CS:4350 Logic in Computer Science

Model Checking

Cesare Tinelli

Spring 2021

Credits

These slides are largely based on slides originally developed by **Andrei Voronkov** at the University of Manchester. Adapted by permission.

Outline

Putting it All Together

When we design a computational system, we would like to be sure that it will satisfy all requirements, including *safety* requirements

Putting it All Together

When we design a computational system, we would like to be sure that it will satisfy all requirements, including *safety* requirements

Now we can treat the safety problem as a logical problem.

Putting it All Together

When we design a computational system, we would like to be sure that it will satisfy all requirements, including *safety* requirements

Now we can treat the safety problem as a logical problem.
We can

- formally represent our system as a transition system
- express the desired properties of the system in temporal logic

Putting it All Together

When we design a computational system, we would like to be sure that it will satisfy all requirements, including *safety* requirements

Now we can treat the safety problem as a logical problem.
We can

- formally represent our system as a transition system
- express the desired properties of the system in temporal logic

What is missing?

The Model Checking Problem

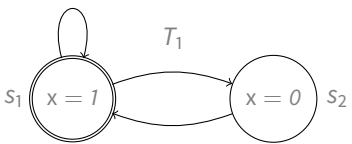
Given

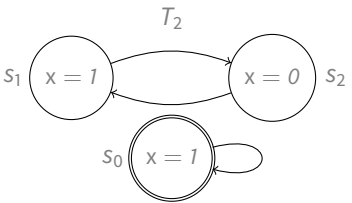
1. a symbolic representation of a transition system
2. a temporal formula F

check if every (some) execution of the system satisfies this formula,
preferably fully automatically

Symbolic Representation and Transition Systems

Consider the transition systems T_1 and T_2:

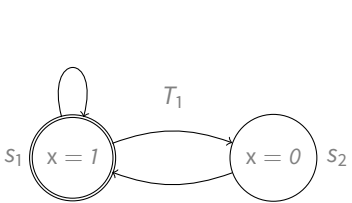


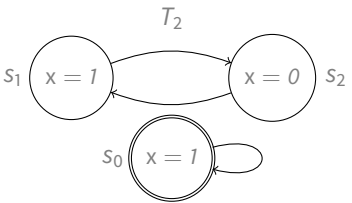


T_1 and T_2 have the same symbolic representation but satisfy different LTL formulas (e.g., $\Diamond \neg x$)

Symbolic Representation and Transition Systems

Consider the transition systems T_1 and T_2:



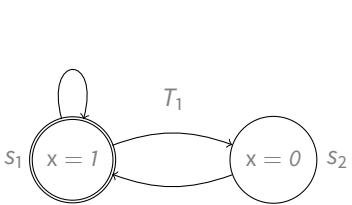


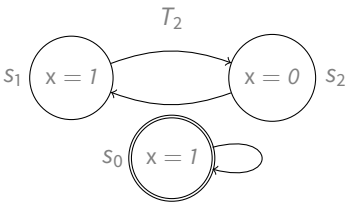
T_1 and T_2 have the same symbolic representation but satisfy different LTL formulas (e.g., $\Diamond\neg x$)

This happens only if one of the transition systems has two states with the same labelling function (e.g., s_0 and s_1 in T_2)

Symbolic Representation and Transition Systems

Consider the transition systems T_1 and T_2:





T_1 and T_2 have the same symbolic representation but satisfy different LTL formulas (e.g., $\Diamond\neg x$)

This happens only if one of the transition systems has two states with the same labelling function (e.g., s_0 and s_1 in T_2)

Such symbolic representations are *inadequate*: one cannot distinguish two different states by a state formula

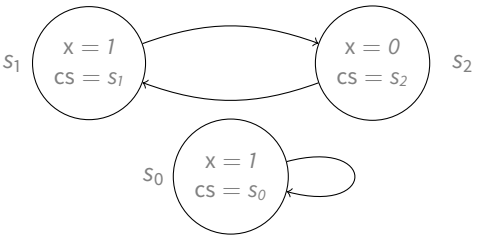
Making an Adequate Representation

If a transition system has different states labeled by the same interpretation, introduce a new state variable to distinguish any such pair of states

Making an Adequate Representation

If a transition system has different states labeled by the same interpretation, introduce a new state variable to distinguish any such pair of states

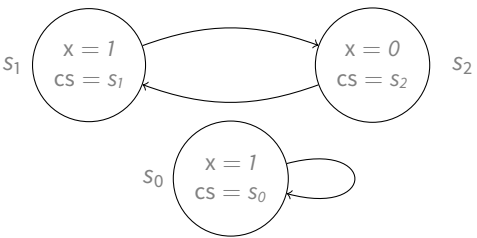
Example: One can add a *current state* variable cs with a unique value for each state



Making an Adequate Representation

If a transition system has different states labeled by the same interpretation, introduce a new state variable to distinguish any such pair of states

Example: One can add a *current state* variable cs with a unique value for each state



We will assume that different states always have different labelings

Reachability and Safety Properties

Reachability property: expressed by a formula for the form

$$\Diamond F$$

where F is a propositional formula[1]

[1] Could be a PLFD. Restriction to PL is for simplicity.

Reachability and Safety Properties

Reachability property: expressed by a formula for the form

$$\Diamond F$$

where F is a propositional formula[1]

Safety/invariance property: expressed by a formula of the form

$$\Box F$$

where F is a propositional formula

[1] Could be a PLFD. Restriction to PL is for simplicity.

Reachability and Safety Properties

Reachability property: expressed by a formula for the form

$$\Diamond F$$

where F is a propositional formula[1]

Safety/invariance property: expressed by a formula of the form

$$\Box F$$

where F is a propositional formula

Most common problems arising in model checking. They are dual to each other:

$$\Box F \equiv \neg\Diamond\neg F \qquad\qquad \Diamond F \equiv \neg\Box\neg F$$

[1] Could be a PLFD. Restriction to PL is for simplicity.

