# Symbolic Model checking:

## Reachability

## One-sided Forward Reachability Algorithm

# Some useful LTL properties

- Reachability and safety properties.
  Let unsafe describe states which are unsafe.
  Then $\Box \neg \mathit{unsafe}$ express a safety requirement.
  Ex: $\Box \neg(\mathit{disp} = \mathit{beer} \wedge \mathit{customer} = \mathit{prof})$
- Mutual exclusion. Two processes are not in the critical section.
  Ex: $\Box \neg(\mathit{critical}_1 \wedge \mathit{critical}_2)$
- Fairness. Ex: $\Box(\mathit{customer} = \mathit{student} \rightarrow \Diamond \mathit{customer} = \mathit{prof})$
- Responsiveness: every request will be eventually acknowledged
  Ex: $\Box(\mathit{request} \rightarrow (\mathit{request}\ \mathcal{U}\ \mathit{ack}))$

# Some useful LTL properties

- Reachability and safety properties.
  Let unsafe describe states which are unsafe.
  Then $\Box\neg\text{unsafe}$ express a safety requirement.
  Ex: $\Box\neg(\text{disp} = \text{beer} \wedge \text{customer} = \text{prof})$
- Mutual exclusion. Two processes are not in the critical section.
  Ex: $\Box\neg(\text{critical}_1 \wedge \text{critical}_2)$
- Fairness. Ex: $\Box(\text{customer} = \text{student} \rightarrow \Diamond\text{customer} = \text{prof})$
- Responsiveness: every request will be eventually acknowledged
  Ex: $\Box(\text{request} \rightarrow (\text{request}\, \mathcal{U}\, \text{ack}))$

# Some useful LTL properties

- Reachability and safety properties.
  Let unsafe describe states which are unsafe.
  Then $\Box\neg\text{unsafe}$ express a safety requirement.
  Ex: $\Box\neg(\text{disp} = \text{beer} \wedge \text{customer} = \text{prof})$
- Mutual exclusion. Two processes are not in the critical section.
  Ex: $\Box\neg(\text{critical}_1 \wedge \text{critical}_2)$
- Fairness. Ex: $\Box(\text{customer} = \text{student} \rightarrow \Diamond\text{customer} = \text{prof})$
- Responsiveness: every request will be eventually acknowledged
  Ex: $\Box(\text{request} \rightarrow (\text{request}\,\mathcal{U}\,\text{ack}))$

# Some useful LTL properties

- Reachability and safety properties.
  Let unsafe describe states which are unsafe.
  Then $\Box\neg\text{unsafe}$ express a safety requirement.
  Ex: $\Box\neg(\text{disp} = \text{beer} \wedge \text{customer} = \text{prof})$
- Mutual exclusion. Two processes are not in the critical section.
  Ex: $\Box\neg(\text{critical}_1 \wedge \text{critical}_2)$
- Fairness. Ex: $\Box(\text{customer} = \text{student} \rightarrow \Diamond\text{customer} = \text{prof})$
- Responsiveness: every request will be eventually acknowledged
  Ex: $\Box(\text{request} \rightarrow (\text{request}\,\mathcal{U}\,\text{ack}))$

# Putting it All Together

When we design a system, we would like to be sure that it will satisfy all requirements, such as safety.

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

- formally represent our system as a transition system:
  - explicit representation
  - symbolic representation
- express the desired properties of the system in temporal logic.

What is next?

# Putting it All Together

When we design a system, we would like to be sure that it will satisfy all requirements, such as safety.

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

- formally represent our system as a transition system:
  - explicit representation
  - symbolic representation
- express the desired properties of the system in temporal logic.

What is next?

# Putting it All Together

When we design a system, we would like to be sure that it will satisfy all requirements, such as safety.

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

- formally represent our system as a transition system:
  - explicit representation
  - symbolic representation
- express the desired properties of the system in temporal logic.

What is next?

# The Symbolic Model Checking Problem

Symbolic Model Checking problem:

Given

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

check if every (some) computation of the system satisfies this formula, preferably in a fully automatic way.

# Reachability and Safety Properties

A reachability property is expressed by a formula

$$\Diamond F,$$

where $F$ is a propositional formula.

A safety property is expressed by a formula

$$\Box F,$$

where $F$ is a propositional formula.

Reachability and safety properties are the most common problems arising in model checking. They are dual to each other: if we can check one of them, we can check the other one too:

- $\Box F \equiv \neg\Diamond\neg F$;
- $\Diamond F \equiv \neg\Box\neg F$.

We cannot reach an unsafe state if and only if all states we can visit are safe.

# Reachability and Safety Properties

A reachability property is expressed by a formula

$$\Diamond F,$$

where $F$ is a propositional formula.

A safety property is expressed by a formula

$$\Box F,$$

where $F$ is a propositional formula.

Reachability and safety properties are the most common problems arising in model checking. They are dual to each other: if we can check one of them, we can check the other one too:

- $\Box F \equiv \neg\Diamond\neg F$;
- $\Diamond F \equiv \neg\Box\neg F$.

We cannot reach an unsafe state if and only if all states we can visit are safe.

# Reachability and Safety Properties

A reachability property is expressed by a formula

$$\Diamond F,$$

where $F$ is a propositional formula.

A safety property is expressed by a formula

$$\Box F,$$

where $F$ is a propositional formula.

Reachability and safety properties are the most common problems arising in model checking. They are dual to each other: if we can check one of them, we can check the other one too:

- $\Box F \equiv \neg\Diamond\neg F$;
- $\Diamond F \equiv \neg\Box\neg F$.

We cannot reach an unsafe state if and only if all states we can visit are safe.

# Reachability and Safety Properties

A reachability property is expressed by a formula

$$\Diamond F,$$

where $F$ is a propositional formula.

A safety property is expressed by a formula

$$\Box F,$$

where $F$ is a propositional formula.

Reachability and safety properties are the most common problems arising in model checking. They are dual to each other: if we can check one of them, we can check the other one too:

- $\Box F \equiv \neg\Diamond\neg F$;
- $\Diamond F \equiv \neg\Box\neg F$.

We cannot reach an unsafe state if and only if all states we can visit are safe.

# Reachability

Fix a transition system $\mathbb{S}$ with the transition relation $T$. We write $s_0 \to s_1$ for $(s_0, s_1) \in T$ (that is, if there is a transition from $s_0$ to $s_1$).

- A state $s$ is reachable in $n$ steps from a state $s_0$ if there exists a sequence of states $s_1, \ldots, s_n$ such that $s_n = s$ and

$$s_0 \to s_1 \to \ldots \to s_n.$$

- A state $s$ is reachable from a state $s_0$ if $s$ is reachable from $s_0$ in $n \geq 0$ steps.

# Reachability

Fix a transition system $\mathbb{S}$ with the transition relation $T$. We write $s_0 \to s_1$ for $(s_0, s_1) \in T$ (that is, if there is a transition from $s_0$ to $s_1$).

