Cyber-Physical Systems (CSC.T431)

Safety Requirements (1)

Instructor: Takuo Watanabe (Department of Computer Science)

Agenda

- Safety Requirements (1)

Course Support & Material

- Slides: OCW-i
- Course Web: https://titech-cps.github.io
- Course Slack: titech-cps.slack.com

Safety Requirements

- Safety requirements: *Nothing bad ever happens*
  - Ex. No two threads can run in the critical section at the same time.
  - The negation of "something bad" is an *invariant* of a *transition system* that models the behaviors of the target system.

- Cf. Liveness requirements: *Something good eventually happens*
  - Ex. If a thread tries to enter the critical section, it will eventually be able to enter.

Transition Systems

- A *transition system* T is a triple $(S, Init, Trans)$ where:
  - S : a finite set of typed *state variables* defining the set Q_S of *states,*
  - $Init$: an *initialization* defining the set $[\![Init]\!] \subseteq Q_S$ of *initial states, and*
  - $Trans$: a *transition description* defining the set $[\![Trans]\!] \subset Q_S \times Q_S$ of *transition* between states.

SRCs as Transition Systems

- $C = (I, O, S, Init, React)$: synchronous reactive component
- We can naturally construct a transition system $(S, Init, Trans)$ in which $Trans$ is obtained from $React$ as follows.
  - Variables in I and O are declared as local variables in $Trans$
  - The values of variables in I are nondeterministically chosen (at each transition).
  - $(s, t) \in [\![Trans]\!]$ if $s \xrightarrow{i/o} t \in [\![React]\!]$ for some $i \in Q_I$ and $o \in Q_O$.

SRCs as Transition Systems

Ex. TriggeredCopy

event(bool) *in* →

nat $x := 0$

if in? **then**
  $\{\ out!\ in;\ x := x + 1\ \}$

→ event(bool) *out*

$$(\{x\}, Init, Trans)$$

local event(bool) in, out;
  in := choose $\{0,1,\bot\}$
  if in? **then**
    $\{out\ !\ in;\ x := x + 1\}$

$$[\![Init]\!] = \{0\}$$

$$[\![Trans]\!] = \{(n,n) \mid n \in \mathbb{N}\} \cup \{(n,n+1) \mid n \in \mathbb{N}\}$$

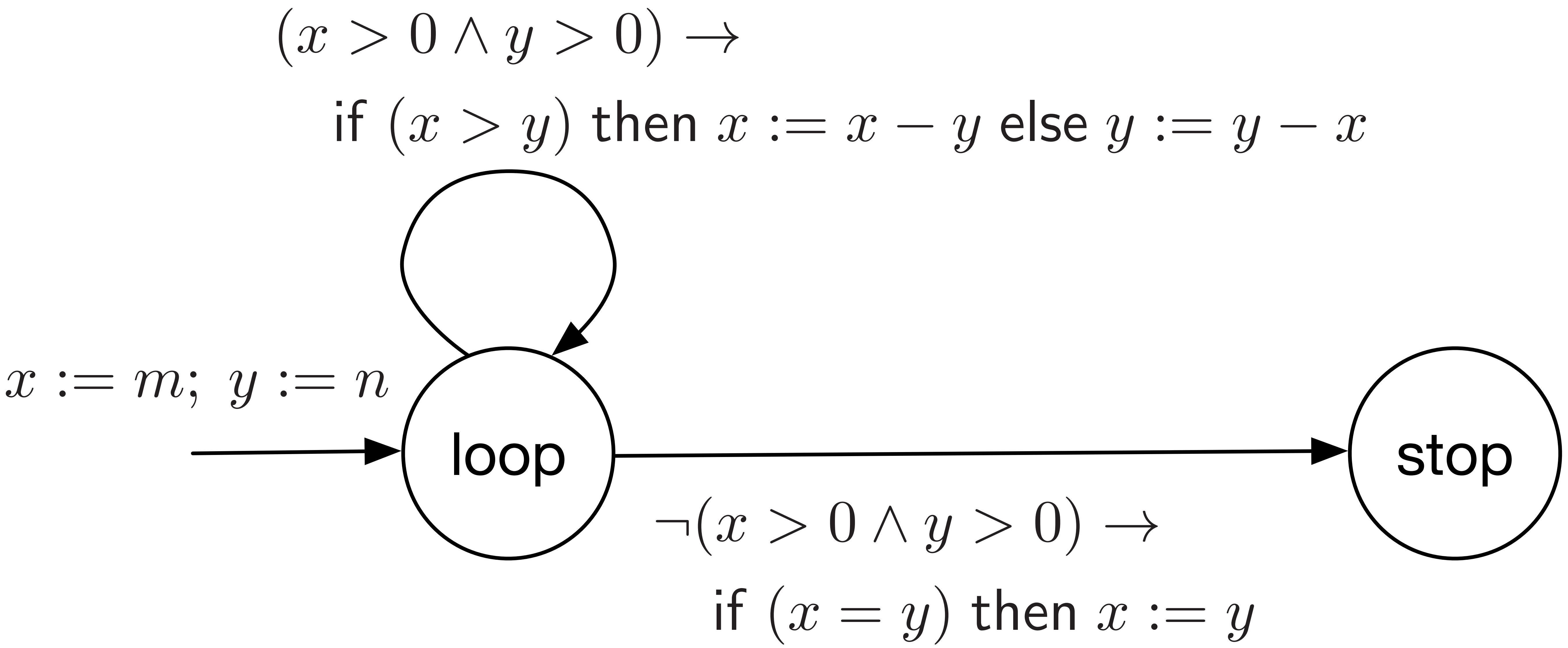
$$\begin{array}{r|ccccccccccccccc}
React & 0 & \xrightarrow{\bot/\bot} & 0 & \xrightarrow{0/0} & 1 & \xrightarrow{1/1} & 2 & \xrightarrow{1/1} & 2 & \xrightarrow{\bot/\bot} & 2 & \xrightarrow{\bot/\bot} & 2 & \xrightarrow{1/1} & 3 \\
\hline
Trans & 0 & \longrightarrow & 0 & \longrightarrow & 1 & \longrightarrow & 2 & \longrightarrow & 2 & \longrightarrow & 2 & \longrightarrow & 2 & \longrightarrow & 3
\end{array}$$

