Formal Methods
Module II: Formal Verification
Ch. 08: **Abstraction in Model Checking**

Roberto Sebastiani

DISI, Università di Trento, Italy – roberto.sebastiani@unitn.it
URL: https://disi.unitn.it/rseba/DIDATTICA/fm2023/
Teaching assistant: Giuseppe Spallitta – giuseppe.spallitta@unitn.it

M.S. in Computer Science, Mathematics, & Artificial Intelligence Systems
Academic year 2022-2023

last update: Thursday 11^{th} May, 2023, 11:52

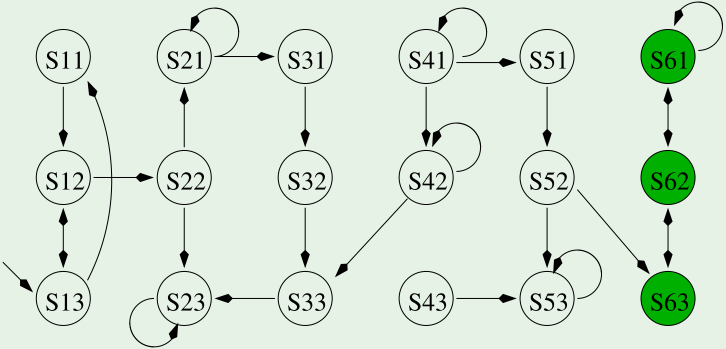
Outline

1. Abstraction
2. Abstraction-Based Symbolic Model Cheching
   - Abstraction
   - Checking the counter-examples
   - Refinement
3. Exercises

Outline

Model Checking Safety Properties: $M \models \mathbf{G}\neg BAD$

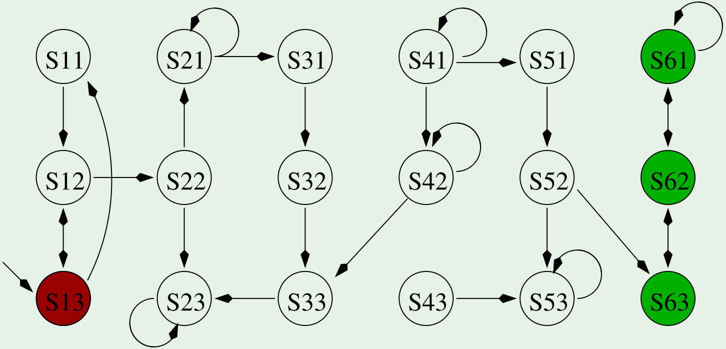
Add reachable states until reaching a fixed-point or a “bad” state



Problem: too many states to handle! (even for symbolic MC)

Model Checking Safety Properties: $M \models \mathbf{G}\neg BAD$

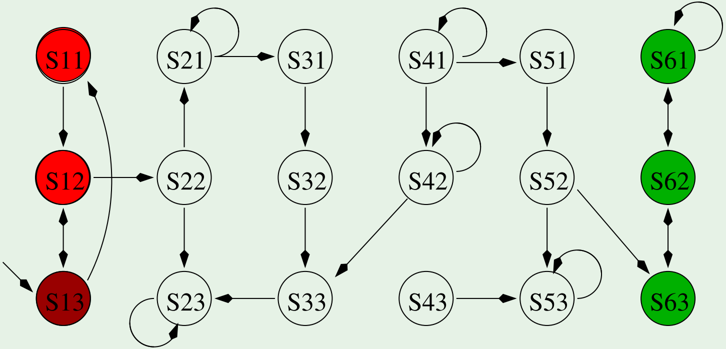
Add reachable states until reaching a fixed-point or a "bad" state



Problem: too many states to handle! (even for symbolic MC)

Model Checking Safety Properties: $M \models \mathbf{G}\neg BAD$

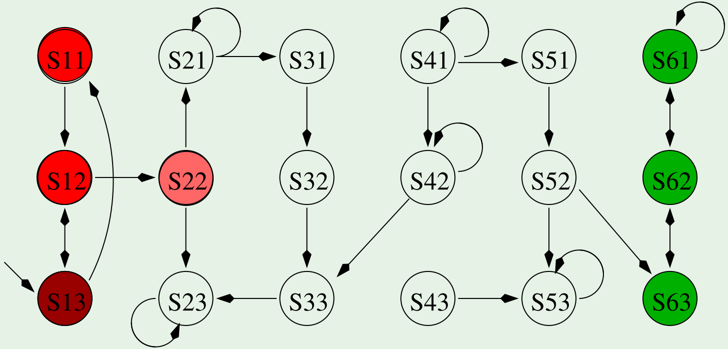
Add reachable states until reaching a fixed-point or a “bad” state



Problem: too many states to handle! (even for symbolic MC)

Model Checking Safety Properties: $M \models \mathbf{G}\neg BAD$

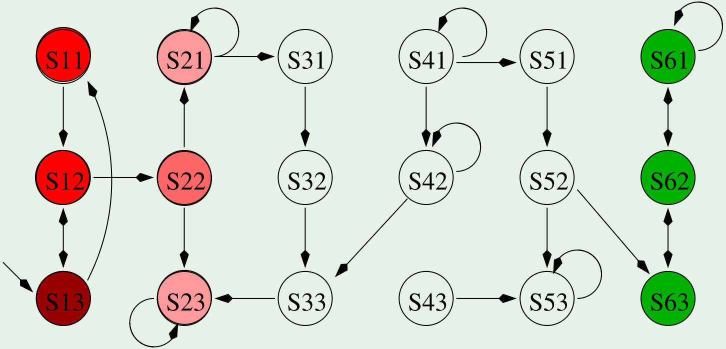
Add reachable states until reaching a fixed-point or a “bad” state



Problem: too many states to handle! (even for symbolic MC)

Model Checking Safety Properties: $M \models \mathbf{G}\neg BAD$

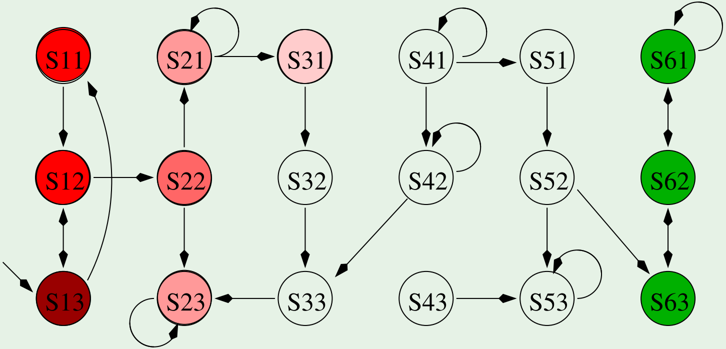
Add reachable states until reaching a fixed-point or a "bad" state



Problem: too many states to handle! (even for symbolic MC)

