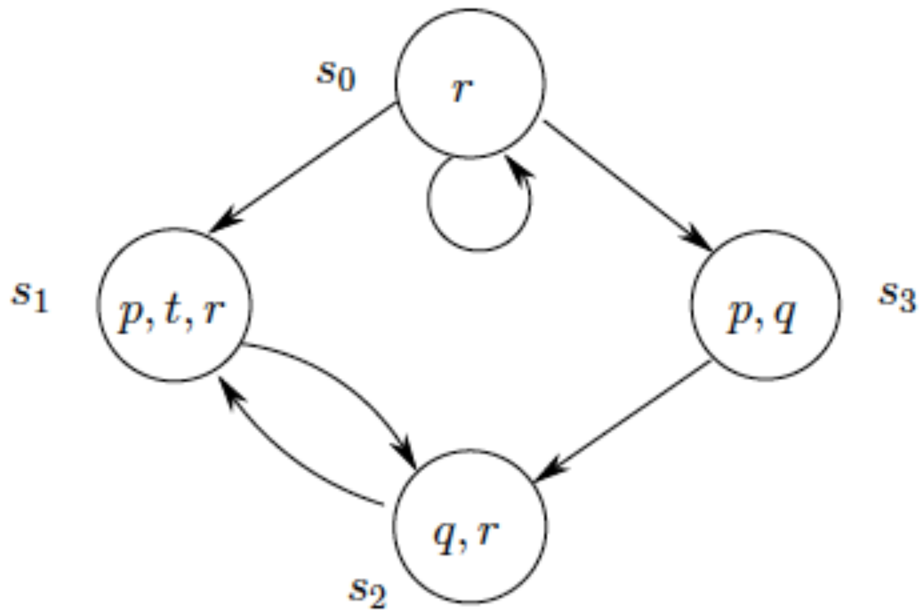
Formal Methods in software development

a.y.2017/2018

Prof. Anna Labella

Exercises



Determine whether $\mathcal{M}, s_0 \vDash \phi$ and $\mathcal{M}, s_2 \vDash \phi$ hold and justify your answer, where ϕ is the LTL or CTL formula:

* (i) $\neg p \to r$

(ii) $\mathrm{F}\, t$

*(iii) $\neg \mathrm{EG}\, r$

(iv) $\mathrm{E}\,(t\ \mathrm{U}\ q)$

(v) $\mathrm{F}\, q$

(vi) $\mathrm{EF}\, q$

(vii) $\mathrm{EG}\, r$

(viii) $\mathrm{G}\,(r \vee q)$.

Exercises

Which of the following pairs of CTL formulas are equivalent? For those which are not, exhibit a model of one of the pair which is not a model of the other:

(a) $\mathrm{EF}\,\phi$ and $\mathrm{EG}\,\phi$

(b) $\mathrm{EF}\,\phi \vee \mathrm{EF}\,\psi$ and $\mathrm{EF}\,(\phi \vee \psi)$

(c) $\mathrm{AF}\,\phi \vee \mathrm{AF}\,\psi$ and $\mathrm{AF}\,(\phi \vee \psi)$

(d) $\mathrm{AF}\,\neg\phi$ and $\neg\mathrm{EG}\,\phi$

(e) $\mathrm{EF}\,\neg\phi$ and $\neg\mathrm{AF}\,\phi$

(f) $\mathrm{A}[\phi_1 \;\mathrm{U}\; \mathrm{A}[\phi_2 \;\mathrm{U}\; \phi_3]]$ and $\mathrm{A}[\mathrm{A}[\phi_1 \;\mathrm{U}\; \phi_2] \;\mathrm{U}\; \phi_3]$, hint: it might make it simpler if you think first about models that have just one path

(g) $\top$ and $\mathrm{AG}\,\phi \to \mathrm{EG}\,\phi$

(h) $\top$ and $\mathrm{EG}\,\phi \to \mathrm{AG}\,\phi$.

Modelling a system and checking it

Temporal logics at work

Example: mutual exclusion

We have two concurrent processes sharing a resource

This means that for each of them there is a critical phase

A process can stay indefinitely in its non critical phase

At any time a process can ask to enter its critical phase

The two processes are not supposed to alternate

Example: mutual exclusion

The solution must satisfy:

- Mutual exclusion: the two processes cannot be in their critical state together

- Progress: each of them cannot stay in its critical phase forever

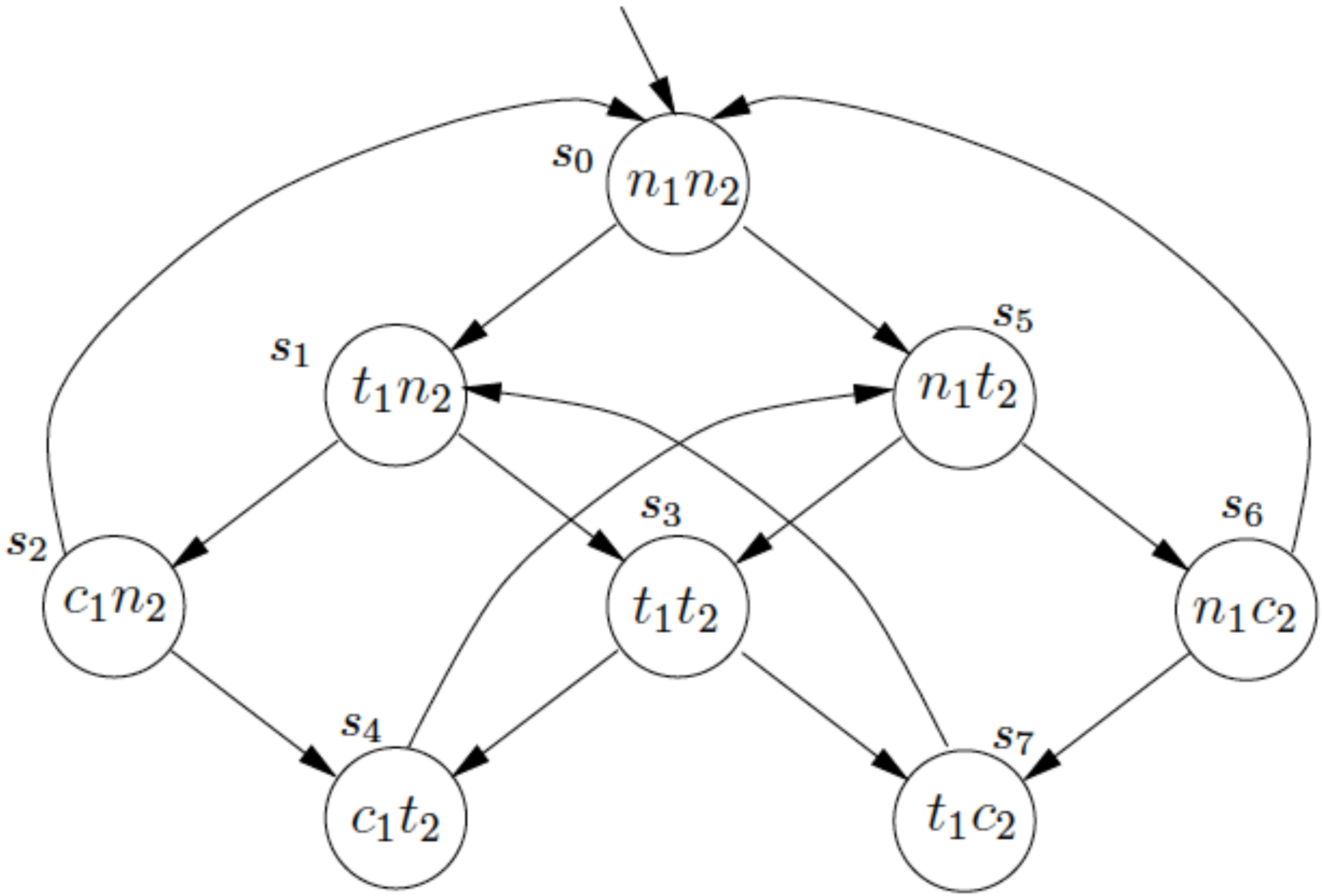
Example: mutual exclusion

The evolution of a single process

n→ t→ c→ n→ t→ c→ ……..

n for "non-critical"
t for "trying to access"
c for "critical"

The first modeling attempt



Esxpressivity of LTL

- safety G $\neg(c_1 \wedge c_2)$ true

- liveness G $(t_i \rightarrow F c_i)$ false

$s_0 \rightarrow s_1 \rightarrow s_3 \rightarrow s_7 \rightarrow s_1 \rightarrow s_3 \rightarrow s_7 \rightarrow \ldots.$

Expressivity of LTL

- Non-blocking

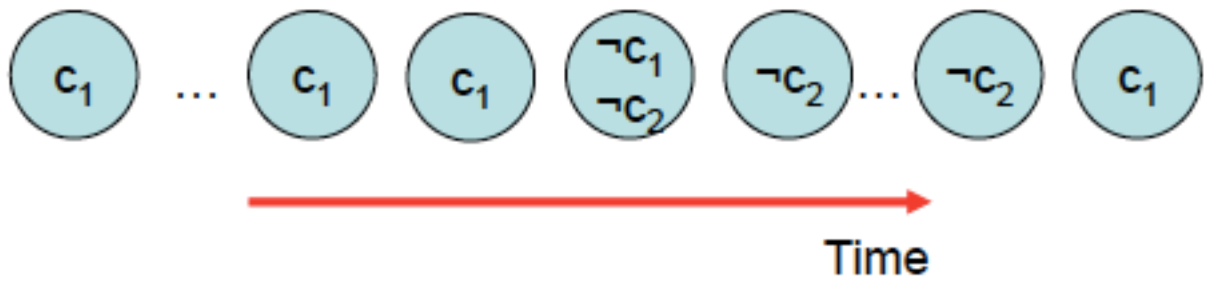
$AG(n_1 \rightarrow EX\ t_1)$

For every state satisfying n_1 there is a successor satisfying t_1

Expressivity of LTL

- no strict sequencing: processes need not enter their critical section in strict sequence