Programs as Transition Systems

Ex. GCD

GCD(m, n):

nat $x := m, y := n$;
while $(x > 0 \wedge y > 0)$
    if $(x > y)$ then $x := x - y$
        else $y := y - x$;
if $(x = 0)$ then $x := y$



$$T_{\mathrm{gcd}} = (\{x, y, mode\}, Init, Trans)$$

$$[\![Init]\!] = \{(m, n, \text{loop})\} \quad (m, n \in \mathbb{N})$$

$$\begin{aligned}[\![Trans]\!] = &\ \{((j, k, \text{loop}), (j-k, k, \text{loop})) \mid j, k \in \mathbb{N} \wedge j > 0 \wedge k > 0 \wedge j > k\} \\ &\cup \{((j, k, \text{loop}), (j, k-j, \text{loop})) \mid j, k \in \mathbb{N} \wedge j > 0 \wedge k > 0 \wedge j \leq k\} \\ &\cup \{((0, k, \text{loop}), (k, k, \text{stop})) \mid k \in \mathbb{N}\} \cup \{((j, 0, \text{loop}), (j, 0, \text{stop})) \mid j \in \mathbb{N}\}\end{aligned}$$

Execution and Reachable States

- Let $T = (S, Init, Trans)$ be a transition system. An *execution* of T is a finite sequence of the form $s_0, s_1, \ldots, s_k$ such that:
    1. $s_j \in Q_S$ for $0 \leq j \leq k$,
    2. $s_0 \in [\![Init]\!]$, and
    3. $(s_{j-1}, s_j) \in [\![Trans]\!]$ for $1 \leq j \leq k$.
- Ex. An execution of T_{gcd}:
    - $(6,4,\text{loop}) \rightarrow (2,4,\text{loop}) \rightarrow (2,2,\text{loop}) \rightarrow (2,0,\text{loop}) \rightarrow (2,0,\text{stop})$
- If $s_0, s_1, \ldots, s_k$ is a sequence of T, s_k is said to be a *reachable state* of T.
    - $Reach(T) \subseteq Q_S$ denotes the set of reachable states of T.

Properties

- $T = (S, Init, Trans)$: a transition system
- A *property* of T is a Boolean-valued expression over S.
  - Ex: $\varphi_{gcd} : \text{gcd}(m, n) = \text{gcd}(x, y)$
- A state $q \in Q_S$ *satisfies* the property φ if φ evaluates to 1 when all variables are assigned values according to q.
  - We say that $q \in Q_S$ *violates* φ if q does not satisfy φ.
- $[\![\varphi]\!] \subseteq Q_S$ is the set of all states that satisfy φ.

Invariants

- $T = (S, Init, Trans)$: a transition system
- A property φ is an *invariant* of T if every reachable state of T satisfies φ.
  - A property φ is an invariant iff $Reach(T) \subseteq [\![\varphi]\!]$.
  - Ex. φ_{gcd} is an invariant of T_{gcd}
  - Ex. $(mode = \text{stop}) \rightarrow (\mathrm{gcd}(m, n) = x)$ is an invariant of T_{gcd}