- A state $s$ is reachable in $n$ steps from a state $s_0$ if there exists a sequence of states $s_1, \ldots, s_n$ such that $s_n = s$ and

$$s_0 \to s_1 \to \ldots \to s_n.$$

- A state $s$ is reachable from a state $s_0$ if $s$ is reachable from $s_0$ in $n \geq 0$ steps.

# Reachability

Fix a transition system $\mathbb{S}$ with the transition relation $T$. We write $s_0 \to s_1$ for $(s_0, s_1) \in T$ (that is, if there is a transition from $s_0$ to $s_1$).

- A state $s$ is reachable in $n$ steps from a state $s_0$ if there exists a sequence of states $s_1, \ldots, s_n$ such that $s_n = s$ and

$$s_0 \to s_1 \to \ldots \to s_n.$$

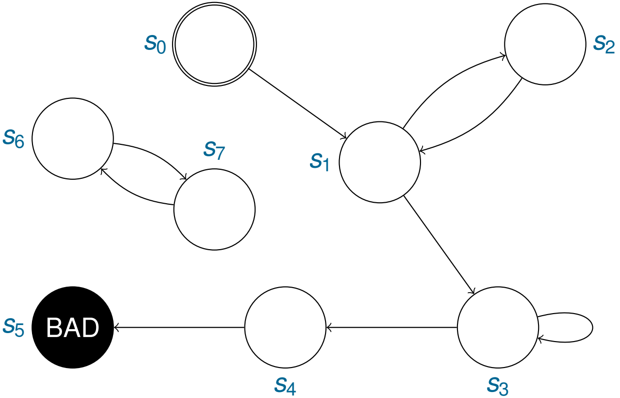
- A state $s$ is reachable from a state $s_0$ if $s$ is reachable from $s_0$ in $n \geq 0$ steps.

# Reachability Properties and Graph Reachability

**Theorem.** Let $F$ be a propositional formula. The formula $\Diamond F$ holds on some computation path if and only if there exists an initial state $s_0$ and a state $s$ such that $s \models F$ and $s$ is reachable from $s_0$.

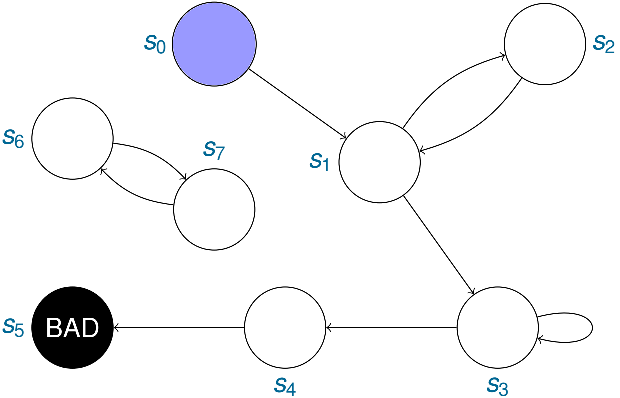
# Reachability in $n$ steps

Number of steps:



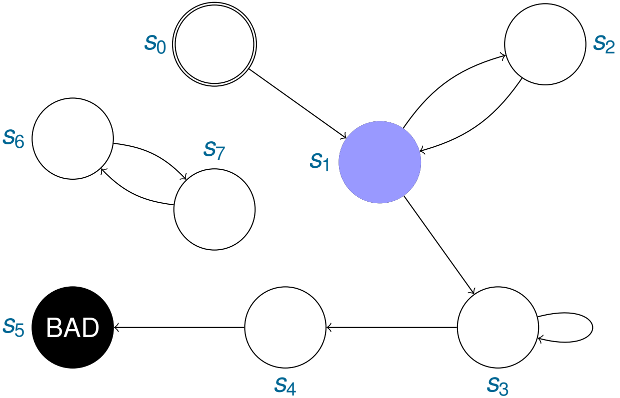
# Reachability in $n$ steps

Number of steps: 0



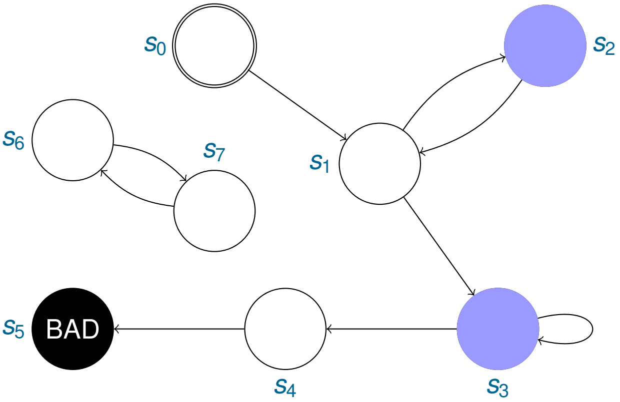
# Reachability in $n$ steps

Number of steps: 1



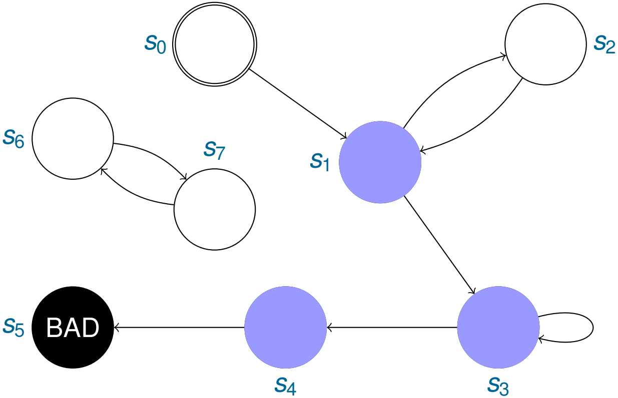
# Reachability in $n$ steps

Number of steps: 2



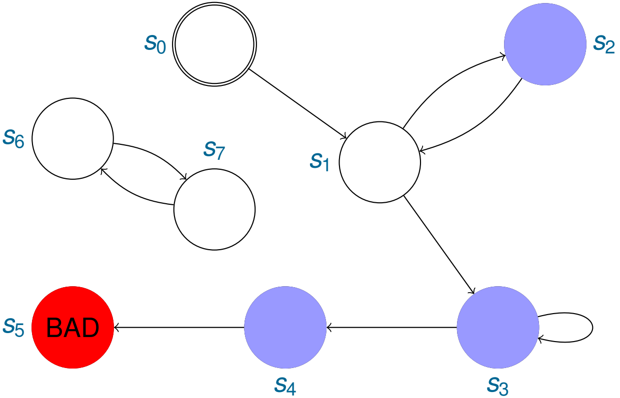
# Reachability in $n$ steps

Number of steps: 3



# Reachability in $n$ steps

Number of steps: 4



# Reformulation of Reachability

Given

1. Initial condition $I$ representing a set of initial states;
2. Final condition $F$ representing a set of final states;
3. formula $Tr$ representing the transition relation of a transition system $\mathbb{S}$,

is any final state reachable from an initial state in $\mathbb{S}$?

