Cyber-Physical Systems (CSC.T431)

Safety Requirements (2)

Instructor: Takuo Watanabe (Department of Computer Science)

Agenda

- Safety Requirements (2)

Course Support & Material

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

Automated Invariant Verification

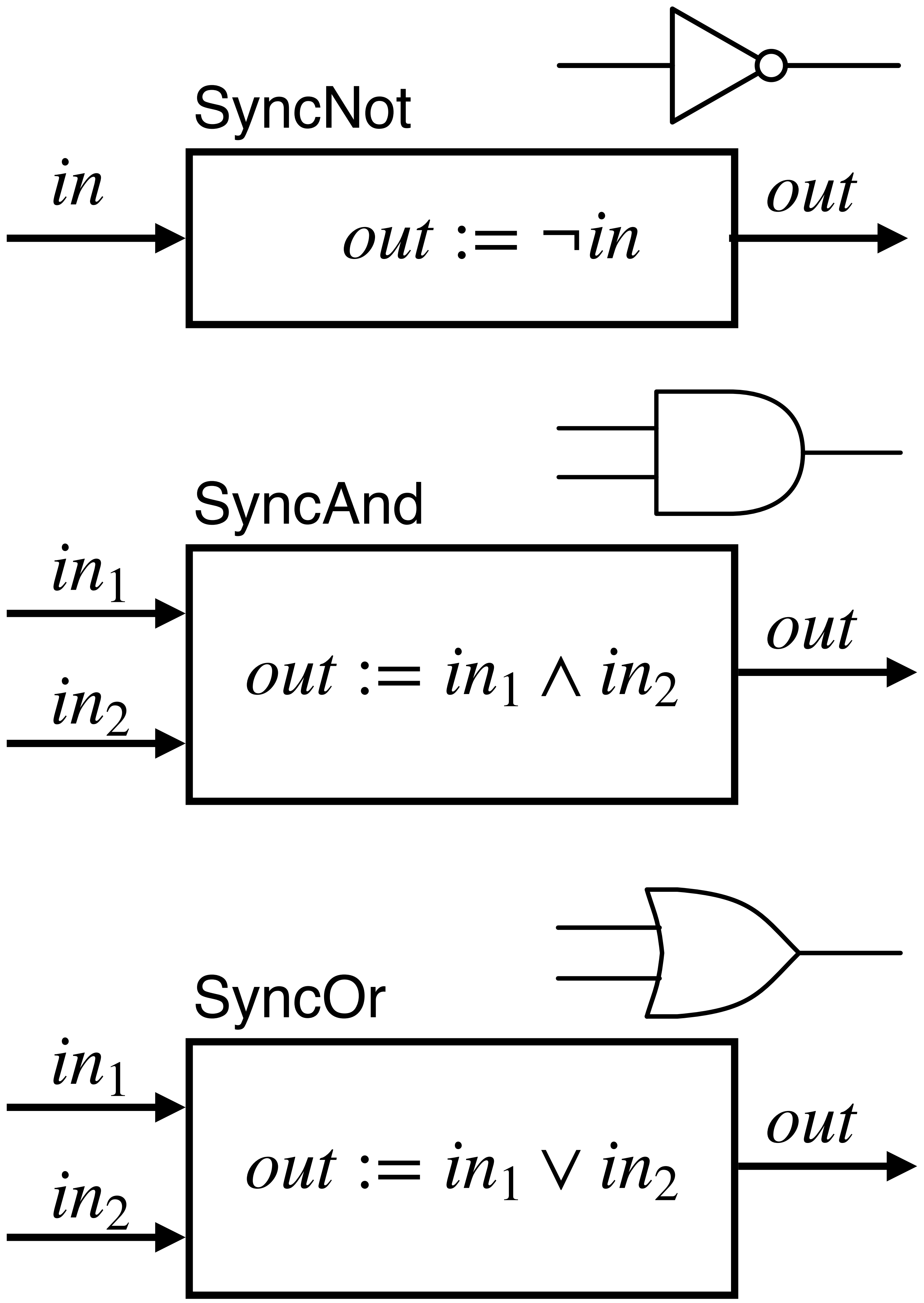
Verification Problem

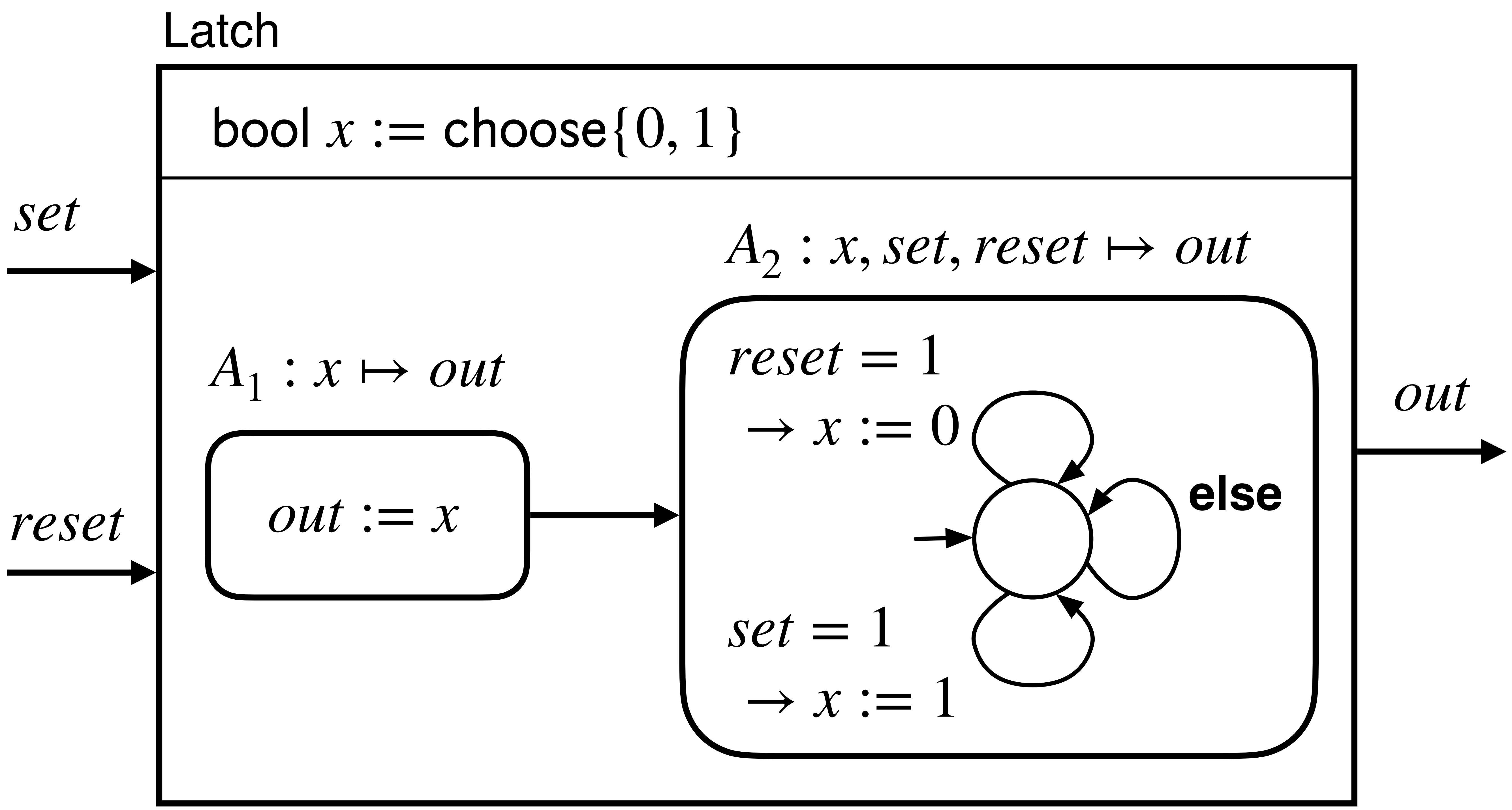
- Verification Problem: to decide whether φ is an invariant of T or not.
  - input: T and φ
  - output: Yes or No (w/ counter example execution)