Model Checking Safety Properties: $M \models \mathbf{G}\neg BAD$

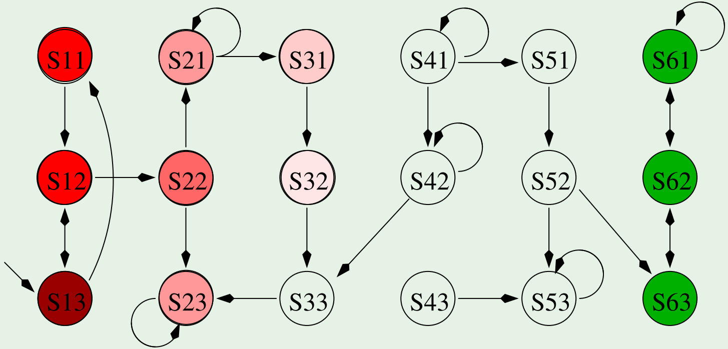
Add reachable states until reaching a fixed-point or a “bad” state



Problem: too many states to handle! (even for symbolic MC)

Model Checking Safety Properties: $M \models \mathbf{G}\neg BAD$

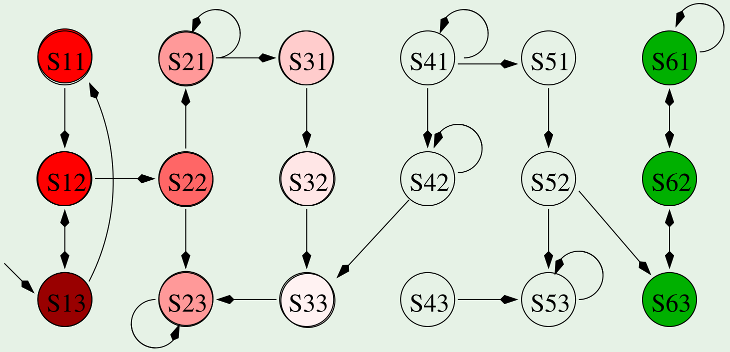
Add reachable states until reaching a fixed-point or a “bad” state



Problem: too many states to handle! (even for symbolic MC)

Model Checking Safety Properties: $M \models \mathbf{G}\neg BAD$

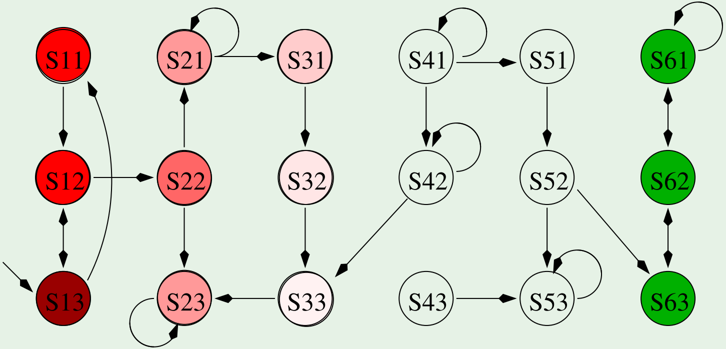
Add reachable states until reaching a fixed-point or a “bad” state



Problem: too many states to handle! (even for symbolic MC)

Model Checking Safety Properties: $M \models \mathbf{G}\neg BAD$

Add reachable states until reaching a fixed-point or a “bad” state

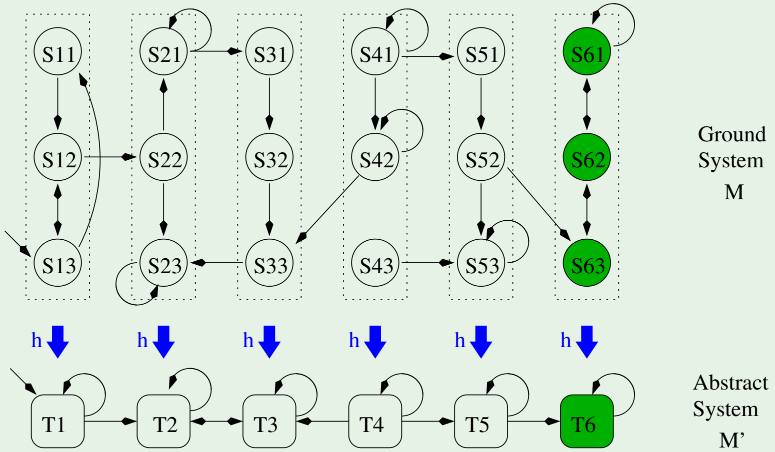


Problem: too many states to handle! (even for symbolic MC)

Idea: Abstraction

Apply a (non-injective) Abstraction Function *h* to M

$\Longrightarrow$ Build an abstract (and much smaller) system M'



Abstraction & Refinement

Abstraction & Refinement

- Let S be the ground (concrete) state space
- Let S' be the abstract state space
- Abstraction: a (typically non-injective) map $h : S \longmapsto S'$
  - h typically a many-to-one function
- Refinement: a map $r : S' \longmapsto 2^S$ s.t. $r(s') \stackrel{\text{def}}{=} \{s \in S \mid s' = h(s)\}$

Simulation and Bisimulation

Simulation

Let $M_1 \stackrel{\text{def}}{=} \langle S_1, I_1, R_1, AP_1, L_1 \rangle$ and $M_2 \stackrel{\text{def}}{=} \langle S_2, I_2, R_2, AP_2, L_2 \rangle$.
Then $p \subseteq S_1 \times S_2$ is a simulation between M_1 and M_2 (M_1 simulates M_2) iff

- for every $s_2 \in I_2$ exists $s_1 \in I_1$ s.t. $\langle s_1, s_2 \rangle \in p$, and
- for every $\langle s_1, s_2 \rangle \in p$:
  - for every transition $\langle s_2, t_2 \rangle \in R_2$, exists a transition $\langle s_1, t_1 \rangle \in R_1$ s.t. $\langle t_1, t_2 \rangle \in p$

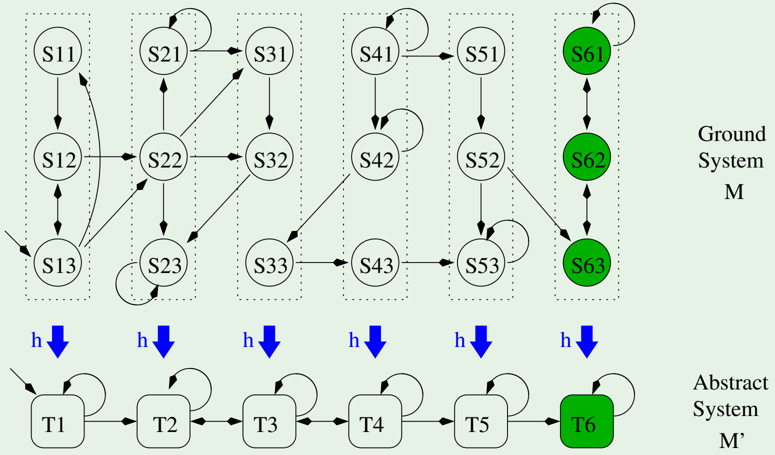
(Intuitively, for every transition in M_2 there is a corresponding transition in M_1.)

