

# 3-Valued Abstraction-Refinement

Sharon Shoham

Academic College of Tel-Aviv Yaffo

# Model Checking

An efficient procedure that receives:

- A **finite-state model** describing a system
- A **temporal logic formula** describing a property

It returns

**yes**, if the system has the property

**no** + **Counterexample**, otherwise

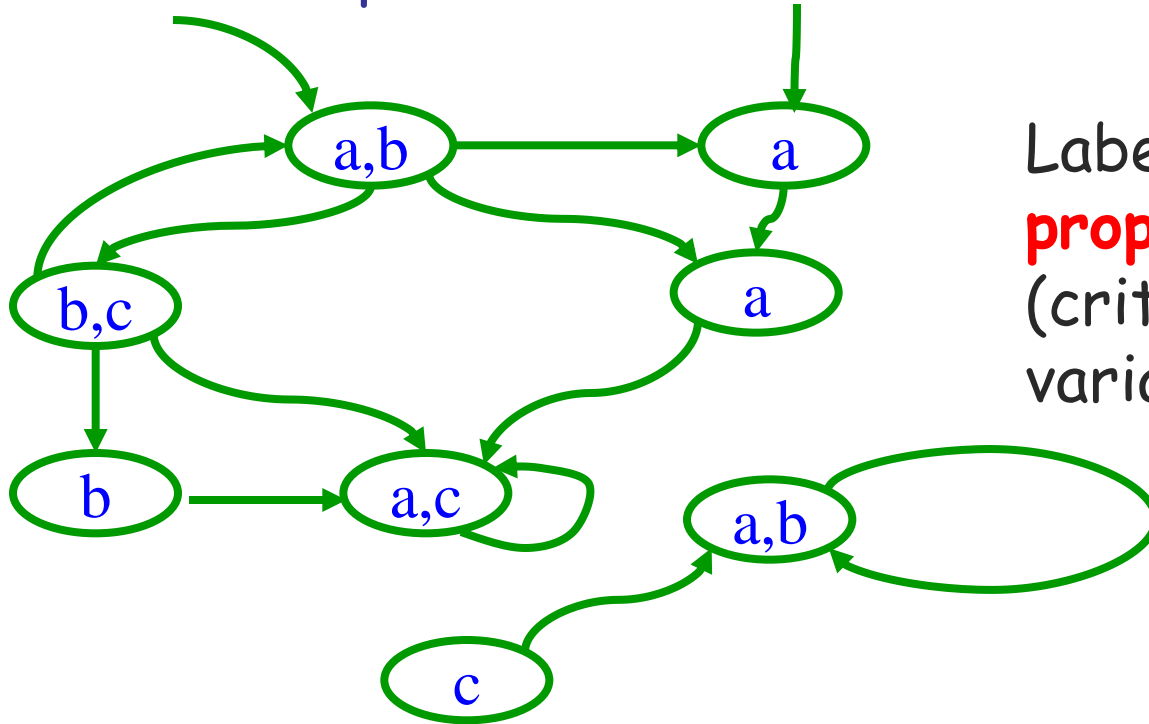
[EC81, QS82]

# Model Checking

- Emerging as an industrial standard tool for verification of **hardware** designs: Intel, IBM, Cadence, ...
- Recently applied successfully also for **software** verification: SLAM (Microsoft), Java PathFinder and SPIN (NASA), BLAST (EPFL), CBMC (Oxford),...

# Model of a System

Kripke structure / transition system



Labeled by **atomic propositions AP**  
(critical section, variable value...)

Notation:  $M = (AP, S, s^0, R, L)$

States

Initial  
state

Transitions  
 $\subseteq S \times S$

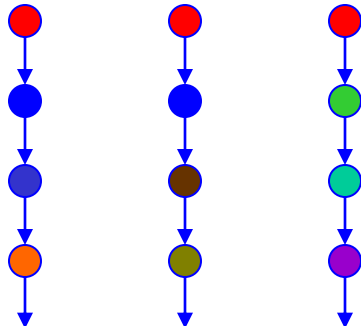
Labeling  
 $S \rightarrow 2^{Lit}$

# Temporal Logics

Express properties of event orderings in time

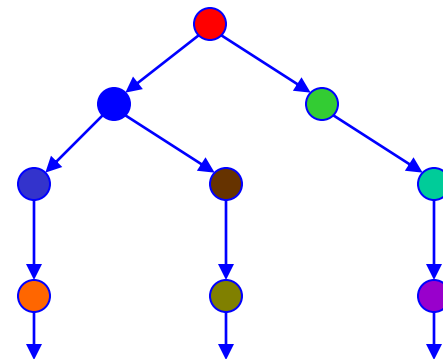
- **Linear Time**

- Every moment has a unique successor
- Infinite sequences (words)
- Linear Time Temporal Logic (LTL)



