Automating safety property verification, intro to liveness

Inductive invariants

A property p of a transition system S is an *inductive invariant of* S if:

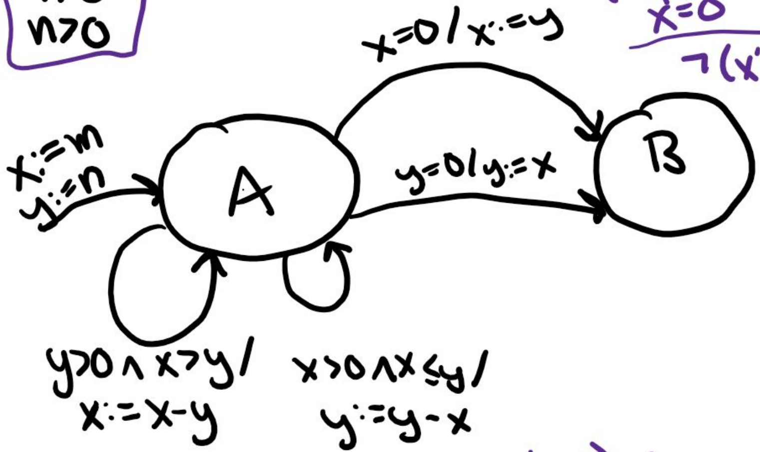
1. The initial state s satisfies p, and
2. If a state s satisfies p, and (s, t) is a transition, then the state t also satisfies p

(Board discussion: Prove $(x >= 0 \wedge y > 0) \vee (x > 0 \wedge y >= 0)$)

Invariant

Monday, November 28, 2022 10:45 AM

$m > 0$
$n > 0$



$(y' > 0 \land x' \geq 0) \lor$
$(x' > 0 \land y' \geq 0)$
$x = 0$

$\neg(x' > 0 \land y' \geq 0)$
$(y' > 0 \land x' \geq 0)$
$y' > 0 \Rightarrow y > 0, x > 0$ ✓

$(y' > 0 \land x' \geq 0) \lor$
$(x' > 0 \land y' \geq 0),$
$y' > 0, \; x' > y'$

$y > 0$
$x' > y'$
$x = x' - y' > 0$ ✓

$(y' > 0 \land x' \geq 0) \lor$
$(x' > 0 \land y' \geq 0)$
$x' > 0, \; x' \leq y'$