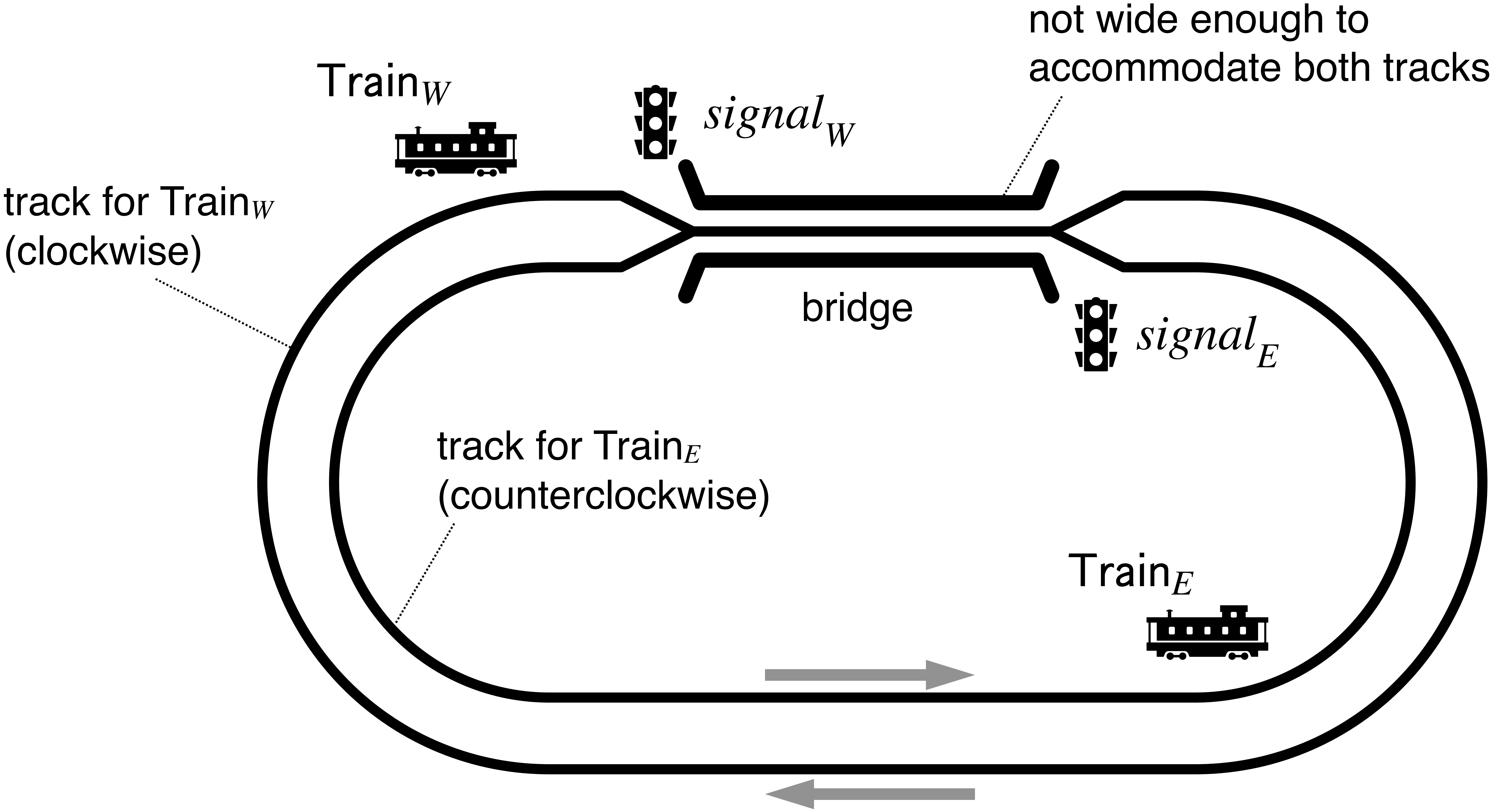
- A property φ is *reachable* if some reachable states of T satisfy φ.
  - A property φ is an reachable iff $Reach(T) \cap [\![\varphi]\!] \neq \emptyset$.
  - Note: Reachability is dual to invariance.

Invariant Verification

- Problem to check whether φ is an invariant of T
  - T: a transition system, φ : a property of T

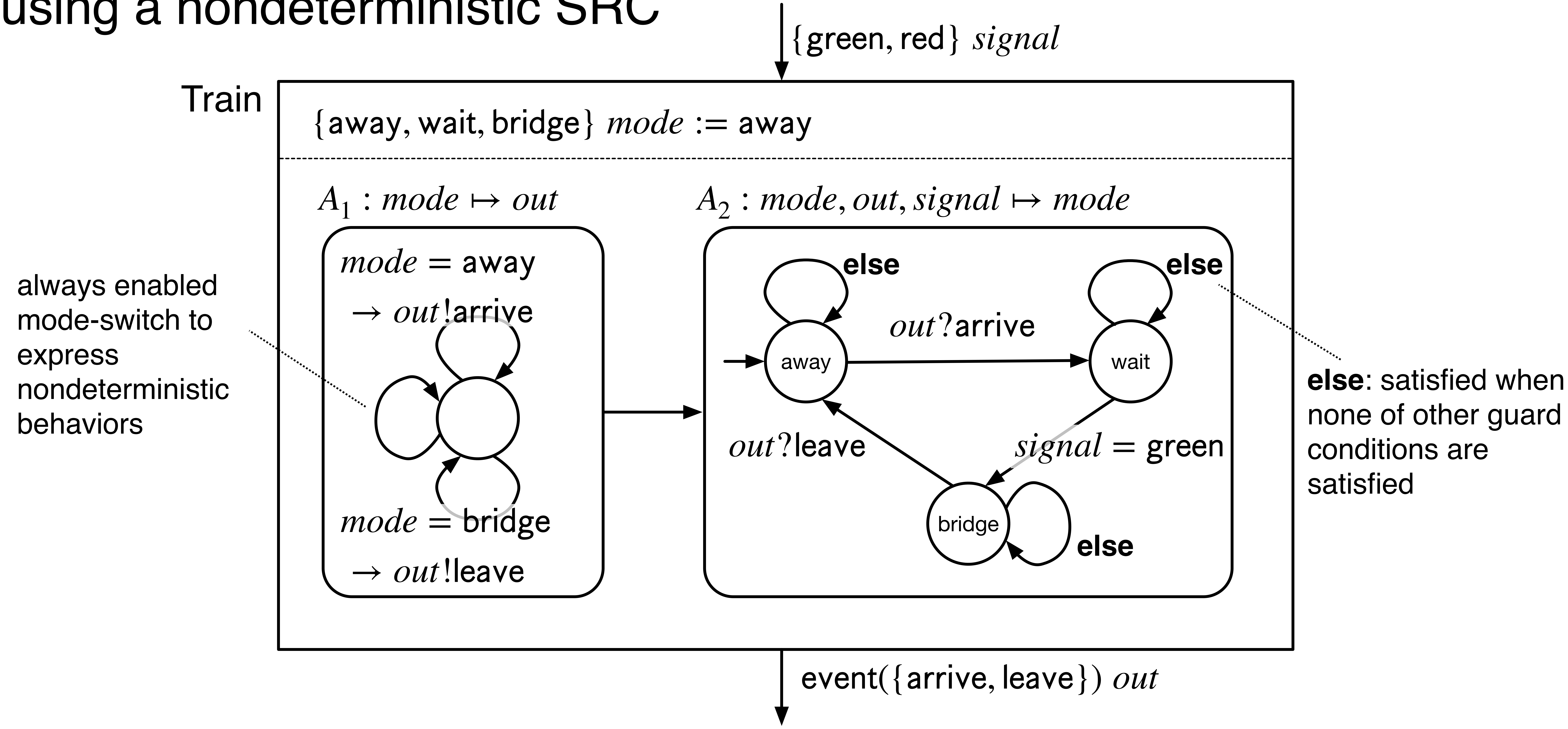
- If φ is NOT an invariant, there must be a state s such that it is reachable and violates φ. In other words, $\neg\varphi$ is reachable.
- An execution $s_0, s_1 \ \ldots, s_k$ is called a *counterexample* of φ (or, *witness* of $\neg\varphi$) if s_k violates φ.