Example of p (spy game): "follower M_1 keeps escaper M_2 at eyesight"

Bisimulation

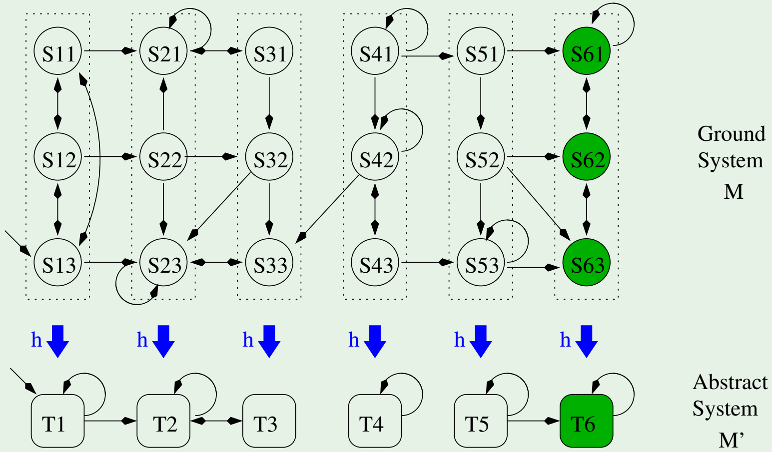
P is a bisimulation between M and M' iff it is both a simulation between M and M' and between M' and M.
We say that M and M' bisimulate each other.

Example I



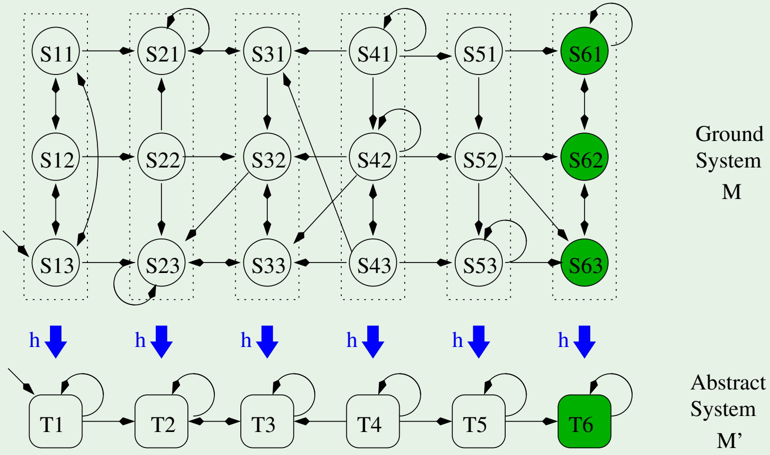
- Does M simulate M'? No: e.g., no arc from $S23$ to any $S3i$.
- Does M' simulate M? Yes

Example II



- Does M simulate M'? Yes
- Does M' simulate M? No: e.g., no arc from *T*4 to *T*3.

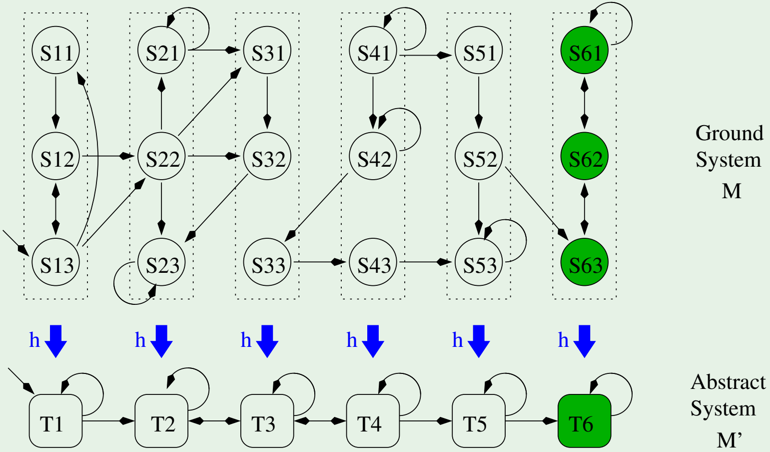
Example III



- Does M simulate M'? Yes
- Does M' simulate M? Yes

Existential Abstraction (Over-Approximation)

An Abstraction from M to M' is an Existential Abstraction (aka Over-Approximation) iff M' simulates M



Model Checking with Existential Abstractions

Preservation Theorem

- Let φ be a universally-quantified property (e.g., in LTL or ACTL)
- Let M' simulate M

Then we have that

$$M' \models \varphi \Longrightarrow M \models \varphi$$

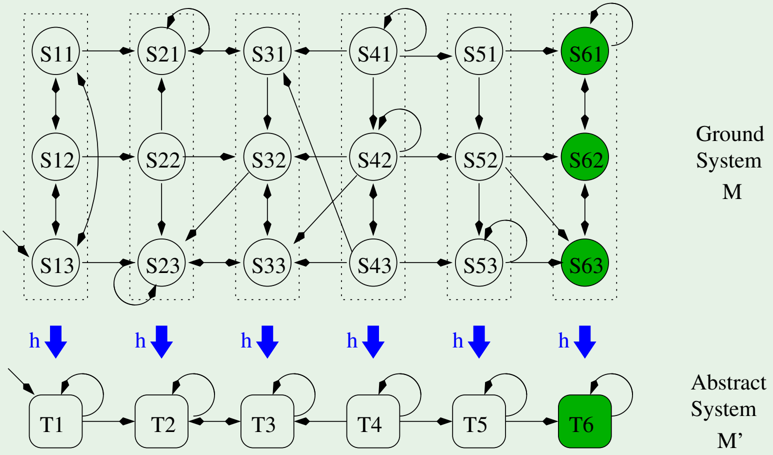
- Intuition: if M has a countermodel, then M' simulates it
- The converse does not hold

$$M \models \varphi \not\Longrightarrow M' \models \varphi$$

$\Longrightarrow$ The abstract counter-example may be spurious
(e.g., in previous figure, $T1 \rightarrow T2 \rightarrow T3 \rightarrow T4 \rightarrow T5 \rightarrow T6$)

Bisimulation Abstraction

An Abstraction from M to M' is a Bisimulation Abstraction iff
M simulates *M′* and *M′* simulates *M*



