Symbolic Reachability

CS60030 FORMAL SYSTEMS

PALLAB DASGUPTA,
FNAE, FASc,
A K Singh Distinguished Professor in AI,
Dept of Computer Science & Engineering
Indian Institute of Technology Kharagpur
Email: pallab@cse.iitkgp.ac.in
Web: http://cse.iitkgp.ac.in/~pallab

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

FMSAFE

FORMAL METHODS FOR SAFETY CRITICAL SYSTEMS

State Transition Systems

In Computer Science we like to model dynamical systems as *state transition systems*.

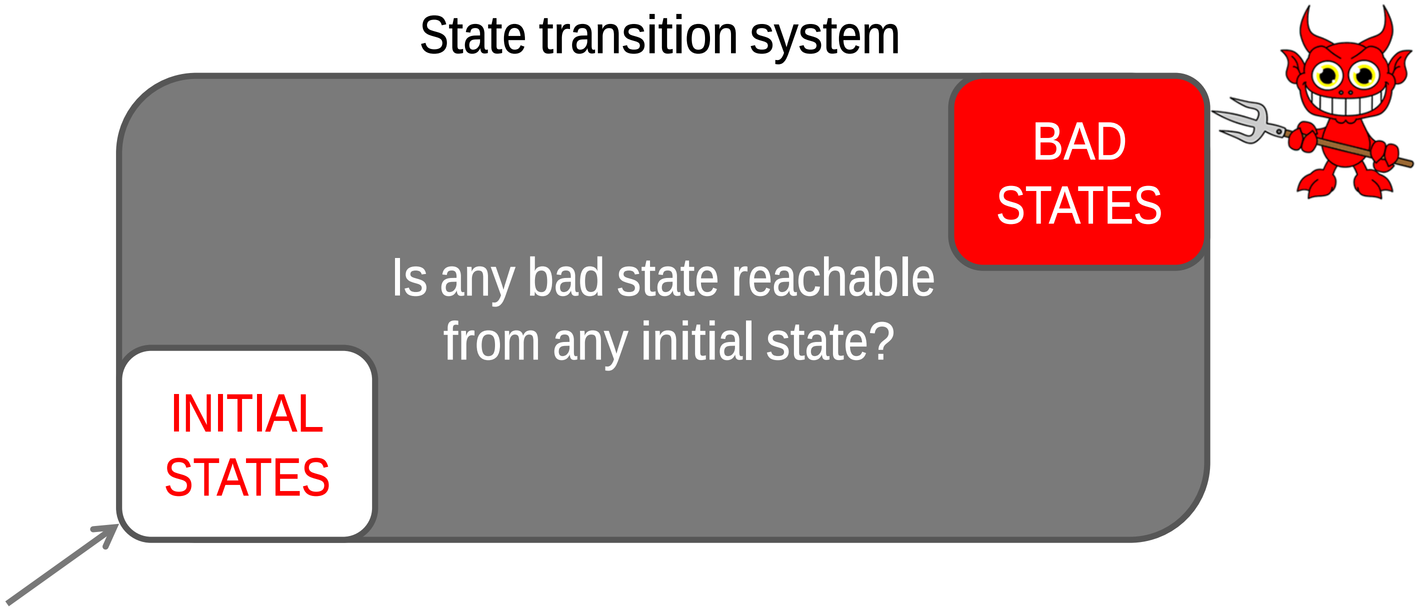
- An STS is a tuple $\langle Q, R, Q_0, Z \rangle$, where
  - Q is the set of states
  - Q_0 is the set of initial states. Obviously $Q_0 \subseteq Q$
  - $R \subseteq Q \times Q$ is a transition relation. Each state has at least one successor.
  - Z is a labeling function that labels each state with the outputs of our interest

- Q may not be finite – we shall discuss this later

- A program is also a STS. The current state of a program is $\langle \boldsymbol{l}, \boldsymbol{v} \rangle$ where $\boldsymbol{l}$ represents the current program location and $\boldsymbol{v}$ represents the current valuation of the program variables.