# Symbolic Reachability Checking

- Idea: build a symbolic representation of the set of reachable states.
- Two main kinds of algorithm:
  - forward reachability;
  - backward reachability.

# Symbolic Reachability Checking

- Idea: build a symbolic representation of the set of reachable states.
- Two main kinds of algorithm:
  - forward reachability;
  - backward reachability.

# Reformulation as a Decision Problem

Given

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

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

1. $s_0 \models I(\bar{x})$;
2. $s_n \models F(\bar{x})$;
3. For all $i = 1, \ldots, n$ we have $(s_{i-1}, s_i) \models T(\bar{x}, \bar{x}')$.

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

# Idea of Reachability-Checking Algorithms

If a final state is reachable from an initial state, then it is reachable from an initial state in some number $n$ of steps.

For a given number $n$, find a symbolic representation of the set of states reachable from from an initial state in $n$ steps. If this formula is not satisfied in a final state, increase $n$ and start again.

# Idea of Reachability-Checking Algorithms

If a final state is reachable from an initial state, then it is reachable from an initial state in some number $n$ of steps.

For a given number $n$, find a symbolic representation of the set of states reachable from from an initial state in $n$ steps. If this formula is not satisfied in a final state, increase $n$ and start again.

# Simple Logical Analysis

## Lemma

*Let $C(\bar{x})$ symbolically represent a set of states $S$. Define*

$$FR(\bar{x}) \stackrel{\text{def}}{=} \exists \bar{x}_1 (C(\bar{x}_1) \wedge T(\bar{x}_1, \bar{x})).$$

*Then $FR(\bar{x})$ represents the set of states reachable from $S$ in one step.*

Define a sequence of formulas $R_n$ for reachability in $n$ states:

$$\begin{array}{rcl} R_0(\bar{x}) & \stackrel{\text{def}}{=} & I(\bar{x}) \\ & \dots & \\ R_n(\bar{x}) & \stackrel{\text{def}}{=} & \exists \bar{x}_{n-1} (R_{n-1}(\bar{x}_{n-1}) \wedge T(\bar{x}_{n-1}, \bar{x})) \end{array}$$

Rearranging using prenexing rules:

$$\begin{array}{rcl} R_0(\bar{x}) & = & I(\bar{x}) \\ R_1(\bar{x}) & = & \exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x})) \\ R_2(\bar{x}) & = & \exists \bar{x}_1 (\exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1)) \wedge T(\bar{x}_1, \bar{x})) = \\ & & \exists \bar{x}_1 \exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x})) \\ & \dots & \\ R_n(\bar{x}) & = & \exists \bar{x}_{n-1} \dots \exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge \dots \wedge T(\bar{x}_{n-1}, \bar{x})) \end{array}$$

# Simple Logical Analysis

## Lemma

*Let $C(\bar{x})$ symbolically represent a set of states $S$. Define*

$$FR(\bar{x}) \stackrel{\text{def}}{=} \exists \bar{x}_1 (C(\bar{x}_1) \wedge T(\bar{x}_1, \bar{x})).$$

*Then $FR(\bar{x})$ represents the set of states reachable from $S$ in one step.*

Define a sequence of formulas $R_n$ for reachability in $n$ states:

$$\begin{array}{rcl} R_0(\bar{x}) & \stackrel{\text{def}}{=} & I(\bar{x}) \\ & \ldots & \\ R_n(\bar{x}) & \stackrel{\text{def}}{=} & \exists \bar{x}_{n-1} (R_{n-1}(\bar{x}_{n-1}) \wedge T(\bar{x}_{n-1}, \bar{x})) \end{array}$$

Rearranging using prenexing rules:

$$\begin{array}{rcl} R_0(\bar{x}) & = & I(\bar{x}) \\ R_1(\bar{x}) & = & \exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x})) \\ R_2(\bar{x}) & = & \exists \bar{x}_1 (\exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1)) \wedge T(\bar{x}_1, \bar{x})) = \\ & & \exists \bar{x}_1 \exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x})) \\ & \ldots & \\ R_n(\bar{x}) & = & \exists \bar{x}_{n-1} \ldots \exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge \ldots \wedge T(\bar{x}_{n-1}, \bar{x})) \end{array}$$

# Simple Logical Analysis

## Lemma

*Let $C(\bar{x})$ symbolically represent a set of states $S$. Define*

$$FR(\bar{x}) \stackrel{\text{def}}{=} \exists \bar{x}_1 (C(\bar{x}_1) \wedge T(\bar{x}_1, \bar{x})).$$

*Then $FR(\bar{x})$ represents the set of states reachable from $S$ in one step.*

Define a sequence of formulas $R_n$ for reachability in $n$ states:

$$\begin{array}{rcl} R_0(\bar{x}) & \stackrel{\text{def}}{=} & I(\bar{x}) \\ & \dots & \\ R_n(\bar{x}) & \stackrel{\text{def}}{=} & \exists \bar{x}_{n-1} (R_{n-1}(\bar{x}_{n-1}) \wedge T(\bar{x}_{n-1}, \bar{x})) \end{array}$$

Rearranging using prenexing rules:

$$\begin{array}{rcl} R_0(\bar{x}) & = & I(\bar{x}) \\ R_1(\bar{x}) & = & \exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x})) \\ R_2(\bar{x}) & = & \exists \bar{x}_1 (\exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1)) \wedge T(\bar{x}_1, \bar{x})) = \\ & & \exists \bar{x}_1 \exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x})) \\ & \dots & \\ R_n(\bar{x}) & = & \exists \bar{x}_{n-1} \dots \exists \bar{x}_0 (I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge \dots \wedge T(\bar{x}_{n-1}, \bar{x})) \end{array}$$

# One-sided Forward Reachability Algorithm

Also called: Bounded Model Checking (BMC).

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

Implementation? Use propositional SAT solvers.

# One-sided Forward Reachability Algorithm

Also called: Bounded Model Checking (BMC).

**procedure** $FReach(I, T, F)$
**input**: formulas $I, T, F$
**output**: "yes" or no output
**begin**

```
  i   :=  0 ;
  R   :=  I(x̄_0) ;
  loop
    if R ∧ F(x̄_i) is satisfiable then return "yes" ;
    R   :=  R ∧ T(x̄_i, x̄_{i+1}) ;
    i   :=  i + 1 ;
  end loop
end
```

Lemma: (symbolic unrolling)

$$I(\bar{x}_0) \land T(\bar{x}_0, \bar{x}_1) \land T(\bar{x}_1, \bar{x}_2) \land \ldots \land T(\bar{x}_{i-1}, \bar{x}_i) \land F(\bar{x}_i)$$

is satisfiable if and only if there is a state $s$ reachable in $i$ steps, such that $s \models F$.

Implementation? Use propositional SAT solvers.

# One-sided Forward Reachability Algorithm

Also called: Bounded Model Checking (BMC).