Model Checking with Bisimulation Abstractions

Preservation Theorem

- Let φ be any ACTL/LTL property
- Let M simulate M' and M' simulate M

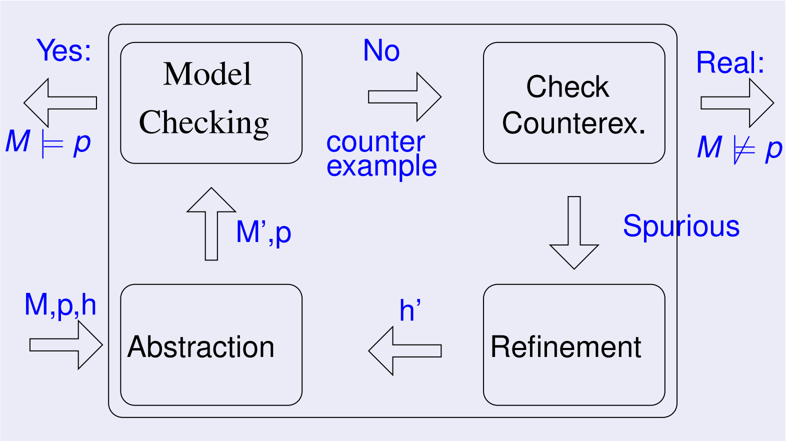
Then we have that

$$M' \models \varphi \Longleftrightarrow M \models \varphi$$

Outline

Counter-Example Guided Abstraction Refinement - CEGAR

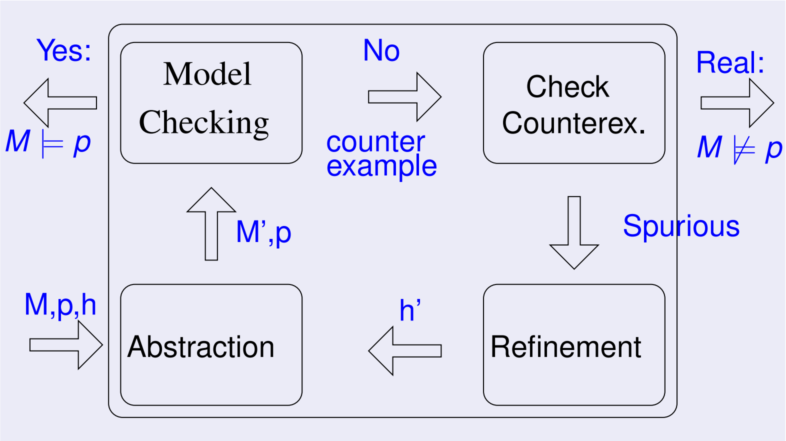
General Schema:



Outline

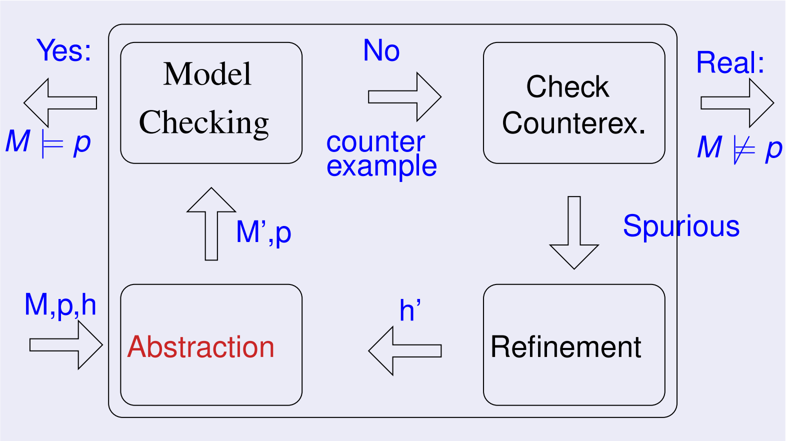
Counter-Example Guided Abstraction Refinement

General Schema:



Counter-Example Guided Abstraction Refinement

General Schema:



A Popular Abstraction for Symbolic MC of $\mathbf{G}\neg BAD$ I

- A.k.a. “Localization Reduction”
- Partition Boolean variables into visible (V) and invisible (I) ones
  - The abstract model built on visible variables only.
  - Invisible variables are made inputs (no updates in the transition relation)
  - All variables occurring in “$\neg BAD$” must be visible
- The abstraction function maps each state to its projection over V.

$\Longrightarrow$ Group ground states with same visible part to a single abstract state.

$$\begin{array}{cc} & \begin{array}{cccc} \multicolumn{2}{c}{\textit{visible}} & \multicolumn{2}{c}{\textit{invisible}} \\ x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{array}{c} S_{11}: \\ S_{12}: \\ S_{13}: \\ S_{14}: \end{array} & \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right] \end{array} \Longrightarrow \left[\, T_1 : \; 0 \quad 0 \,\right]$$

A Popular Abstraction for Symbolic MC of $\mathbf{G}\neg BAD$ I

- A.k.a. "Localization Reduction"
- Partition Boolean variables into visible (V) and invisible (I) ones
  - The abstract model built on visible variables only.
  - Invisible variables are made inputs (no updates in the transition relation)
  - All variables occurring in "$\neg BAD$" must be visible
- The abstraction function maps each state to its projection over V.

$\Longrightarrow$ Group ground states with same visible part to a single abstract state.

$$
\left[
\begin{array}{ccccc}
 & \multicolumn{2}{c}{\textit{visible}} & \multicolumn{2}{c}{\textit{invisible}} \\
 & x_1 & x_2 & x_3 & x_4 \\
\hline
S_{11}: & 0 & 0 & 0 & 0 \\
S_{12}: & 0 & 0 & 0 & 1 \\
S_{13}: & 0 & 0 & 1 & 0 \\
S_{14}: & 0 & 0 & 1 & 1
\end{array}
\right]
\Longrightarrow
\left[\; T_1 : \; 0 \quad 0 \; \right]
$$

A Popular Abstraction for Symbolic MC of $\mathbf{G}\neg BAD$ I

- A.k.a. "Localization Reduction"
- Partition Boolean variables into visible (V) and invisible (I) ones
  - The abstract model built on visible variables only.
  - Invisible variables are made inputs (no updates in the transition relation)
  - All variables occurring in "$\neg BAD$" must be visible
- The abstraction function maps each state to its projection over V.

$\Longrightarrow$ Group ground states with same visible part to a single abstract state.

$$\begin{array}{c}\begin{array}{ccccc} & \multicolumn{2}{c}{\textit{visible}} & \multicolumn{2}{c}{\textit{invisible}} \end{array}\\ \left[\begin{array}{ccccc} & x_1 & x_2 & x_3 & x_4 \\ \hline S_{11}: & 0 & 0 & 0 & 0 \\ S_{12}: & 0 & 0 & 0 & 1 \\ S_{13}: & 0 & 0 & 1 & 0 \\ S_{14}: & 0 & 0 & 1 & 1 \end{array}\right]\end{array} \Longrightarrow \left[\begin{array}{ccc} T_1: & 0 & 0 \end{array}\right]$$

A Popular Abstraction for Symbolic MC of $\mathbf{G}\neg BAD$ II

M' can be computed efficiently if M is in functional form
(e.g. sequential circuits).

$$\left[\begin{array}{l} next(x_1) := f_1(x_1, x_2, x_3, x_4) \\ next(x_2) := f_2(x_1, x_2, x_3, x_4) \\ next(x_3) := f_3(x_1, x_2, x_3, x_4) \\ next(x_4) := f_4(x_1, x_2, x_3, x_4) \end{array}\right] \Longrightarrow \left[\begin{array}{l} next(x_1) := f_1(x_1, x_2, x_3, x_4) \\ next(x_2) := f_2(x_1, x_2, x_3, x_4) \end{array}\right]$$

Note: The next values of invisible variables, $next(x_3)$ and $next(x_4)$, can assume every value nondeterministically
$\Longrightarrow$ do not constrain the transition relation

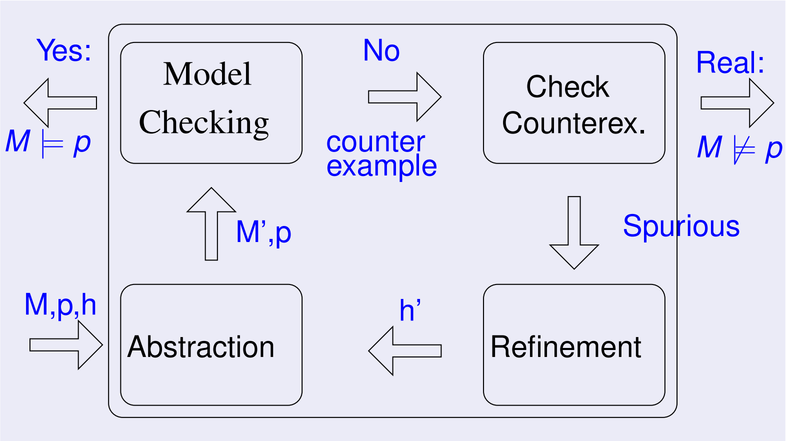
Since M' obviously simulates M, this is an Existential Abstraction

