CS:4980
Foundations of Embedded Systems

Safety Requirements
Part III

Requirements-based Design

Systematic approach to design of systems

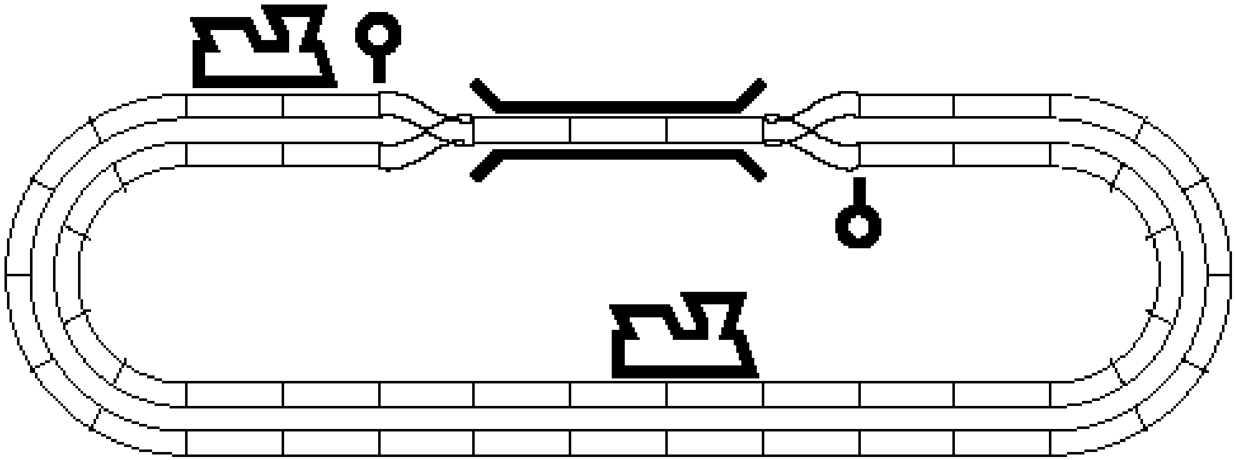
Given:

- Input/output interface of system/component C to be designed
- Model E of the environment
- Safety properties $P_1, \dots, P_n$ of the composite system

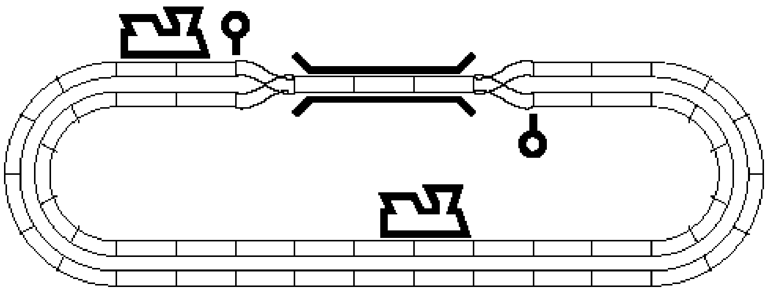
Design problem:

- Fill in details of C (state variables, initialization, and update) so that $P_1, \dots, P_n$ are invariant for $C \| E$

