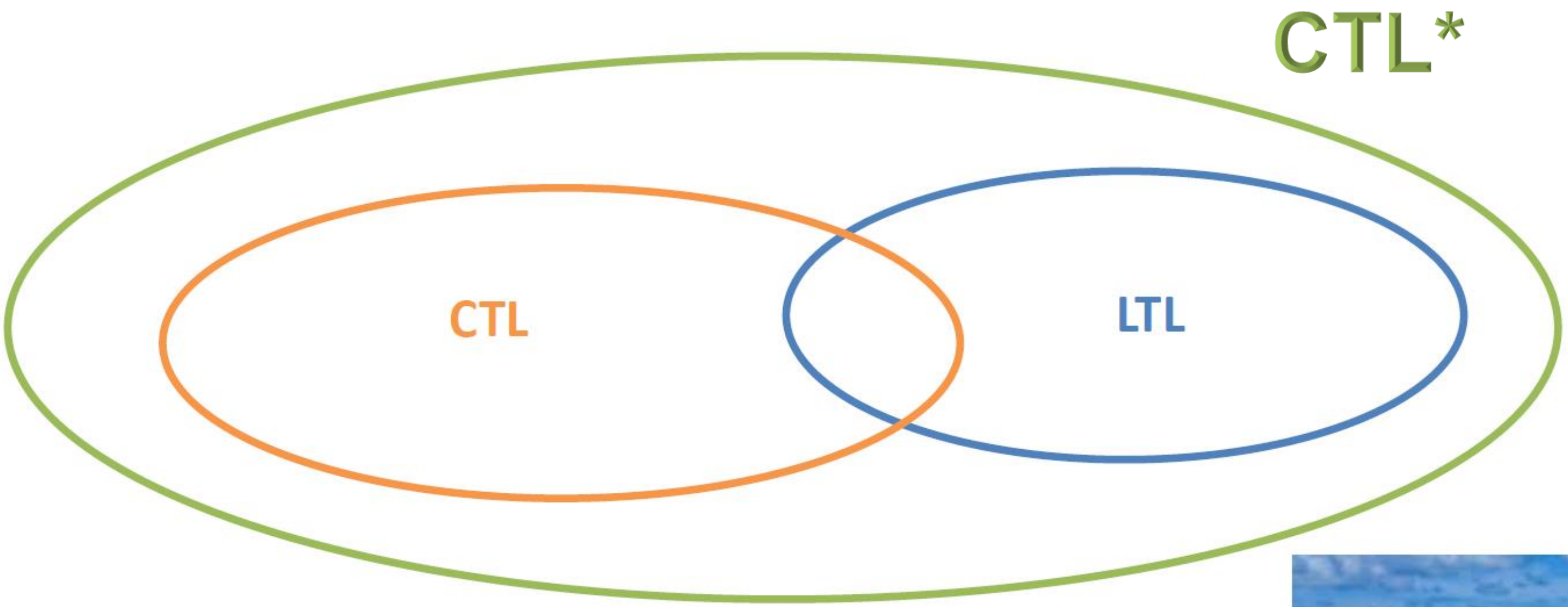
7 All reachable states can result in 10 cups of coffee eventually.

$$f_{7,CTL} = EG\ AF\,(EF\,(serve\_10\_strong \vee serve\_10\_medium))$$

Expressiv Power of LTL and CTL



Expressiv Power of LTL and CTL

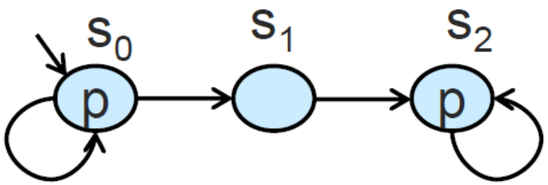


LTL vs CTL

- Exercise:
  - Dies the LTL formula $\boldsymbol{F G\ p}$ has an equivalent in CTL?