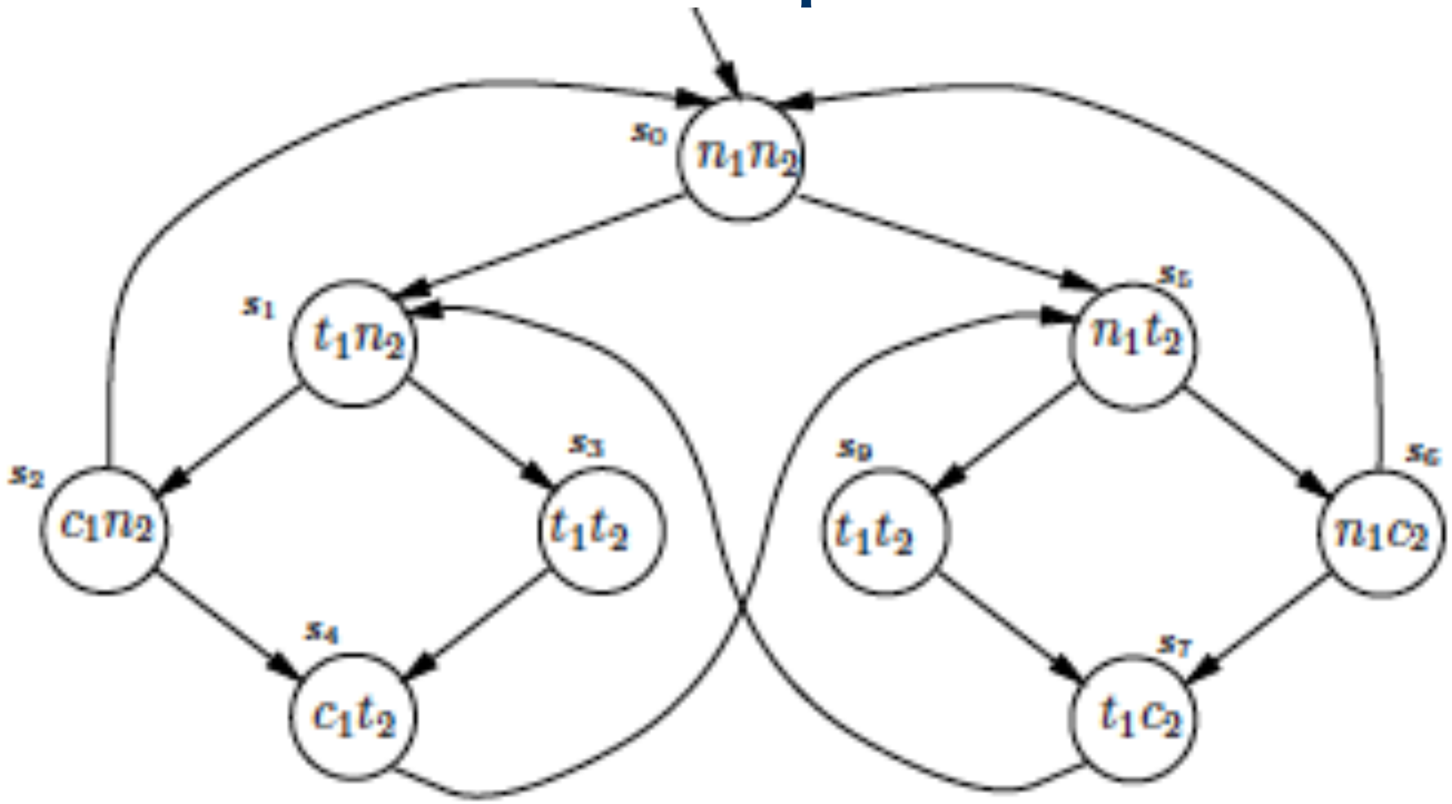
There exists at least one path with no strict sequencing:



$G\ (\ c_1 \rightarrow (\ c_1 W\ (\ \neg c_1 \wedge (\neg c_1 W\ c_2)\ ))$ false

$s_0 \rightarrow s_5 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \rightarrow s_3 \rightarrow s_4 \rightarrow \ldots$

The second attempt



But we cannot model the fact that a process can stay for a while
In its critical section