Reachability and Safety Properties

Reachability property: expressed by a formula for the form

$$\Diamond F$$

where F is a propositional formula[1]

Safety/invariance property: expressed by a formula of the form

$$\Box F$$

where F is a propositional formula

Most common problems arising in model checking. They are dual to each other:

$$\Box F \equiv \neg\Diamond\neg F \qquad\qquad \Diamond F \equiv \neg\Box\neg F$$

Cannot reach an unsafe state iff all reachable states are safe

[1] Could be a PLFD. Restriction to PL is for simplicity.

Reachability

Fix a transition system $\mathbb{S}$ with transition relation T over states S

We write $s_0 \to s_1$ if $(s_0, s_1) \in T$, i.e., if there is a transition from state s_0 to state s_1

Let $s \in S$

- s is *reachable in n steps from a state* $s_0 \in S$ if there exist states $s_1, \ldots, s_n \in S$ such that $s_n = s$ and $s_0 \to s_1 \to \cdots \to s_n$
- $s \in S$ is *reachable from a state* $s_0 \in S$ if s is reachable from s_0 in $n \geq 0$ steps
- $s \in S$ is *reachable in* $\mathbb{S}$ if s is reachable from some initial state of $\mathbb{S}$

Reachability

Fix a transition system $\mathbb{S}$ with transition relation T over states S

We write $s_0 \to s_1$ if $(s_0, s_1) \in T$, i.e., if there is a transition from state s_0 to state s_1

Let $s \in S$

- s is *reachable in n steps from a state* $s_0 \in S$ if there exist states $s_1, \ldots, s_n \in S$ such that $s_n = s$ and $s_0 \to s_1 \to \cdots \to s_n$
- $s \in S$ is *reachable from a state* $s_0 \in S$ if s is reachable from s_0 in $n \geq 0$ steps
- $s \in S$ is *reachable in* $\mathbb{S}$ if s is reachable from some initial state of $\mathbb{S}$

Reachability

Fix a transition system $\mathbb{S}$ with transition relation T over states S

We write $s_0 \to s_1$ if $(s_0, s_1) \in T$, i.e., if there is a transition from state s_0 to state s_1

Let $s \in S$

- s is *reachable in n steps from a state* $s_0 \in S$ if there exist states $s_1, \ldots, s_n \in S$ such that $s_n = s$ and $s_0 \to s_1 \to \cdots \to s_n$
- $s \in S$ is *reachable from a state* $s_0 \in S$ if s is reachable from s_0 in $n \geq 0$ steps
- $s \in S$ is *reachable in* $\mathbb{S}$ if s is reachable from some initial state of $\mathbb{S}$

Reachability

Fix a transition system $\mathbb{S}$ with transition relation T over states S

We write $s_0 \to s_1$ if $(s_0, s_1) \in T$, i.e., if there is a transition from state s_0 to state s_1

Let $s \in S$

- s is *reachable in n steps from a state* $s_0 \in S$ if there exist states $s_1, \ldots, s_n \in S$ such that $s_n = s$ and $s_0 \to s_1 \to \cdots \to s_n$
- $s \in S$ is *reachable from a state* $s_0 \in S$ if s is reachable from s_0 in $n \geq 0$ steps
- $s \in S$ is *reachable in* $\mathbb{S}$ if s is reachable from some initial state of $\mathbb{S}$

Reachability Properties and Graph Reachability

Theorem 1

A reachability property $\Diamond F$ holds on some computation path iff $s \models F$ for some reachable state s.

Reformulation of Reachability

Given

1. An *initial condition* I denoting the set of initial states of a transition system $\mathbb{S}$
2. A *final condition* F denoting a set of final states
3. A *transition formula* Tr denoting the transition relation of $\mathbb{S}$

is any final state reachable from an initial state?

Reformulation of Reachability

Given

1. An *initial condition* I denoting the set of initial states of a transition system $\mathbb{S}$
2. A *final condition* F denoting a set of final states
3. A *transition formula* Tr denoting the transition relation of $\mathbb{S}$

is any final state reachable from an initial state?

Note: this reformulation does not use temporal logic

Symbolic Reachability Checking

Main Idea: build a symbolic representation of the set of reachable states

Two main kinds of algorithm:

- forward reachability
- backward reachability

Symbolic Reachability Checking

Main Idea: build a symbolic representation of the set of reachable states

Two main kinds of algorithm:

- forward reachability
- backward reachability

Reachability as a Decision Problem

Let $\boldsymbol{x} = x_1, \ldots, x_n$ be state variables

Given

1. a formula $I(\boldsymbol{x})$, the *initial condition*
2. a formula $F(\boldsymbol{x})$, the *final condition*
3. formula $T(\boldsymbol{x}, \boldsymbol{x}')$, the *transition formula*

is there a sequence of states $s_0, \ldots, s_n$ such that

1. $s_0 \models I(\boldsymbol{x})$
2. $(s_{i-1}, s_i) \models T(\boldsymbol{x}, \boldsymbol{x}')$ for all $i = 0, \ldots, n-1$
3. $s_n \models F(\boldsymbol{x})$

Note that in this case s_n is reachable from s_0 in n steps

Reachability as a Decision Problem

Let $\boldsymbol{x} = x_1, \ldots, x_n$ be state variables

Given

1. a formula $I(\boldsymbol{x})$, the *initial condition*
2. a formula $F(\boldsymbol{x})$, the *final condition*
3. formula $T(\boldsymbol{x}, \boldsymbol{x}')$, the *transition formula*

is there a sequence of states $s_0, \ldots, s_n$ such that

1. $s_0 \models I(\boldsymbol{x})$
2. $(s_{i-1}, s_i) \models T(\boldsymbol{x}, \boldsymbol{x}')$ for all $i = 0, \ldots, n-1$
3. $s_n \models F(\boldsymbol{x})$