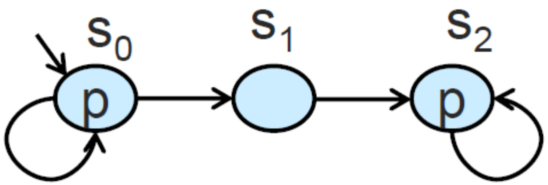
LTL vs CTL

- Exercise:
  - Dies the LTL formula $\boldsymbol{FG\ p}$ has an equivalent in CTL?
- Solution: No
  - Failed attempts: **AFAG**p
    - "in every path there is a point from which all reachable states satisfy p"



LTL vs CTL

- Exercise:
  - Dies the LTL formula $\boldsymbol{FG\,p}$ has an equivalent in CTL?
- Solution: No
  - Failed attempts: **AFAG**p
    - “in every path there is a point from which all reachable states satisfy p”



All paths satisfy **FG**p

- $s_0, s_0, s_0, \ldots$
- $s_0, s_0, \ldots s_0, s_1, s_2, s_2, s_2 \ldots$

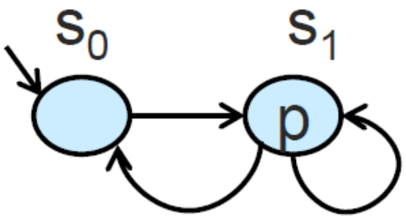
But first one does not sat **FAG**p

LTL vs CTL

- Exercise:
  - Dies the LTL formula $\boldsymbol{FG\ p}$ has an equivalent in CTL?
- Solution: No

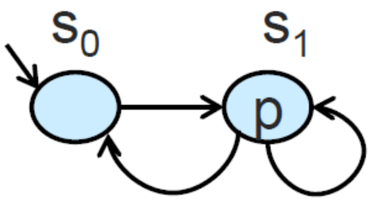
ToDo What about **AFEG p?**

Hint:



LTL vs CTL

- Exercise:
  - Dies the LTL formula $\boldsymbol{FG\ p}$ has an equivalent in CTL?
- Solution: No
  - What about **AFEG p?**
    - "in every path there is a point from which there is a path where p globally holds"



All paths satisfy **FEG**p

- since s_1 sat **EG**p

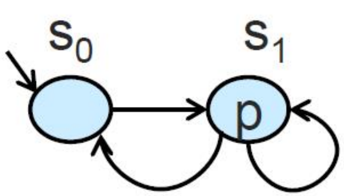
But $s_0,s_1,s_0,s_1,s_0,s_1\ldots$ does not satisfy **FG**p

LTL vs CTL

- Exercise:
  - Does **AG(EF p)** has an LTL equivalent?

LTL vs CTL

- Exercise:
  - Does **AG(EF p)** has an LTL equivalent?
- Solution: No
  - Failed attempt:
    - **GF**p : “in all paths, p holds infinitely many times.”



All reachable states (s_0, s_1) satisfy **EF**p

But $s_0,s_0,s_0,\ldots$ does not satisfy **GF**p

LTL vs CTL

- The expressive powers of LTL and CTL are incomparable. That is,
  - There is an LTL formula that has no equivalent CTL formula
  - There is a CTL formula that has no equivalent LTL formula

- CTL* is more expressive than either of them

Counterexamples

- Given M and φ, such that $M \not\models \varphi$, a counterexample is a behavior of M, demonstrating the violation of φ in M

- To be useful for debugging it should
  - have finite representation
  - be easy-to-understand by human

Examples of Counterexamples

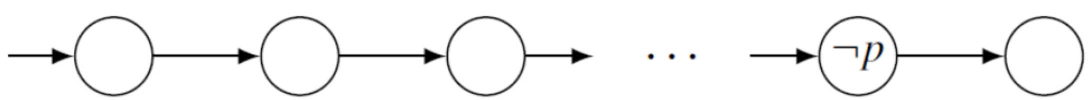
- For **AX**p:
  A transition from an initial state to a state violating p
  - Counterexample for **AX**p is a witness for **EX**$\neg$ p

Examples of Counterexamples

- For **AX**p:
  A transition from an initial state to a state violating p
  - Counterexample for **AX**p is a witness for **EX**¬ p

- For **AG**p:
  A finite path from an initial state to a state violating p
  - Counterexample for **AG**p is a witness for **EF**¬ p

$\mathbf{F}\,\neg p$



Examples of Counterexamples

- For **AF**p:
  An infinite path, all of its states violating p (satisfying ¬p)
  - Counterexample for **AF**p is a witness for **EG**¬ p

Exercise:

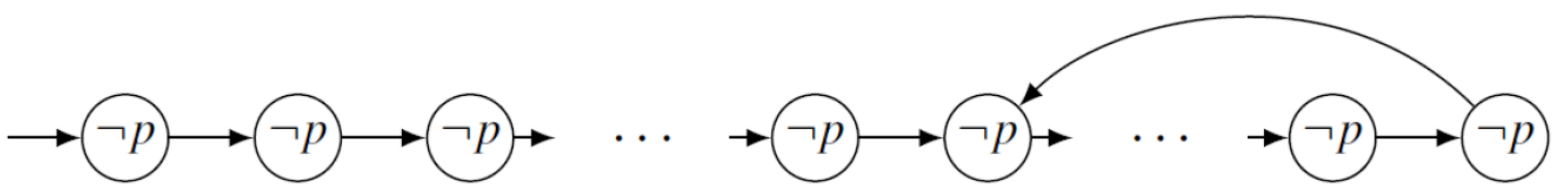
- How do we get a finite representation for the CE?

Examples of Counterexamples

- For **AF**p:
  An infinite path, all of its states violating p (satisfying $\neg$p)
  - Counterexample for **AF**p is a witness for **EG**$\neg$ p

- A finite representation for violation of AFp:
  - A lasso, which is a path of the form $\pi = \pi_0 (\pi_1)^\omega$
  - π_0 and π_1 are finite paths
  - ω indicates infinitely many repetitions of π_1

$\mathbf{G}\neg p$



Safety and Liveness Properties

Safety and Liveness Properties

Informally,

- Safety properties guarantee that “something wrong will never happen”
  - Typical example: **AG**p

Safety and Liveness Properties

Informally,

- **Safety** properties guarantee that "something wrong will never happen"
  - Typical example: **AG**p
- **Liveness** properties guarantee that "something good will eventually happen"
  - Typical examples: **AF**p, **A**(p**U**q)

Safety Properties

- Nothing 'bad' will happen.
  - Example: $G(p \rightarrow X\ q)$

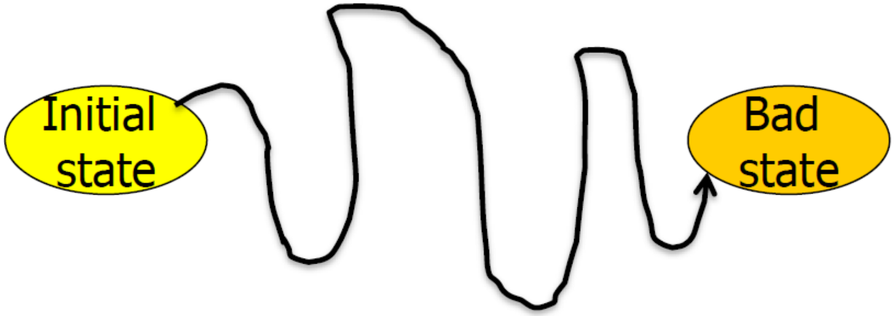
ToDo Exercise:

- How does a CE for a **safety** property look like?

Safety Properties

- Nothing 'bad' will happen.
  - Example: G(p → X q)

- Counterexamples for a safety property is a finite (loop-free) path



Liveness Properties

- Something ‘good’ will happen.
  - Example: F p

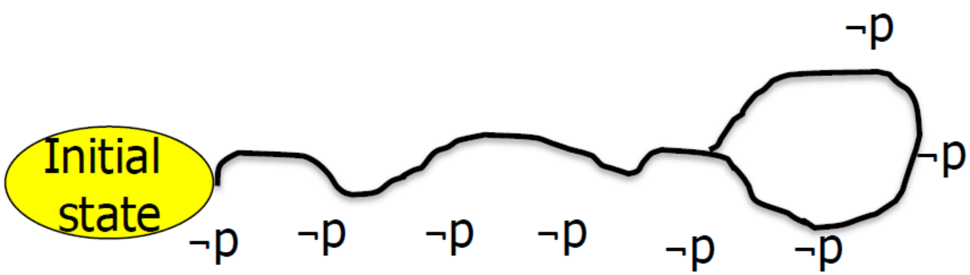
Exercise:

- How does a CE for a **Liveness** property look like?