Railroad Controller Example

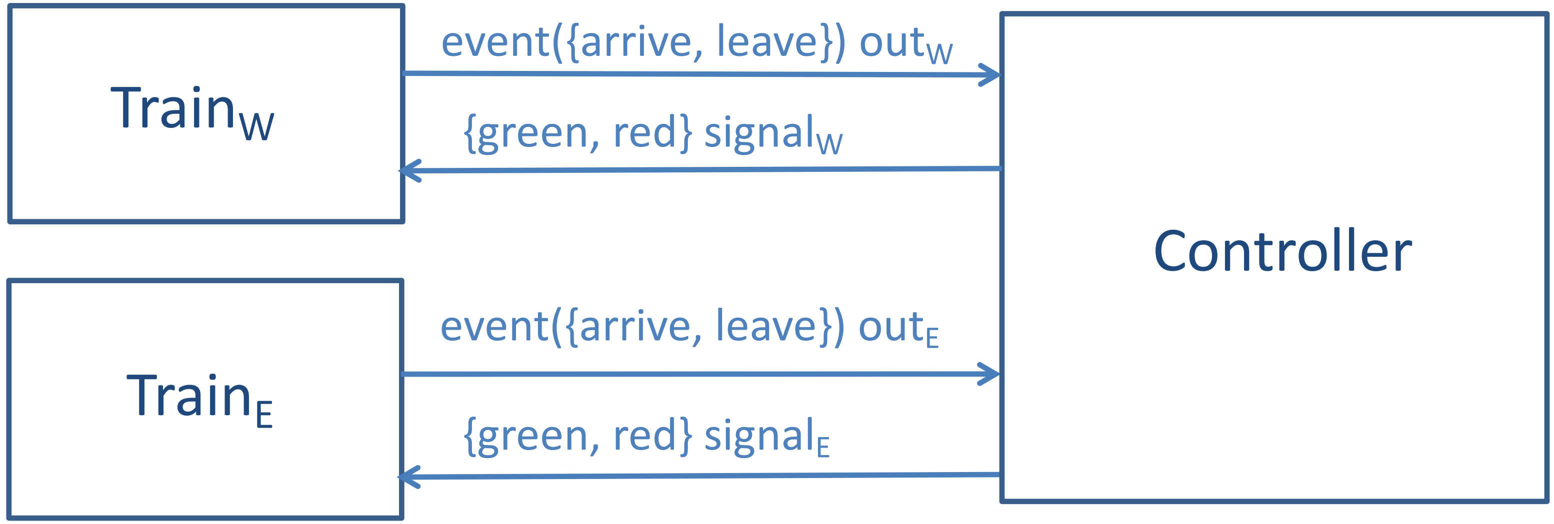


Train Model



- Each train is initially away from bridge
- Train can be in away state for an arbitrarily long period
- When the train gets close, it communicates with the traffic controller via an event, say, arrive, and now it is in a different state, say, wait
- When near, train is monitoring the signal on the bridge:
  - If the signal is green, it enters the bridge
  - If the signal is red, it continues to wait
- A train can stay on bridge for a duration that is no exactly known (and not directly under the control of the traffic controller)
- When the train leaves the bridge, it communicates with the controller via an event, say, leave, and goes back to away state
- This behavior repeats: an away train may again request entry
- The two trains have symmetric behavior