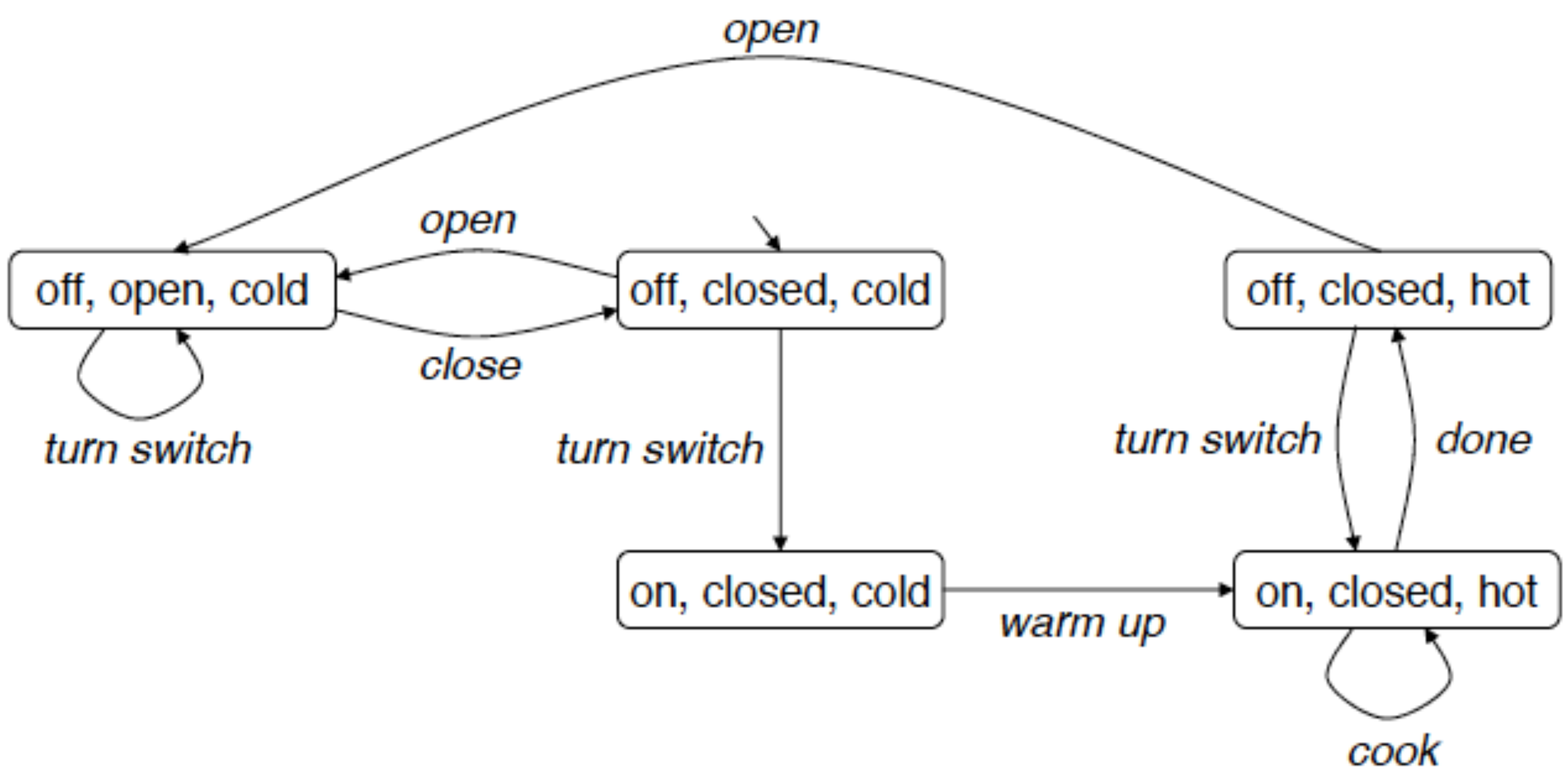
A modeling example

- The following is a description of a microwave oven:

- The oven has the following components:
  - a switch (which is either on or off, initially off);
  - a door (which is either open or closed, initially closed);
  - a plate (which is either hot or cold, initially cold).

- The user may open or close the door.
- He may turn the switch when the oven is off.
- Turning the switch when the door is open has no effect (to prevent accidents).
- When the oven is turned on, it first warms up the plate,
  then cooks until the dish is ready, and then automatically turns itself off.
- Opening the door makes the heat dissipate.

Example
Behavior of the system



Example
Specification for the oven

The manufacturer wants us to check the following property :

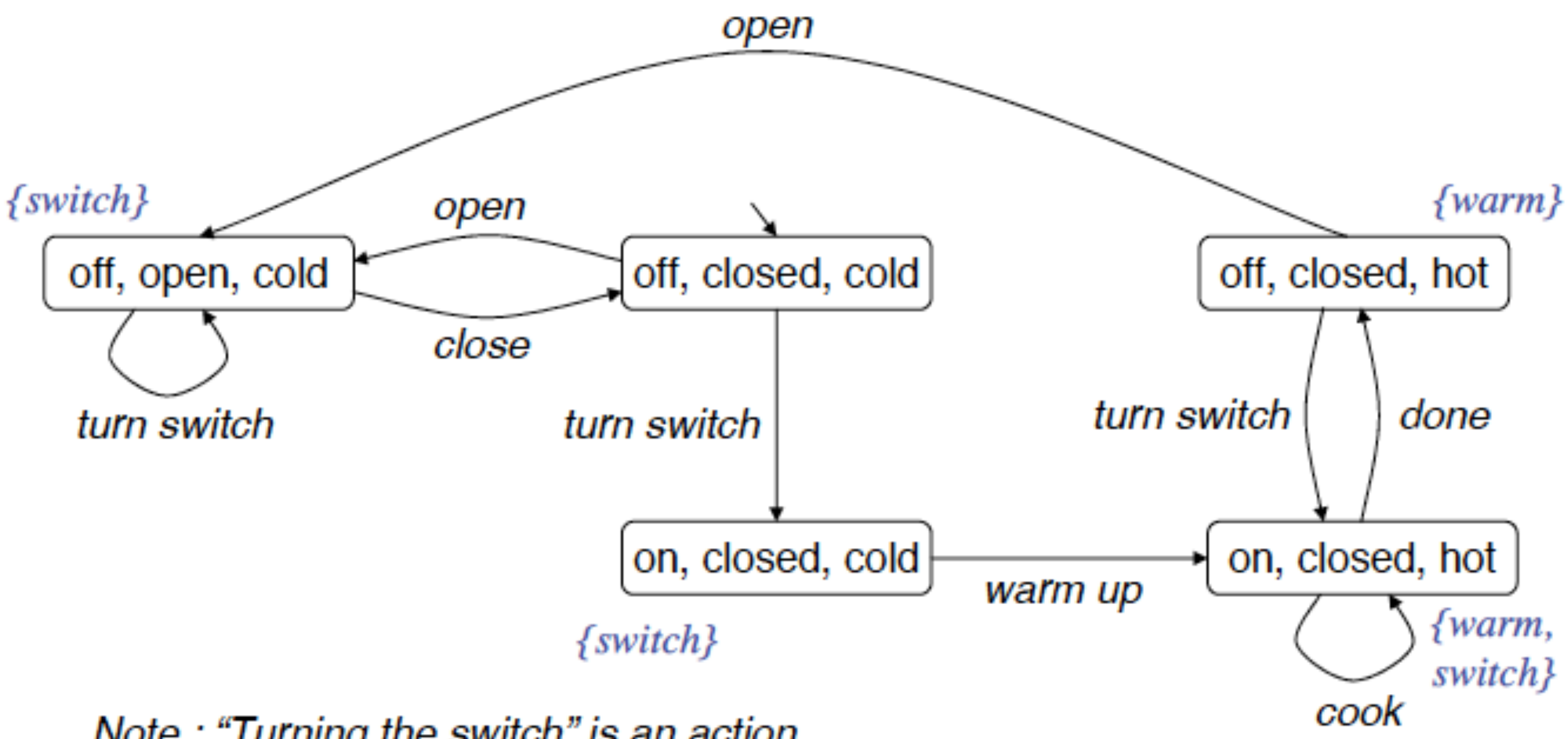
"Whenever the user turns the switch,
the plate will eventually become warm."