Note that in this case s_n is reachable from s_0 in n steps

Idea of Reachability-Checking Algorithms

Note: If a final state is reachable from an initial state, it is reachable (from an initial state) in some number n of steps

Approach: For given number $n \geq 0$, find a formula denoting the set of states reachable in n steps

If this formula is not satisfied in a final state, increase n and start again

Idea of Reachability-Checking Algorithms

Note: If a final state is reachable from an initial state, it is reachable (from an initial state) in some number n of steps

Approach: For given number $n \geq 0$, find a formula denoting the set of states reachable in n steps

If this formula is not satisfied in a final state, increase n and start again

Idea of Reachability-Checking Algorithms

Note: If a final state is reachable from an initial state, it is reachable (from an initial state) in some number n of steps

Approach: For given number $n \geq 0$, find a formula denoting the set of states reachable in n steps

If this formula is not satisfied in a final state, increase n and start again

Idea of Reachability-Checking Algorithms

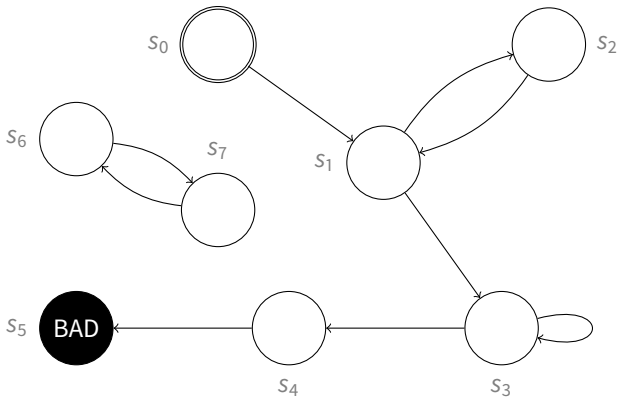
Note: If a final state is reachable from an initial state, it is reachable (from an initial state) in some number n of steps

Approach: For given number $n \geq 0$, find a formula denoting the set of states reachable in n steps

If this formula is not satisfied in a final state, increase n and start again

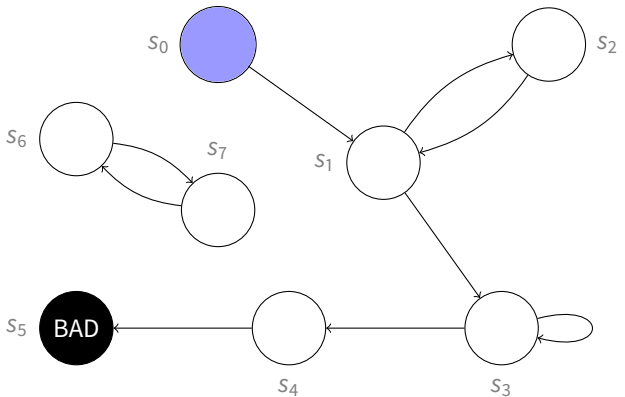
When does this process terminate?