- Clearly the problem is undecidable.
  - State space may be unbound.

- How about finite-state systems?

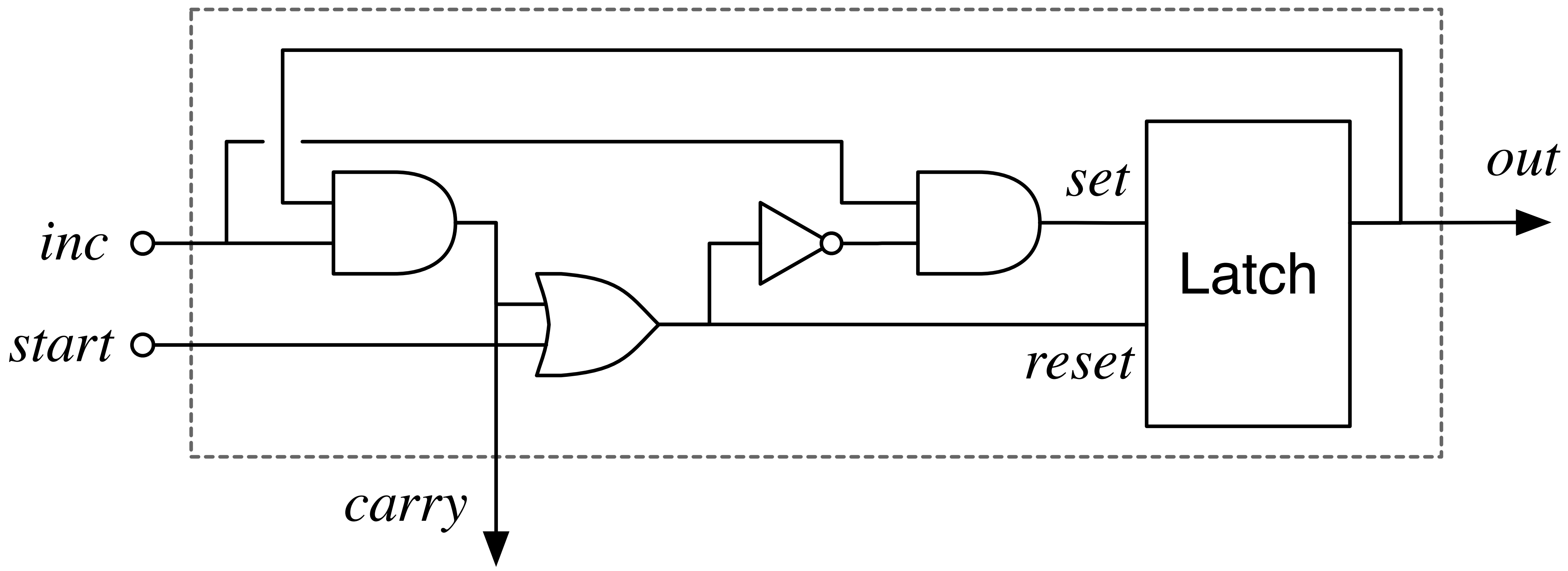
Ex. Synchronous Circuits as SRCs



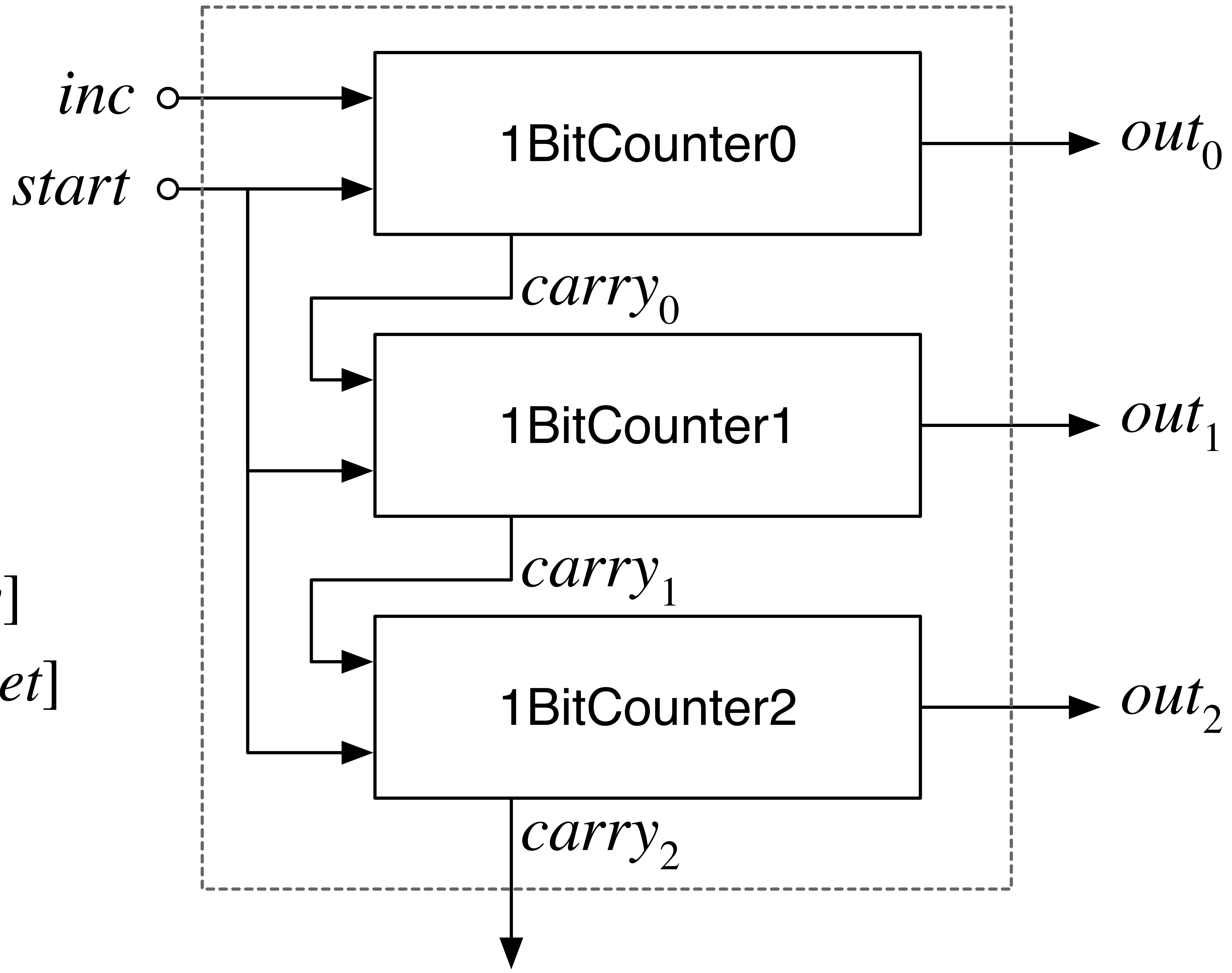


Ex. Synchronous Circuits as SRCs

1BitCounter



3BitCounter



$$
\begin{aligned}
\mathsf{1BitCounter} = (&\mathsf{SyncAnd}[in_1 \mapsto out, in_2 \mapsto inc, out \mapsto carry] \\
\| \;& \mathsf{SyncOr}[in_1 \mapsto carry, in_2 \mapsto start, out \mapsto reset] \\
\| \;& \mathsf{SyncNot}[in \mapsto reset, out \mapsto tmp] \\
\| \;& \mathsf{SyncAnd}[in_1 \mapsto inc, in_2 \mapsto tmp, out \mapsto set] \\
\| \;& \mathsf{Latch}\ ) \setminus \{tmp, set, reset\}
\end{aligned}
$$

Complexity of Invariant Verification

Ex. Synchronous Circuits

- Let C be a sequential circuit that consists of instances of SyncNot, SyncAnd, SyncOr, and Latch, and φ be a property specified as a Boolean expression over the state variables corresponding to the instances of Latch.
- The computational problem of checking whether φ is an invariant of the transition system corresponding C is PSPACE-complete.