- $M' \models \varphi \Longrightarrow M \models \varphi$
- may produce spurious counter-examples

Outline

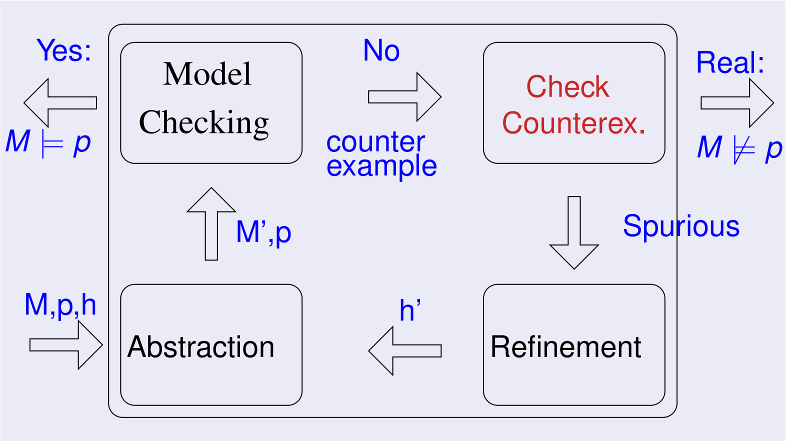
Counter-Example Guided Abstraction Refinement

General Schema:



Counter-Example Guided Abstraction Refinement

General Schema:



Checking the Abstract Counter-Example I

The problem

- Let $c_0, ..., c_m$ counter-example in the abstract space
  - Note: each c_i is a truth assignment on the visible variables
- Problem: check if there exist a corresponding ground counterexample $s_0, ..., s_m$ s.t. $c_i = h(s_i)$, for every i

Checking the Abstract Counter-Example II

Idea

- Simulate the counterexample on the concrete model
- Use Bounded Model Checking:

$$\Phi \stackrel{\text{def}}{=} I(s_0) \wedge \bigwedge_{i=0}^{m-1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=0}^{m} \mathit{visible}(s_i) = c_i$$

If satisfiable, the counter example is real, otherwise it is spurious

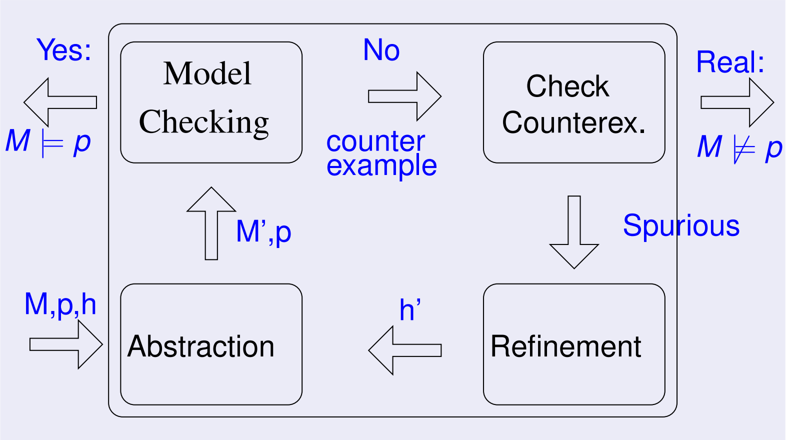
Note: much more efficient than the direct BMC problem:

$$\Phi \stackrel{\text{def}}{=} I(s_0) \wedge \bigwedge_{i=0}^{m-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{m} \neg BAD_i$$

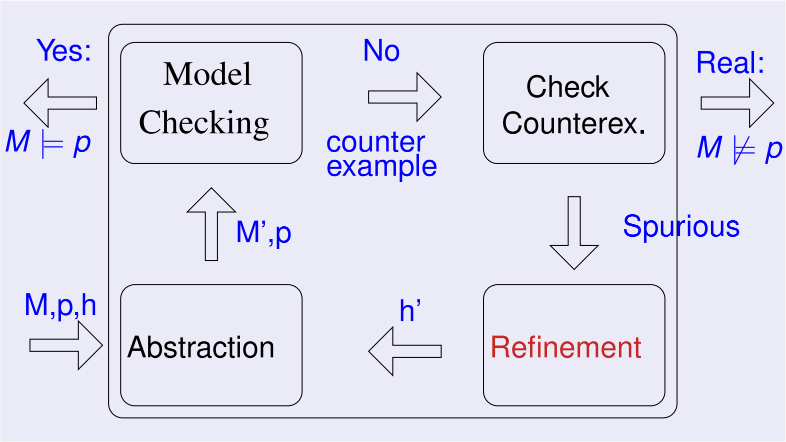
$\Longrightarrow$ cuts a $2^{(m+1)\cdot|V|}$ factor from the Boolean search space.

Outline

Counter-Example Guided Abstraction Refinement



Counter-Example Guided Abstraction Refinement



The cause of spurious counter-examples I

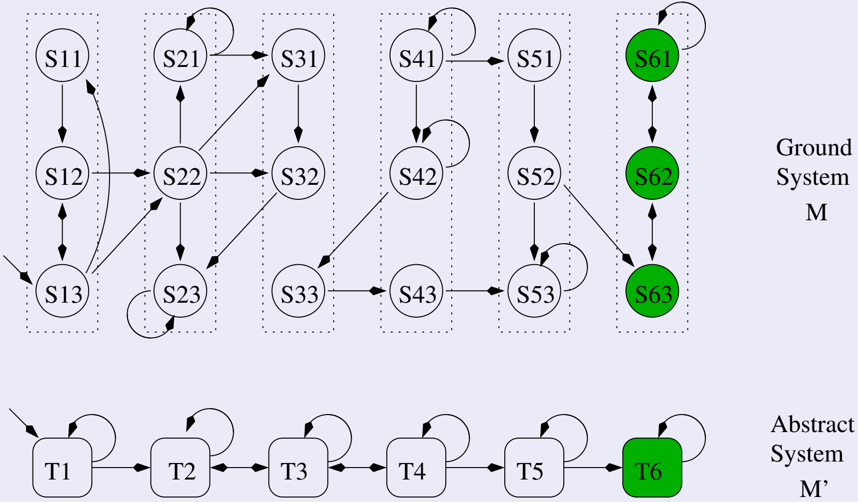
Problem

There is a state in the abstract counter-example (failure state) s.t. two different and un-connected kinds of ground states are mapped into it:

- Deadend states: reachable states which do not allow to proceed along a refinement of the abstract counter-example
- Bad states: un-reachable states which allow to proceed along a refinement of the abstract counter-example