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

Lemma: (symbolic unrolling)

$$I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x}_2) \wedge \ldots \wedge T(\bar{x}_{i-1}, \bar{x}_i) \wedge F(\bar{x}_i)$$

is satisfiable if and only if there is a state $s$ reachable in $i$ steps, such that $s \models F$.

Implementation? Use propositional SAT solvers.

# One-sided Forward Reachability Algorithm

Also called: Bounded Model Checking (BMC).

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

Lemma: (symbolic unrolling)

$$I(\bar{x}_0) \wedge T(\bar{x}_0, \bar{x}_1) \wedge T(\bar{x}_1, \bar{x}_2) \wedge \ldots \wedge T(\bar{x}_{i-1}, \bar{x}_i) \wedge F(\bar{x}_i)$$

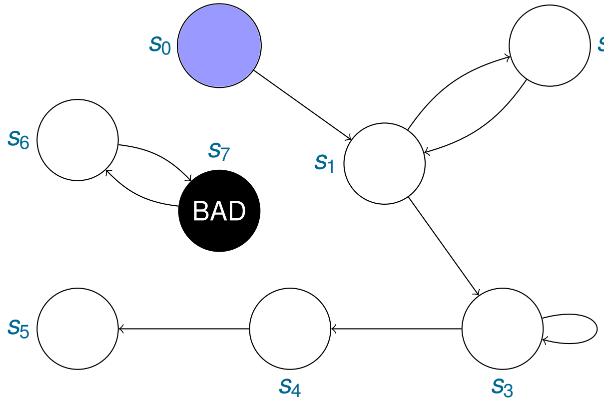
is satisfiable if and only if there is a state $s$ reachable in $i$ steps, such that $s \models F$.

Implementation? Use propositional SAT solvers.