Reachability in n steps



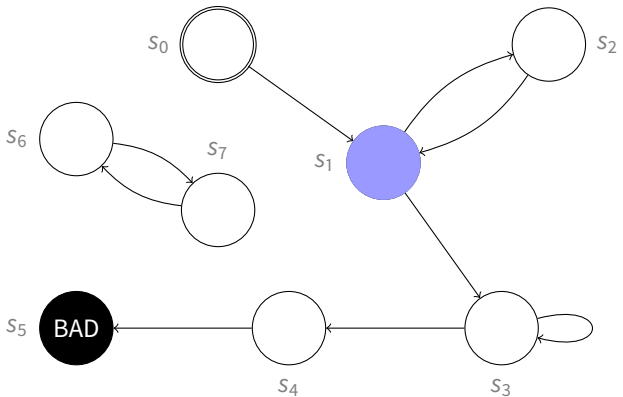
Reachability in n steps

Number of steps: 0



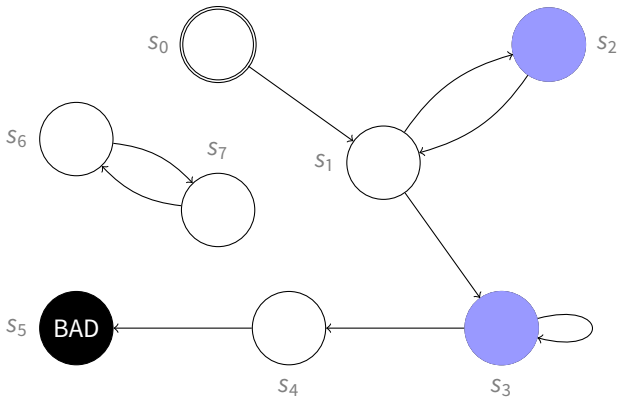
Reachability in n steps

Number of steps: 1



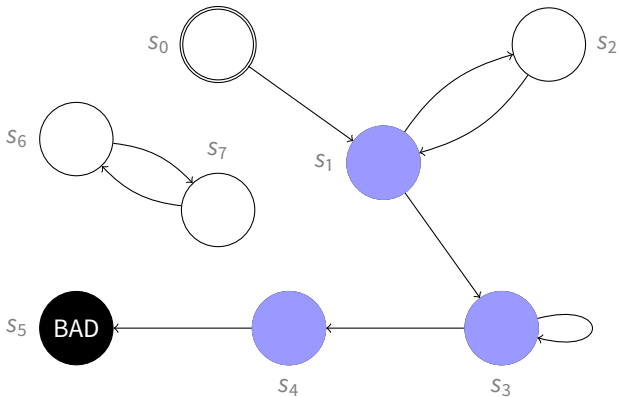
Reachability in n steps

Number of steps: 2



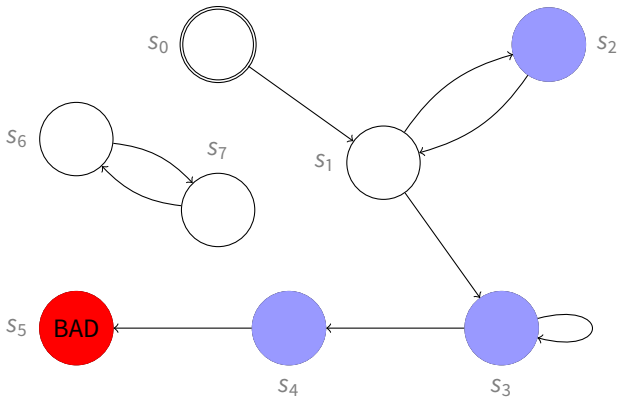
Reachability in n steps

Number of steps: 3



Reachability in n steps

Number of steps: 4



Simple Logical Analysis

Notation If $\boldsymbol{z} = (z_1, \ldots, z_n)$ is a tuple of variables, $\exists \boldsymbol{z} F$ abbreviates $\exists z_1 \cdots \exists z_n F$

Lemma 2

Let $C(\boldsymbol{x})$ symbolically represent a set of states S_C. The formula

$$FR(\boldsymbol{x}) \quad \stackrel{\text{def}}{=} \quad \exists \boldsymbol{z}(C(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$$

represents the set of states reachable from S_C in one step.

Simple Logical Analysis

Notation If $\boldsymbol{z} = (z_1, \ldots, z_n)$ is a tuple of variables, $\exists \boldsymbol{z} F$ abbreviates $\exists z_1 \cdots \exists z_n F$

Lemma 2

Let $C(\boldsymbol{x})$ symbolically represent a set of states S_C. The formula

$$FR(\boldsymbol{x}) \stackrel{\text{def}}{=} \exists \boldsymbol{z}(C(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$$

represents the set of states reachable from S_C in one step.

Each formula R_n defined inductive as follows:

$$\begin{aligned} R_0(\boldsymbol{x}) &\stackrel{\text{def}}{=} I(\boldsymbol{x}) \\ R_{n+1}(\boldsymbol{x}) &\stackrel{\text{def}}{=} \exists \boldsymbol{z}(R_n(\boldsymbol{z}) \wedge T(\boldsymbol{x}, \boldsymbol{z})) \end{aligned}$$

denotes the set of states reachable in n steps

Simple Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or no output
begin
 $i := 0$
 $R := I(\boldsymbol{x}_0)$
 loop
  if $R \wedge F(\boldsymbol{x}_i)$ is satisfiable **then return** "yes"
  $R := R \wedge T(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$
  $i := i + 1$
 end loop
end

Simple Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or no output
begin
 $i := 0$
 $R := I(\boldsymbol{x}_0)$
 loop
  if $R \wedge F(\boldsymbol{x}_i)$ is satisfiable **then return** "yes"
  $R := R \wedge T(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$
  $i := i + 1$
 end loop
end

How do we check the satisfiability of $R \wedge F(\boldsymbol{x}_i)$?

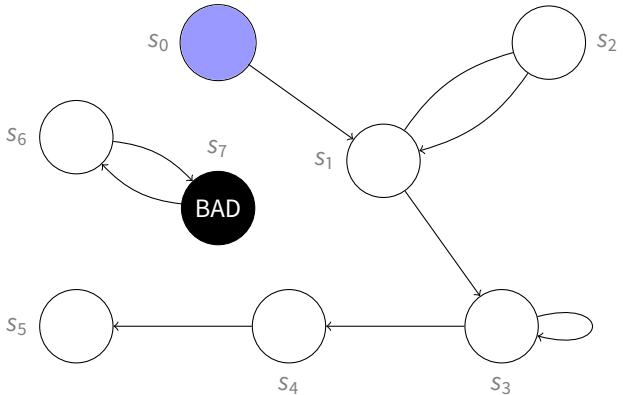
Simple Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or no output
begin
 $i := 0$
 $R := I(\boldsymbol{x}_0)$
 loop
  if $R \wedge F(\boldsymbol{x}_i)$ is satisfiable **then return** "yes"
  $R := R \wedge T(\boldsymbol{x}_i, \boldsymbol{x}_{i+1})$
  $i := i + 1$
 end loop
end

How do we check the satisfiability of $R \wedge F(\boldsymbol{x}_i)$? Using SAT solvers!