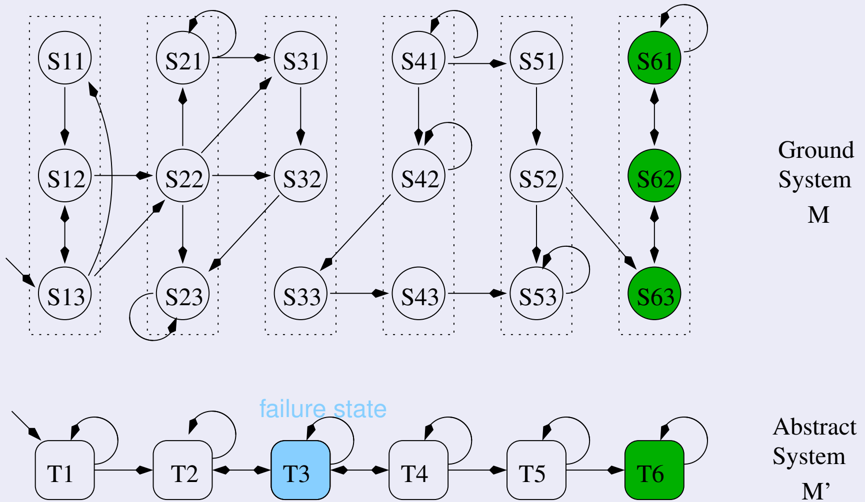
The cause of spurious counter-examples II

For the spurious counter-example: $T1 \to T2 \to T3 \to T4 \to T5 \to T6$



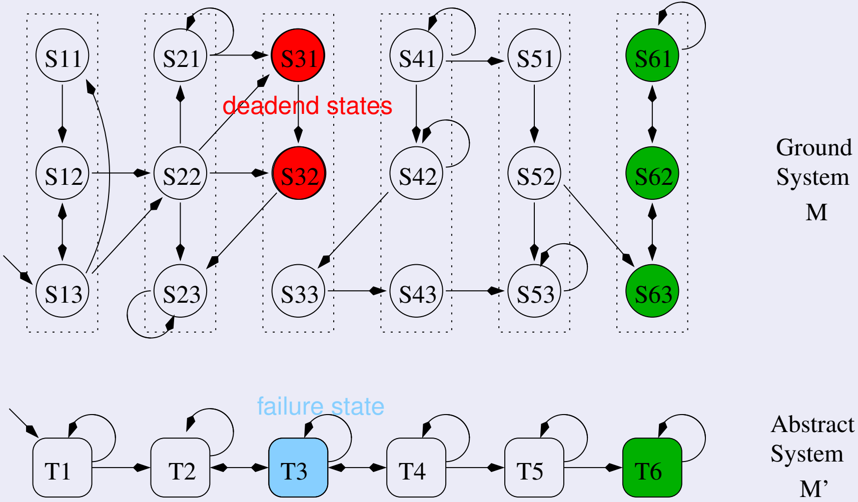
The cause of spurious counter-examples II

For the spurious counter-example: $T1 \rightarrow T2 \rightarrow T3 \rightarrow T4 \rightarrow T5 \rightarrow T6$



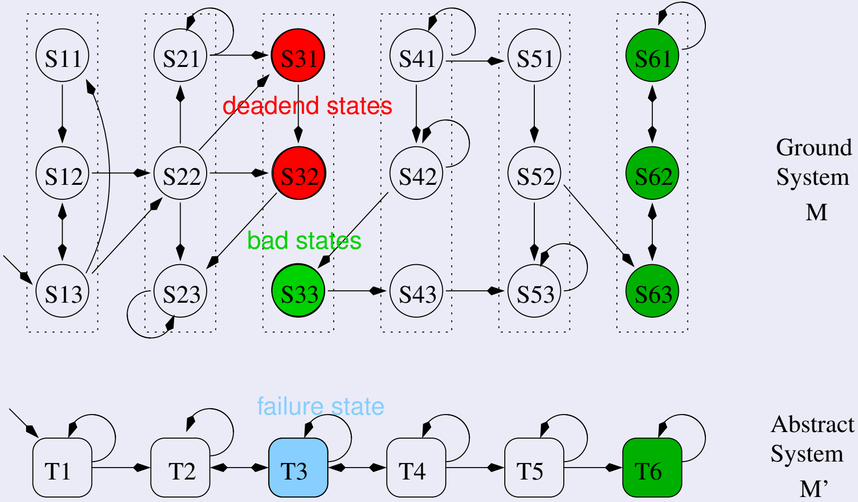
The cause of spurious counter-examples II

For the spurious counter-example: $T1 \rightarrow T2 \rightarrow T3 \rightarrow T4 \rightarrow T5 \rightarrow T6$



The cause of spurious counter-examples II

For the spurious counter-example: $T1 \rightarrow T2 \rightarrow T3 \rightarrow T4 \rightarrow T5 \rightarrow T6$



The cause of spurious counter-examples III

Problem

There is a state in the abstract counter-example (failure state) s.t. two different and un-connected kinds of ground states are mapped into it:

- Deadend states: reachable states which do not allow to proceed along a refinement of the abstract counter-example
- Bad states: un-reachable states which allow to proceed along a refinement of the abstract counter-example

Solution: Refine the abstraction function.

1. identify the failure state and its deadend and bad states
2. refine the abstraction function s.t. deadend and bad states are mapped into different abstract state

Identify the failure state and its deadend & bad states

- The failure state is the state of maximum index f in the abstract counter-example s.t. the following formula is satisfiable:

$$\Phi_D \stackrel{\text{def}}{=} I(s_0) \wedge \bigwedge_{i=0}^{f-1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=0}^{f} visible(s_i) = c_i$$

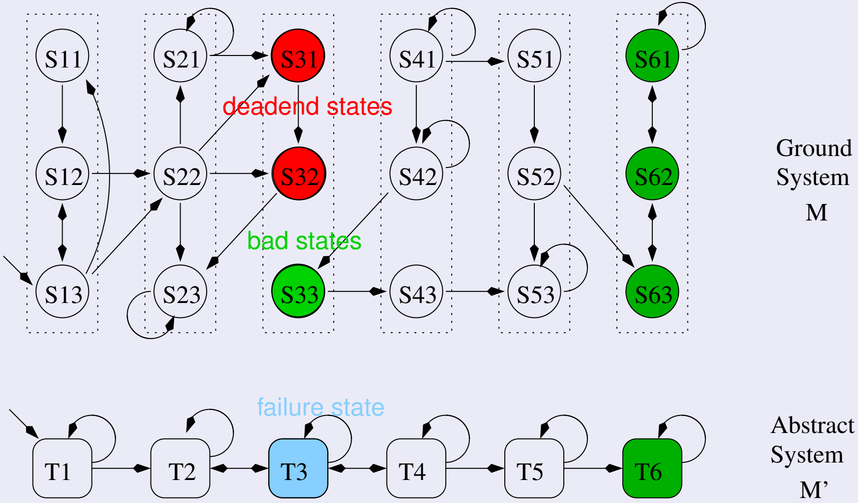
- The (restriction on index f of the) models of Φ_D identify the deadend states $\{d_1, ..., d_k\}$
  - can be identified by projected AllSAT enumeration over variables s_f
- The bad states $\{b_1, ..., b_n\}$ are identified by the (restriction on index f of the) models of the following formula:

$$\Phi_B \stackrel{\text{def}}{=} R(s_f, s_{f+1}) \wedge visible(s_f) = c_f \wedge visible(s_{f+1}) = c_{f+1}$$

  - can be identified by projected AllSAT enumeration over variables s_f

Identify the failure state and its deadend & bad states

For the spurious counter-example: $T1 \rightarrow T2 \rightarrow T3 \rightarrow T4 \rightarrow T5 \rightarrow T6$



Refinement: Separate deadend & bad states

The state separation problem

- Input: sets $D \stackrel{\text{def}}{=} \{d_1, ..., d_k\}$ and $B \stackrel{\text{def}}{=} \{b_1, ..., b_n\}$ of states
- Output: (possibly smallest) set $U \subseteq I$ of invisible variables s.t.

$$\forall d_i \in D,\ \forall b_j \in B,\ \exists u \in U\ s.t.\ d_i(u) \neq b_j(u)$$

$\Longrightarrow$ the truth values of U allow for separating each pair $\langle d_i, b_j \rangle$

$\Longrightarrow$ The refinement h' is obtained by adding U to V.