Formal Verification

State transition system



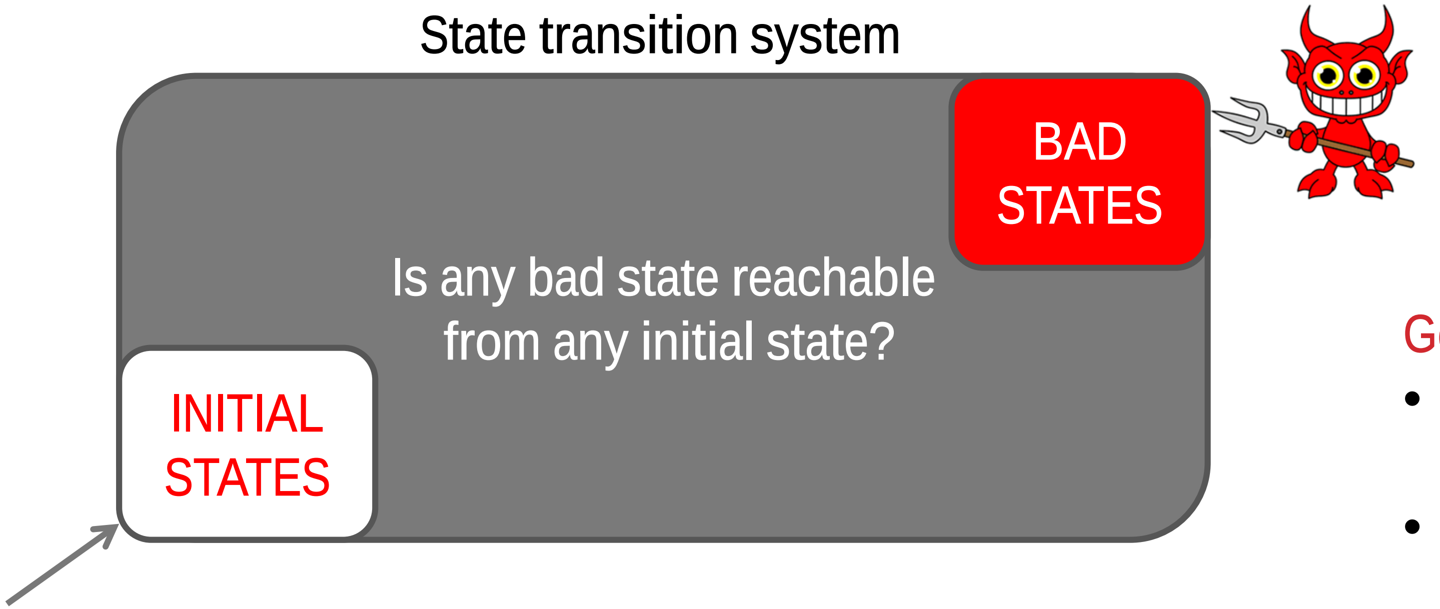
Simulation / Bug Hunting:

- Will explore only certain paths in the transition system
- May miss a path leading to a bad state

Goal of Formal Verification:

- To find a path leading to a bad state if it exists, or
- Guarantee that bad states are not reachable

Symbolic Search



Goal:

- To find a path leading to a bad state if it exists, or
- Guarantee that bad states are not reachable

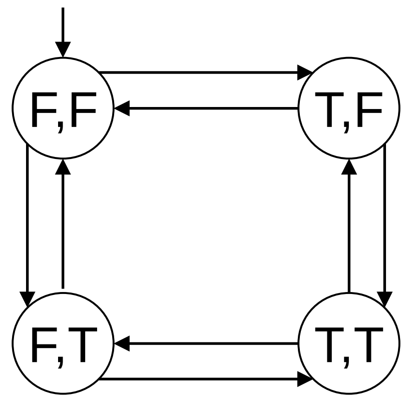
Will standard search techniques work?

- We could perform DFS or BFS from the set of initial states for example.

This will not work in general, because:

- The state space is too big (could be infinite also) – the state transition graph will not fit in memory
- But we have to know when we have seen all states reachable from the initial states (to terminate)
- We need search techniques that can work on a compact *symbolic* representation of the STS

A Simple Example



Variables: x, y: boolean

Set of states:

Q = {(F,F), (F,T), (T,F), (T,T)}

Initial condition:

$Q_0 \equiv \neg x \wedge \neg y$

Transition relation (negates one variable at a time):

$R \equiv [\ (x' = \neg x) \wedge (y' = y)\ ] \vee [\ (x' = x) \wedge (y' = \neg y)\ ]$ (= means $\leftrightarrow$)

x' is the next value of x, and y' is the next value of y

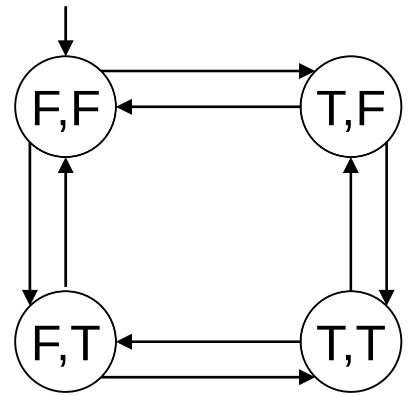
The Simple Example Contd.

FORWARD SEARCH: Start from the initial state and search for paths to the bad states.

BACKWARD SEARCH: Start from the bad states and work backwards to see whether we reach an initial state.

CORE STEP IN FORWARD SEARCH: *Find the set of successors of a given set state, S.*

Recall that sets of states can be modeled by Boolean functions.