# Complete Forward Reachability Algorithm

# Termination ?

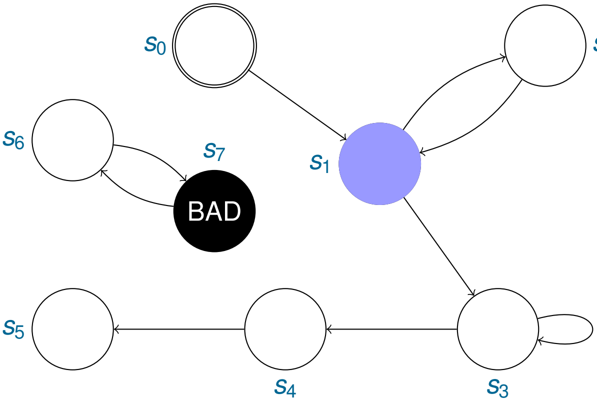
Number of steps: 0




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

# Termination ?

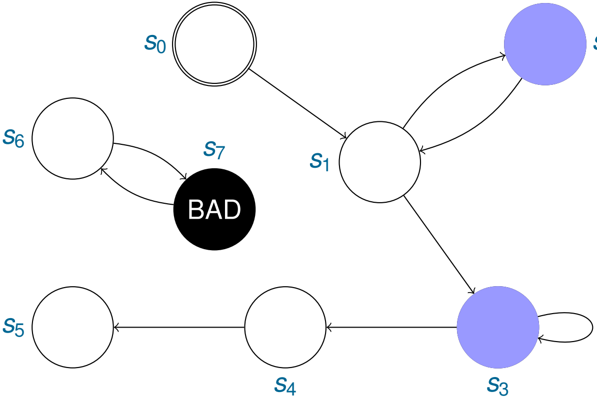
Number of steps: 1




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

# Termination ?

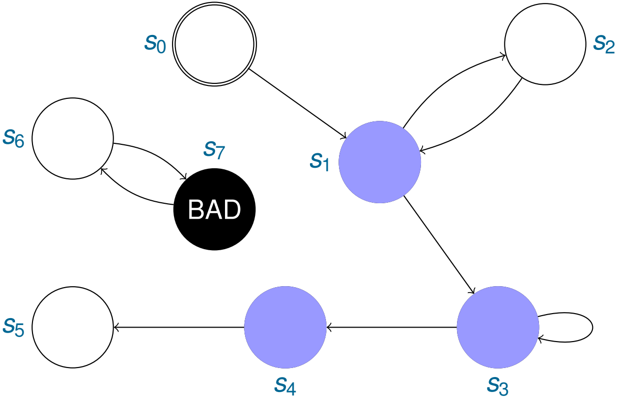
Number of steps: 2




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

# Termination ?

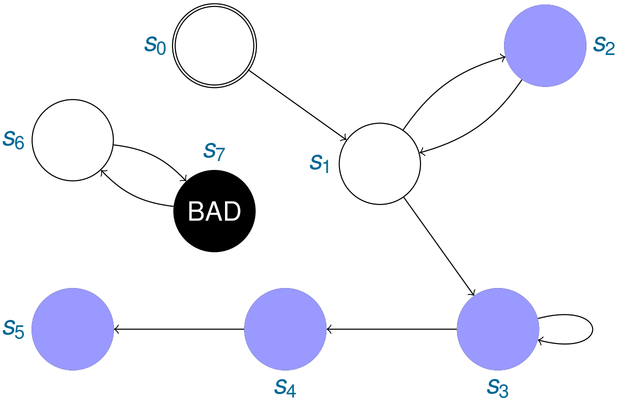
Number of steps: 3




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

# Termination ?

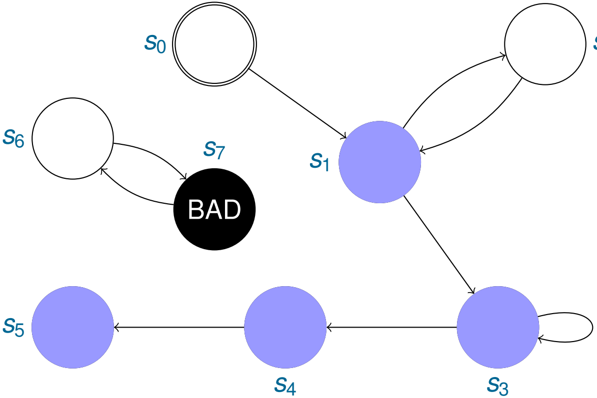
Number of steps: 4




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

# Termination ?

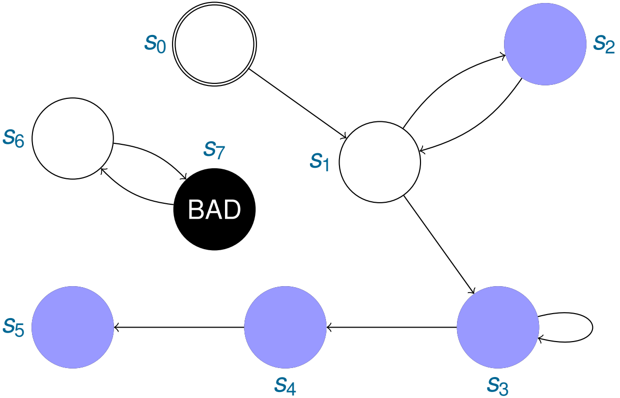
Number of steps: 5




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

# Termination ?

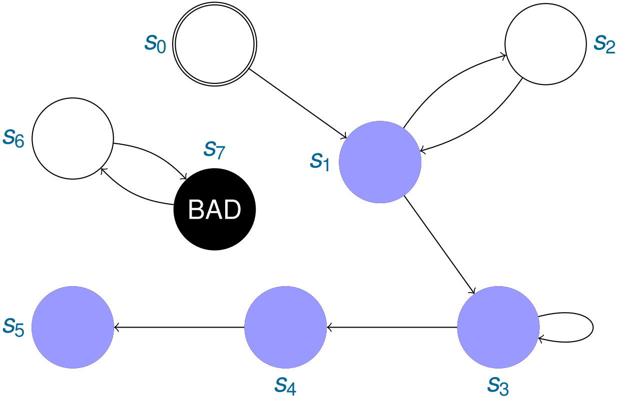
Number of steps: 6




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

# Termination ?

Number of steps: 7




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

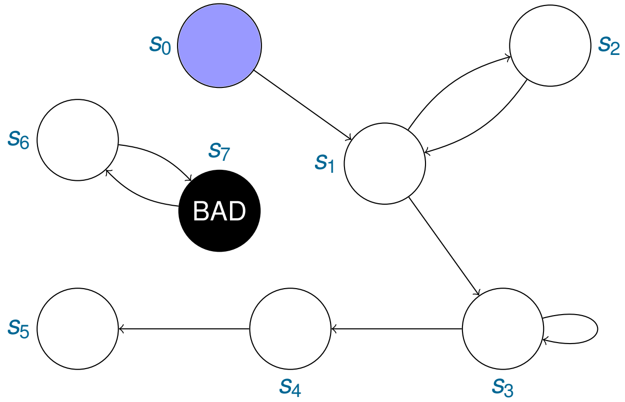
# Reachability in $\leq n$ steps

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

$$
\begin{array}{rcl}
R_{\leq 0}(\bar{x}) & \stackrel{\text{def}}{=} & I(\bar{x}) \\
 & \ldots & \\
R_{\leq n}(\bar{x}) & \stackrel{\text{def}}{=} & R_{\leq n-1}(\bar{x}) \vee \exists \bar{x}_{n-1}(R_{\leq n-1}(\bar{x}_{n-1}) \wedge T(\bar{x}_{n-1}, \bar{x}))
\end{array}
$$

# Reachability in $\leq n$ steps

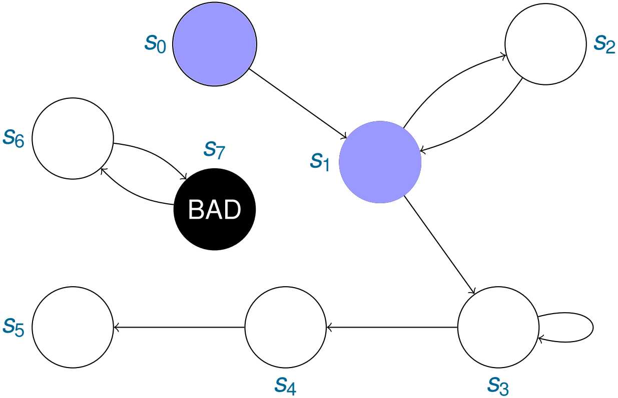
Number of steps: 0




The set of states will change no more.

# Reachability in $\leq n$ steps

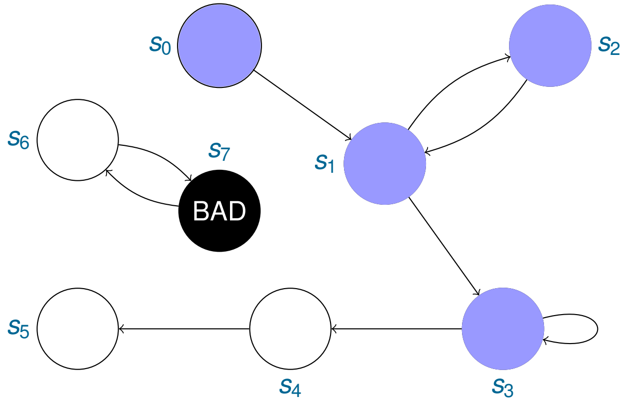
Number of steps: 1




The set of states will change no more.

# Reachability in $\leq n$ steps

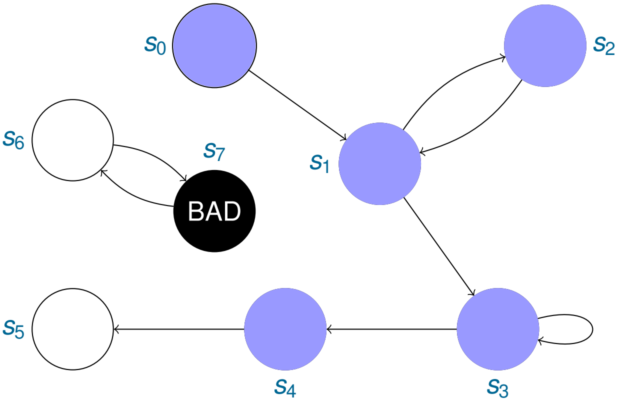
Number of steps: 2




The set of states will change no more.

# Reachability in $\leq n$ steps

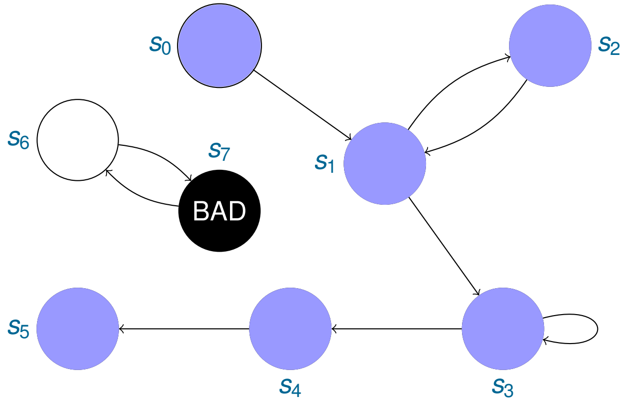
Number of steps: 3




The set of states will change no more.

# Reachability in $\leq n$ steps

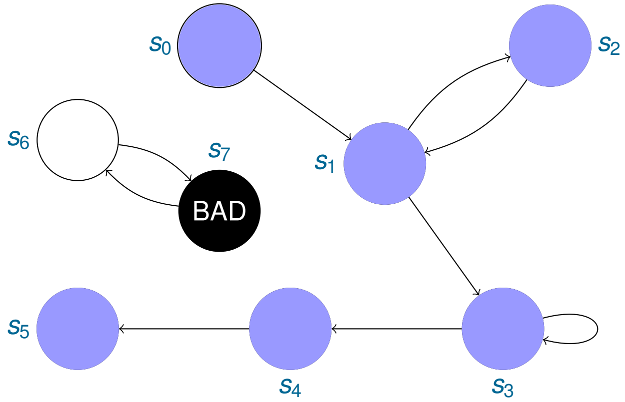
Number of steps: 4




The set of states will change no more.