Example

visible, invisible

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

- differentiating d_1, b_1: make x_4 visible
- differentiating d_1, b_2: make x_5 visible
- differentiating d_2, b_1: make x_7 visible
- differentiating d_2, b_2: already different

$\Longrightarrow$ $U = \{x_4, x_5, x_7\}$, h' keeps only x_6 invisible

Goal: Keep U as small as possible!

Example

visible, invisible

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

- differentiating d_1, b_1: make x_4 visible
- differentiating d_1, b_2: make x_5 visible
- differentiating d_2, b_1: make x_7 visible
- differentiating d_2, b_2: already different

$\Longrightarrow$ $U = \{x_4, x_5, x_7\}$, h' keeps only x_6 invisible

Goal: Keep U as small as possible!

Example

visible, invisible

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

- differentiating d_1, b_1: make x_4 visible
- differentiating d_1, b_2: make x_5 visible
- differentiating d_2, b_1: make x_7 visible
- differentiating d_2, b_2: already different

$\Longrightarrow$ $U = \{x_4, x_5, x_7\}$, h' keeps only x_6 invisible

Goal: Keep U as small as possible!

Example

visible, invisible

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

- differentiating d_1, b_1: make x_4 visible
- differentiating d_1, b_2: make x_5 visible
- differentiating d_2, b_1: make x_7 visible
- differentiating d_2, b_2: already different

$\Longrightarrow$ $U = \{x_4, x_5, x_7\}$, h' keeps only x_6 invisible

Goal: Keep U as small as possible!

Example

visible, invisible

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

- differentiating d_1, b_1: make x_4 visible
- differentiating d_1, b_2: make x_5 visible
- differentiating d_2, b_1: make x_7 visible
- differentiating d_2, b_2: already different

$\Longrightarrow$ $U = \{x_4, x_5, x_7\}$, h' keeps only x_6 invisible

Goal: Keep U as small as possible!

Example

visible, invisible

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

- differentiating d_1, b_1: make x_4 visible
- differentiating d_1, b_2: make x_5 visible
- differentiating d_2, b_1: make x_7 visible
- differentiating d_2, b_2: already different

$\Longrightarrow$ $U = \{x_4, x_5, x_7\}$, h' keeps only x_6 invisible

Goal: Keep U as small as possible!

Example

visible, invisible

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

- differentiating d_1, b_1: make x_4 visible
- differentiating d_1, b_2: make x_5 visible
- differentiating d_2, b_1: make x_7 visible
- differentiating d_2, b_2: already different

$\Longrightarrow$ $U = \{x_4, x_5, x_7\}$, h' keeps only x_6 invisible

Goal: Keep U as small as possible!

Two Separation Methods

- Separation based on Decision-Tree Learning
  - Not optimal.
  - Polynomial.
- ILP-based separation
  - Minimal separating set.
  - Computationally expensive.

Separation with decision tree (Example)

Idea: expand the decision tree until no $\langle d_i, b_j \rangle$ pair belongs to set.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

$\{d_1, d_2, b_1, b_2\}$

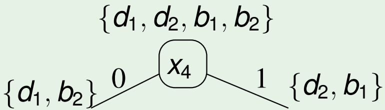
- differentiating d_1, b_1: x_4
- differentiating d_1, b_2: x_5
- differentiating d_2, b_1: x_7

$\Longrightarrow U = \{x_4, x_5, x_7\}$

Separation with decision tree (Example)

Idea: expand the decision tree until no $\langle d_i, b_j \rangle$ pair belongs to set.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

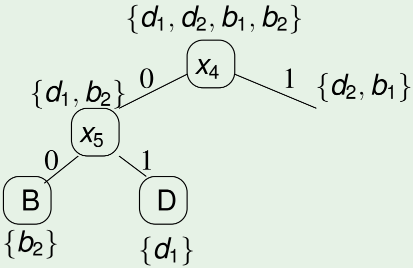


- differentiating d_1, b_1: x_4
- differentiating d_1, b_2: x_5
- differentiating d_2, b_1: x_7
  $\Longrightarrow U = \{x_4, x_5, x_7\}$

Separation with decision tree (Example)

Idea: expand the decision tree until no $\langle d_i, b_j \rangle$ pair belongs to set.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

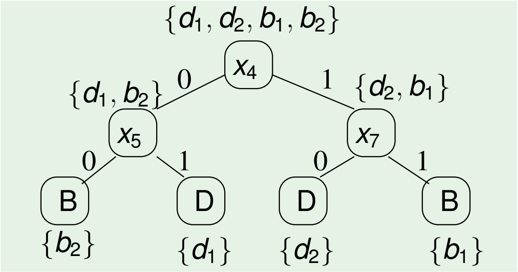


- differentiating d_1, b_1: x_4
- differentiating d_1, b_2: x_5
- differentiating d_2, b_1: x_7
  $\Longrightarrow U = \{x_4, x_5, x_7\}$

Separation with decision tree (Example)

Idea: expand the decision tree until no $\langle d_i, b_j \rangle$ pair belongs to set.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1



- differentiating d_1, b_1: x_4
- differentiating d_1, b_2: x_5
- differentiating d_2, b_1: x_7
  $\Longrightarrow U = \{x_4, x_5, x_7\}$

Separation with 0-1 ILP

Idea

- Encode the problem as a 0-1 ILP problem

$$\min \sum_{x_k \in I} v_k, \qquad \textit{subject to :}$$

$$\sum_{\substack{x_k \in I \\ d(x_k) \neq b(x_k)}} v_k \geq 1 \qquad \forall d \in D,\ \forall b \in B,$$

- intuition: $v_k = \top$ iff x_k must me made visible
- one constraint for every pair $\langle d_i, b_j \rangle$

Separation with 0-1 ILP: Example

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
d_1	0	1	0	0	1	0	1
d_2	0	1	0	1	1	1	0
b_1	0	1	0	1	1	1	1
b_2	0	1	0	0	0	0	1