Suppose $S \equiv \neg y$ (therefore this set contains the states (F,F) and (T,F)

Post-Image(S) $\equiv \exists x \exists y \; S \wedge R$

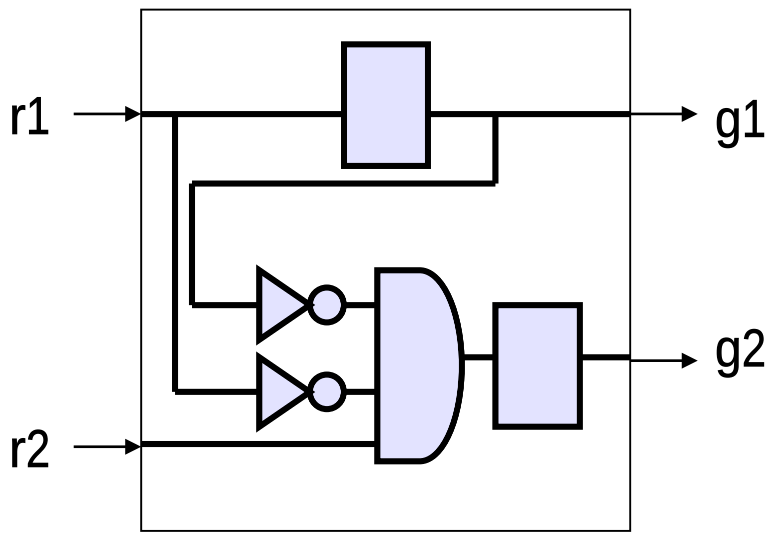
$\equiv \exists x \exists y \; (\neg y) \wedge [(x' = \neg x \wedge y' = y) \vee (x' = x \wedge y' = \neg y)]$

$\equiv \exists x \exists y \; (\neg y) \wedge [(x' = \neg x \wedge \neg y') \vee (x' = x \wedge y')]$

$\equiv [(x' \wedge \neg y') \vee (\neg x' \wedge y')] \vee [(\neg x' \wedge \neg y') \vee (x' \wedge y')] \equiv$ True

This formula represents the set of states {(T,F), (F,T), (F,F), (T,T)}, which is the set of successor states of S

One step of forward reachability (with BDDs)



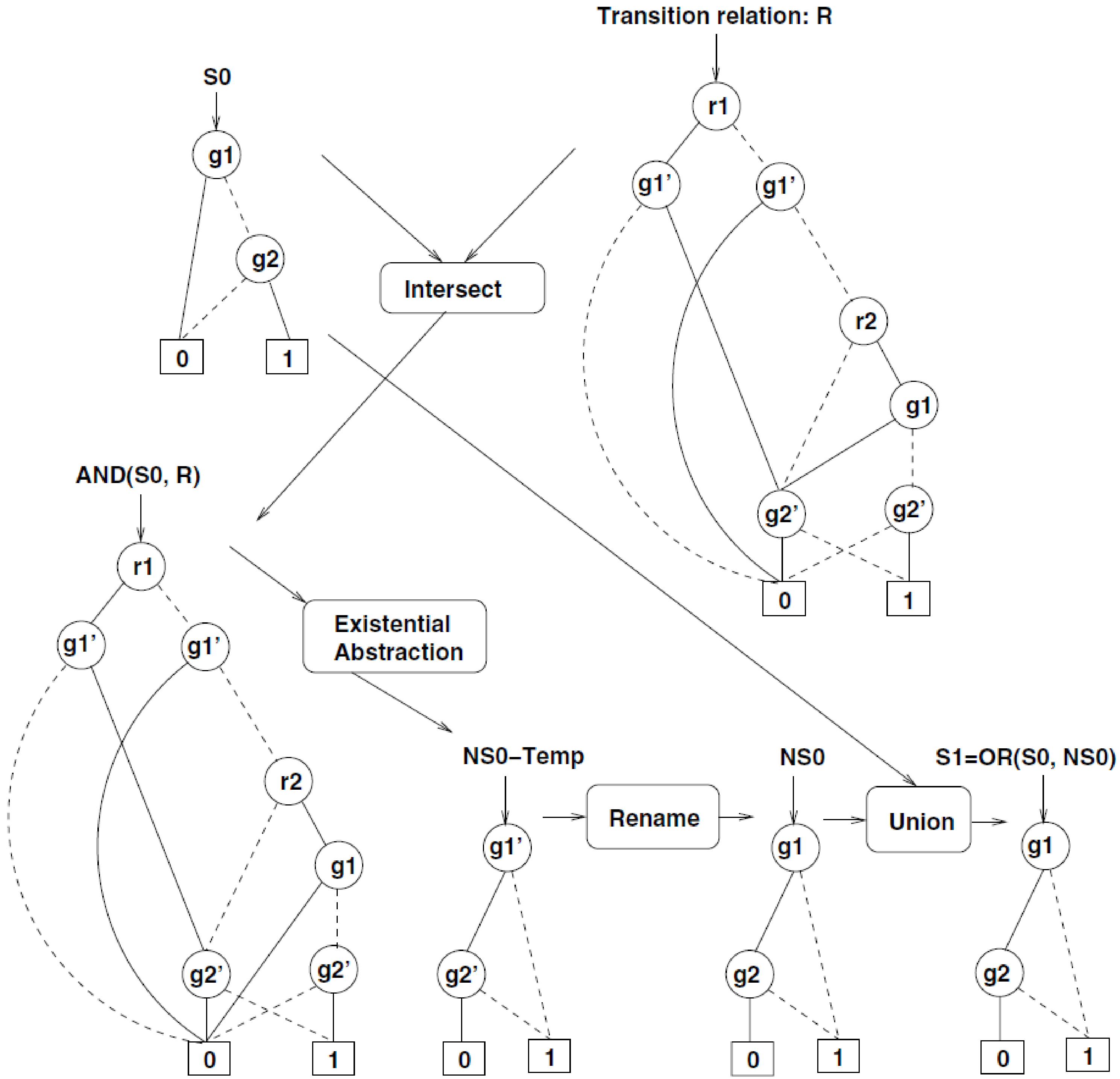
Transition Relation:

$g'_1 \Leftrightarrow r_1$

$g'_2 \Leftrightarrow \neg r_1 \wedge r_2 \wedge \neg g_1$

Set of next states of $\neg g_1 \wedge g_2$ is $\neg g_1 \vee \neg g_2$

Set of states reachable in at most one transition is also $\neg g_1 \vee \neg g_2$



Symbolic Forward Traversal

- We start with the set of initial states, I
- Then we successively compute:

$Z_0 = I$

$Z_1 = Z_0 \vee \text{Post-Image}(Z_0)$ // Z_1 represents all states reachable in zero or one step

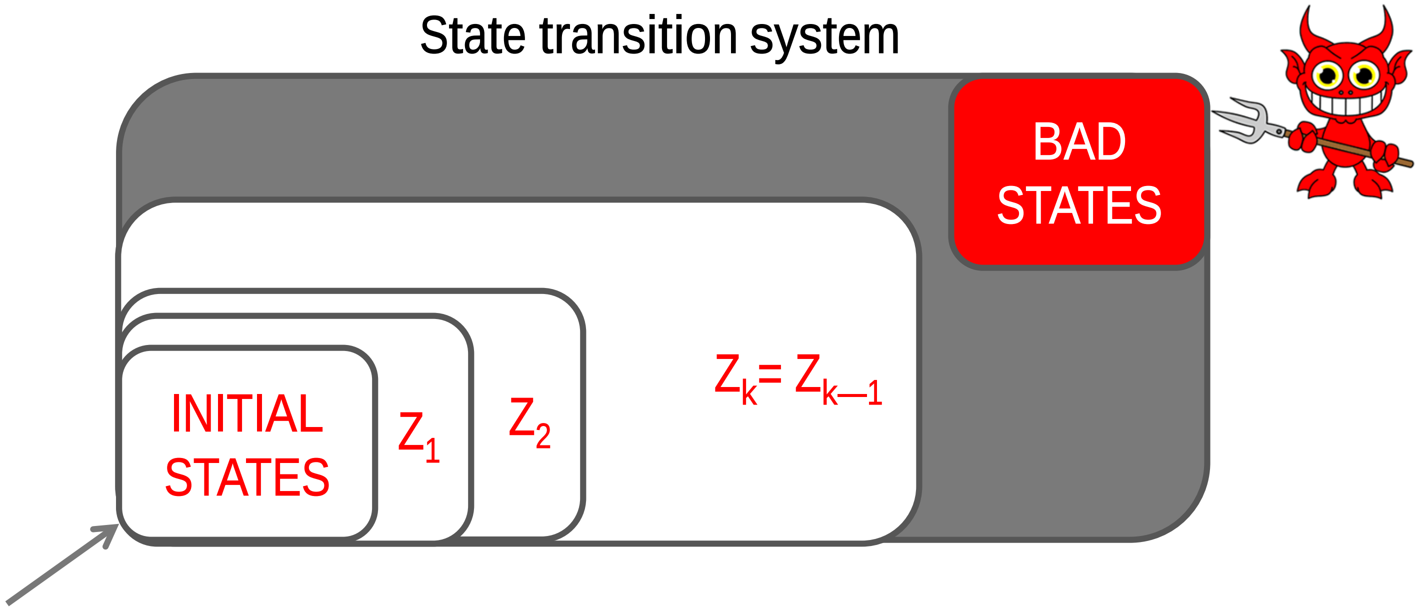
$Z_2 = Z_1 \vee \text{Post-Image}(Z_1)$ // Z_2 represents all states reachable in at most two steps

…

$Z_k = Z_{k-1} \vee \text{Post-Image}(Z_{k-1})$ // Z_k represents all states reachable in at most *k* steps

- Since the state machine has a finite number of states, we will reach an iteration where $Z_k = Z_{k-1}$
- This is called the fixpoint of the transition function, and Z_k represents the set of reachable states starting from the initial states in I.