  - A decision problem is PSPACE if it can be solved using a memory whose size is polynomial on the input size. A decision problem is PSPACE-complete if it is PSPACE and any PSPACE problems can be transformed to the problem in polynomial time.

Simulation-Based Analysis

- Let T be a transition system and φ be a property of T.
- For a given $k > 0$, generate an execution $s_0, s_1, \ldots, s_k$ to checke if s_j satisfies φ for all $j \in \{0,\ldots,k\}$.
  - Nondeterminism: choices of initial states / choices of transitions

- Let $s_0, s_1, \ldots, s_k$ be a generated execution. If there is a state that violates φ, we can say that φ is not an invariant of T. However, if all states in the execution satisfy φ, we cannot say more than that.

Simulation-Based Analysis

Invariant Falsification

```
array[state] exec;
nat j := 0;
state s := ChooseInitState(T);
if s = null then return;
exec[j] := s;
if Satisfies(s, φ) = 0 then return exec;
for j = 1 to k do {
    s := ChooseSuccState(s, T);
    if s = null then return;
    exec[j] := s;
    if Satisfies(s, φ) = 0 then return exec;
}
```

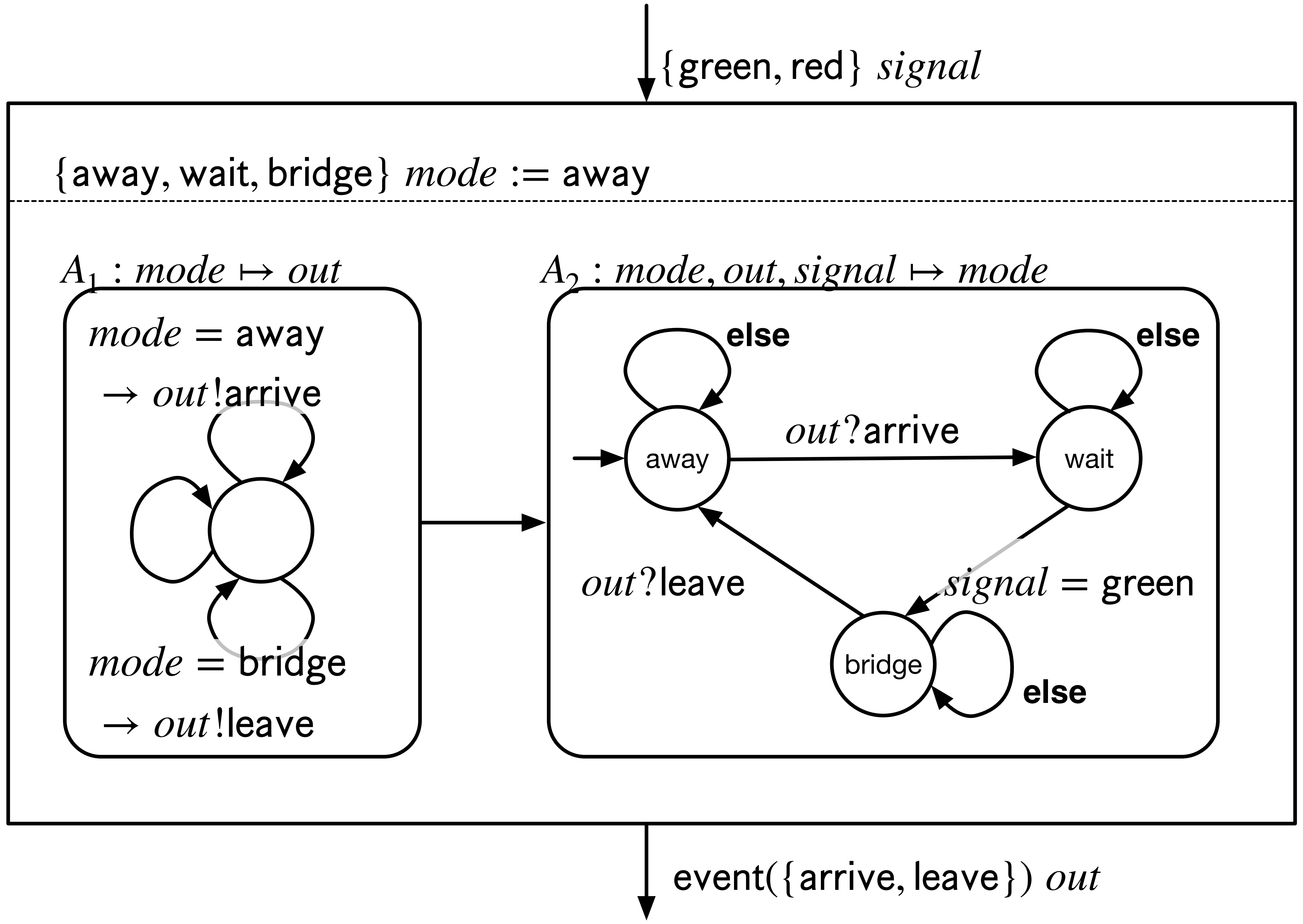
- Input: T, φ, and $k > 0$.
- Output: if the algorithm encounters a state s_i $(0 \leq i \leq k)$ that violates φ, it returns the counterexample execution $s_0, s_1, \ldots, s_i$.