Controller Design Problem

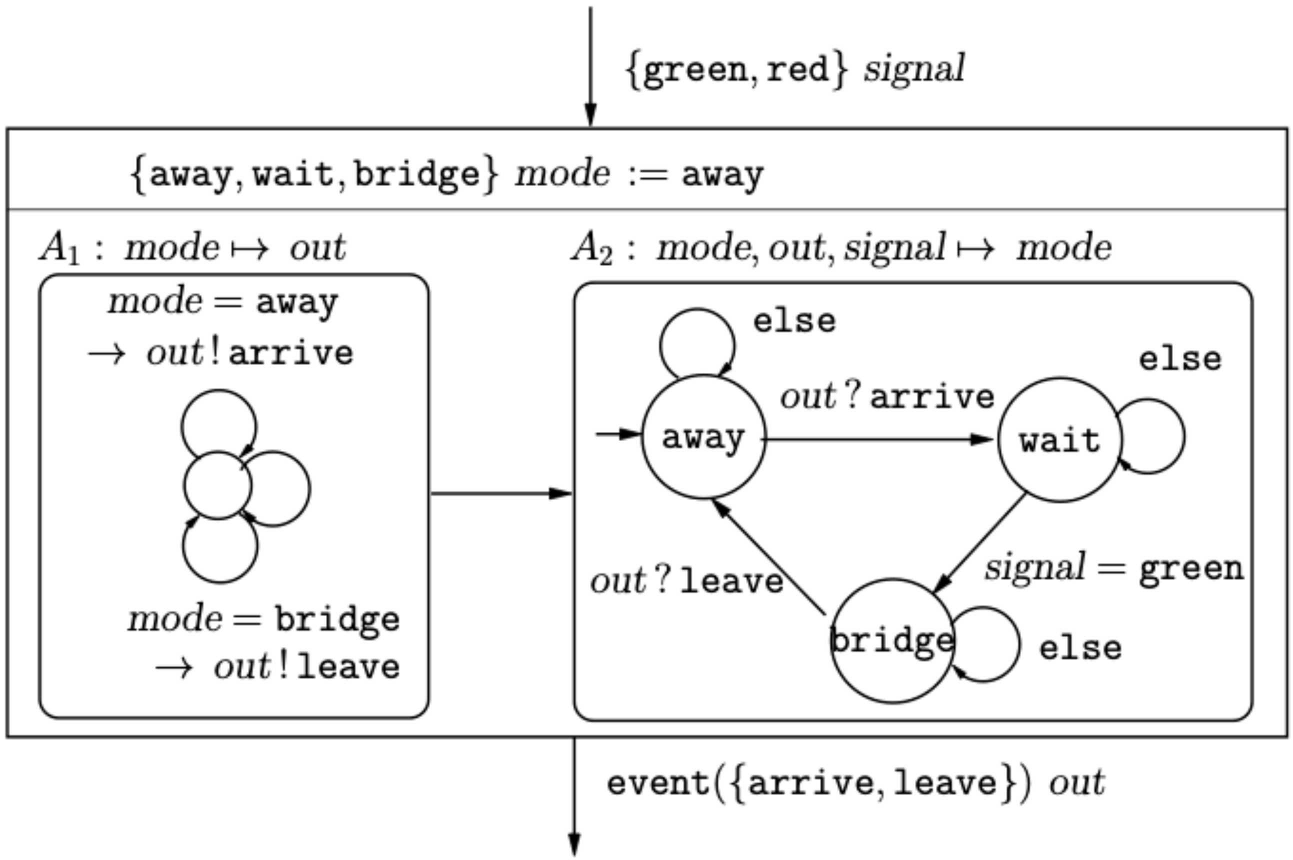


Safety Requirement: Trains should not be on bridge simultaneously

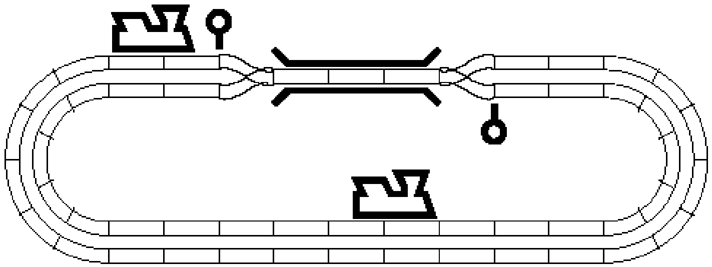
Formally, the following should be an invariant of the system:

$$\neg(mode_W = bridge \ \wedge \ mode_E = bridge)$$

Synchronous Component Train



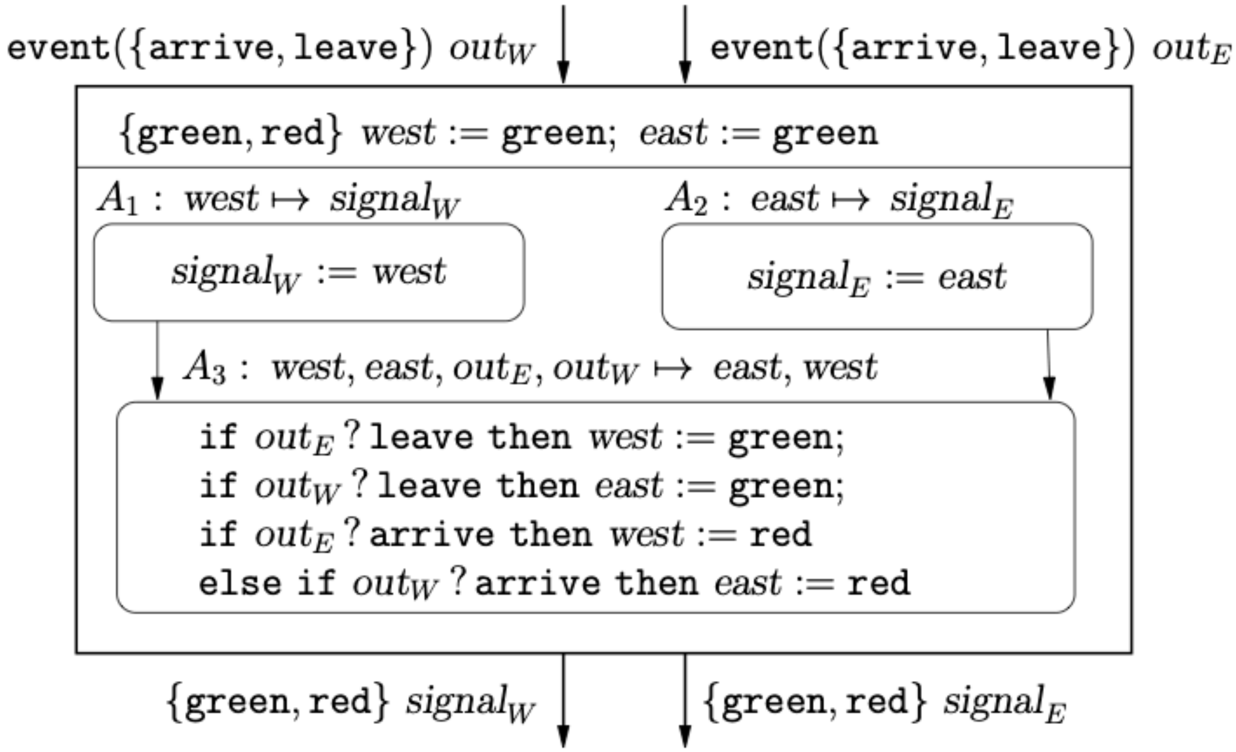
First Attempt at Controller Design

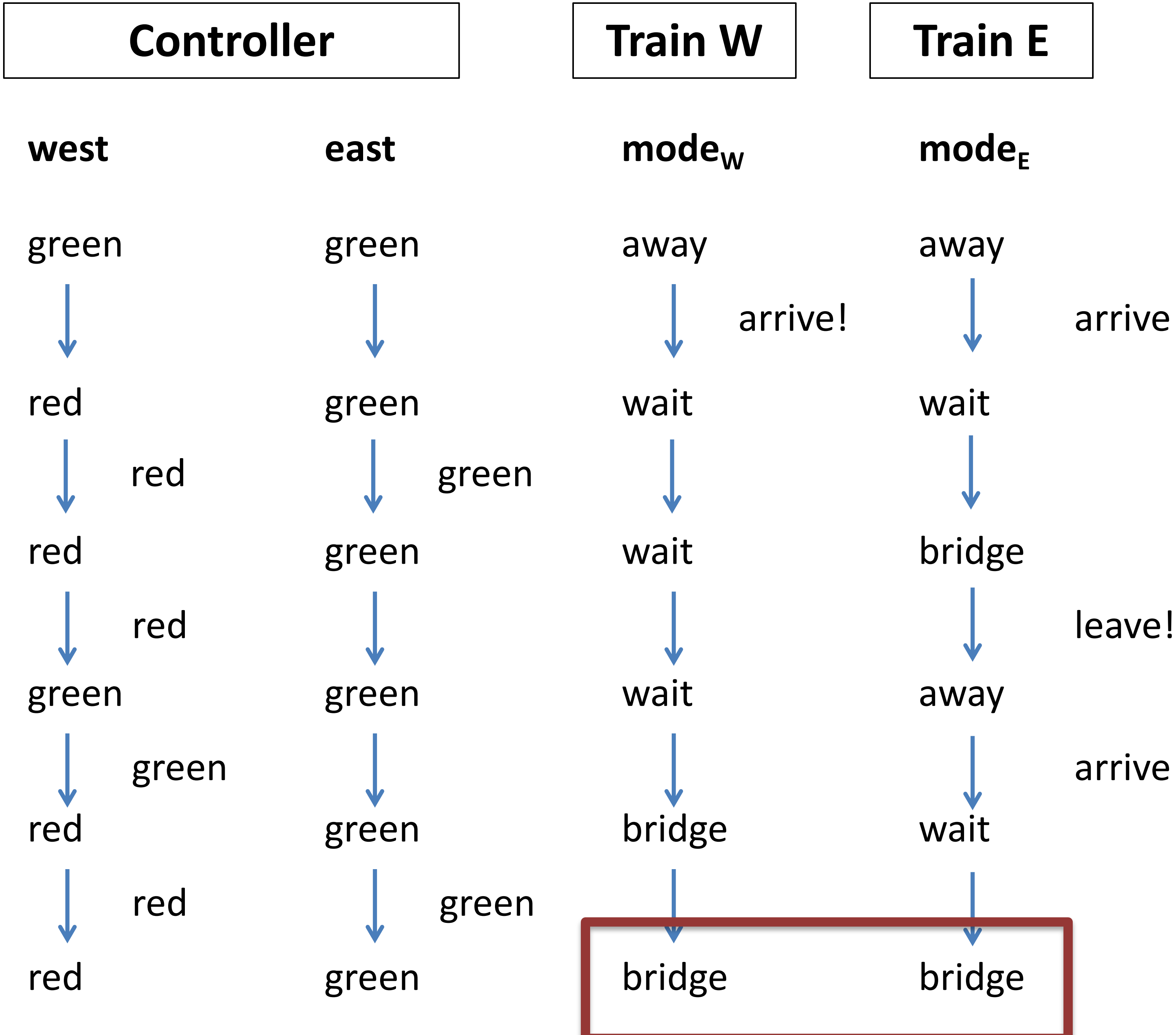


- Controller maintains state variables east, west to track the state of each signal
- Both state variables are initially green
- The output for the signals is based on the corresponding state vars
- When a train arrives, update the opposite signal var to red to block the other train from entering the bridge
- When a train leaves, reset the opposite signal var to green
- What happens if both trains arrive simultaneously?