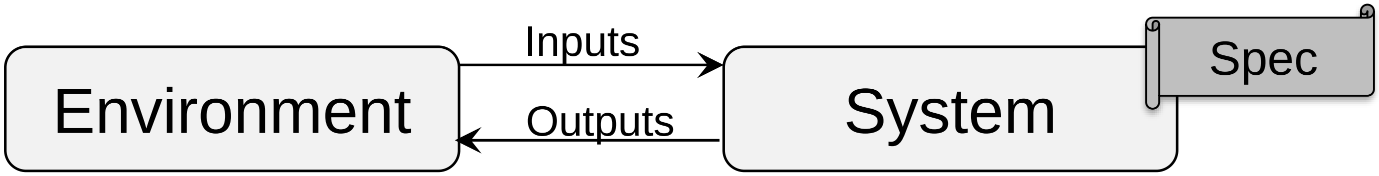
Liveness Properties

- Something 'good' will happen.
  - Example: F p

  - A counterexample is an infinite trace with lasso shape, showing that this good thing **NEVER** happens.



Reactive Systems

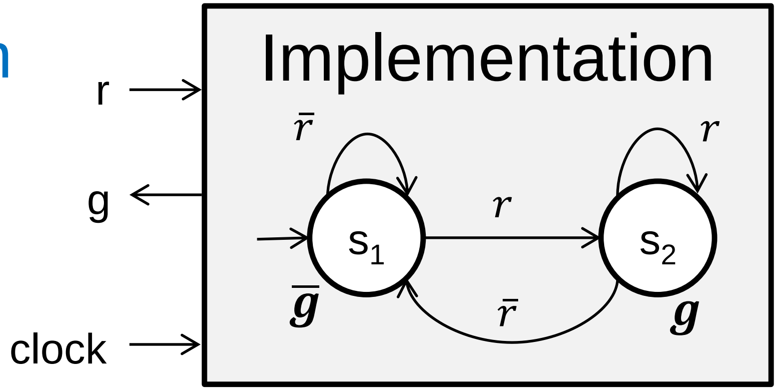


- $I = \{i_1, i_2, \ldots, i_m\}$ is a set of Boolean inputs
- $O = \{o_1, o_2, \ldots, o_n\}$ is a set of Boolean outputs

Implementation: What is Inside?

Moore Machine:

- Input alphabet: $\Sigma_I = 2^I$
- Output alphabet: $\Sigma_O = 2^O$
- A Moore Machine is a tuple $(S, s_0, \Sigma_I, \Sigma_O, \delta, o)$
  - S is a finite set of states
  - s_0 is an initial state
  - $\delta: S \times \Sigma_I \rightarrow S$ is the transition function
  - $o: S \rightarrow \Sigma_O$ is the output function

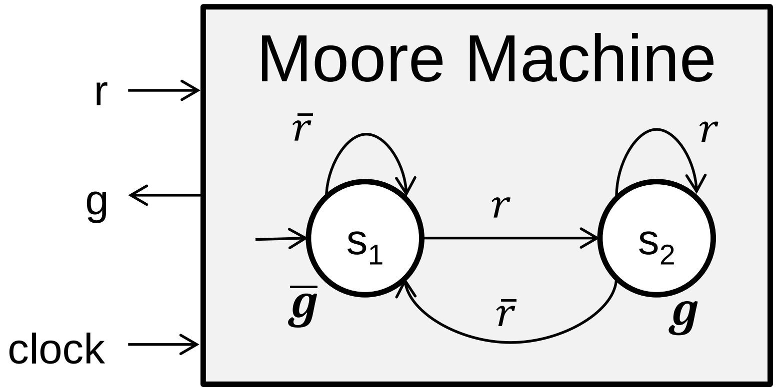


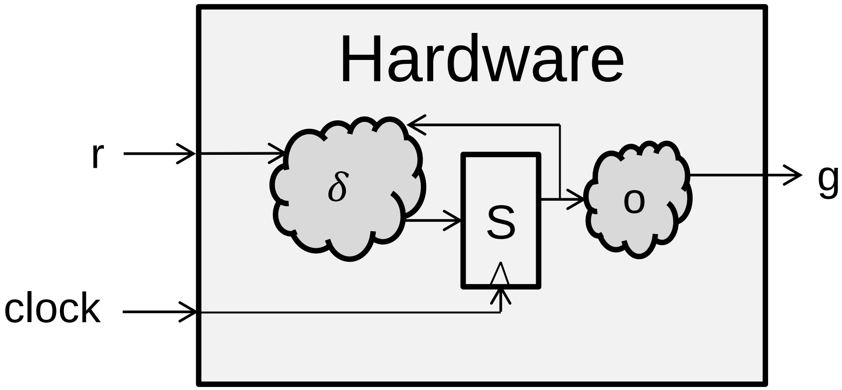
SCOS
Secure & Correct Systems

From a Moore Machine to Hardware

Simple Transformation:

- The current state $s \in S$ is stored in flip-flops
- $\delta: S \times \Sigma_\mathrm{I} \rightarrow S$ and $o: S \rightarrow \Sigma_\mathrm{O}$ are combinatorial circuits





Example: Arbiter

Input: r0, r1

Output: g0, g1

ToDo **Draw Moore Machine**

Guarantee G1: $G(r0 \rightarrow Fg0)$

Guarantee G2: $G(r1 \rightarrow Fg1)$

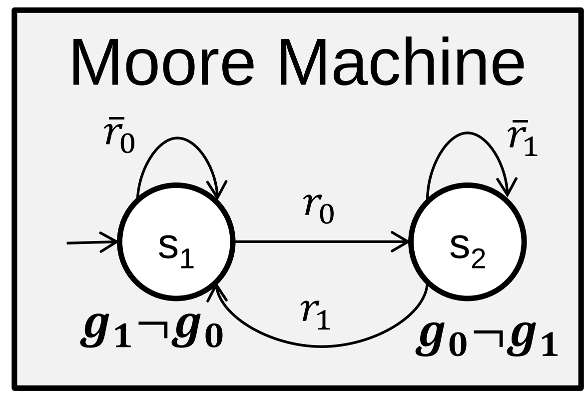
Guarantee G3: $G(\neg g0 \vee \neg g1)$

$$\varphi := G_1 \wedge G_2 \wedge G_3$$

Example: Arbiter

Input: r0, r1

Output: g0, g1



Guarantee G1: G(r0 → **F**g0)

Guarantee G2: G(r1 → **F**g1)

Guarantee G3: G(¬g0 ∨ ¬g1)

$$\varphi := G_1 \wedge G_2 \wedge G_3$$

Example: Arbiter

Input: r0, r1

Output: g0, g1

Draw Moore Machine

Guarantee G1: $G(r0 \rightarrow Xg0)$

Guarantee G2: $G(r1 \rightarrow Xg1)$

Guarantee G3: $G(\neg g0 \vee \neg g1)$

$$\varphi \coloneqq G_1 \wedge G_2 \wedge G_3$$

Example: Arbiter

Input: r0, r1

Output: g0, g1

Unrealizable!

Guarantee G1: G(r0 $\rightarrow$ **X**g0)

Guarantee G2: G(r1 $\rightarrow$ **X**g1)

Guarantee G3: G($\neg$g0 $\vee$ $\neg$g1)

$$\varphi := G_1 \wedge G_2 \wedge G_3$$

Example: Arbiter

Input: r0, r1

Output: g0, g1

Draw Moore Machine

Guarantee G1: G(r0 → **X**g0)

Guarantee G2: G(r1 → **X**g1)

Guarantee G3: G(¬g0 ∨ ¬g1)

Assumption A1: G(¬g0 ∨ ¬g1)

$$\varphi := A_1 \rightarrow G_1 \wedge G_2 \wedge G_3$$

Example: Arbiter

Input: r0, r1

Output: g0, g1

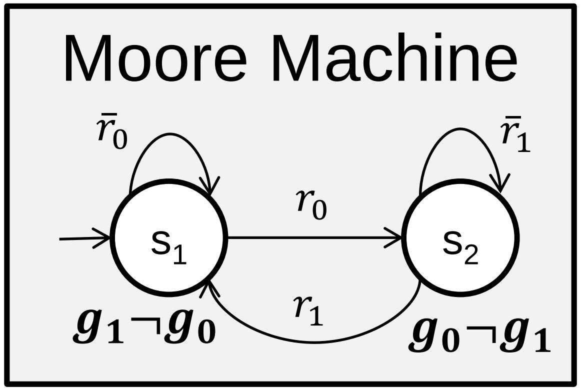
Guarantee G1: G(r0 → **X**g0)

Guarantee G2: G(r1 → **X**g1)

Guarantee G3: G(¬g0 ∨ ¬g1)

Assumption A1: G(¬g0 ∨ ¬g1)

$$\varphi := A_1 \rightarrow G_1 \wedge G_2 \wedge G_3$$




THANK
YOU!