- We can only say that φ is not an invariant of T if the algorithm returns a counterexample.

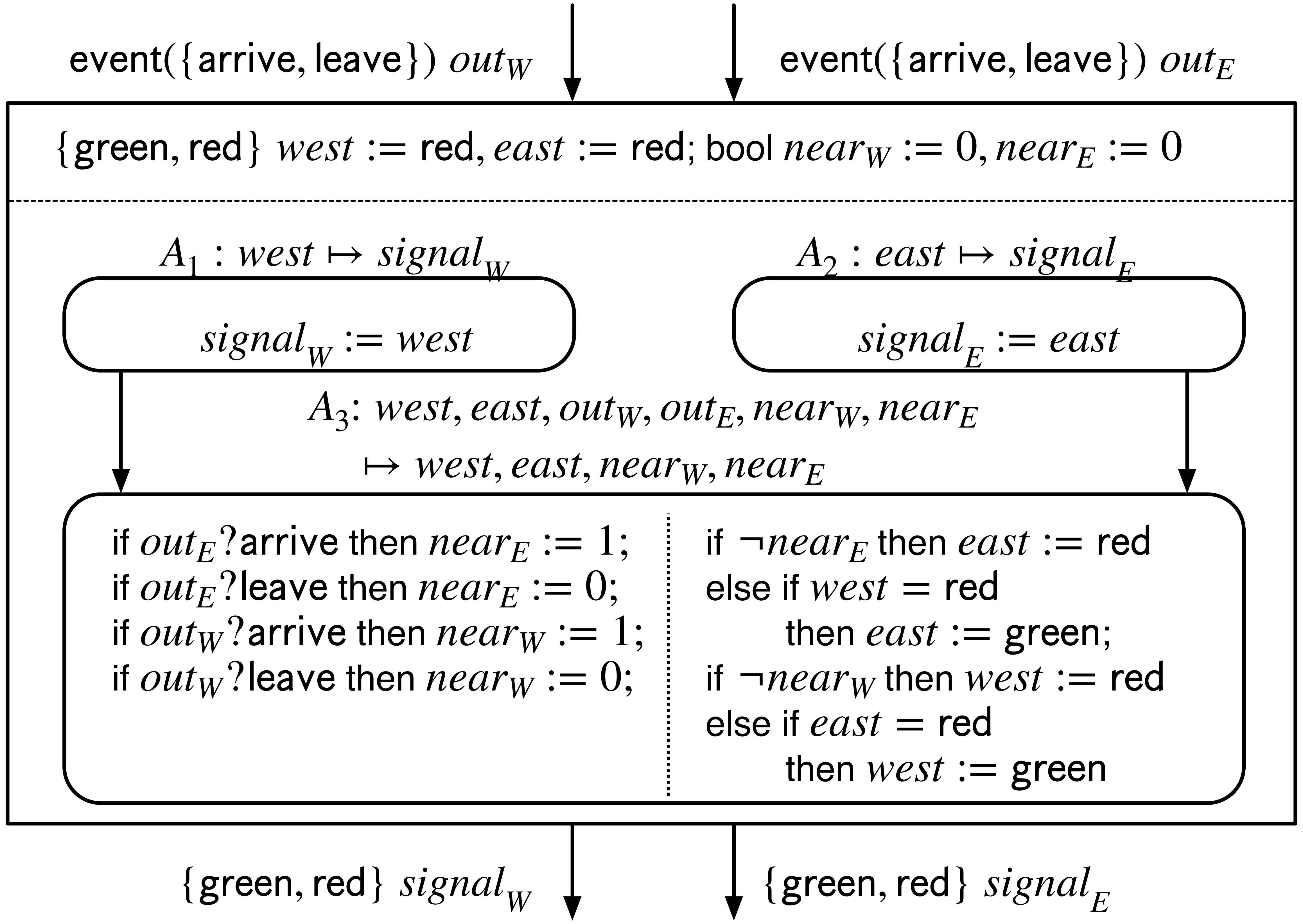
Enumerative Search

- Let T be a transition system and φ be a property of T.
- To check φ is an invariant of T, we check whether $\neg\varphi$ is reachable.
- An exhaustive search gives the answer of the verification problem.
  - Simulation-Based Analysis
    - randomly chooses one possible initial state
    - randomly chooses one possible transition
  - Enumerative Search
    - systematically enumerates all possible initial states
    - systematically enumerates all possible transitions

RailRoadSystem2
$= \mathsf{Train}_W \parallel \mathsf{Train}_E \parallel \mathsf{Controller2}$



Train



Controller2

Safety

Is TrainSafety an invariant of RailRoadSystem2?

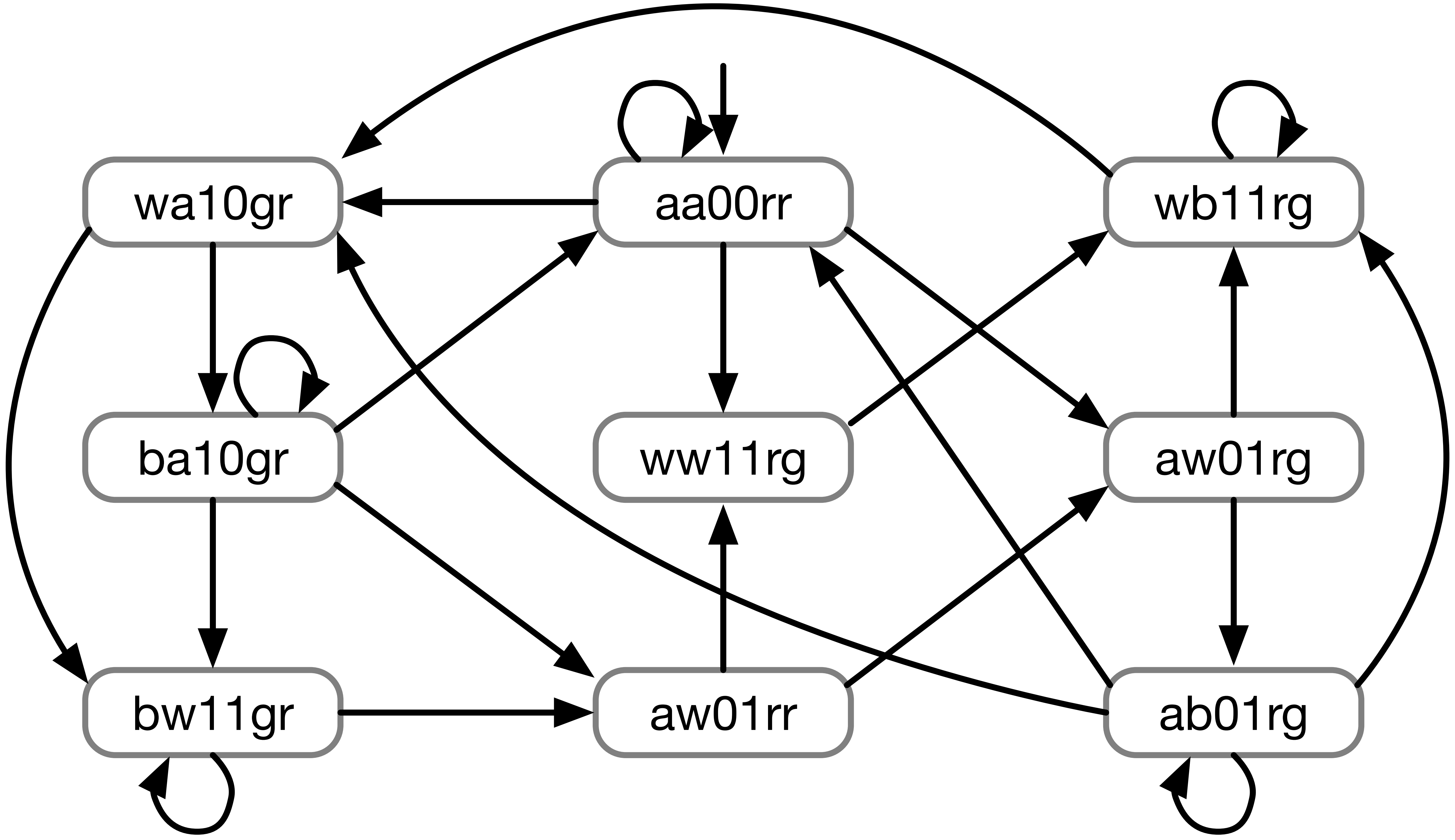
- TrainSafety $= \neg(mode_W = \text{bridge} \wedge mode_E = \text{bridge})$