  Give priority to east train: set west signal var to red

Synchronous Component Controller1

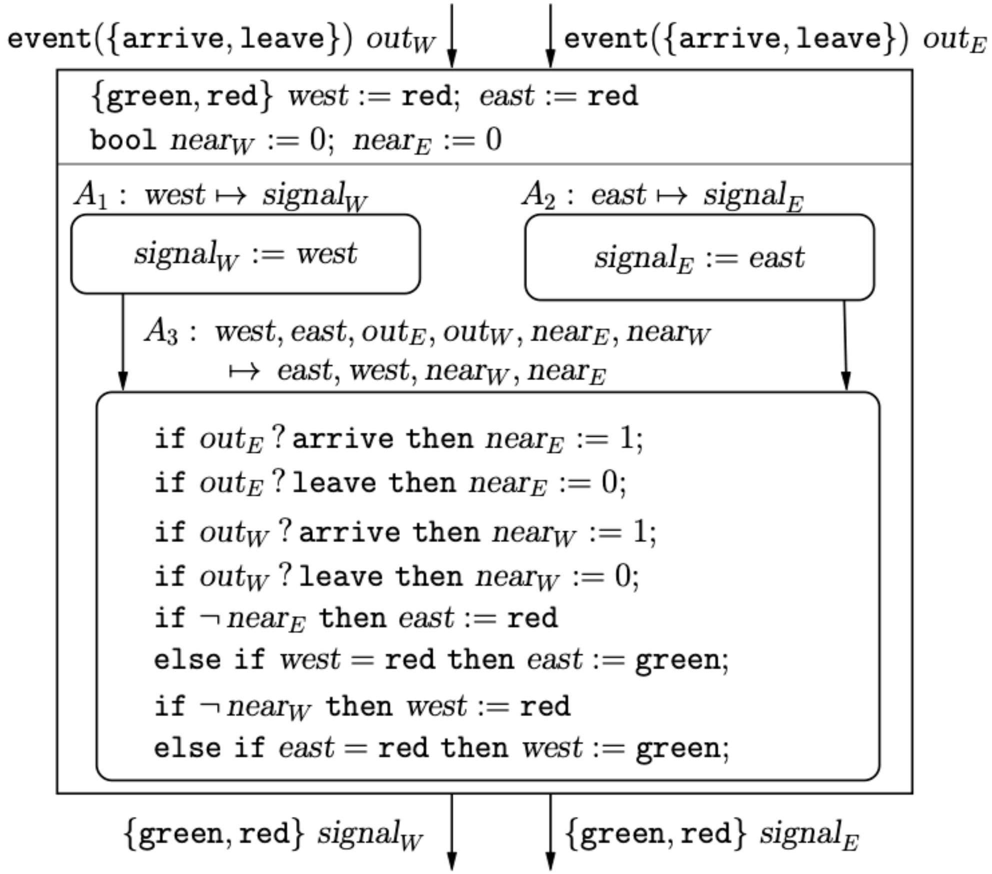



Controller
Train W
Train E
west
east
modeW
modeE
green
green
away
away
arrive!
arrive!
red
green
wait
wait
red
green
red
green
wait
bridge
red
leave!
green
green
wait
away
green
arrive!
red
green
bridge
wait
red
green
red
green
bridge
bridge

Second Attempt at Controller Design

- What went wrong the first time? Controller did not remember whether a train was waiting at each entrance
- Boolean variable $near_W$ remembers whether the west train wants to use the bridge
  - Initially 0
  - When the west train issues arrive, changed to 1
  - When the west train issues leave, reset back to 0
- Invariant: $mode_W = away \Leftrightarrow near_W = 0$
- Variable $near_E$ is symmetric
- Let's also start with both signals red
- A signal is changed to green if the corresponding train is near and the other signal is red; it is changed back to red when train is away
- Need still to resolve simultaneous arrivals by preferring one train

Second Attempt at Controller Design



Properties of Controller2

- The system RailRoadSystem2 = Controller2 || $Train_W$ || $Train_E$ satisfies the safety property

$$\neg(mode_W = bridge \ \wedge \ mode_E = bridge)$$

- What about some additional properties?
  1. *If the west train is waiting, then west signal will eventually become green*
  2. *If the west train is waiting for its signal to turn green, other train should not be allowed on bridge more than once*
- Requirement 1 is a liveness requirement (see Chap. 5 of text)
- Requirement 2 is a safety requirement
  - Its violation can be demonstrated by a (finite) execution in which east train enters, leaves, and enters again while west train keeps waiting with its signal red
  - But it cannot be encoded as an invariant on system state vars!