# Reachability in $\leq n$ steps

Number of steps: 5




The set of states will change no more.

# Termination

Denote by $S_n$ the set of states reachable from an initial state in $\leq n$ steps.
Key properties for termination.

- $S_i \subseteq S_{i+1}$ for all $i$;
- the system has a finite number of states;
- therefore, there exists a number $k$ such that $S_k = S_{k+1}$;
- for such $k$ we have $R_{\leq k}(\bar{x}) \equiv R_{\leq k+1}(\bar{x})$.

# Complete Forward Reachability Algorithm

**procedure** $CFReach(I, T, F)$
**input**: formulas $I, T, F$
**output**: “yes” or “no”
**begin**
  $i \;:=\; 0$ ;
  $R_{\leq 0}(\bar{x}) \;:=\; I(\bar{x})$ ;
  **loop**
    **if** $R_{\leq i}(\bar{x}) \wedge F(\bar{x})$ is satisfiable **then** **return** “yes” ;

  **end** **loop**
**end**

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Complete Forward Reachability Algorithm

**procedure** $CFReach(I, T, F)$
**input**: formulas $I, T, F$
**output**: "yes" or "no"
**begin**
 $i \ := \ 0$ ;
 $R_{\leq 0}(\bar{x}) \ := \ I(\bar{x})$ ;
 **loop**
  **if** $R_{\leq i}(\bar{x}) \land F(\bar{x})$ is satisfiable **then** **return** "yes" ;
  $R_{\leq i+1}(\bar{x}) \ := \ R_{\leq i}(\bar{x}) \lor \exists \bar{x}_i (R_{\leq i}(\bar{x}_i) \land T(\bar{x}_i, \bar{x}))$ ;

 **end** **loop**
**end**

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Complete Forward Reachability Algorithm

**procedure** $CFReach(I, T, F)$
**input**: formulas $I, T, F$
**output**: “yes” or “no”
**begin**
$i \ := \ 0$ ;
$R_{\leq 0}(\bar{x}) \ := \ I(\bar{x})$ ;
**loop**
**if** $R_{\leq i}(\bar{x}) \wedge F(\bar{x})$ is satisfiable **then** **return** “yes” ;
$R_{\leq i+1}(\bar{x}) \ := \ R_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (R_{\leq i}(\bar{x}_i) \wedge T(\bar{x}_i, \bar{x}))$ ;
**if** $R_{\leq i}(\bar{x}) \equiv R_{\leq i+1}(\bar{x})$ **then** **return** “no” ;

**end** **loop**
**end**

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Complete Forward Reachability Algorithm

**procedure** $CFReach(I, T, F)$
**input**: formulas $I, T, F$
**output**: "yes" or "no"
**begin**
  $i := 0$ ;
  $R_{\leq 0}(\bar{x}) := I(\bar{x})$ ;
  **loop**
    **if** $R_{\leq i}(\bar{x}) \wedge F(\bar{x})$ is satisfiable **then** **return** "yes" ;
    $R_{\leq i+1}(\bar{x}) := R_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (R_{\leq i}(\bar{x}_i) \wedge T(\bar{x}_i, \bar{x}))$ ;
    **if** $R_{\leq i}(\bar{x}) \equiv R_{\leq i+1}(\bar{x})$ **then** **return** "no" ;
    $i := i + 1$ ;
  **end** **loop**
**end**

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Complete Forward Reachability Algorithm

**procedure** $CFReach(I, T, F)$
**input**: formulas $I, T, F$
**output**: "yes" or "no"
**begin**
  $i \ := \ 0$ ;
  $R_{\leq 0}(\bar{x}) \ := \ I(\bar{x})$ ;
  **loop**
    **if** $R_{\leq i}(\bar{x}) \wedge F(\bar{x})$ is satisfiable **then** **return** "yes" ;
    $R_{\leq i+1}(\bar{x}) \ := \ R_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (R_{\leq i}(\bar{x}_i) \wedge T(\bar{x}_i, \bar{x}))$ ;
    **if** $R_{\leq i}(\bar{x}) \equiv R_{\leq i+1}(\bar{x})$ **then** **return** "no" ;
    $i \ := \ i + 1$ ;
  **end** **loop**
**end**

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Complete Forward Reachability Algorithm

**procedure** $CFReach(I, T, F)$
**input**: formulas $I, T, F$
**output**: “yes” or “no”
**begin**
$i := 0$ ;
$R_{\leq 0}(\bar{x}) := I(\bar{x})$ ;
**loop**
**if** $R_{\leq i}(\bar{x}) \wedge F(\bar{x})$ is satisfiable **then** **return** “yes” ;
$R_{\leq i+1}(\bar{x}) := R_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (R_{\leq i}(\bar{x}_i) \wedge T(\bar{x}_i, \bar{x}))$ ;
**if** $R_{\leq i}(\bar{x}) \equiv R_{\leq i+1}(\bar{x})$ **then** **return** “no” ;
$i := i + 1$ ;
**end** **loop**
**end**

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Complete Forward Reachability Algorithm

**procedure** *CFReach*$(I, T, F)$
**input**: formulas $I, T, F$
**output**: “yes” or “no”
**begin**
  $i \ := \ 0$ ;
  $R_{\leq 0}(\bar{x}) \ := \ I(\bar{x})$ ;
  **loop**
    **if** $R_{\leq i}(\bar{x}) \wedge F(\bar{x})$ is satisfiable **then** **return** “yes” ;
    $R_{\leq i+1}(\bar{x}) \ := \ R_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (R_{\leq i}(\bar{x}_i) \wedge T(\bar{x}_i, \bar{x}))$ ;
    **if** $R_{\leq i}(\bar{x}) \equiv R_{\leq i+1}(\bar{x})$ **then** **return** “no” ;
    $i \ := \ i + 1$ ;
  **end** **loop**
**end**

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Complete Forward Reachability Algorithm

**procedure** $CFReach(I, T, F)$
**input**: formulas $I, T, F$
**output**: "yes" or "no"
**begin**
$i := 0$ ;
$R_{\leq 0}(\bar{x}) := I(\bar{x})$ ;
**loop**
**if** $R_{\leq i}(\bar{x}) \wedge F(\bar{x})$ is satisfiable **then** **return** "yes" ;
$R_{\leq i+1}(\bar{x}) := R_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (R_{\leq i}(\bar{x}_i) \wedge T(\bar{x}_i, \bar{x}))$ ;
**if** $R_{\leq i}(\bar{x}) \equiv R_{\leq i+1}(\bar{x})$ **then** **return** "no" ;
$i := i + 1$ ;
**end** **loop**
**end**

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Complete Forward Reachability Algorithm

**procedure** *CFReach*$(I, T, F)$
**input**: formulas $I, T, F$
**output**: "yes" or "no"
**begin**
  $i := 0$ ;
  $R_{\leq 0}(\bar{x}) := I(\bar{x})$ ;
  **loop**
    **if** $R_{\leq i}(\bar{x}) \wedge F(\bar{x})$ is satisfiable **then** **return** "yes" ;
    $R_{\leq i+1}(\bar{x}) := R_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (R_{\leq i}(\bar{x}_i) \wedge T(\bar{x}_i, \bar{x}))$ ;
    **if** $R_{\leq i}(\bar{x}) \equiv R_{\leq i+1}(\bar{x})$ **then** **return** "no" ;
    $i := i + 1$ ;
  **end** **loop**
**end**

Implementation?