- States of RailRoadSystem2
  - $(mode_W, mode_E, near_W, near_E, west, east)$
  - # of possible states : 144
  - # of reachable states may be smaller than 144.

- Is $\neg$TrainSafety reachable?

Reachable Subgraph

of RailRoadSystem2



- State: aa00rr is an abbreviation of $(\mathsf{away}, \mathsf{away}, 0, 0, \mathsf{red}, \mathsf{red})$.
  - a : away, w : wait, b : bridge
  - r : red, g : green

- No states of the form bb**** are reachable from the initial state.
- Thus, TranSafety is an invariant of RailRoadSystem2.

On-the-Fly Depth-First Search

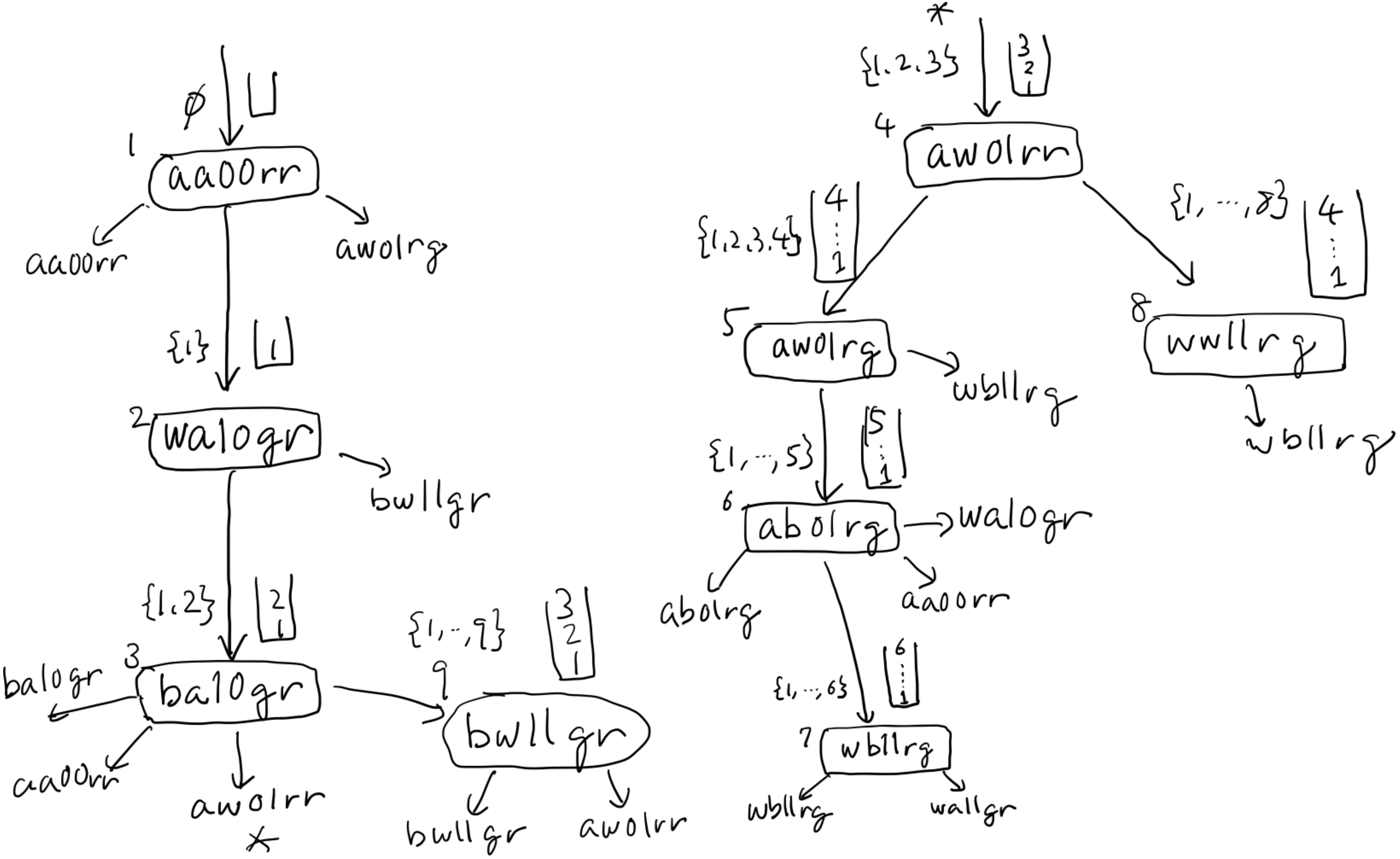
- An algorithm to check if φ is reachable in a countably branching T
  - If the algorithm returns 0, φ is not reachable in T.
  - If the algorithm returns a sequence of states $s_0, s_1, \ldots, s_k$, then s_k satisfies φ.
    - such sequence is called a *witness*.
  - If the number of reachable states of T is finite, then the algorithm terminates.
    The number of calls to DFS is bounded by the number of reachable states.
- To check if φ is an invariant of T, we apply the algorithm to $\neg\varphi$ and T.
- Note: T is said to be a *countably branching* transition system if the choices of initial states and choices of transitions in T are countable.