Safety Monitor

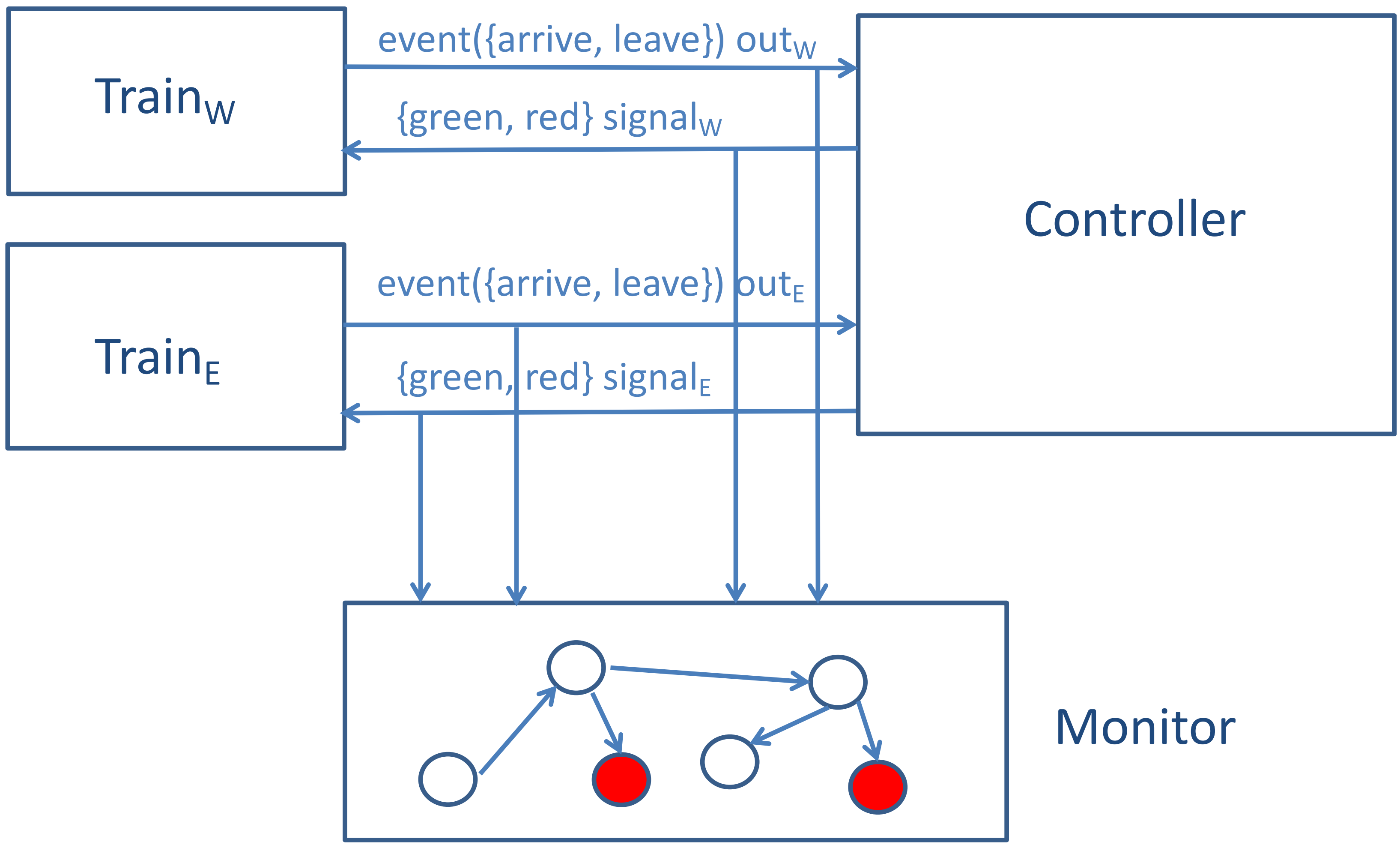
- Monitor M for a system observes its inputs/outputs, and enters an error state if undesirable behavior is detected

- Monitor M is specified as extended state machine
  1. The set of input variables of M = input/output variables of system being monitored
  2. An output of M cannot be an input to system (monitor does not influence what the system does)
  3. A subset F of modes of state-machine declared as accepting

