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 $(x \geq 0 \land y > 0) \lor (x > 0 \land y \geq 0)$)
Invariant

\[ (y' > 0 \land x' > 0) \lor (x > 0 \land y' > 0) \]
\[ x = 0 \]
\[ \neg (x' > 0 \land y' > 0) \]
\[ y > 0 \Rightarrow y > 0, x > 0 \]

\[ p = (y' > 0 \land x' > 0) \lor (x > 0 \land y' > 0) \]

1. base case
   \[ A, m, n \]
   \[ \checkmark \]

2. inductive case
   \[ x > 0, x' \leq y' \]
   \[ 0 \leq y' - x' = y \]
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 $B \Rightarrow x > 0 \land y > 0$)
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 $5400 + 25 \times (\text{control knob reading}) / 8$ with an error of at most 3 degrees F (300 centidegrees).
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

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 \textit{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 $\delta$ for closed finite-state system $M$
Output: Set $R$ of reachable states of $M$

1 Initialize: Stack $\Sigma$ to contain a single state $s_0$; Current set of reached states $R := \{s_0\}$.
2 \textbf{DFS_Search()} \{ 
3 \textbf{while Stack }$\Sigma$\textbf{ is not empty do} 
4 \hspace{1em} Pop the state $s$ at the top of $\Sigma$
5 \hspace{1em} Compute $\delta(s)$, the set of all states reachable from $s$ in one transition
6 \hspace{1em} \textbf{for each }$s' \in \delta(s)$\textbf{ do} 
7 \hspace{2em} \textbf{if }$s' \notin R$\textbf{ then} 
8 \hspace{3em} $R := R \cup \{s'\}$
9 \hspace{3em} Push $s'$ onto $\Sigma$
10 \hspace{1em} \textbf{end}
11 \hspace{1em} \textbf{end}
12 \textbf{end}
13 \}

Algorithm 15.1: Computing the reachable state set by depth-first explicit-state search.
DFS board example for AC

1. AC_ON

(mils-saved_m) < 2 ∨ (cur_temp > 70)
/ mils := mils + 1

(mils-saved_m) < 2 ∨ (cur_temp <= 70)
/ cur_temp := cur_temp - 1

(mils-saved_m) >= 2 ∧ (cur_temp > 70)
/ saved_m := mils

(mils-saved_m) >= 2 ∧ (cur_temp <= 70)
/ cur_temp := cur_temp - 1

2. AC_OFF

(mils-saved_m) < 2 ∨ (cur_temp <= 70)
/ mils := mils + 1
1. AC_ON

2. AC_OFF

\[
mils \geq 2 \land (\text{cur\_temp} > 70) / mils := 0
\]

\[
mils < 2 \vee (\text{cur\_temp} > 70) / mils := \text{mils} + 1
\]

\[
\text{cur\_temp} := \text{cur\_temp} - 1
\]

\[
mils \geq 2 \land (\text{cur\_temp} \leq 70) / \text{mils} := 0
\]

\[
mils < 2 \lor (\text{cur\_temp} \leq 70) / \text{mils} := \text{mils} + 1
\]

\[
\text{cur\_temp} := \text{cur\_temp} + 1
\]

\[
mils < 2 \lor (\text{cur\_temp} > 70) / \text{cur\_temp} := \text{cur\_temp} - 1
\]

\[
mils < 2 \lor (\text{cur\_temp} > 70) / \text{mils} := \text{mils} + 1
\]

\[
\text{cur\_temp} := \text{cur\_temp} - 1
\]
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 $\delta$ for closed finite-state system $M$

Output: Set $R$ of reachable states of $M$

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

2. DFS_Search() {
   3. while Stack $\Sigma$ is not empty do
      4. Pop the state $s$ at the top of $\Sigma$
      5. Compute $\delta(s)$, the set of all states reachable from $s$ in one transition
      6. for each $s' \in \delta(s)$ do
         7. if $s' \notin R$ then
            8. $R := R \cup \{s'\}$
            9. Push $s'$ onto $\Sigma$
      10. end
   11. end
   12. }

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