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Complete Forward Reachability Algorithm

**procedure** $\mathit{CFReach}(I, T, F)$
**input**: formulas $I, T, F$
**output**: "yes" or "no"
**begin**
$i := 0$ ;
$R_{\leq 0}(\bar{x}) := I(\bar{x})$ ;
**loop**
**if** $R_{\leq i}(\bar{x}) \wedge F(\bar{x})$ is satisfiable **then** **return** "yes" ;
$R_{\leq i+1}(\bar{x}) := R_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (R_{\leq i}(\bar{x}_i) \wedge T(\bar{x}_i, \bar{x}))$ ;
**if** $R_{\leq i}(\bar{x}) \equiv R_{\leq i+1}(\bar{x})$ **then** **return** "no" ;
$i := i + 1$ ;
**end** **loop**
**end**

Implementation?
Use OBDDs and OBDD algorithms

Conjunction and disjunction
Quantification
Satisfiability checking
Equivalence checking

# Backward Reachability

# Main Problems with the Forward Reachability Algorithms

Forward reachability behave in the same way independently of the set of final states.

In other words, they are not goal oriented.

# 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.

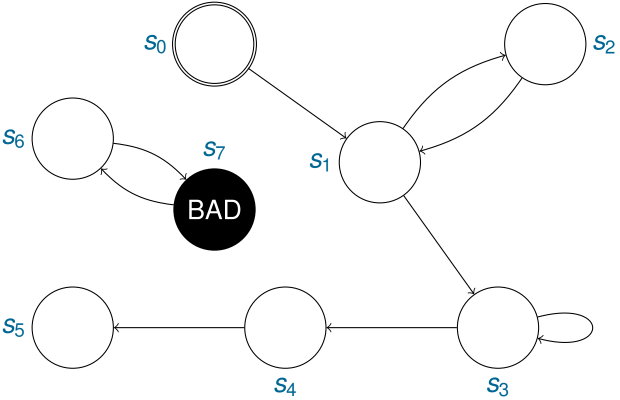
Number of backward steps:

# 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

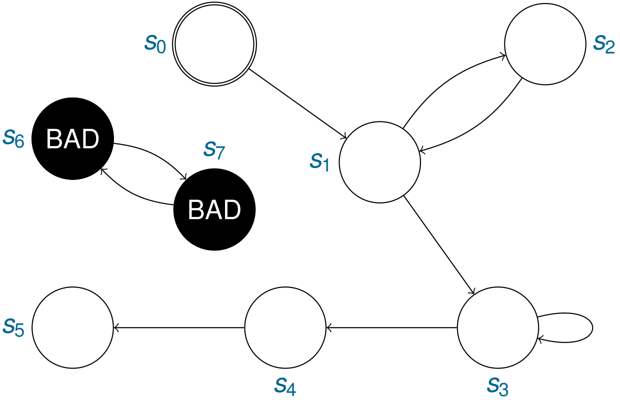


# Backward Reachability in $\leq n$ steps

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

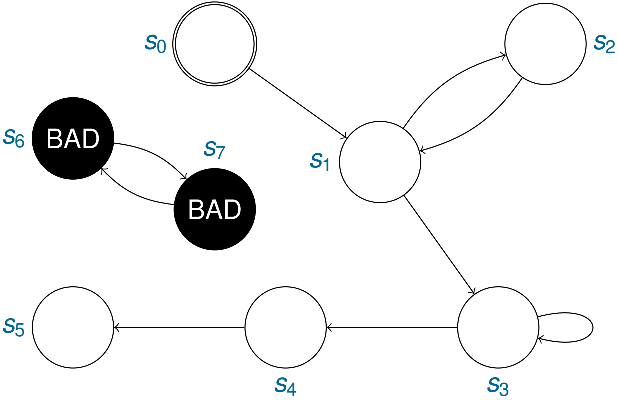


# Backward Reachability in $\leq n$ steps

Idea:

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

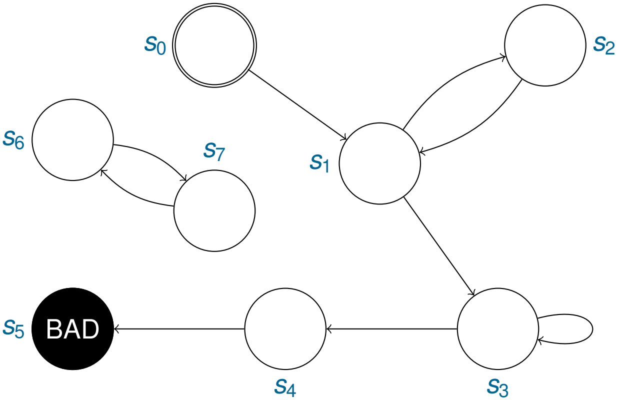
Number of backward steps: 1




Unreachable!