Symbolic Forward Search



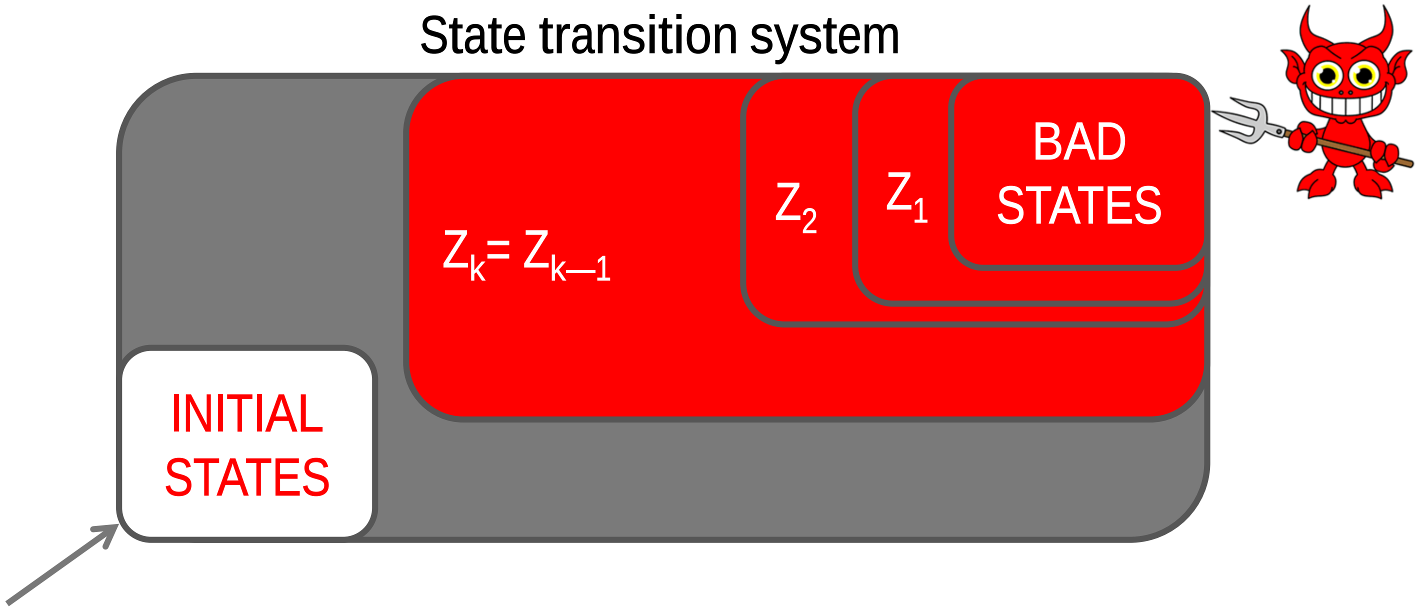
If no bad state is reachable, then we reach the fix point, Z_k and still $Z_k \cap$ BadStates = $\varnothing$

- This leads us to conclude that the bad states are not reachable from the initial states

If a bad state is reachable, then $Z_j \cap$ BadStates $\neq \varnothing$ for some $j \leq k$

- A satisfiability check on $Z_j \cap$ BadStates will reveal whether a bad state is reachable from some initial state
- We need to produce a counter-example. This will be taken up later.

Symbolic Backward Search



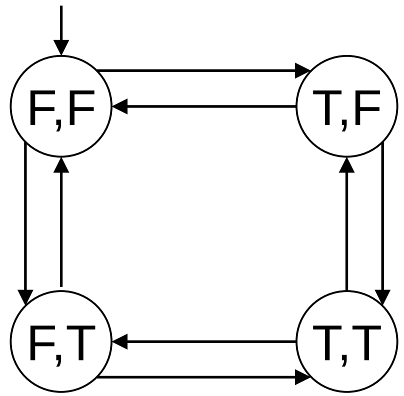
- Since we know the set of bad states (such as all green signals in a traffic intersection), we could represent the BadStates as a Boolean formula.
- We could also work backward from the bad states to see whether we can reach the initial states. See the next slide.
- Could we go backward in simulation?

The Simple Example – Now we try backward search

Suppose $p \equiv x \wedge y$ defines the set of bad states.

BACKWARD SEARCH: Start from the bad states and work backwards to see whether we reach an initial state.