- We first formulate the property in CTL:

$$AG\,(switch \Rightarrow AF\,warm)$$

- To check the property, we label the Kripke structure with the two atomic propositions: *warm* and *switch*

Example (continued)



Note : "Turning the switch" is an action,
which we cannot directly model in our state-based semantics,
therefore we take all the states which are the target of a switching transition.

Example (continued)

- So, we rewrite the formula :

$$f_{CTL} = G\,(switch \Rightarrow AF\ warm)$$

or equivalently,

$$f_{CTL} = \neg(true\ EU\ (switch \wedge EG\ \neg warm))$$

- Checking the property yields that the formula is not true – because the system may stay forever in left most state.

  The erroneous behavior happens when
  the user keeps turning the switch while the door is open.

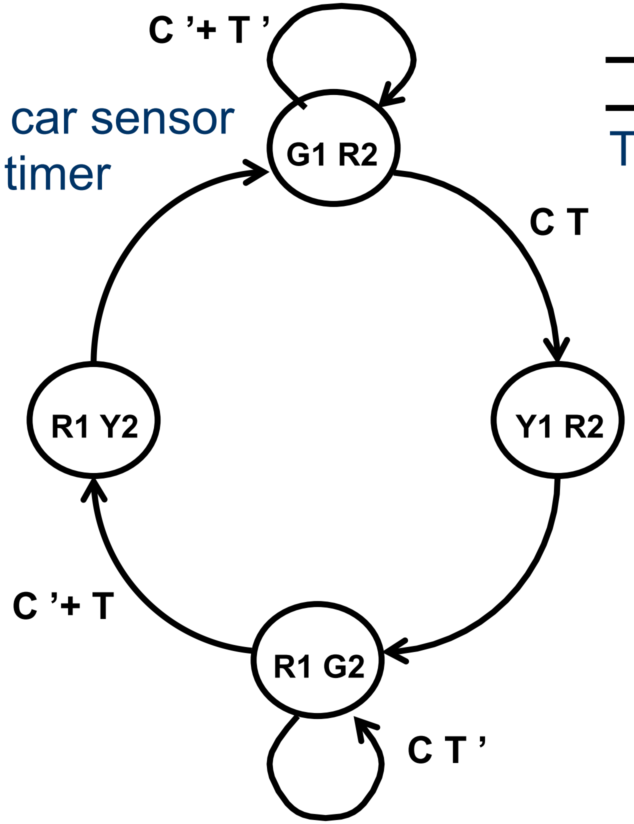
- A “reasonable” user should eventually realize that
  turning the switch when the door is open does no good. So,
  consider only executions that do not stay forever in this state.

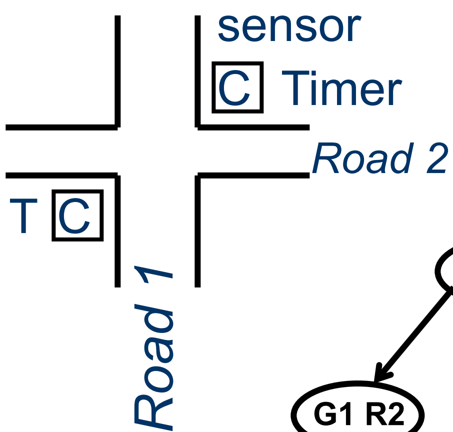
not expressible in CTL
but in CTL fair ….