Ex. Railroad System

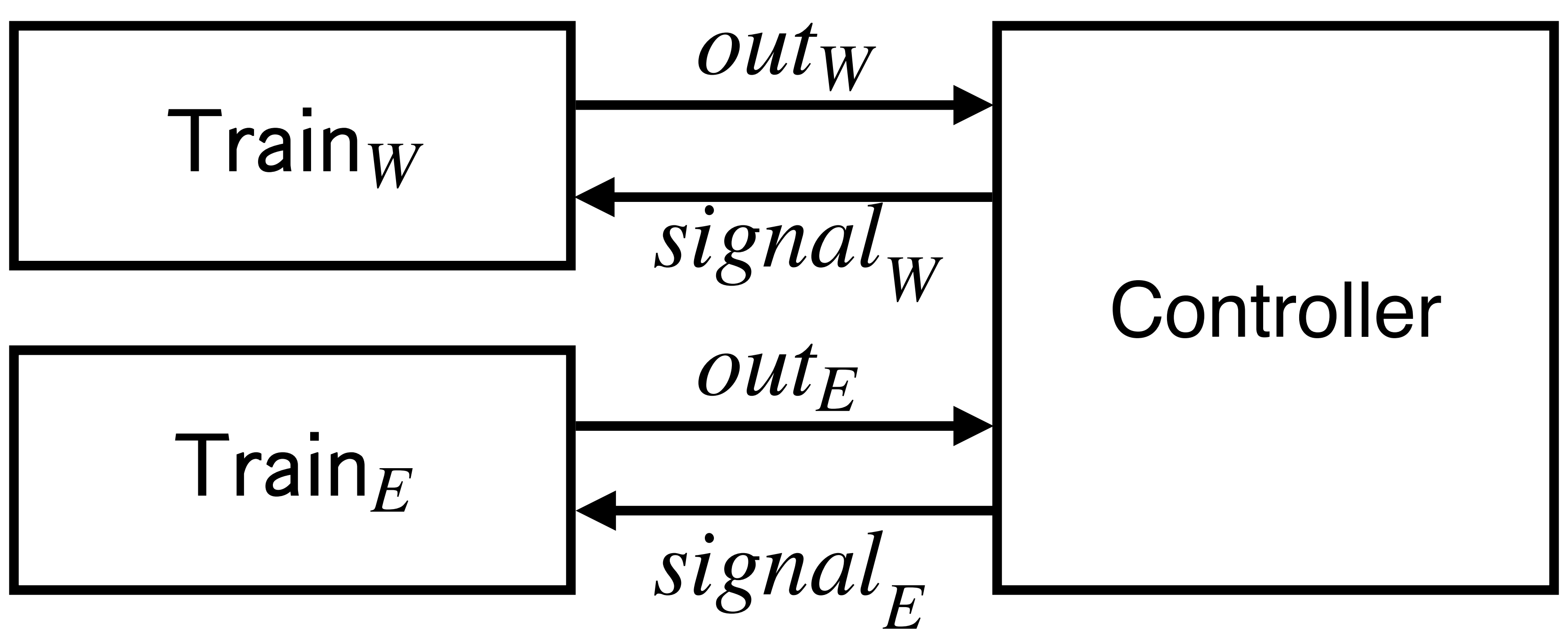


Modeling Trains

using a nondeterministic SRC



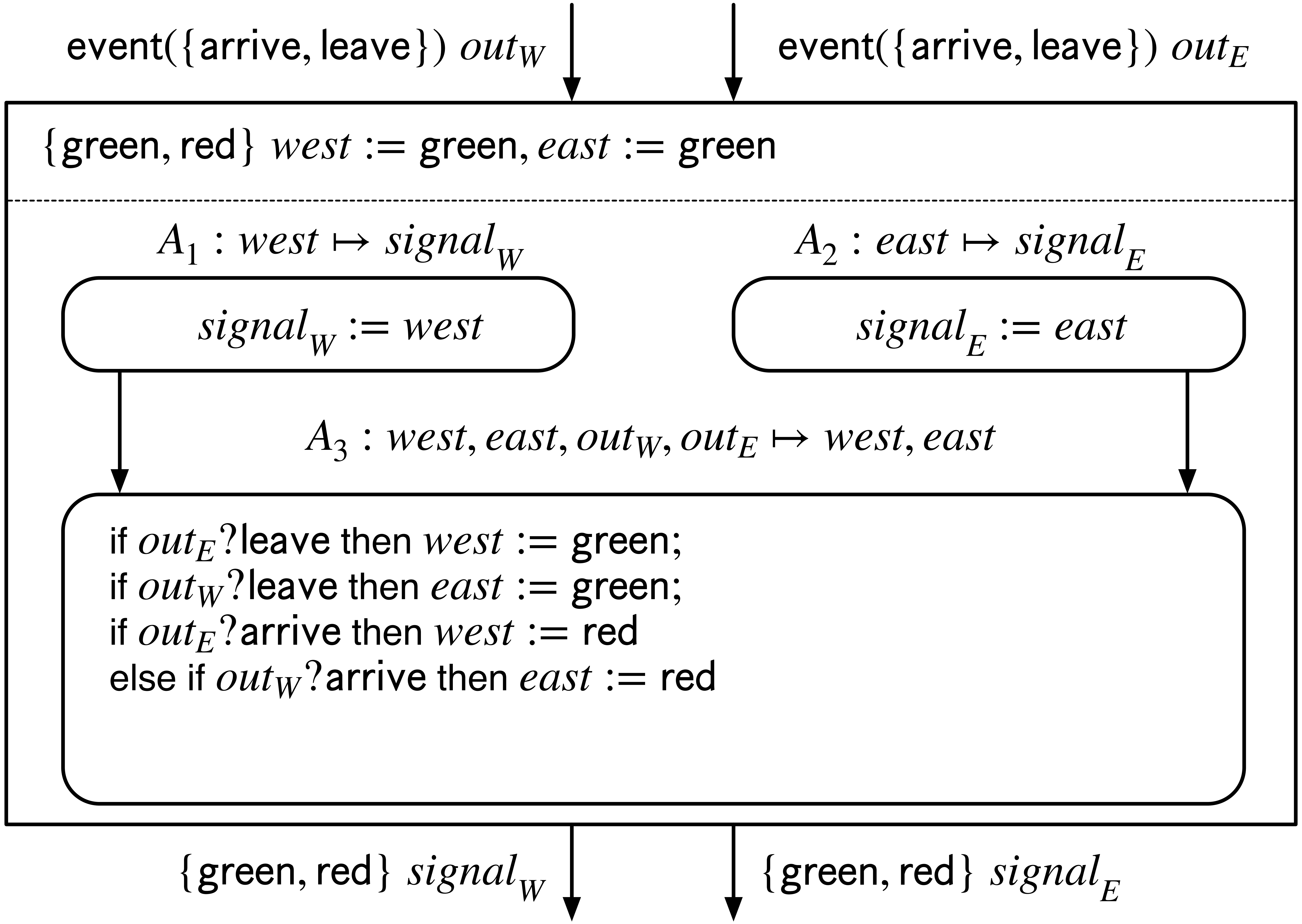
Composite System & Safety Requirement



- $\mathsf{RailroadSystem} = \mathsf{Controller} \parallel \mathsf{Train}_W \parallel \mathsf{Train}_E$
  - $\mathsf{Train}_W = \mathsf{Train}[out \mapsto out_W, signal \mapsto signal_W]$, $\mathsf{Train}_E = \cdots$

- Safety Requirement: Both trains should not be on the bridge at the same time.
  - $\mathsf{TrainSafety} = \neg(mode_W = \mathsf{bridge} \wedge mode_E = \mathsf{bridge})$