On-the-Fly Depth-First Search Algorithm

Input: T, φ
Output: if φ is reachable, return a witness
otherwise return 0

set(state) $Reach$:= EmptySet;
stack(state) $Pending$:= EmptyStack;
state s := $FirstInitState(T)$;
while $s \neq$ null **do** {
  if Contains($Reach, s$) $= 0$ **then**
    if DFS(s) $= 1$ **then**
      return Reverse($Pending$);
  s := $NextInitState(s, T)$;
};
return 0.

bool **function** DFS(state s)
  Insert($s, Reach$);
  Push($s, Pending$);
  if $Satisfies(s, \varphi) = 1$ **then return** 1;
  state t := $FirstSuccState(s, T)$;
  while $t \neq$ null **do** {
    if Contains($Reach, t$) $= 0$ **then**
      if $DFS(t) = 1$ **then return** 1;
    t := $NextSuccState(s, t, T)$;
  };
  Pop($Pending$);
  return 0.


∅
1
aa0orr
aaoorr
awolrg
{1}
2
wa1ogr
bwllgr
{1,2}
3
ba1ogr
balogr
aa0orr
awolrr
*
{1,…,9}
9
bwllgr
bwllgr
awolrr
*
{1,2,3}
4
aw0lrr
{1,2,3,4}
5
aw0lrg
wbllrg
{1,…,5}
6
ab0lrg
walogr
abolrg
aaoorr
{1,…,6}
7
wb1lrg
wbllrg
wallgr
{1,…,8}
8
wwllrg
wbllrg

Symbolic Transition Systems

- Let $T = (S, Init, Trans)$ be a transition system.
- A symbolic representation of T is a triple $(S, \varphi_I, \varphi_T)$ where:
- φ_I : a logical formula (Boolean expression over S) representing $Init$
- φ_T : a logical formula (Boolean expression over S) representing $Trans$

- To define φ_T, we need to relate the values of state variables before and after a transition. For $v \in S$, we use v and v' to denote the variable before and after the transition respectively.

Symbolic Transition Systems

Converting Transitions into Logical Formulas

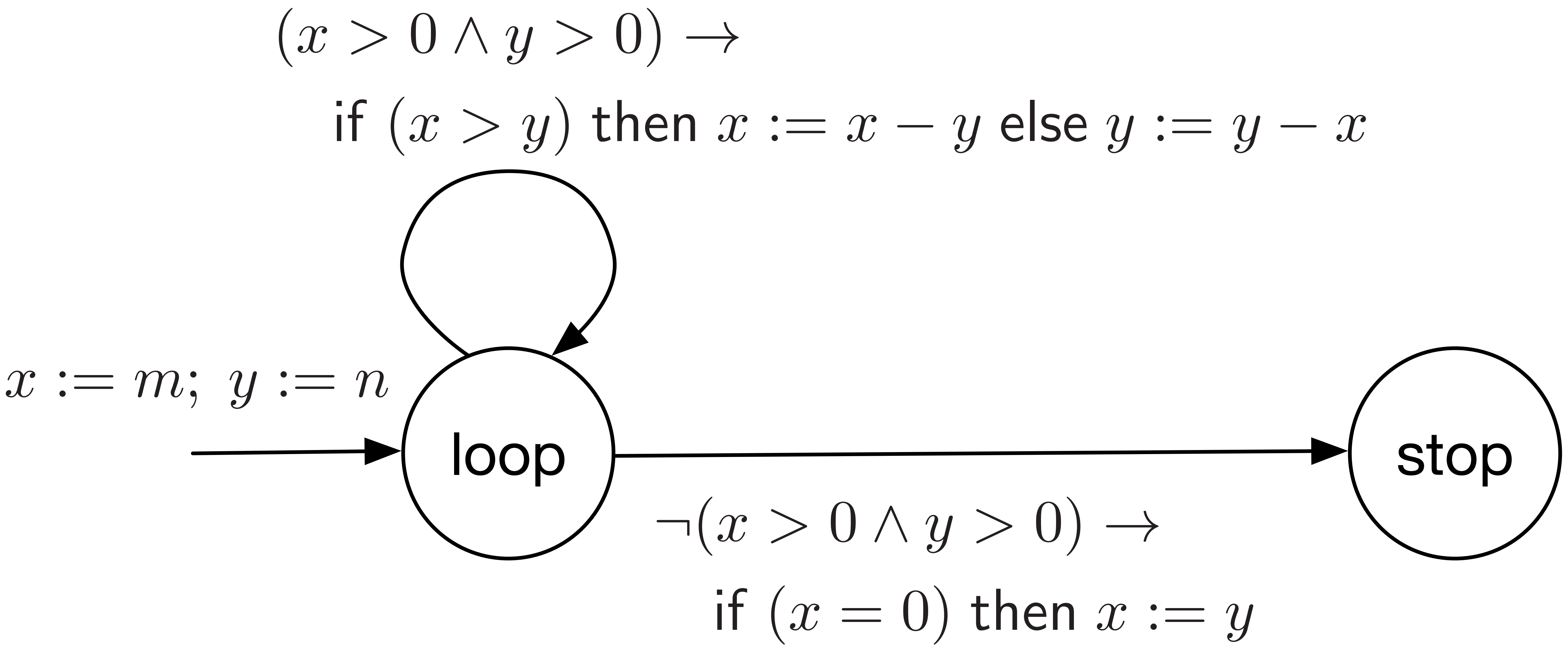
- Assignments
  - $x := e \;\Rightarrow\; x' = e$
  - $x :=$ choose$\{e_1, e_2\} \;\Rightarrow\; x = e_1 \vee x = e_2$
    - If x is a Boolean variable, $x :=$ choose$\{0,1\}$ becomes 1 (true).
  - $x := e_1;\; y := e_2 \;\Rightarrow\; x' = e_1 \wedge y' = e_2[x \mapsto x']$
- Condition
  - if e then c_1 else $c_2 \;\Rightarrow\; (e \wedge \psi_1) \vee (\neg e \wedge \psi_2)$ (where $c_1 \Rightarrow \psi_1$ and $c_2 \Rightarrow \psi_2$)

- Note: the above are not complete rules for the conversion.