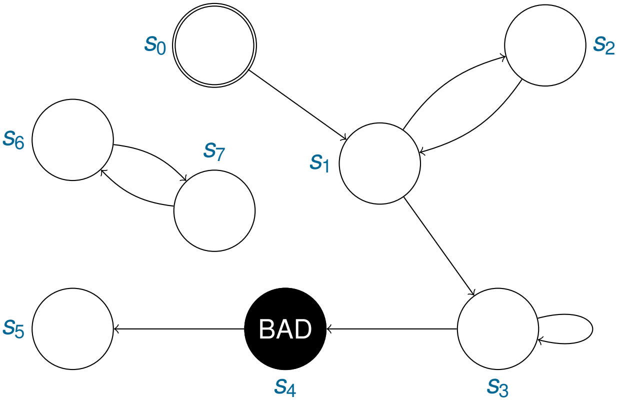
# Backward Reachability in *n* steps

Number of backward steps: 0



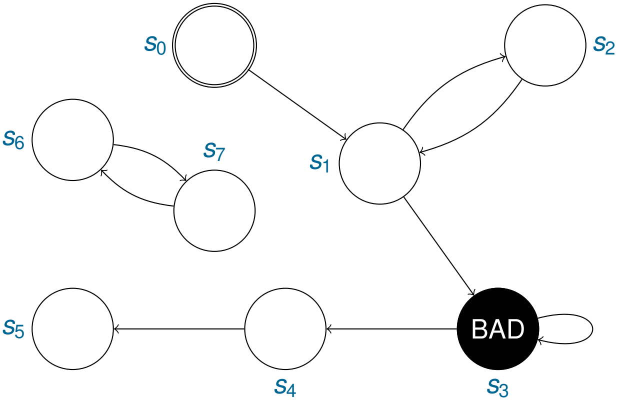
# Backward Reachability in *n* steps

Number of backward steps: 1



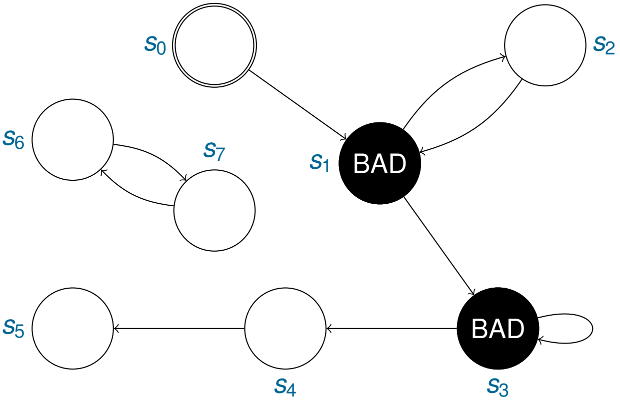
# Backward Reachability in *n* steps

Number of backward steps: 2



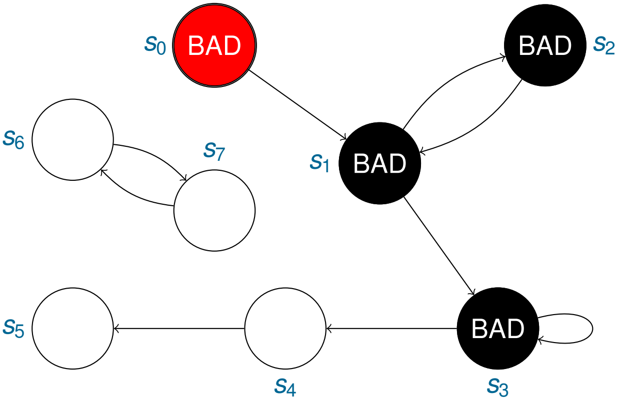
# Backward Reachability in $n$ steps

Number of backward steps: 3



# Backward Reachability in *n* steps

Number of backward steps: 4




Reachable!

# Backward Reachability

If $S_n$ is reachable from $S_0$ in $n$ steps, we say that $S_0$ is backward reachable from $S_0$ in $n$ steps.

# Backward Reachability

If $S_n$ is reachable from $S_0$ in $n$ steps, we say that $S_0$ is backward reachable from $S_0$ in $n$ steps.

## Lemma

*Let $C(\bar{x})$ symbolically represent a set of states $S$. Define*

$$BR(\bar{x}) \quad \stackrel{\text{def}}{=} \quad \exists \bar{x}_1 (C(\bar{x}_1) \wedge T(\bar{x}, \bar{x}_1)).$$

*Then $BR(\bar{x})$ represents the set of states backward reachable from $S$ in one step.*

# Complete 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**
  $i \; := \; 0$ ;
  $BR_{\leq 0}(\bar{x}) \; := \; F(\bar{x})$ ;
  **loop**
    **if** $BR_{\leq i}(\bar{x}) \wedge I(\bar{x})$ is satisfiable **then** **return** "yes" ;
    $BR_{\leq i+1}(\bar{x}) \; := \; BR_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (BR_{\leq i}(\bar{x}_i) \wedge T(\bar{x}, \bar{x}_i))$ ;
    **if** $BR_{\leq i}(\bar{x}) \equiv BR_{\leq i+1}(\bar{x})$ **then** **return** "no" ;
    $i \; := \; i + 1$ ;
  **end** **loop**
**end**

# Complete 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**
  $i$ := $0$ ;
  $BR_{\leq 0}(\bar{x})$ := $F(\bar{x})$ ;
  **loop**
    **if** $BR_{\leq i}(\bar{x}) \wedge I(\bar{x})$ is satisfiable **then return** "yes" ;
    $BR_{\leq i+1}(\bar{x})$ := $BR_{\leq i}(\bar{x}) \vee \exists \bar{x}_i (BR_{\leq i}(\bar{x}_i) \wedge T(\bar{x}, \bar{x}_i))$ ;
    **if** $BR_{\leq i}(\bar{x}) \equiv BR_{\leq i+1}(\bar{x})$ **then return** "no" ;
    $i$ := $i + 1$ ;
  **end loop**
**end**

# Summary

- model checking
- safety properties as reachability
- symbolic reachability checking
- one-sided forward reachability (satisfiability algorithms)
- full forward/backward reachability (QBF/OBDD)

# Short summary of the course

# Short summary of the course (I)

Propositional Logic:

- satisfiability, validity, equivalence
- formalising problems
- splitting algorithm, polarity, pure atom
- CNF, CNF transformation
- clausal form, definitional clausal form transformation
- satisfiability of sets of clauses: DPLL, splitting+unit propagation, pure literal, tautology removal, Horn clauses.
- satisfiability of general formulas: semantic tableaux

Probabilistic analysis of satisfiability:

- random clause generation, transition function
- sharp transitions, easy-hard problems
- randomized algorithms for satisfiability: GSAT, WSAT, GSAT with Random Walks

# Short summary of the course (II)

OBDDs: compact representation of Boolean functions

- BDT, OBDDs, building OBDDs, if-then-else normal form
- satisfiability, validity, equivalence checking for OBDDs
- alg. on OBDDs: disjunction, conjunc., quantification

QBF: Quantified Boolean Formulas

- syntax, semantics
- bound and free occurrences of variables
- rectification, prenex form, CNF transformation
- sat., validity can be reduced to evaluation of closed formulas
- evaluating QBF formulas:
  splitting, DPLL, pure literal, universal literal
- OBDD representation of QBF

# Short summary of the course (III)

Propositional Logic of Finite Domains (PLFD):

- syntax, semantics
- translation of propositional logic into PLFD and back
- satisfiability checking: semantic tableaux (new rules)

Transition Systems:

- states, transitions
- symbolic representation of sets of states, transitions
- preconditions, postconditions, frame problem

# Short summary of the course (IV)

Linear Temporal Logic LTL:
reasoning about temporal properties of transition systems

- syntax, semantics, temporal operators $\bigcirc$, $\Diamond$, $\Box$, $\mathcal{U}$, $\mathcal{R}$
- properties that can be expressed by LTL
- checking whether properties true/false on all/some paths of a transition system
- equivalence of LTL formulas, how to show non-equivalence

Model Checking:

- checking reachability and safety
- symbolic representation of reachable states using QBF
- forward symbolic model checking of the reachability property
- one-sided forward reachability (using satisfiability algorithms)
- full forward/backward reachability (QBF/OBDD)