CORE STEP IN BACKWARD SEARCH: *Find the states that have a successor satisfying p*



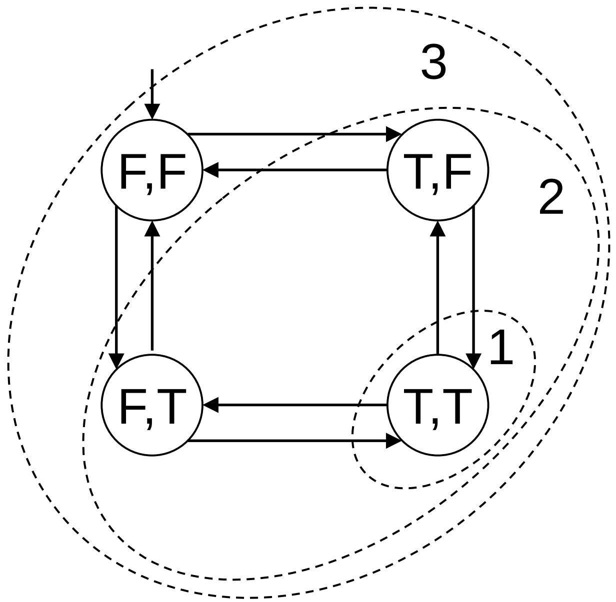
$\text{Pre-Image}(p) \equiv \exists x' \; \exists y' \; R \wedge (x' \wedge y')$

$\equiv \exists x' \; \exists y' \; [(x' = \neg x \wedge y' = y) \vee (x' = x \wedge y' = \neg y)] \wedge (x' \wedge y')$

$\equiv [\neg x \wedge y] \vee [x \wedge \neg y]$

This formula represents the set of states {(F,T), (T,F)}, which is the set of states having a successor satisfying p

The Simple Example Contd.



Suppose $p \equiv x \wedge y$ defines the set of bad states.

Pre-Image(p) $\equiv [\neg x \wedge y] \vee [x \wedge \neg y]$

FIXPOINT COMPUTATION for BACWARD REACHABILITY

$Z_0 = p$

$Z_1 = Z_0 \vee$ Pre-Image(Z_0)

$Z_2 = Z_1 \vee$ Pre-Image(Z_1)

... and so on, until we have $Z_k = Z_{k-1}$ for some *k*. We call it Z*

Then Z_k is a Boolean formula that represents the set of states that can reach the bad states.

We have a bug if $Q_0 \wedge Z_k$ is satisfiable.

A liveness property

We have been discussing *safety properties* so far. With safety properties we wish to prove that something bad will *never* happen.

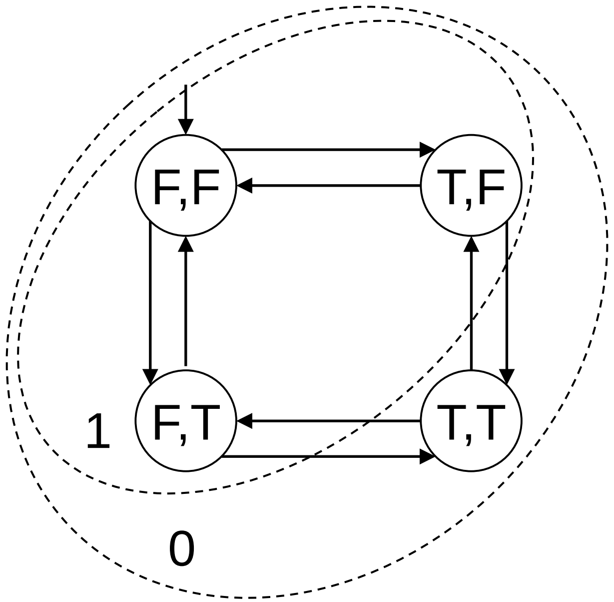
Lets now consider a *liveness property*. A liveness property is used to express that something good will *eventually* happen. This means that we wish to prove that good states will always be reached.

Suppose the good states we wish to reach is given by $(x \wedge y)$.

We shall search for an infinite path (that is, a path which loops) where no state satisfies $(x \wedge y)$.

- If such a path exists then that (infinite) path is a counter-example
- Otherwise, the liveness property holds.

Checking the Liveness Property



Suppose $p \equiv x \wedge y$ defines the set of good states.