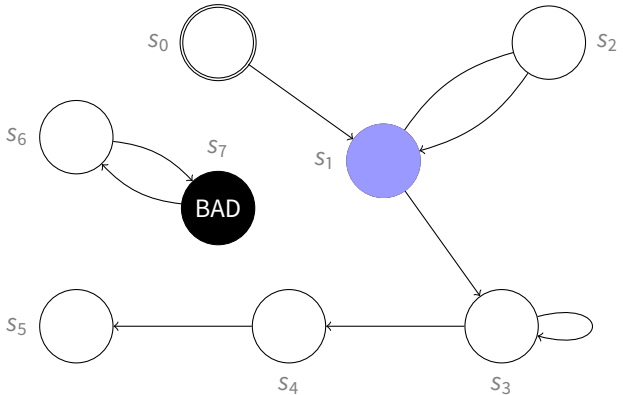
Termination

Number of steps: 0



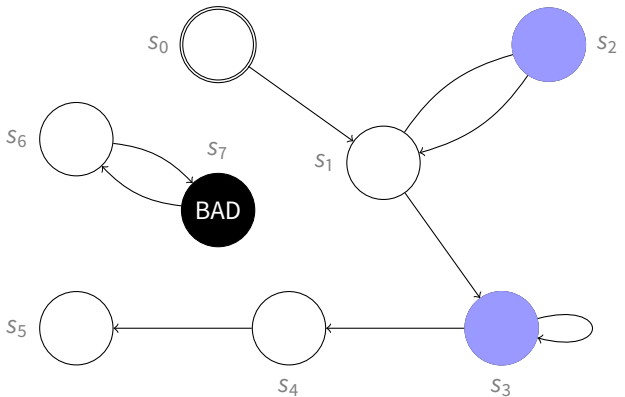
Termination

Number of steps: 1



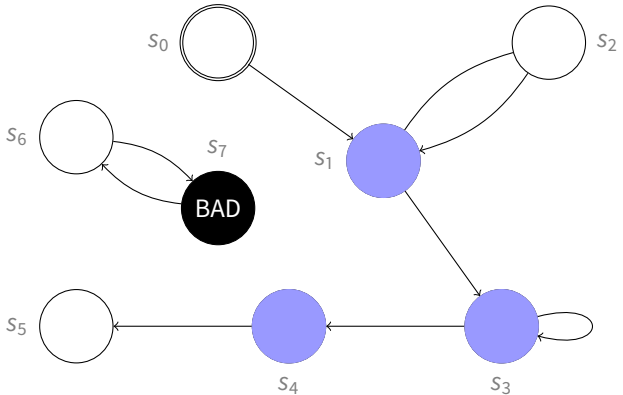
Termination

Number of steps: 2



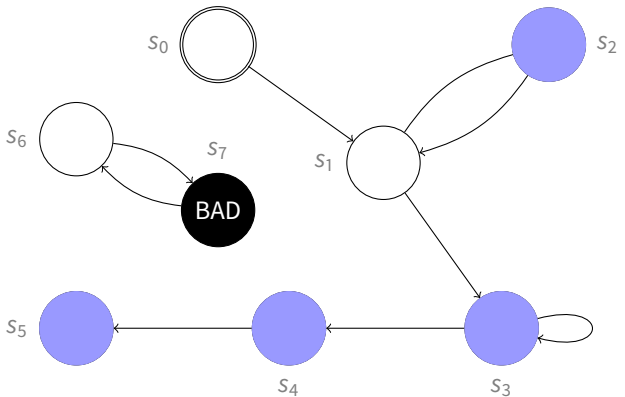
Termination

Number of steps: 3



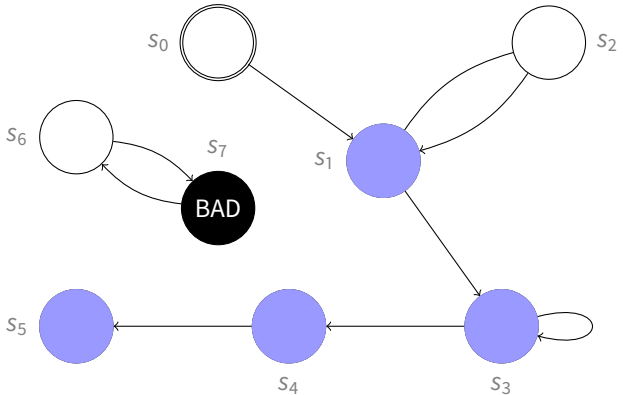
Termination

Number of steps: 4



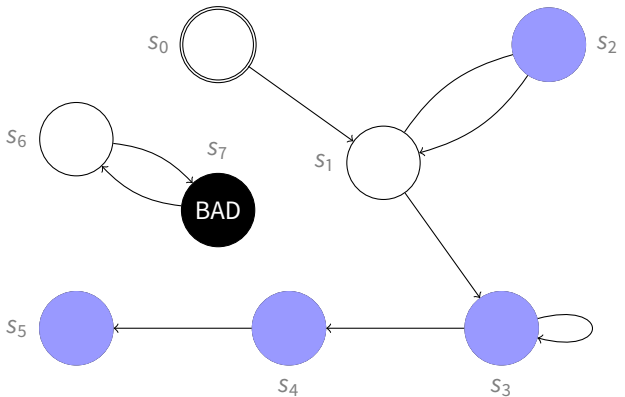
Termination

Number of steps: 5



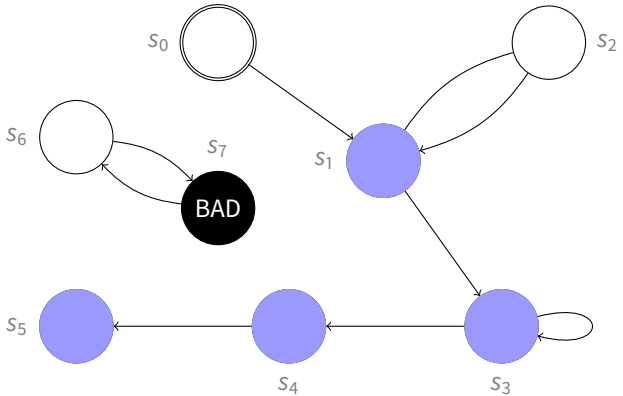
Termination

Number of steps: 6



Termination

Number of steps: 7



When no final state is reachable, the algorithm does not terminate!

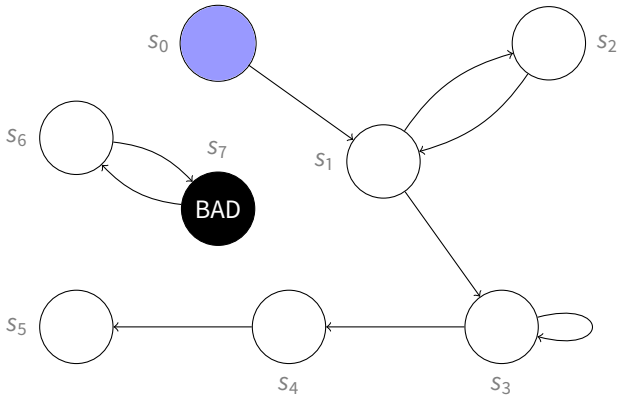
Reachability in $\leq n$ steps

Define a sequence of formulas $R_{\leq n}$ for reachability in at most n states:

$$\begin{aligned} R_{\leq 0}(\boldsymbol{x}) &\stackrel{\text{def}}{=} I(\boldsymbol{x}) \\ R_{\leq n+1}(\boldsymbol{x}) &\stackrel{\text{def}}{=} R_{\leq n}(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R_{\leq n}(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x})) \end{aligned}$$