$x > 0$
$x' \leq y'$
$0 \leq y' - x' = y$

$p = (y > 0 \land x \geq 0) \lor$
$(x > 0 \land y \geq 0)$

① base case

A, m, n ✓

② inductive case

Proving non-inductive invariants

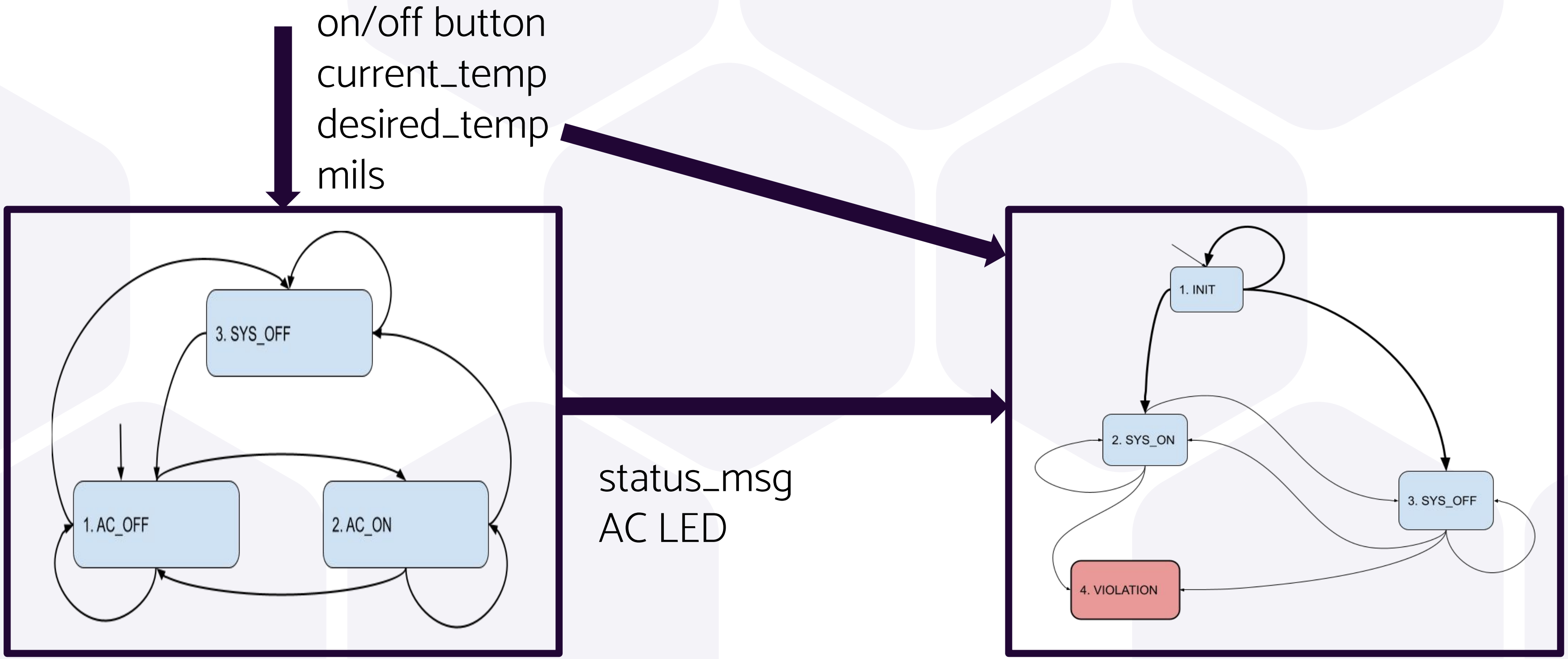
To establish that a property *p* is an invariant of the transition system *S*, find a property *q* that:

1. *q* is an inductive invariant of *S*, and
2. the property *q* implies the property *p* (that is, a state satisfying *q* is guaranteed to satisfy *p*)

(Board discussion: Prove B => x > 0 ^ y > 0)

How would you deal with this invariant?

*8) If the system is on and the control knob hasn't changed for 290 ms, the desired temperature as sent by status message obeys the formula 5400 + 25 * (control knob reading) / 8 with an error of at most 3 degrees F (300 centidegrees).*


on/off button
current_temp
desired_temp
mils
3. SYS_OFF
1. AC_OFF
2. AC_ON
status_msg
AC LED
1. INIT
2. SYS_ON
3. SYS_OFF
4. VIOLATION

Stateful invariants

For a transition system *S*, Create a *safety monitor FSM* called *M* where:

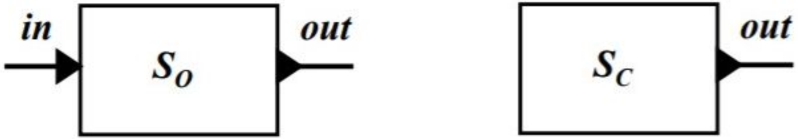
- inputs of *M* are a subset of the inputs and outputs of *S*
- Some subset *E* of the states of *M* are designated as "error" states
- The behavior of *M* is designed such that if the sequence of inputs to *M* leads *M* to an error state in *E*, this is an invariant violation

Compose *M* and *S*. The invariant becomes that any state in *E* is not reachable

What similarities do you see between the safety monitor FSM definition and the runtime monitor you wrote in lab 8?