Controller1



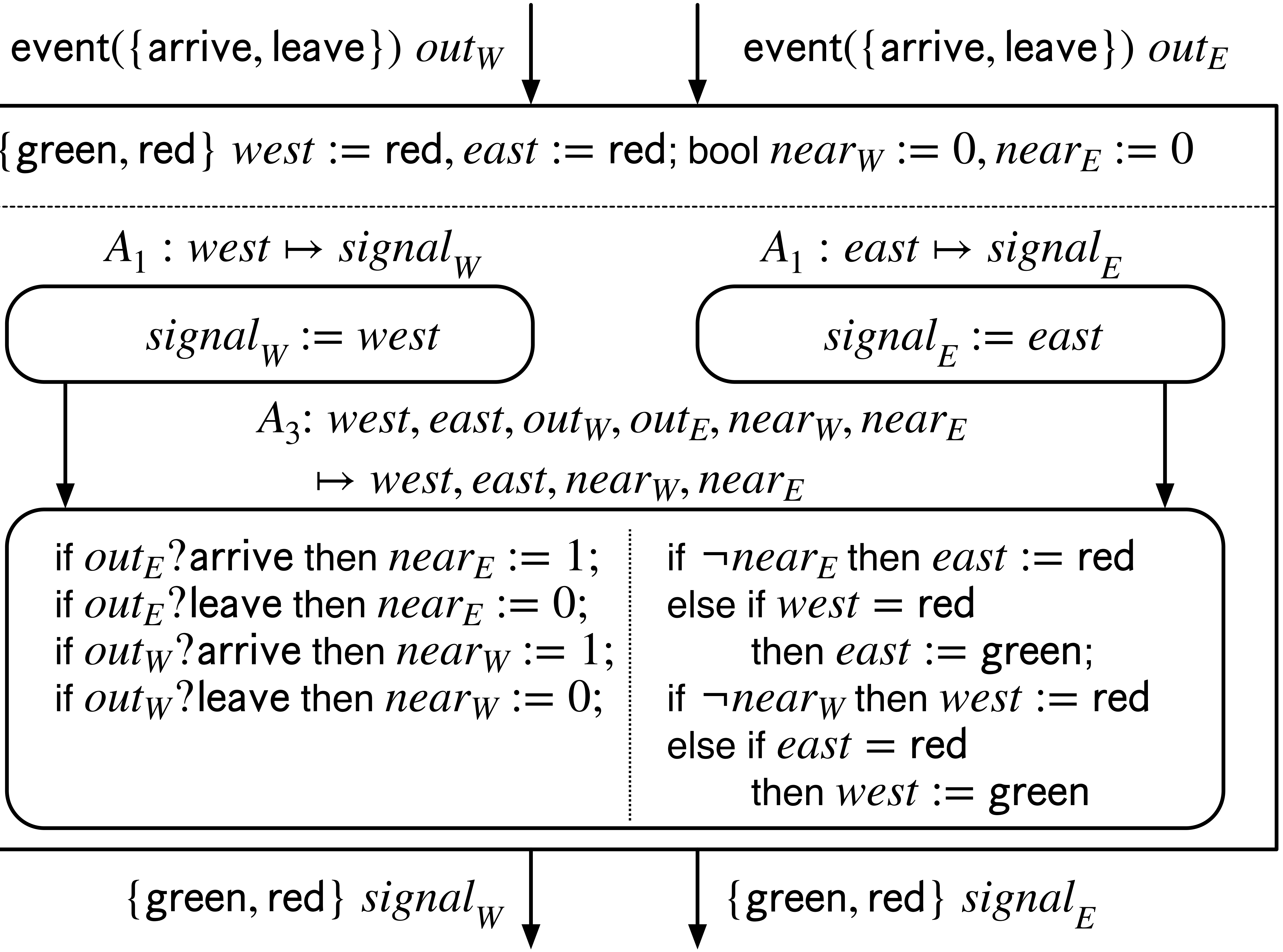
Example Execution

of RailroadSystem1 $=$ Controller1 $\|$ Train$_W$ $\|$ Train$_E$

		init	1	2	3	4	5
Controller	$west$	green	red	red	green	red	red
	$east$	green	green	green	green	green	green
	$signal_W$		green	red	red	green	red
	$signal_E$		green	green	green	green	green
Train$_W$	$mode_W$	away	wait	wait	wait	bridge	bridge
	out_W		arrive	$\perp$	$\perp$	$\perp$	$\perp$
Train$_E$	$mode_E$	away	wait	bridge	away	wait	bridge
	out_E		arrive	$\perp$	leave	arrive	$\perp$

TrainSafety is not an invariant of RailroadSystem1.

Controller2



Safety

of RailroadSystem2 $=$ Controller2 ‖ Train$_W$ ‖ Train$_E$

- TrainSafety is an invariant of RailroadSystem2

- How can we prove it?
  - Do you think you need to enumerate all the reachable states of the composite system to check them? (How many?)
  - How should we do if the number of the states is infinite?

Safety Monitor

- Ex. A sort of "fairness" requirement: *Suppose that a train arrives at a bridge. While it waits for the signal to turn green, the other train should not be allowed to enter the bridge repeatedly.*
  - I.e., While a train is waiting at a bridge with its signal red, the other train should not leave the bridge twice.
- It is difficult to formulate the above requirement as an invariant directly.
- To express such an invariant, we introduce another component classified as *safety monitors*.