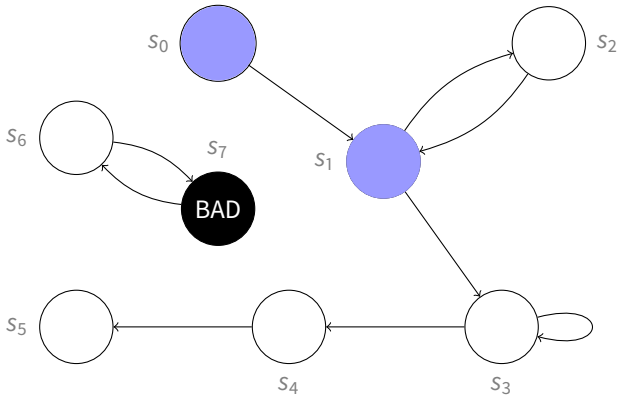
Reachability in $\leq n$ steps

Number of steps: 0



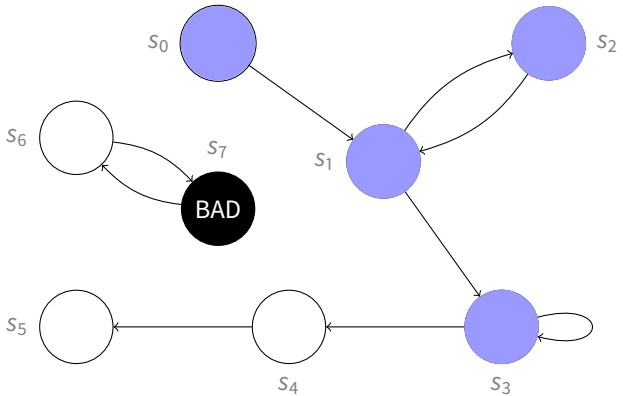
Reachability in $\leq n$ steps

Number of steps: 1



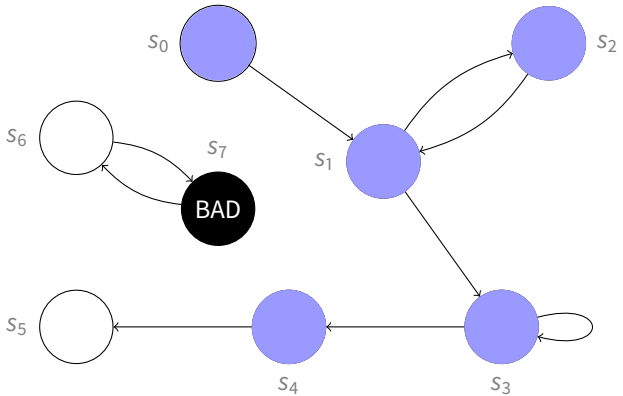
Reachability in $\leq n$ steps

Number of steps: 2



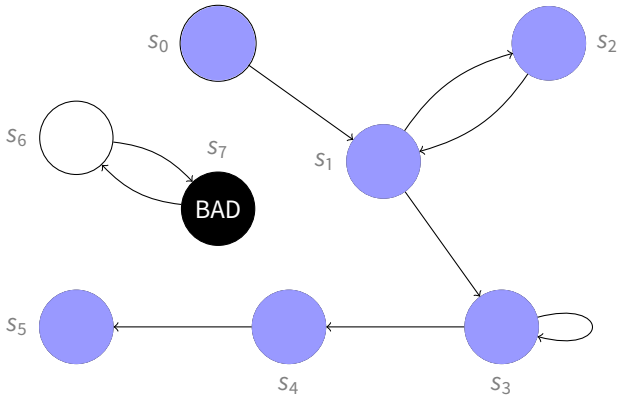
Reachability in $\leq n$ steps

Number of steps: 3



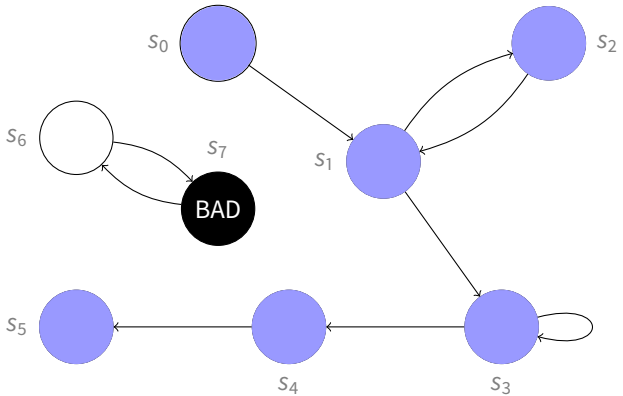
Reachability in $\leq n$ steps

Number of steps: 4



Reachability in $\leq n$ steps

Number of steps: 5



Full set of reachable states has been determined

Termination

Let S_n the set of states reachable in $\leq n$ steps

Key properties for termination:

1. $S_i \subseteq S_{i+1}$ for all i
2. the state space is finite

Consequences:

- there is k such that $S_k = S_{k+1}$
- for such k we have $R_{\leq k}(x) = R_{\leq k+1}(x)$

Termination

Let S_n the set of states reachable in $\leq n$ steps

Key properties for termination:

1. $S_i \subseteq S_{i+1}$ for all i
2. the state space is finite

Consequences:

- there is k such that $S_k = S_{k+1}$
- for such k we have $R_{\leq k}(\boldsymbol{x}) \equiv R_{\leq k+1}(\boldsymbol{x})$

Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) \;:=\; I(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge F(\boldsymbol{x})$ is satisfiable **then return** "yes"
  $R'(\boldsymbol{x}) \;:=\; R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) \;:=\; R'(\boldsymbol{x})$
 end loop
end

Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) \;:=\; I(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge F(\boldsymbol{x})$ is satisfiable **then return** "yes"
  $R'(\boldsymbol{x}) \;:=\; R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) \;:=\; R'(\boldsymbol{x})$
 end loop
end

Implementation?

Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) \;:=\; I(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge F(\boldsymbol{x})$ is satisfiable **then return** "yes"
  $R'(\boldsymbol{x}) \;:=\; R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) \;:=\; R'(\boldsymbol{x})$
 end loop
end

Conjunction and disjunction

Implementation?

Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) \;:=\; I(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge F(\boldsymbol{x})$ is satisfiable **then return** "yes"
  $R'(\boldsymbol{x}) \;:=\; R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) \;:=\; R'(\boldsymbol{x})$
 end loop
end

Implementation?

Conjunction and disjunction
Quantification

Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) \;:=\; I(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge F(\boldsymbol{x})$ is satisfiable **then return** "yes"
  $R'(\boldsymbol{x}) \;:=\; R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) \;:=\; R'(\boldsymbol{x})$
 end loop
end

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking

Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
$R(\boldsymbol{x}) := I(\boldsymbol{x})$
loop
if $R(\boldsymbol{x}) \wedge F(\boldsymbol{x})$ is satisfiable **then return** "yes"
$R'(\boldsymbol{x}) := R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$
if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
$R(\boldsymbol{x}) := R'(\boldsymbol{x})$
end loop
end

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

Forward Reachability Algorithm

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) \;:=\; I(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge F(\boldsymbol{x})$ is satisfiable **then return** "yes"
  $R'(\boldsymbol{x}) \;:=\; R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) \;:=\; R'(\boldsymbol{x})$
 end loop
end

Implementation?
Use OBDDs and OBDD algorithms

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

Main Issues with Forward Reachability Algorithms

Forward reachability behaves in the same way, independently of the set of final states

In other words, they are not goal oriented

Backward Reachability

Idea:

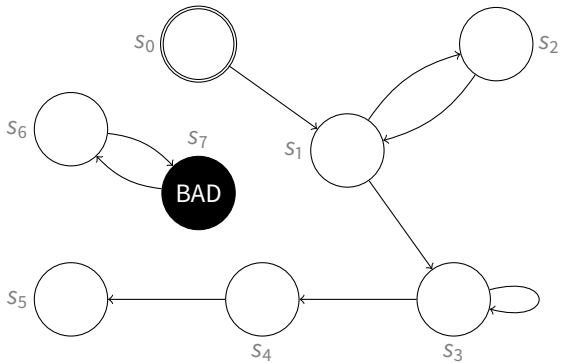
- instead of going forward in the state transition graph, go backward
- swap initial and final states and invert the transition relation

Backward Reachability in $\leq n$ steps

Idea:

- instead of going forward in the state transition graph, go backward
- swap initial and final states and invert the transition relation

Number of backward steps: 0

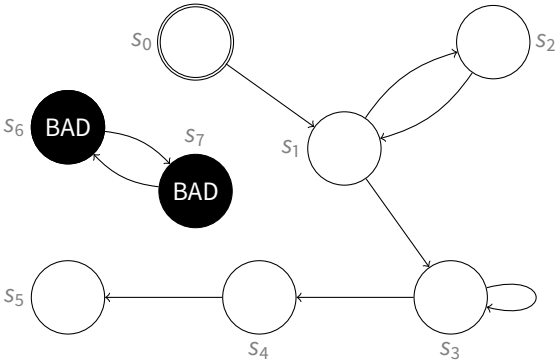


Backward Reachability in $\leq n$ steps

Idea:

- instead of going forward in the state transition graph, go backward
- swap initial and final states and invert the transition relation

Number of backward steps: 1

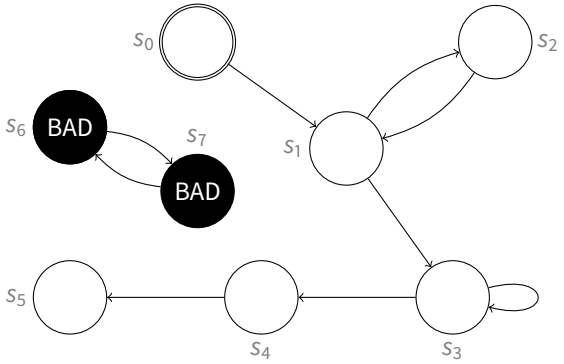


Backward Reachability in $\leq n$ steps

Idea:

- instead of going forward in the state transition graph, go backward
- swap initial and final states and invert the transition relation

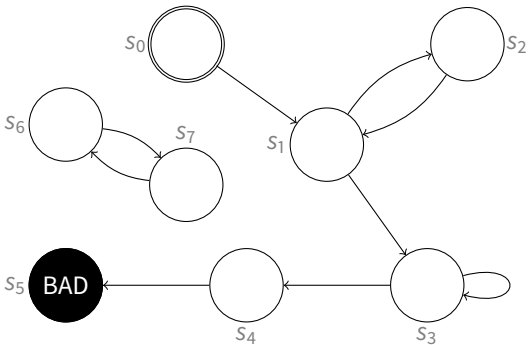
Number of backward steps: 1



Bad states unreachable!