Open and closed systems

To automate invariant verification, we need to work with a closed system



(a) Open system (b) Closed system

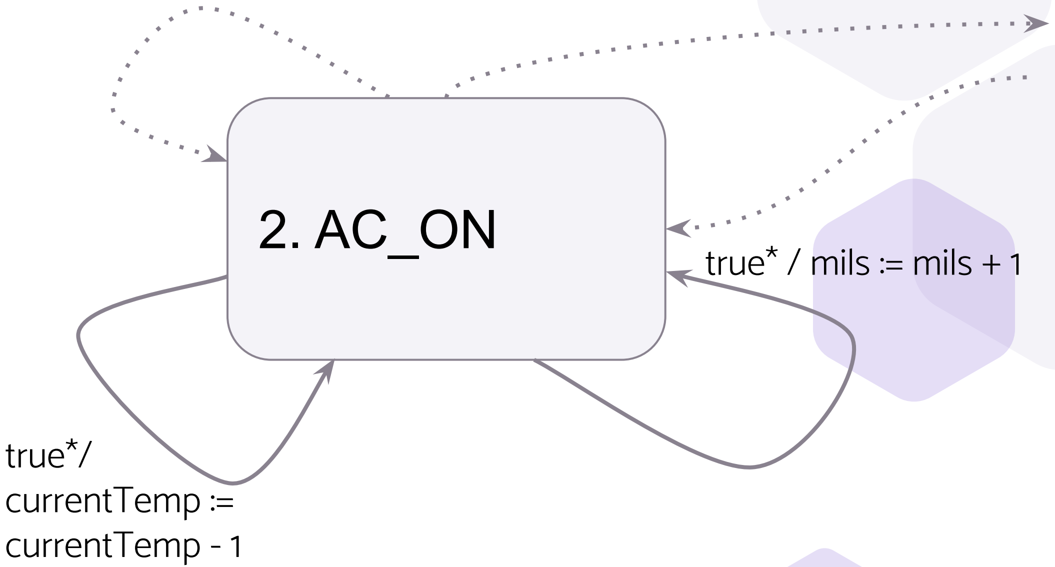
Figure 15.1: Open and closed systems.

[Lee/Seshia, chapter 15]

Reminder: closed AC model

Environment:

- Time
- Button
- Current temp
- Desired temp



Note: for the logics/computation models we are talking about here, we are using *discrete* systems (but not necessarily deterministic!)

Automated reachability analysis

A property *p* of a transition system* *S* is an *invariant* of *S* if every **reachable** state of *S* satisfies *p*

How would you automatically determine the set of reachable states?

Assume a system of finite states

(Verification for a system of infinite states is *undecidable*)

Depth-first search

Input : Initial state s_0 and transition relation δ for closed finite-state system M