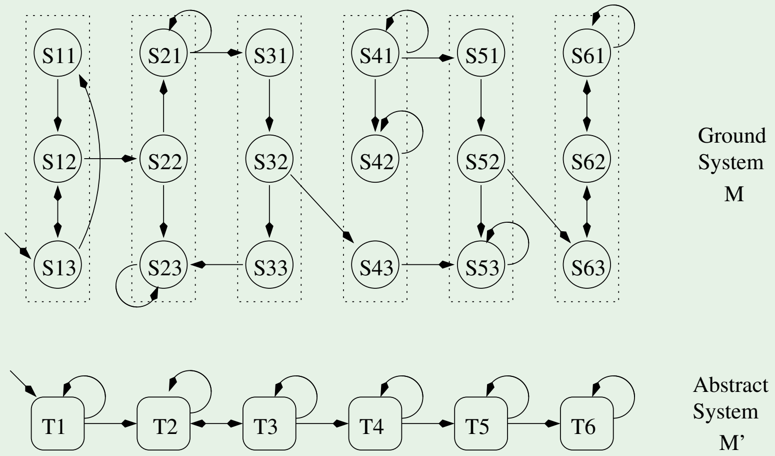
$$min\ \{v_4 + v_5 + v_6 + v_7\} \qquad subject\ to:$$

$$\begin{cases} v_4 + \quad\quad v_6 \quad\quad \geq 1 & \text{// separating } d_1, b_1 \\ \quad\quad v_5 \quad\quad\quad\quad \geq 1 & \text{// separating } d_1, b_2 \\ \quad\quad\quad\quad\quad\quad v_7 \geq 1 & \text{// separating } d_2, b_1 \\ v_4 + v_5 + v_6 + v_7 \geq 1 & \text{// separating } d_2, b_2 \end{cases}$$

Outline

Ex: Simulation

Consider the following pair of ground and abstract machines M and M', and the abstraction $\alpha : M \longmapsto M'$ which, for every $j \in \{1, ..., 6\}$, maps $Sj1, Sj2, Sj3$ into Tj.



Ex: Simulation [cont.]

For each of the following facts, say which is true and which is false.

(a) M simulates M'.

[Solution: False. E.g.,: if M is in $S23$, M' is in $T2$ and M' switches to $T3$, there is no transition in M from $S23$ to any state $S3i$, $i \in \{1, 2, 3\}$.]

(b) M' simulates M.

[Solution: true]

(c) for every $j \in \{1, ..., 6\}$ and $i \in \{1, ..., 3\}$, if Tj is reachable in M', then Sji is reachable in M

[Solution: False. E.g., $T4$ is reachable but $S42$ is not.]

(d) for every $j \in \{1, ..., 6\}$ and $i \in \{1, ..., 3\}$, if Sji is reachable in M, then Tj is reachable in M'.

[Solution: true]

Ex: Abstraction-based MC

Consider the following pair of ground and abstract machines M and M', and the abstraction $\alpha : M \longmapsto M'$ which makes the variable `z` invisible.

M:

```
MODULE main
VAR
  x : boolean;
  y : boolean;
  z : boolean;
ASSIGN
  init(x) := FALSE;
  init(y) := FALSE;
  init(z) := TRUE;
TRANS
  (next(x) <-> y) &
  (next(y) <-> z) &
  (next(z) <-> x)
```

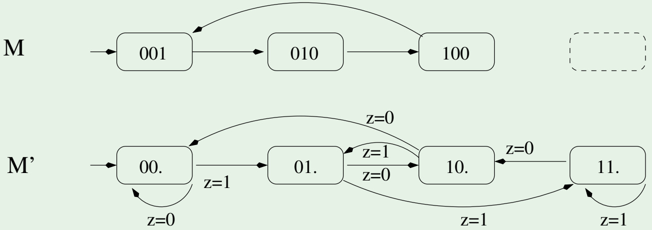
M':

```
MODULE main
VAR
  x : boolean;
  y : boolean;
  z : boolean;
ASSIGN
  init(x) := FALSE;
  init(y) := FALSE;

TRANS
  (next(x) <-> y) &
  (next(y) <-> z)
```

Ex: Abstraction-based MC [cont.]

(a) Draw the FSM's for M and M' (n.b.: in M' only v_1 and v_2 are state variables).
[Solution: (We label states with xyz and xy. respectively. "$z = 0$" and "$z = 1$" are comments.)



]

(b) Does M simulate M'? [Solution: No. E.g. the M' execution looping on (00) cannot be simulated in M.]

(c) Does M' simulate M? [Solution: Yes]

(d) Is α a suitable abstraction for solving the MC problem $M \models \mathbf{G}\neg(v_1 \wedge v_2)$?
If yes, explain why. If no, produce a spurious counter-example.
[Solution: No, since $M \models \mathbf{G}\neg(v_1 \wedge v_2)$ but $M' \not\models \mathbf{G}\neg(v_1 \wedge v_2)$. A spurious counter-example is $C \stackrel{\text{def}}{=} (00) \Longrightarrow (01) \Longrightarrow (11)$.]

Ex: Abstraction-based MC [cont.]

(e) Use the SAT-based refinement technique to show that the abstract counter-example $C \stackrel{\text{def}}{=} (00) \Longrightarrow (01) \Longrightarrow (11)$ is spurious.

[Solution: We generate the following formula and feed it to a SAT solver:

$$
\begin{array}{lcl}
(\neg x_0 \wedge \neg y_0 \wedge z_0) & \wedge & // \; I(x_0, y_0, z_0) \wedge \\
((x_1 \leftrightarrow y_0) \wedge (y_1 \leftrightarrow z_0) \wedge (z_1 \leftrightarrow x_0)) & \wedge & // \; T(x_0, y_0, z_0, x_1, y_1, z_1) \wedge \\
((x_2 \leftrightarrow y_1) \wedge (y_2 \leftrightarrow z_1) \wedge (z_2 \leftrightarrow x_1)) & \wedge & // \; T(x_1, y_1, z_1, x_2, y_2, z_2) \wedge \\
(\neg x_0 \wedge \neg y_0) & \wedge & // \; (visible(s_0) = c_0) \wedge \\
(\neg x_1 \wedge \; y_1) & \wedge & // \; (visible(s_1) = c_1) \wedge \\
(\; x_2 \wedge \; y_2) & & // \; (visible(s_2) = c_2)
\end{array}
$$

$\Longrightarrow$ $\{\neg x_0, \neg y_0, \; z_0, \neg x_1, \; y_1, \neg z_1, \; x_2, \neg y_2, \neg z_2\}$ are unit-propagated due to the first three rows

$\Longrightarrow$ UNSAT

$\Longrightarrow$ spurious counter-example.

]

Ex: Separation problem

In a counter-example-guided-abstraction-refinement model checking process using localization reduction, variables $x_3, x_4, x_5, x_6, x_7, x_8$ are made invisible.
Suppose the process has identified a spurious counterexample with an abstract failure state [00], two ground deadend states d_1, d_2 and two ground bad states b_1, b_2 as described in the following table:

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
d_1	0	0	0	0	0	1	1	1
d_2	0	0	0	1	1	1	1	0
b_1	0	0	1	1	1	1	0	1
b_2	0	0	0	1	0	0	0	0

Identify a minimum-size subset of invisible variables which must be made visible in the next abstraction to avoid the above failure. Briefly explain why.
[Solution: The minimum-size subset is $\{x_7\}$. In fact, if x_7 is made visible, then both d_1, d_2 are made different from both b_1, b_2.]