Pre-Image(p) $\equiv [\neg x \wedge y] \vee [x \wedge \neg y]$

FIXPOINT COMPUTATION

Z_0= True

$Z_1 = \neg p$

$Z_2 = Z_1 \wedge$ Pre-Image(Z_1)   // Set of states that do not satisfy p and have a successor not satisfying p

$Z_1 = Z_2 \wedge$ Pre-Image(Z_2)

… and so on, until we have $Z_k = Z_{k-1}$ for some *k*. We call it Z*

$Z^* \equiv \neg x \vee \neg y$

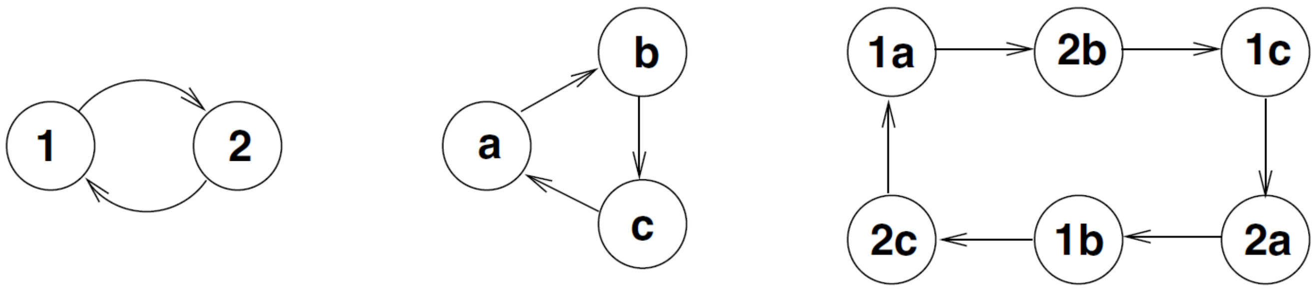
Since $Q_0 \wedge EG(\neg(x \wedge y)) \neq \varnothing$ we conclude that the liveness property does not hold.

Checking Invariants

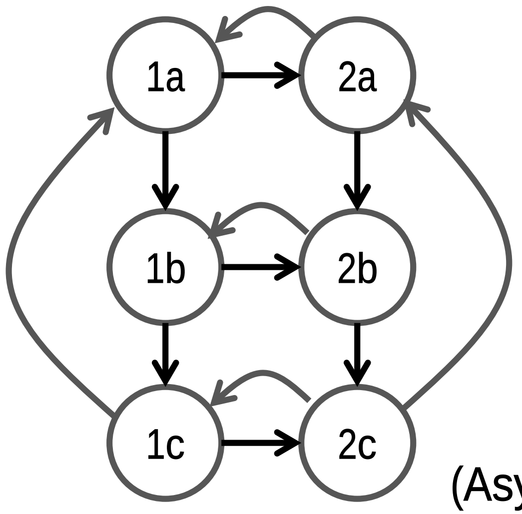
- An invariant is a property that must hold in all reachable states.
  - For example safety properties which are state properties, such as *the two traffic lights at a crossing must never be green together*

- Using symbolic reachability
  - Find the set Z_k of reachable states
  - Model the property as a Boolean formula P over the state variables
  - Check whether $Z_k \wedge \neg P$ is satisfiable. If not, then P is an invariant

A note on Asynchronous Composition



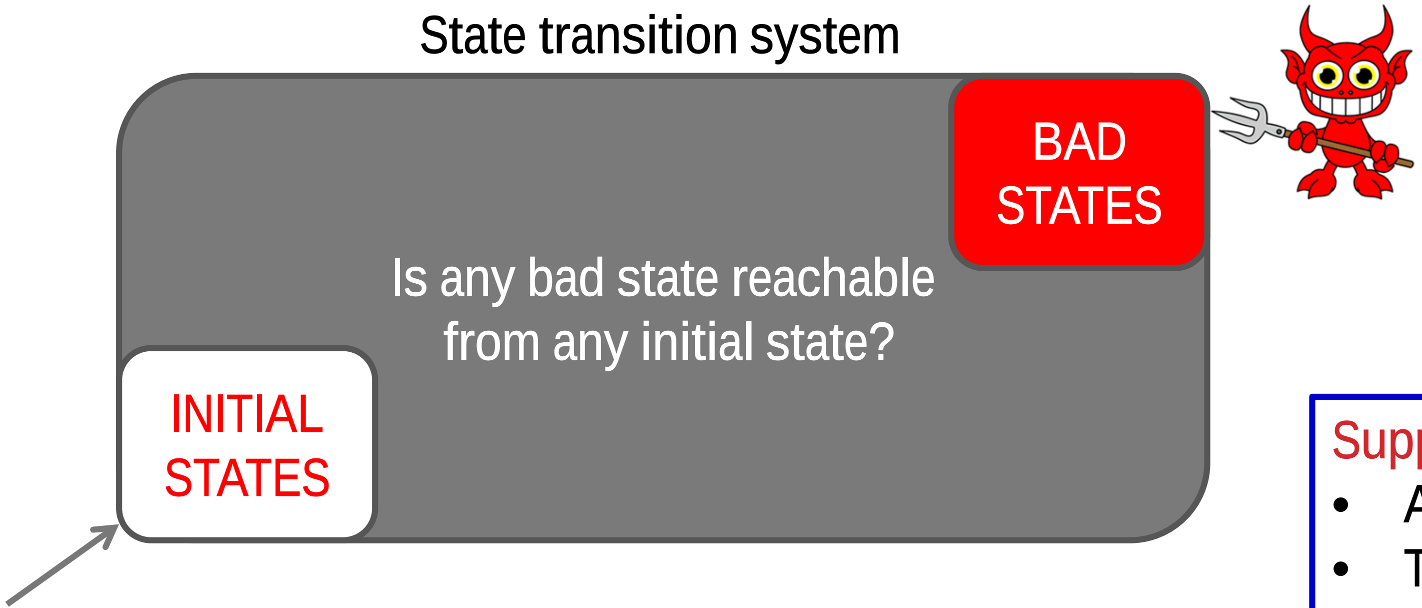
M1 **M2** **M1 X M2**
(Synchronous)