Ex. GCD as a Transition System

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 $x := y$



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

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

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

Converting GCD

- c_1 : if $x > y$ then $x := x - y$ else $y := y - x$
  $\Rightarrow \ [x > y \wedge x' = x - y \wedge y' = y] \vee [\neg(x > y) \wedge x' = x \wedge y' = y - x] \ \ (\psi)$
- c_2 : if $x = 0$ then $x := y$
  $\Rightarrow \ [x = 0 \wedge x' = y \wedge y' = y] \vee [\neg(x = 0) \wedge x' = x \wedge y' = y] \ \ (\psi')$
- while $(x > 0 \wedge y > 0)$ do $c_1;\ c_2$
  $\Rightarrow \ [(x > 0 \wedge y > 0) \wedge mode = \text{loop} \wedge \psi \wedge mode' = \text{loop}]$
  $\vee \ [\neg(x > 0 \wedge y > 0) \wedge mode = \text{loop} \wedge \psi' \wedge mode' = \text{stop}]$

GCD as a Symbolic Transition System

- $S = \{x, y, mode\}$
- $\varphi_I = [x = m \land y = n \land mode = \text{loop}]$
- $\varphi_T = \psi_1 \lor \psi_2$
  - $\psi_1 = [(x > 0 \land y > 0) \land mode = \text{loop} \land \psi \land mode' = \text{loop}]$
  - $\psi = [x > y \land x' = x - y \land y' = y] \lor [\neg(x > y) \land y' = y - x \land x' = x]$
  - $\psi_2 = [\neg(x > 0 \land y > 0) \land mode = \text{loop} \land \psi' \land mode' = \text{stop}]$
  - $\psi' = [x = 0 \land x' = y \land y' = y] \lor [\neg(x = 0) \land x' = x \land y = y]$

SRCs as Symbolic Transition Systems

Ex. Delay

- Delay $= (I, O, S, Init, React)$ where $I = \{in\}, O = \{out\}, S = \{x\}$
- Symbolic representation of Delay: $(I, O, S, \varphi_I, \varphi_R)$

Reaction Formula

  - $\varphi_I : x = 0$ (a formula over S)
  - $\varphi_R : out = x \wedge x' = in$ (a formula over $S \cup I \cup O \cup S'$, where $S' = \{v' \mid v \in S\}$)
- Delay as a transition system: $T = (S, Init, Trans)$
- Symbolic representation of T: $(S, \varphi_I, \varphi_T)$
  - $\varphi_T : \exists in \,.\, \exists out \,.\, (out = x \wedge x' = in)$
    - The localized input and output variables are quantified.
  - For any x and x', φ_T evaluates to 1. *I.e.*, every pair of states can be a transition.

local in, out;
 $in :=$ choose$\{0,1\}$;
 $out := x$; $x := in$

SRCs as Transition Systems

Ex. TriggeredCopy

event(bool) *in* →

nat $x := 0$

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

→ event(bool) *out*

$\Downarrow$

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

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

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

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

$React$	0	$\xrightarrow{\perp/\perp}$	0	$\xrightarrow{0/0}$	1	$\xrightarrow{1/1}$	2	$\xrightarrow{1/1}$	2	$\xrightarrow{\perp/\perp}$	2	$\xrightarrow{\perp/\perp}$	2	$\xrightarrow{1/1}$	3
$Trans$	0	$\longrightarrow$	0	$\longrightarrow$	1	$\longrightarrow$	2	$\longrightarrow$	2	$\longrightarrow$	2	$\longrightarrow$	2	$\longrightarrow$	3

SRCs as Symbolic Transition Systems

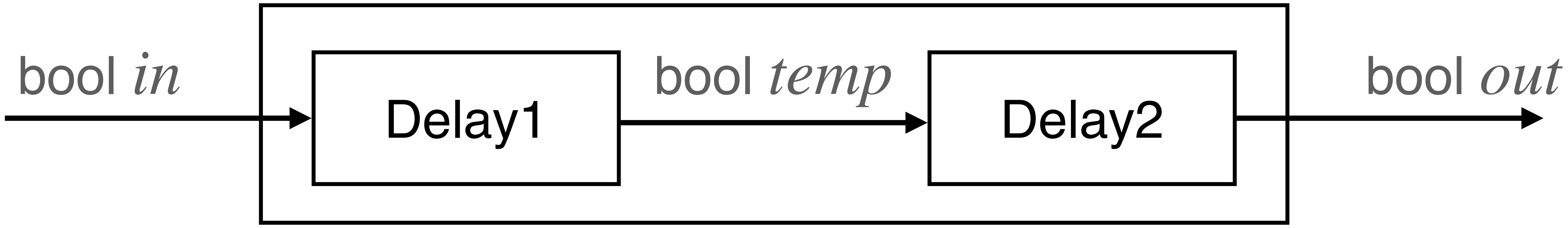
Ex. TriggeredCopy

- Symbolic Representation: $(\{in\}, \{out\}, \{x\}, \varphi_I, \varphi_R)$
  - $\varphi_I : x = 0$
  - $\varphi_R : (in? \land out = in \land x' = x + 1) \lor (\neg in? \land out = \bot \land x' = x)$

- As a Symbolic Transition System: $(\{x\}, \varphi_I, \varphi_T)$
  - $\varphi_T : \exists in \,.\, \exists out \,.\, \varphi_R : (in? \land out = in \land x' = x + 1) \lor (\neg in? \land out = \bot \land x' = x)$
  - φ_T can be simplified to $x' = x + 1 \lor x' = x$.

Composing Symbolic Representations

Ex. DoubleDelay



- DoubleDelay $= (\text{Delay1} \parallel \text{Delay2}) \setminus temp$
  - Delay1 $=$ Delay$[out \mapsto temp]$
  - Delay2 $=$ Delay$[in \mapsto temp]$

Composing Symbolic Representations

Ex. DoubleDelay

- DoubleDelay = Delay1 ‖ Delay2
  - Delay1 = $(\{in\}, \{temp\}, \{x_1\}, Init_1, React_1)$
  - Delay2 = $(\{temp\}, \{out\}, \{x_2\}, Init_2, React_2)$
- Symbolic Representations of Delay1 and Delay2
  - $C_1 = (\{in\}, \{temp\}, \{x_1\}, \varphi_I^1, \varphi_R^1)$ where $\varphi_I^1 : x_1 = 0$ and $\varphi_R^1 : temp = x_1 \wedge x_1' = in$.
  - $C_2 = (\{temp\}, \{out\}, \{x_2\}, \varphi_I^2, \varphi_R^2)$ where $\varphi_I^2 : x_2 = 0$ and $\varphi_R^2 : out = x_2 \wedge x_2' = temp$.
- Symbolic Representation of DoubleDelay
  - $C = (\{in\}, \{temp, out\}, \{x_1, x_2\}, \varphi_I, \varphi_R)$, where $\varphi_I : x_1 = 0 \wedge x_2 = 0$ and $\varphi_R : temp = x_1 \wedge x_1' = in \wedge out = x_2 \wedge x_2' = temp$.

Composing Symbolic Representations

- Let C_1 and C_2 be compatible SRCs.
- If φ_I^1 and φ_I^2 are the respective initialization formulas for the symbolic representations of C_1 and C_2, then the initialization formula for the symbolic representation of $C_1 \| C_2$ is $\varphi_I^1 \wedge \varphi_I^2$.
- If φ_R^1 and φ_R^2 are the respective reaction formulas for the symbolic representations of C_1 and C_2, then the reaction formula for the symbolic representation of $C_1 \| C_2$ is $\varphi_R^1 \wedge \varphi_R^2$.