Output: Set R of reachable states of M

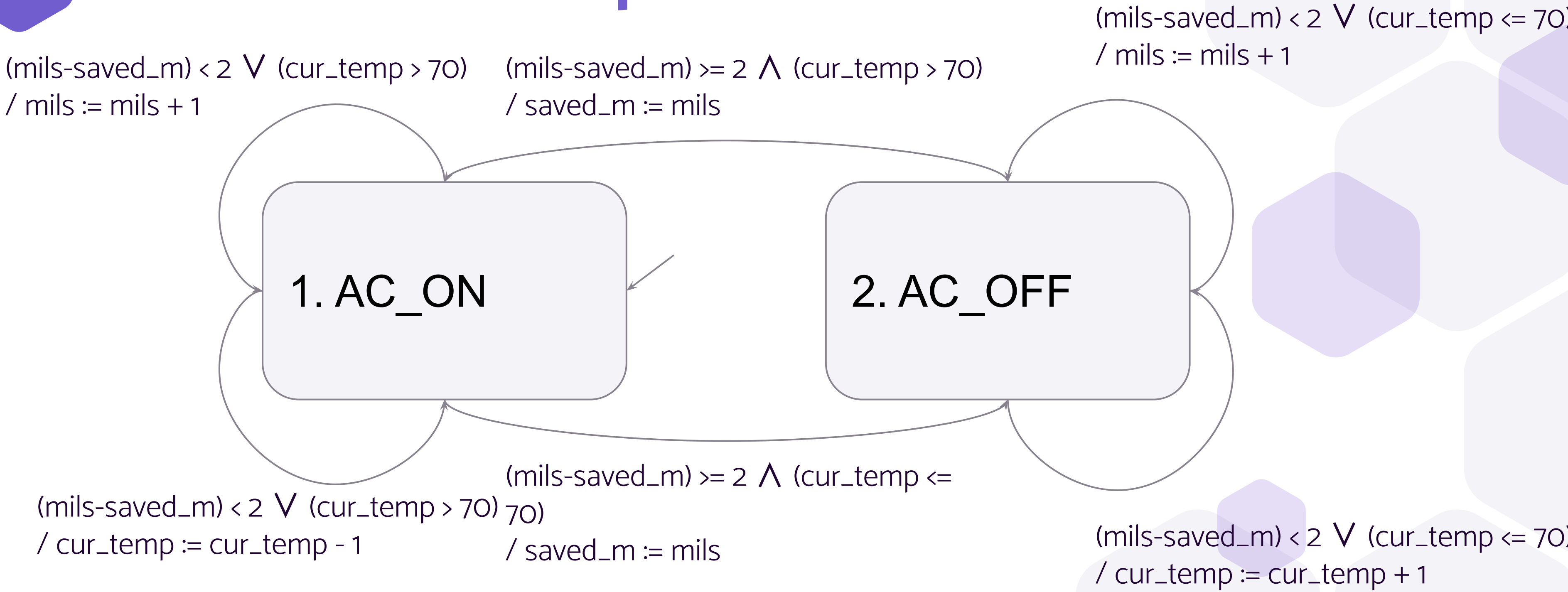
Initialize: *Stack* Σ *to contain a single state* s_0; *Current set of reached states* $R := \{s_0\}$.

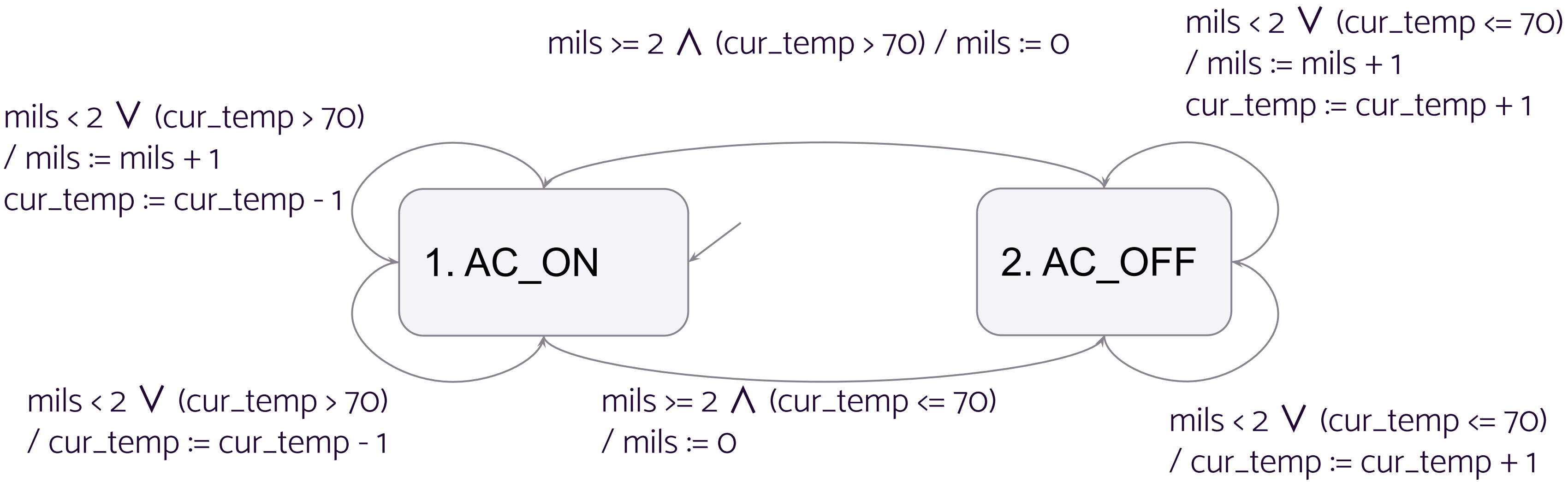
DFS_Search() {
while *Stack* Σ *is not empty* **do**
    Pop the state s at the top of Σ
    Compute $\delta(s)$, the set of all states reachable from s in one transition
    for *each* $s' \in \delta(s)$ **do**
        if $s' \notin R$ **then**
            $R := R \cup \{s'\}$
            Push s' onto Σ
        end
    end
end
}