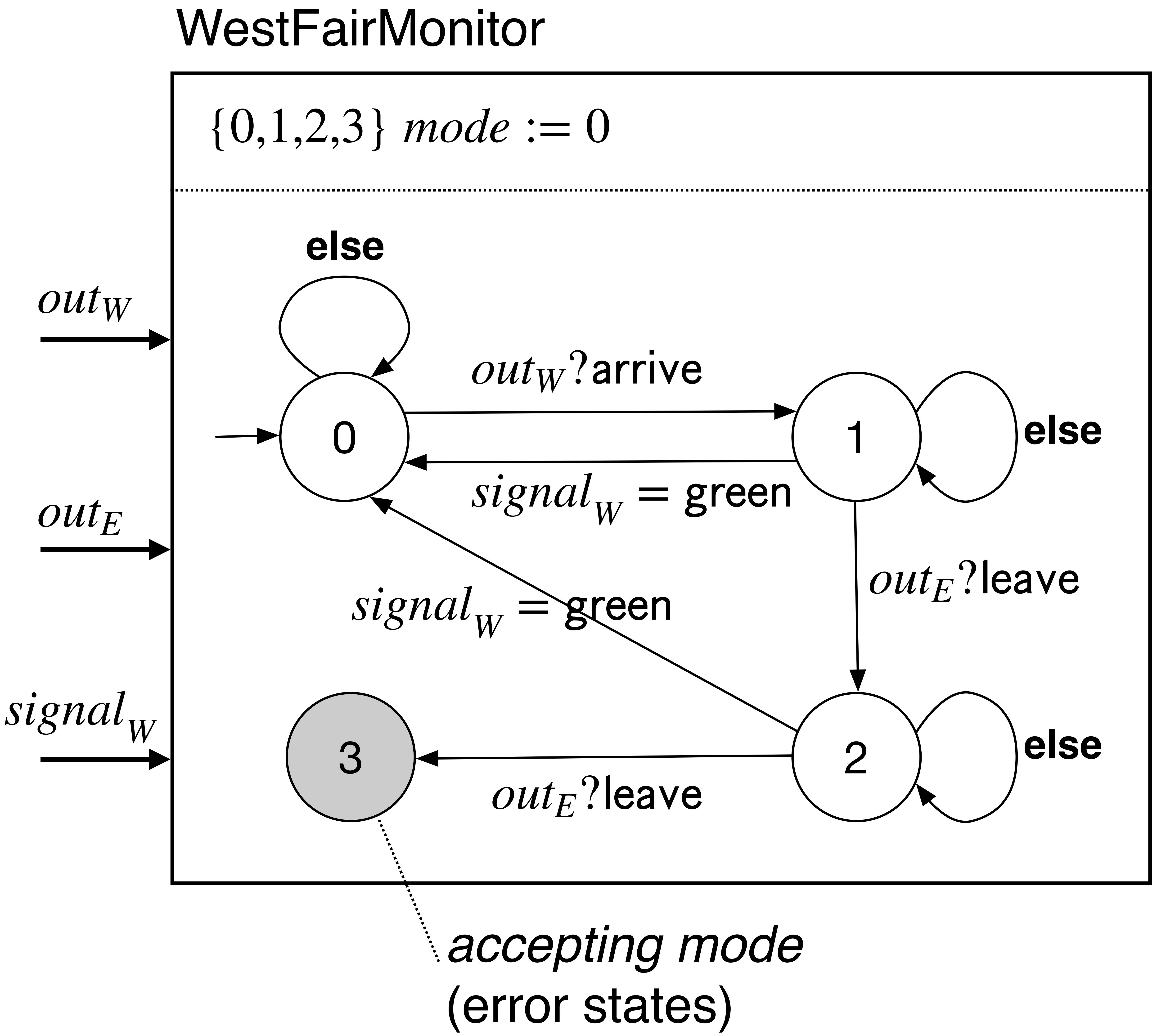
Safety Monitor

Ex. Fairness on Train$_W$

While Train$_W$ is waiting with the signal red, Train$_E$ should not leave the bridge twice.

$\Updownarrow$

WestFairMonitor . $mode \neq 3$ is an invariant of RailroadSystem ∥ WestFairMonitor.



Safety Monitor

Definition

- A safety monitor of an SRC $C = (I_C, O_C, S_C, Init_C, React_C)$ consists of an SRC $M = (I_M, O_M, S_M, Init_M, React_M)$ such that:
  - $I_M \subseteq I_C \cup O_C$,
  - $O_M \cap (I_C \cup O_C) = \emptyset$, and
  - $React_M$ is given as an extended-state machine along with a subset F of the modes declared as *accepting*.
- C satisfies the monitor specification if $M.mode \notin F$ is an invariant of $C \| M$.

Inductive Invariants

- Let $T = (S, Init, Trans)$ be a transition system.
- φ is an *inductive invariant* of T if:
  1. for all $s \in [\![Init]\!]$, s satisfies φ, and
  2. for all $s, t \in Q_S$, if s satisfies φ and $(s, t) \in [\![Trans]\!]$, then t satisfies φ.

- If φ is an inductive invariant of T, it is an invariant of T.