Composing Symbolic Representations

Ex. DoubleDelay

- The output variable $temp$ should be hidden: $\text{DoubleDelay} = (\text{Delay1} \parallel \text{Delay2}) \setminus temp$
- The corresponding symbolic representation is:
  - $C = (\{in\}, \{out\}, \{x_1, x_2\}, \varphi_I, \varphi_R)$ where $\varphi_I : x_1 = 0 \wedge x_2 = 0$ and $\varphi_R : \exists temp\,.\,(temp = x_1 \wedge x_1' = in \wedge out = x_2 \wedge x_2' = temp)$.
  - φ_R can be simplified to $x_1' = in \wedge out = x_2 \wedge x_2' = x_1$.

Composing Symbolic Representations

- Let C be an SRC and y be an output variable of C.
- If φ_I is the initialization formula for the symbolic representation of C, then the initialization formula for the symbolic representation of $C \setminus y$ is φ_I.
- If φ_R is the reaction formula for the symbolic representation of C, then the reaction formula for the symbolic representation of $C \setminus y$ is $\exists y \,.\, \varphi_R$.

Symbolic Breadth-First Search

Symbolic Algorithm for Invariant Verification

- **Region**: Symbolically Represented Set of States
  - Ex. If x is a Boolean state variable and $[\![Init]\!] = \{0\}$, then the region for the initialization is represented by the formula $x = 0$.
  - Ex. If x is a real state variable and $[\![Init]\!] = \{v \in \mathbb{R} \mid 0 \leq v \leq 10\}$, then the region for the initialization is represented by the formula $0 \leq x \leq 10$.
- Transitions are presented by regions over $S \cup S'$.
  - Ex. If $[\![Trans]\!]$ is given by an assignment $x := 2x + 1$, its symbolic representation $x' = 2x + 1$ also represents the region for $[\![Trans]\!]$.

Operations on Regions (1/2)

- Let V be a set of variables, and A, B be regions over V.
- $\mathsf{Disj}(A, B)$ returns the region for the states that are either A or B.
- $\mathsf{Conj}(A, B)$ returns the region for the states that are in both A and B.
- $\mathsf{Diff}(A, B)$ returns the region for the states that are in A but not in B.
- $\mathsf{IsEmpty}(A)$ returns 1 if A contains no states and 0 otherwise.
- Let $X \subseteq V$. $\mathsf{Exists}(A, X)$ returns the region A projected onto over $V \setminus X$.
  - If s is a valuation (state) in $\mathsf{Exists}(A, X)$, then s is a valuation over $V \setminus X$ and there exists a valuation t over X such that the valuation over V obtained by combining s and t is in A.
  - $\mathsf{Exists}(A, X)$ is given by $\exists x_1 \,.\, \ldots \,.\, \exists x_n \,.\, A$ for $x_1, \ldots, x_n \in X$.

Operations on Regions (2/2)

- Let $X = \{x_1, \ldots, x_n\}$ be a list of variables in V and $Y = \{y_1, \ldots, y_n\}$ be a list of variables not in V such that x_j and y_j have the same type for $j = 1, \ldots, n$.
- Rename(A, X, Y) returns the region obtained by renaming x_j to y_j for $j = 1, \ldots, n$.
  - Rename(A, X, Y) contains a valuation t over $(V \cup Y) \setminus X$ exactly when there exists a valuation s in A such that $t(y_j) = s(x_j)$ for $j = 1, \ldots, n$ and $t(z) = s(z)$ for $z \in V \setminus X$.

Symbolic Image Computation

- S : the set of state variables of transition system T
- $Trans$: the region over $S \cup S'$ representing the transitions of T
- Let A be a region over S. The *post-image* of A is defined by

$$\mathsf{Post}(A, Trans) = \mathsf{Rename}(\mathsf{Exists}(\mathsf{Conj}(A, Trans), S), S', S).$$

  - $\mathsf{Post}(A, Trans)$ is a region over S.
  - If $\mathsf{Post}(A, Trans)$ contains a state t, there exists a transition (s, t) in T for some $s \in A$.

Image Computation

Ex. 1

- Consider a transition system T with a single state variable x of type real, and the transition region is given by $x' = 2x + 1$ (corresponds to $x := 2x + 1$).
- Let A be a region given by $0 \leq x \leq 10$.
- Post$(A, Trans)$ can be calculated step-by-step as follows:
    - Conj$(A, Trans) = 0 \leq x \leq 10 \wedge x' = 2x + 1$
    - Exists(Conj$(A, Trans), \{x\}) = \exists x\,.\,(0 \leq x \leq 10 \wedge x' = 2x + 1) = (1 \leq x' \leq 21)$
    - Rename(Exists(Conj$(A, Trans), \{x\}), \{x'\}, \{x\}) = (1 \leq x \leq 21)$

Image Computation

Ex. 2 (1/2)

- Consider a transition system T with state variables x and y of type int.
- The transitions are defined with the statement

$$\text{if } (y > 0) \text{ then } x := x + 1 \text{ else } y := y - 1.$$

- The region for the transitions $Trans$ can be represented as the formula

$$[y > 0 \wedge x' = x + 1 \wedge y' = y] \vee [y \leq 0 \wedge x' = x \wedge y' = y - 1]$$

- Let A be a region given by the formula $2 \leq x - y \leq 5$.

Image Computation

Ex. 2 (2/2)

- Step-by-step calculation of $Post(A, Trans)$
  - Conj$(A, Trans) = [y > 0 \land x' = x + 1 \land y' = y \land 2 \leq x - y \leq 5]$
    $\lor [y \leq 0 \land x' = x \land y' = y - 1 \land 2 \leq x - y \leq 5]$.
  - Exists(Conj$(A, Trans), \{x, y\}) = [y' > 0 \land 2 \leq x' - 1 - y' \leq 5]$
    $\lor [y' + 1 \leq 0 \land 2 \leq x' - y' - 1 \leq 5]$.
  - Rename(Exists(Conj$(A, Trans), \{x, y\}), \{x', y'\}, \{x, y\}) =$ $(3 \leq x - y \leq 6) \land (y \neq 0)$.

Symbolic Breadth-First Search Algorithm

for Reachability

reg $Reach$:= $Init$;
reg New := $Init$;
while IsEmpty(New) = 0 **do** {
    if IsEmpty(Conj(New, φ)) = 0
        then return 1;
    New := Diff(Post(New, $Trans$), $Reach$);
    $Reach$:= Disj($Reach$, New)
};
return 0

- Input: transition system T and property φ
  - $Init$: region for initial state of T
  - $Trans$: region for transitions in T
- Output: 1 if φ is reachable in T, 0 otherwise

Symbolic Breadth-First Search Algorithm

- If the algorithm terminates, then the return value correctly indicates whether φ is reachable in T.
- If φ is reachable in T, then the algorithm terminates after j iterations of the while loop, where j is the length of the shortest witness to the reachability of φ.
- If there exists a natural number j, such that every reachable state of T is reachable by an execution with at most j transitions, then the algorithm terminates after at most j iterations of the while loop.

Summary

- Safety Requirements (2)
  - Automated Invariant Verification, Complexity
  - Simulation-Based Analysis, Falsification
  - Enumerative Search, Reachable Subgraph, On-the-Fly Depth-First Search
  - Symbolic Search, Symbolic Transition Systems, Regions, Symbolic Breadth-First Search