Algorithm 15.1: Computing the reachable state set by depth-first explicit-state search.

[Lee/Seshia, chapter 15]

DFS board example for AC




mils >= 2 ⋀ (cur_temp > 70) / mils := 0
mils < 2 ⋁ (cur_temp <= 70)
/ mils := mils + 1
cur_temp := cur_temp + 1
mils < 2 ⋁ (cur_temp > 70)
/ mils := mils + 1
cur_temp := cur_temp - 1
1. AC_ON
2. AC_OFF
mils < 2 ⋁ (cur_temp > 70)
/ cur_temp := cur_temp - 1
mils >= 2 ⋀ (cur_temp <= 70)
/ mils := 0
mils < 2 ⋁ (cur_temp <= 70)
/ cur_temp := cur_temp + 1

How would you modify the DFS algorithm to either produce a "YES" or a counterexample for a property p?

Reference for DFS question

Input : Initial state s_0 and transition relation δ for closed finite-state system M
Output: Set R of reachable states of M

Initialize: *Stack* Σ *to contain a single state* s_0*; Current set of reached states* $R := \{s_0\}$.

DFS_Search() {
while *Stack* Σ *is not empty* **do**
    Pop the state s at the top of Σ
    Compute $\delta(s)$, the set of all states reachable from s in one transition
    for *each* $s' \in \delta(s)$ **do**
        if $s' \notin R$ **then**
            $R := R \cup \{s'\}$
            Push s' onto Σ
        end
    end
end
}

Algorithm 15.1: Computing the reachable state set by depth-first explicit-state search.

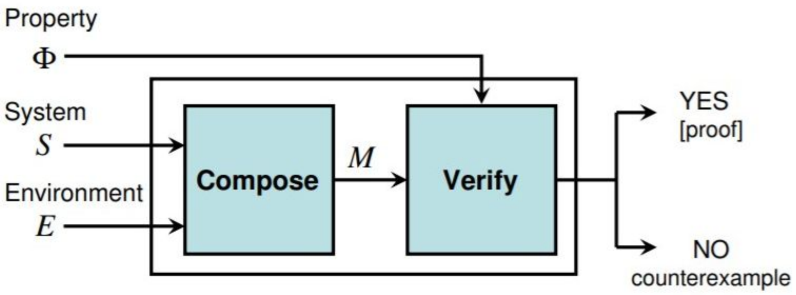


Figure 15.2: Formal verification procedure.

Safety requirements vs liveness requirements

Safety: nothing bad *ever* happens

Liveness: something good *eventually* happens

- Means system is functioning as intended
- System requirements are often liveness requirements

What are some liveness requirements for the AC?

*How would you **monitor** that a liveness requirement is fulfilled?*

Verifying some liveness properties

Saying something *eventually* happens is the same thing as saying that it is *not* the case that it always *doesn’t* happen