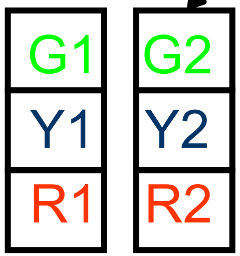
Model Checking Example

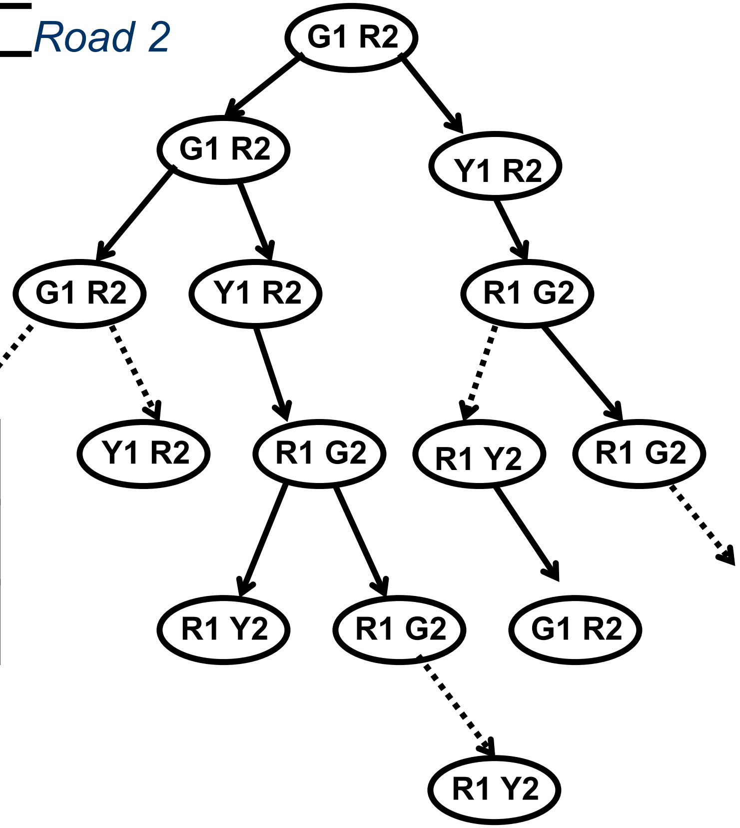
Traffic light controller
(simplified)

C = car sensor
T = timer









Traffic light controller - Model Checking

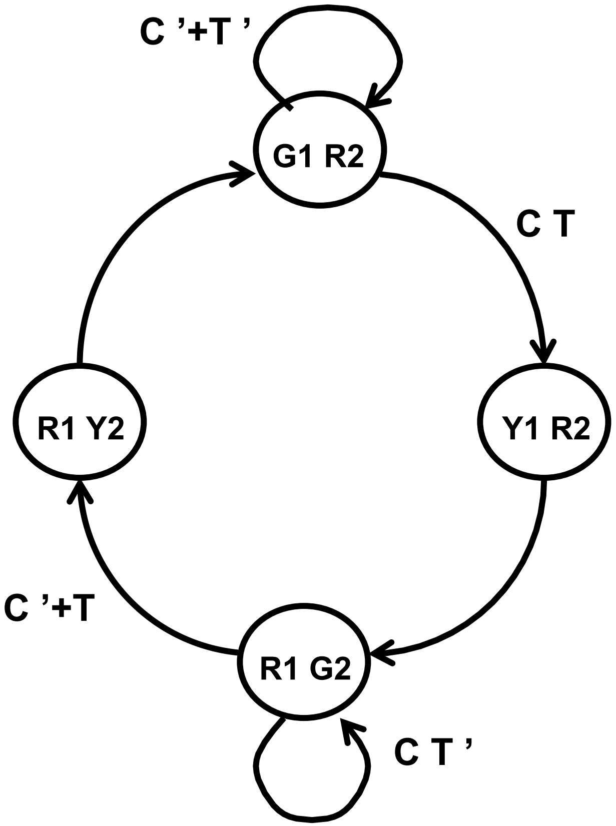
- Model Checking task: check
  - safety condition
  - fairness conditions

- *Safety condition*: no green lights on both roads at the same time

  A G ¬ (G1 ∧ G2)

- *Fairness condition*: eventually one road has green light

  E F (G1 ∨ G2)



Checking conditions

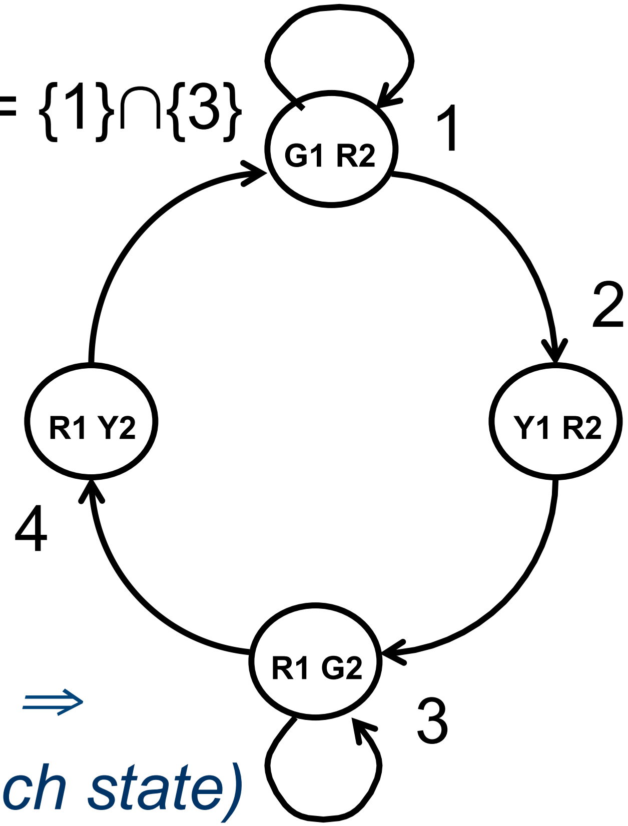
We can associate with each formula the set of states satisfying it