M1 X M2
(Asynchronous)

- Composition is the primary cause of state explosion
- Can we do reachability analysis without composition of M1, M2?
- For asynchronous composition, we can independently find the reachable states of M1 and the reachable states of M2, and then take their product.

The intuitive basis for induction



Suppose we prove the following:

- All initial states are good, and
- The transition relation does not allow any transition from a good state to a bad state

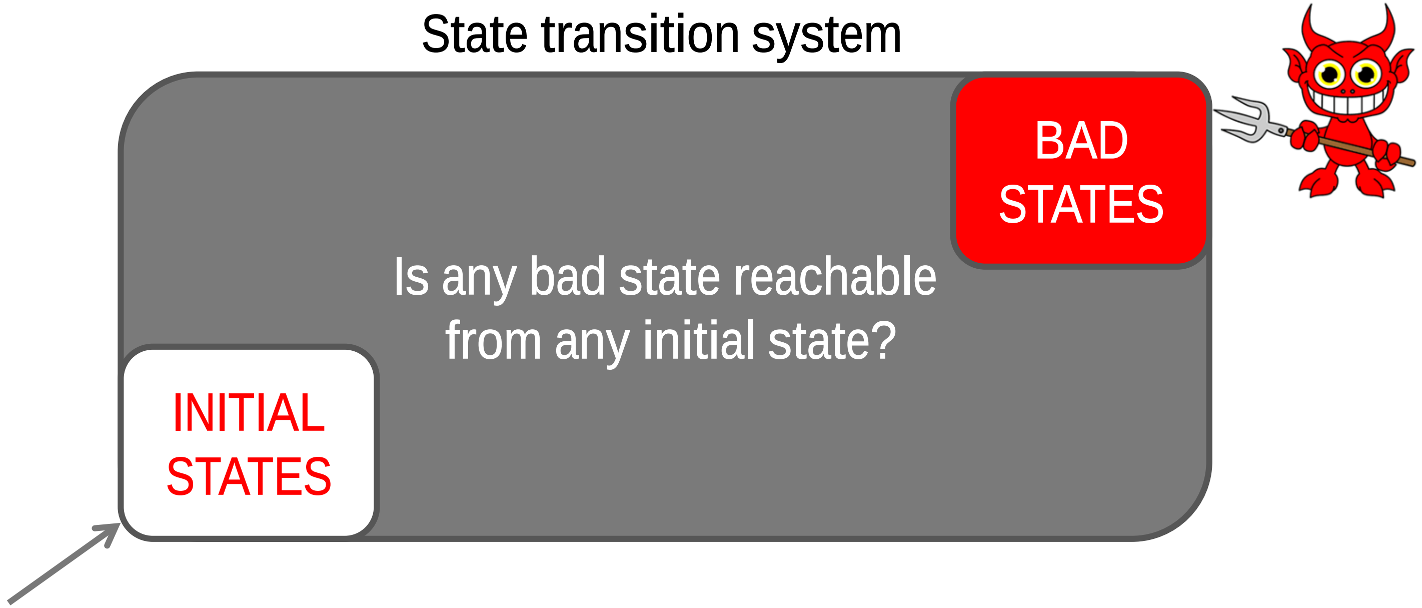
Then inductively, we are safe

Let p be the formula representing bad states
Then we check:

1. Whether $Q_0 \wedge p$ is empty
2. Whether PreImage(p) $\wedge \neg p$ is empty

If both are true, then we have inductively shown that bad states are unreachable

The notion of k-induction



For k= 0, 1, ...

1. Check whether any state reachable from Q_0 in k or fewer steps is bad.

   If so, report counterexample and exit.
2. Check whether R guarantees that there is no transition to a bad state after k safe steps

   If so, exit with success.
3. Otherwise continue to the next iteration

For finite state systems we can guarantee that the above will terminate in a finite number of iterations.