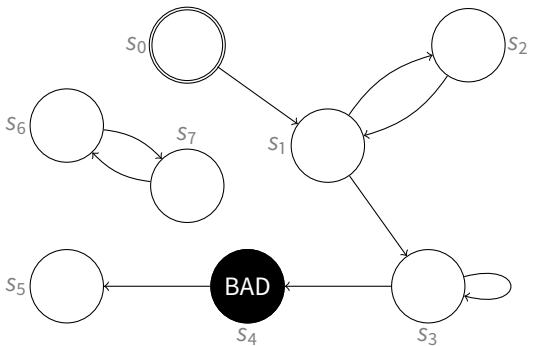
Backward Reachability in n steps

Number of backward steps: 0



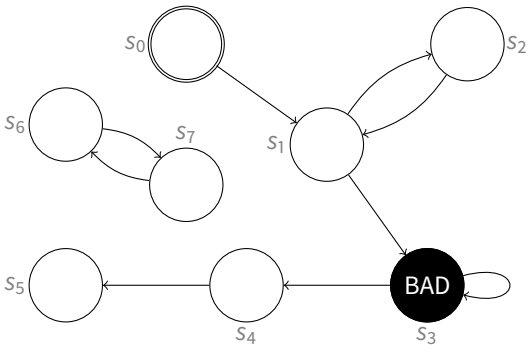
Backward Reachability in n steps

Number of backward steps: 1



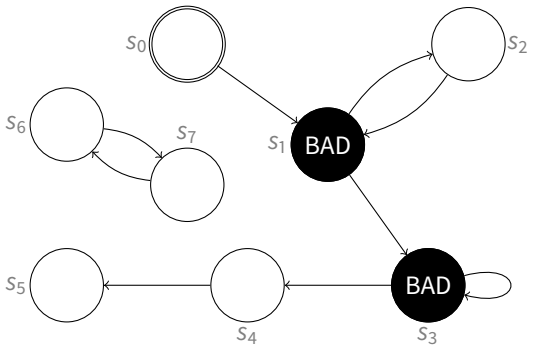
Backward Reachability in n steps

Number of backward steps: 2



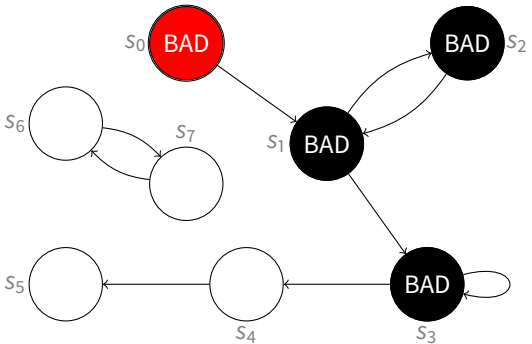
Backward Reachability in n steps

Number of backward steps: 3



Backward Reachability in n steps

Number of backward steps: 4



Bad states reachable!

Backward Reachability

S_0 is *backward reachable from F in n steps* if F is reachable from S_0 in n steps

Backward Reachability

S_0 is *backward reachable from* F *in* n *steps* if F is reachable from S_0 in n steps

Lemma 3

Let $C(\boldsymbol{x})$ *symbolically represent a set of states* S_C*. The formula*

$$BR(\boldsymbol{x}) \quad \stackrel{\text{def}}{=} \quad \exists \boldsymbol{z}(T(\boldsymbol{x}, \boldsymbol{z}) \wedge C(\boldsymbol{z}))$$

denotes the set of states backward reachable from S_C *in one step.*

Backward Reachability Algorithm

Same as the forward reachability algorithms, but

- swap I with F
- use the inverse of the transition relation T

Backward Reachability Algorithm

Same as the forward reachability algorithms, but

- swap I with F
- use the inverse of the transition relation T

procedure $BReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) := F(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge I(\boldsymbol{x})$ is satisfiable **then**
   return "yes"
  $R'(\boldsymbol{x}) := R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(T(\boldsymbol{x}, \boldsymbol{z}) \wedge R(\boldsymbol{z}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) := R'(\boldsymbol{x})$
 end loop
end

Backward Reachability Algorithm

Same as the forward reachability algorithms, but

- swap I with F
- use the inverse of the transition relation T

procedure $BReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) := F(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge I(\boldsymbol{x})$ is satisfiable **then**
   return "yes"
  $R'(\boldsymbol{x}) := R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(T(\boldsymbol{x}, \boldsymbol{z}) \wedge R(\boldsymbol{z}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) := R'(\boldsymbol{x})$
 end loop
end

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) := I(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge F(\boldsymbol{x})$ is satisfiable **then**
   return "yes"
  $R'(\boldsymbol{x}) := R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) := R'(\boldsymbol{x})$
 end loop
end

Backward Reachability Algorithm

Same as the forward reachability algorithms, but

- swap I with F
- use the inverse of the transition relation T

procedure $BReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) := F(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge I(\boldsymbol{x})$ is satisfiable **then**
   return "yes"
  $R'(\boldsymbol{x}) := R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(T(\boldsymbol{x}, \boldsymbol{z}) \wedge R(\boldsymbol{z}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) := R'(\boldsymbol{x})$
 end loop
end

procedure $FReach(I, T, F)$
input: formulas I, T, F
output: "yes" or "no"
begin
 $R(\boldsymbol{x}) := I(\boldsymbol{x})$
 loop
  if $R(\boldsymbol{x}) \wedge F(\boldsymbol{x})$ is satisfiable **then**
   return "yes"
  $R'(\boldsymbol{x}) := R(\boldsymbol{x}) \vee \exists \boldsymbol{z}(R(\boldsymbol{z}) \wedge T(\boldsymbol{z}, \boldsymbol{x}))$
  if $R(\boldsymbol{x}) \equiv R'(\boldsymbol{x})$ **then return** "no"
  $R(\boldsymbol{x}) := R'(\boldsymbol{x})$
 end loop
end

Extensions of Model Checking

- There are model-checking algorithms for properties other than reachability
- there is a general model-checking algorithm for arbitrary LTL properties
- there are extensions of model-checking techniques for infinite-state systems

Extensions of Model Checking

- There are model-checking algorithms for properties other than reachability
- there is a general model-checking algorithm for arbitrary LTL properties
- there are extensions of model-checking techniques for infinite-state systems

Extensions of Model Checking

- There are model-checking algorithms for properties other than reachability
- there is a general model-checking algorithm for arbitrary LTL properties
- there are extensions of model-checking techniques for infinite-state systems

Extensions of Model Checking

- There are model-checking algorithms for properties other than reachability
- there is a general model-checking algorithm for arbitrary LTL properties
- there are extensions of model-checking techniques for infinite-state systems
- they will not be considered in this course