Checking the Safety Condition

$\mathbf{A\ G} \neg (G1 \wedge G2) = \neg\ \mathbf{E\ F}\ (G1 \wedge G2)$

- $S(G1 \wedge G2) = S(G1) \cap S(G2) = \{1\} \cap \{3\} = \varnothing$
- $S(\mathbf{EF}\ (G1 \wedge G2)) = \varnothing$
- $S(\neg\ \mathbf{EF}\ (G1 \wedge G2)) = \neg\varnothing$

$= \{1, 2, 3, 4\}$

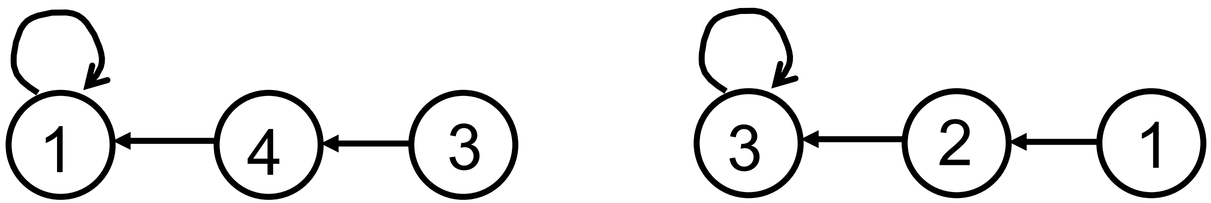


Each state is included in {1,2,3,4} $\Rightarrow$
the safety condition is true (for each state)

Checking the Fairness Condition

E F (G1 v G2) = **E**(*true* **U** (G1 v G2))

- S(G1 v G2) = S(G1)US(G2) = {1} U {3} = {1,3}
- S(**EF** (G1 v G2)) = {1,2,3,4}
  (going *backward* from {1,3}, find predecessors)

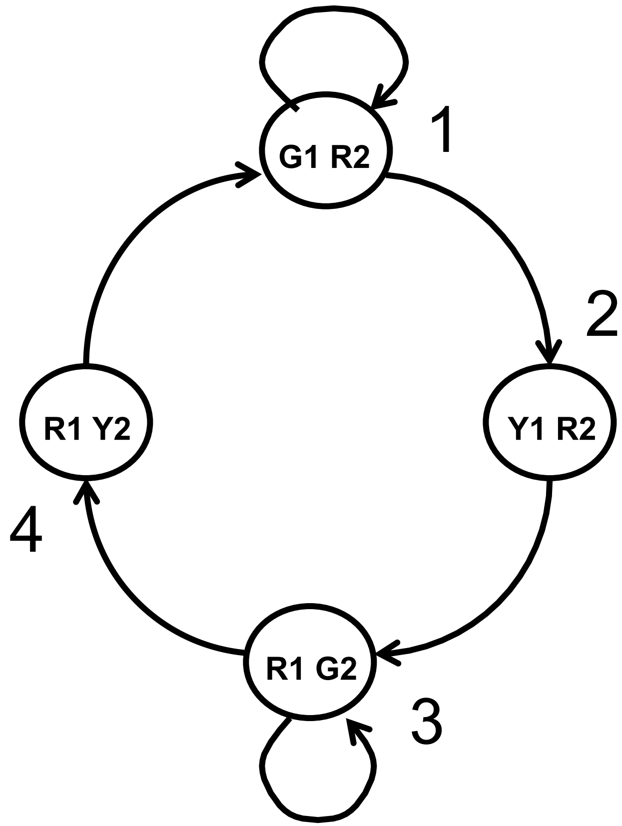


Since {1,2,3,4} contains all states, the condition is true for all the *states*

Another Check

$(\mathbf{E\ X})^2$ (Y1) = **E X** (**E X** (Y1))

(starting at S_1=G1R2, is there a path s.t. Y1 is true in 2 steps ?)