Inductive Invariants

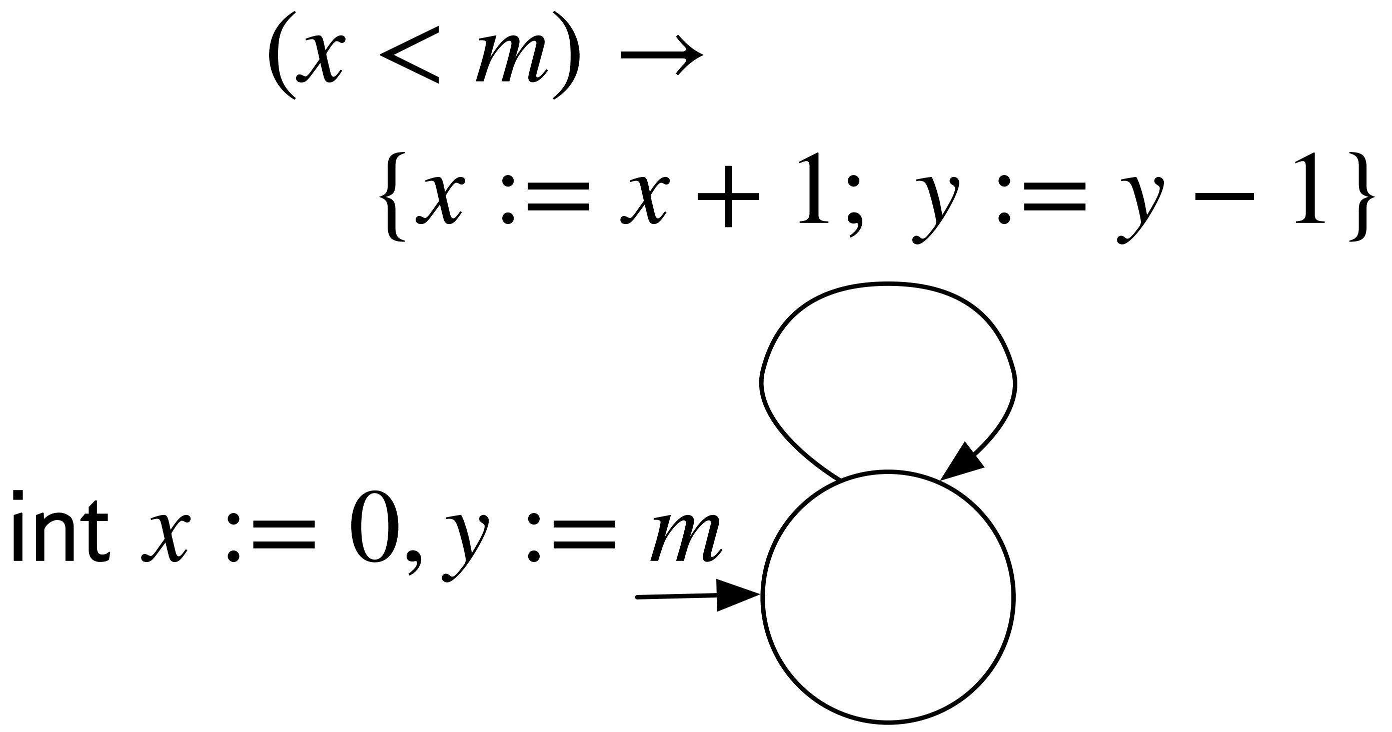
Ex. GCD

$$
\begin{aligned}
T_{\mathrm{gcd}} &= (\{x, y, mode\}, Init, Trans) \\
[\![Init]\!] &= \{(m, n, \mathsf{loop})\} \quad (m, n \in \mathbb{N}) \\
[\![Trans]\!] &= \{((j, k, \mathsf{loop}), (j-k, k, \mathsf{loop})) \mid j, k \in \mathbb{N} \land j > 0 \land k > 0 \land j > k\} \\
&\cup \{((j, k, \mathsf{loop}), (j, k-j, \mathsf{loop})) \mid j, k \in \mathbb{N} \land j > 0 \land k > 0 \land j \leq k\} \\
&\cup \{((0, k, \mathsf{loop}), (k, k, \mathsf{stop})) \mid k \in \mathbb{N}\} \cup \{((j, 0, \mathsf{loop}), (j, 0, \mathsf{stop})) \mid j \in \mathbb{N}\}
\end{aligned}
$$

- Let φ_{gcd} be a property given by $\mathrm{gcd}(m, n) = \mathrm{gcd}(x, y)$.
- We show that φ_{gcd} is an inductive invariant of T_{gcd}.
- If $s \in [\![Init]\!]$, clearly s satisfies φ_{gcd} because $s(x) = m$ and $s(y) = n$.
- Suppose that $s = (j, k, \mathsf{loop})$ and s satisfies φ_{gcd}.
  - If $j > 0 \land k > 0 \land j > k$ and $(s, t) \in [\![Trans]\!]$ where $t = (j-k, k, \mathsf{loop})$, then t satisfies φ_{gcd} since $\mathrm{gcd}(j-k, k) = \mathrm{gcd}(j, k)$ ($\because$ any divisor of both j and k is also a divisor of $j-k$ and any divisor of both j and $j-k$ is also a divisor of k).
  - Similar arguments can be used for other cases.

Strengthening Invariants

Ex. IncDec(m)



- Let $m \in \mathbb{N}$. IncDec(m) = $(\{x, y\}, Init, Trans)$ is a transition system where:
  - $[\![Init]\!] = \{(0, m)\}$, and
  - $[\![Trans]\!] = \{((a, b), (a + 1, b - 1)) \mid a, b \in \mathbb{Z} \wedge a < m\}$.

- $\varphi_x : 0 \leq x \leq m$
- To show φ_x is an invariant of IncDec(m), we show that it is an inductive invariant of IncDec(m).
  - If $s \in [\![Init]\!]$, clearly s satisfies φ_x because $s(x) = 0$.
  - If $(s, t) \in [\![Trans]\!]$ and s satisfies φ_x (*i.e.*, $0 \leq s(x) \leq m$), then t satisfies φ_x because $t(x) = s(x) + 1 \leq m$ ($\because$ $s(x) < m$).

Strengthening Invariants

Ex. IncDec(m)

- $\varphi_y : 0 \leq y \leq m$