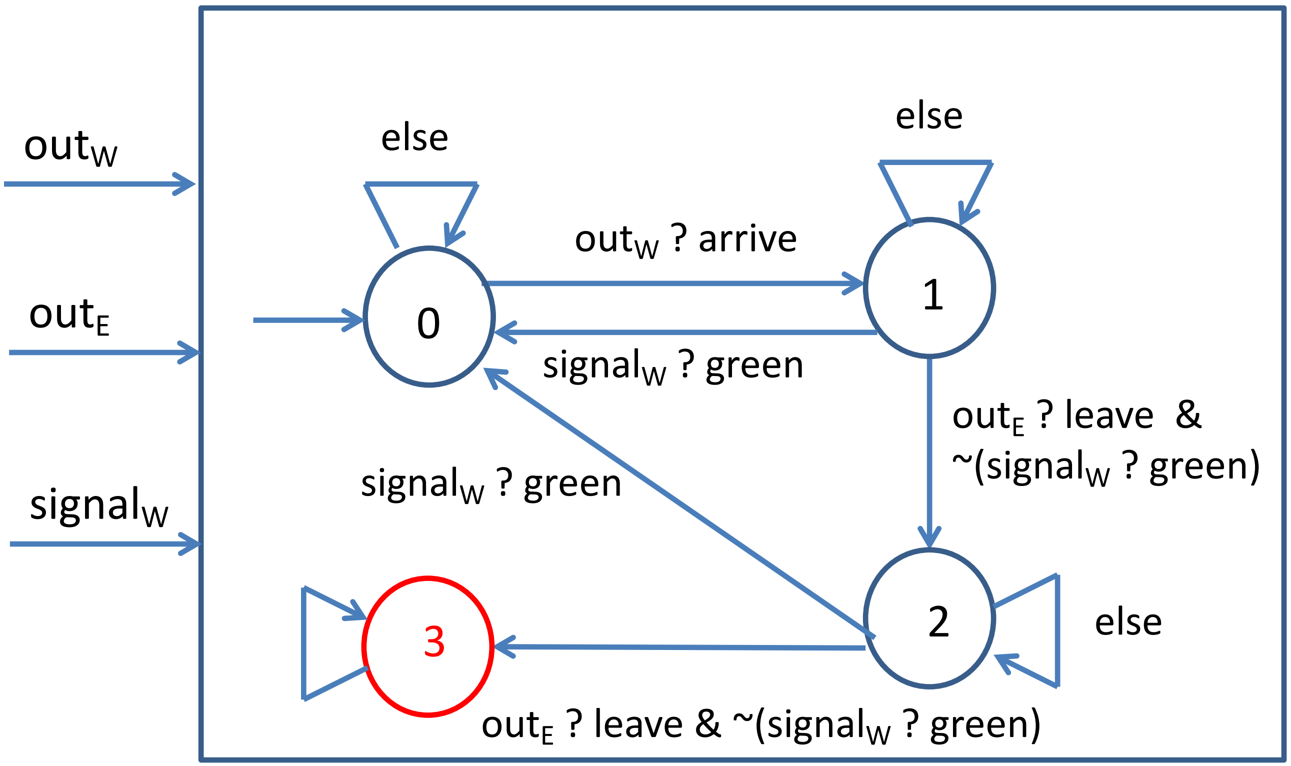
- Undesirable behavior: An execution that leads monitor state to F

- Safety verification: Check whether (M.mode not in F) is an invariant of system C || M

Safety Monitors



Monitor to check fairness for railroad



Error execution:

As west train waits, east train is allowed on bridge twice

Exercise: Leader Election

- Suppose we want to check that at most one of the nodes declares itself to be the leader
- Design a monitor M
  - Input variables: {undecided, leader, follower} $status_n$, for each node n
  - M should enter error state iff for two distinct nodes m and n
    1. there exists a round r_1 in which $status_m$ = leader and
    2. there exists a round r_2 in which $status_n$ = leader
- Consider the requirement: eventually $status_n$!= undecided

  Why can't we design a monitor that enters an error state if this requirement is violated?

Credits

Notes based on Chapter 3 of

Principles of Cyber-Physical Systems
by Rajeev Alur
MIT Press, 2015