- S (Y1) = {2}
- S (**EX** (Y1)) = {1}
  (predecessor of 2)
- S (**EX** (**EX**(Y1)) = {1,4}
  (predecessors of 1)

Property E X^2 (Y1) *is true for states {1,4}, hence true*

Explicit Model Checking - complexity

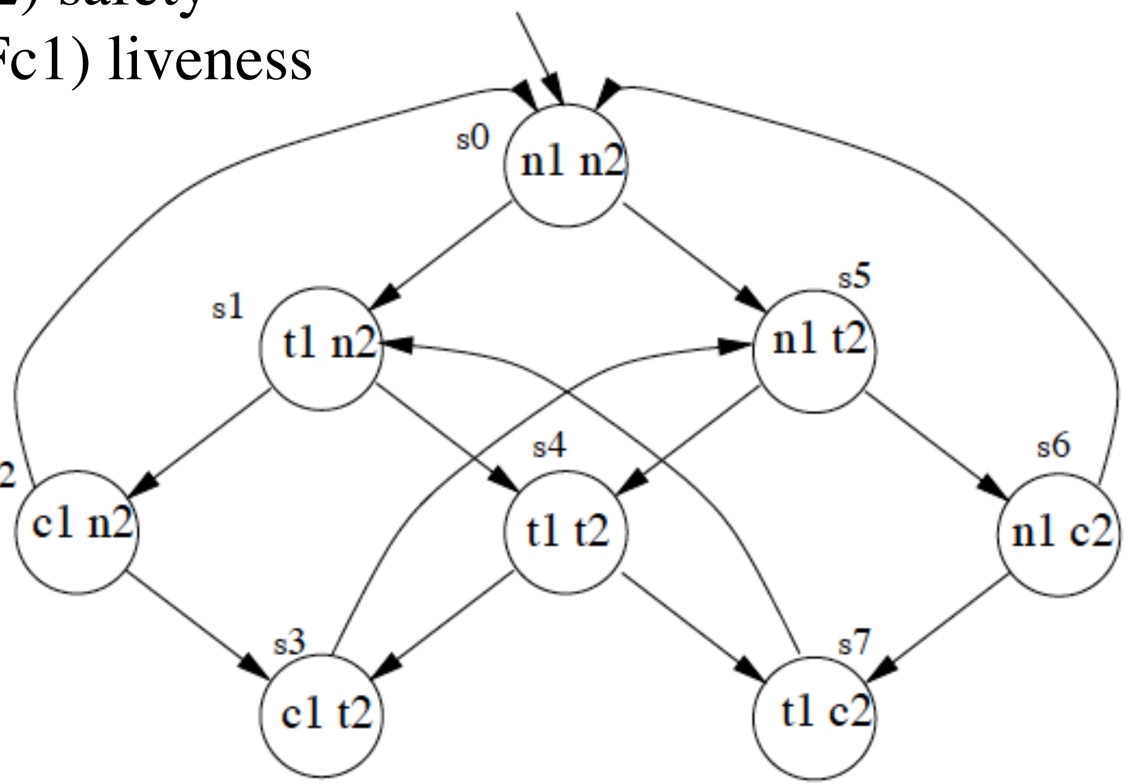
- CTL model checking is *linear* in the size of the formula and the size of the *structure* M
- Not a good news:
  - what if you have 10^{50} states?
  - Number of states grows exponentially with number of variables
  - Explicit model checking limited to … 10^{9} states
- *Symbolic* model checking can do much better

Mutual exclusion again (in CTL)

$AG\neg(c1 \wedge c2)$ safety

$AG(t1 \Rightarrow AFc1)$ liveness

The first modeling attempt



Does it work?

Mutual exclusion again (in CTL)

Lack of fairness may result into a violation of liveness

The problem is solved by using CTL with fairness
i.e. by restricting ourselves to fair paths

In LTL $GF\,\phi$ or $GF\psi \rightarrow GF\phi$ but not in CTL

CTL with fairness

- Formally, we consider the problem where we are given K and ϕ as before and additionally a fairness constraint $F \subseteq S$.
- We call a run fair (w.r.t. F) iff it contains infinitely many states from F.
- The problem is to compute $S_K(\phi)$ for the case where the operators *EG* and *EU* consider only fair runs (w.r.t. F).
- Therefore, we introduce the following modified operators:

$S_K(EG_f\ \phi) = \{\ s \mid \exists$ **a fair run ρ of K** s.t.
$\rho(0) = s$ and $\forall i \geq 0,\ \rho(i) \in S_K(\phi)\ \}$

$S_K(\phi_1\ EU_f\ \phi_2) = \{\ s \mid \exists$ **a fair run ρ of K** s.t.
$\rho(0) = s$ and $\exists i$ such that $\rho(i) \in S_K(\phi_2)$ and $\forall k < i,\ \rho(k) \in S_K(\phi_1)\ \}$

CTL with fairness

- First, observe that fair runs have the following properties:
  1. ρ is a fair run iff ρ^i is fair for all $i \geq 0$.
  2. ρ is a fair run iff ρ has a fair suffix ρ^i for some i.
- Using this, we can rewrite the EU_f operator as follows:

$$\phi_1\ EU_f\ \phi_2 \quad \equiv \quad \phi_1\ EU\ (\phi_2 \wedge EG_f\ true)$$

- Thus, it is enough to provide a new algorithm for EG_f

Exercises

Express the following properties in CTL and LTL whenever possible. If neither is possible, try to express the property in CTL*:

(a) Whenever p is followed by q (after finitely many steps), then the system enters an 'interval' in which no r occurs until t.

(b) Event p precedes s and t on all computation paths. (You may find it easier to code the negation of that specification first.)

(c) After p, q is never true. (Where this constraint is meant to apply on all computation paths.)

(d) Between the events q and r, event p is never true.

(e) Transitions to states satisfying p occur at most twice.

(f) Property p is true for every second state along a path.