- To show that φ_y is an invariant of IncDec(m), we would like to show that it is an inductive invariant of IncDec(m).
    - Obviously $(0, m) \in [\![Init]\!]$ satisfies φ_y.
    - Let $(s, t) \in [\![Trans]\!]$ and s satisfies φ_y. Is it possible to show that t satisfies φ_y?
        - The answer is NO.
        - What we should show is $(0 \leq s(y) \leq m) \rightarrow (0 \leq s(y) - 1 \leq m)$. However, it does not hold. For example, consider $((0, 0), (1, -1)) \in [\![Trans]\!]$. $(0, 0)$ satisfies φ_y, but $(1, -1)$ does not.

Strengthening Invariants

Ex. IncDec(m)

- $\varphi_{xy} : 0 \leq y \leq m \wedge x + y = m$
  - It is clear that $\varphi_{xy} \rightarrow \varphi_y$. Thus, if we can show that φ_{xy} is an inductive invariant of IncDec(m), then we can say that φ_y is an invariant of IncDec(m).
- Proof: φ_{xy} is an inductive invariant of IncDec(m).
  - Clearly $(0, m) \in [\![Init]\!]$ satisfies φ_{xy}.
  - Suppose $s = (a, b)$ that satisfies φ_{xy} (i.e., $0 \leq b \leq m \wedge a + b = m$) and $a < m$. Then $(s, t) \in [\![Trans]\!]$ if $t = (a + 1, b - 1)$. Since $a + b = m$ and $a < m$, we can say $b > 0$. It follows that $0 \leq b - 1 \leq m$ and $(a + 1) + (b - 1) = m$. Thus, t satisfies φ_{xy}.

Proof Rule for Invariants

- To show that φ is an invariant of T, find a property ψ such that:
  1. ψ is an inductive invariant of T, and
  2. $\psi \to \varphi$.

- The rule is sound and complete:
  - Soundness: If ψ is an inductive invariant, then all reachable states of T satisfy ψ. Since $\psi \to \varphi$, every state satisfying ψ also satisfies φ. So φ is an invariant of T.
  - Completeness: If φ is an invariant of T, there exist an inductive invariant ψ that implies φ.
- Identifying ψ sometimes requires expertise.

Summary

- Safety Requirements (1)
  - What are safety requirements
  - Transition Systems
    - SRCs as Transition Systems, Programs as Transition Systems
    - Execution, Reachable States, Properties, Invariants
    - Safety Monitors
  - Verifying Invariants
    - Inductive Invariants
    - Proof Rule for Invariants