- **Branching Time**

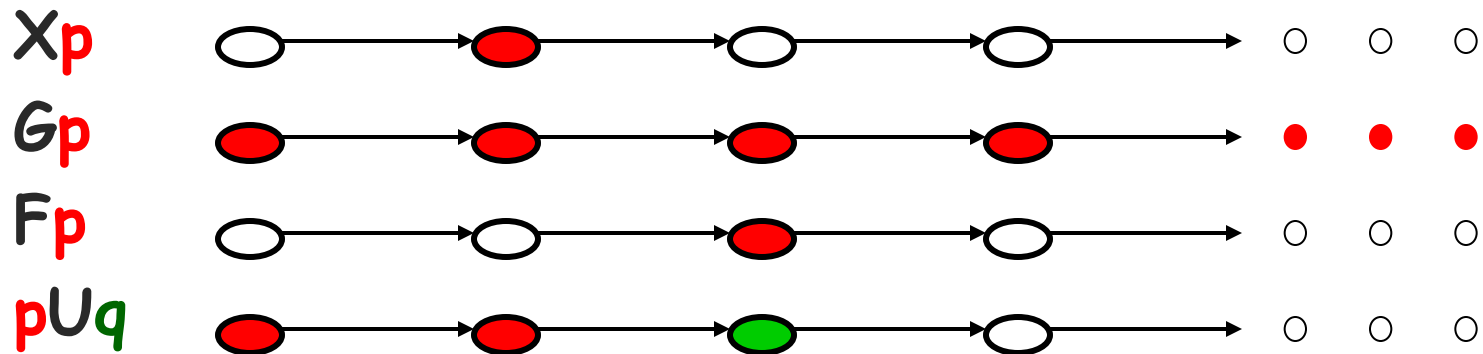
- Every moment has several successors
- Infinite tree
- Computation Tree Logic (CTL), CTL\*,  $\mu$ -calculus



# Propositional Temporal Logic

**AP** - a set of atomic propositions

Temporal operators:



Path quantifiers: **A** for all paths

**E** there exists a path

# LTL / CTL / CTL\*

**LTL** - of the form  $A\psi$

$\psi$  - path formula, contains **no** path **quantifiers**

- interpreted over infinite computation paths

**CTL** - path quantifiers and temporal operators appear in pairs: **AG, AU, AF, AX, EG, EU, EF, EX**

- interpreted over infinite computation trees

**CTL\*** - Allows any combination of temporal operators and path quantifiers. Includes both LTL and CTL

## ACTL / ACTL\*

The **universal** fragments of the logics, with only universal path quantifiers

- Negation is allowed only on atomic propositions



# Main Limitation of Model Checking

## The state explosion problem:

Model checking is efficient w.r.t. to the state space of the model. But...

The number of states in the system model grows exponentially with

- the number of variables
- the number of components in the system

# “Solutions” to the State Explosion Problem

Symbolic model checking:


The model is represented symbolically

- BDD-based model checking
- SAT-based Bounded/ Unbounded model checking

Small models replace the full, concrete model:

- Abstraction
- Compositional verification
- Partial order reduction
- Symmetry

# Outline

- Background:
  - Model Checking 
  - Abstraction
  - CEX guided abstraction-refinement (CEGAR)
- 3-Valued Abstraction Refinement (TVAR)
- Example: TVAR for CTL
- Investigation of abstract models used in TVAR
  - Monotonicity of Refinement
  - Completeness
  - Precision
  - Efficiency

# Abstraction-Refinement

- **Abstraction**: removes or simplifies details that are irrelevant to the property under consideration

Can reduce the number of states

- from **large** to **small**
- from **infinite** to **finite**

- **Refinement** might be needed

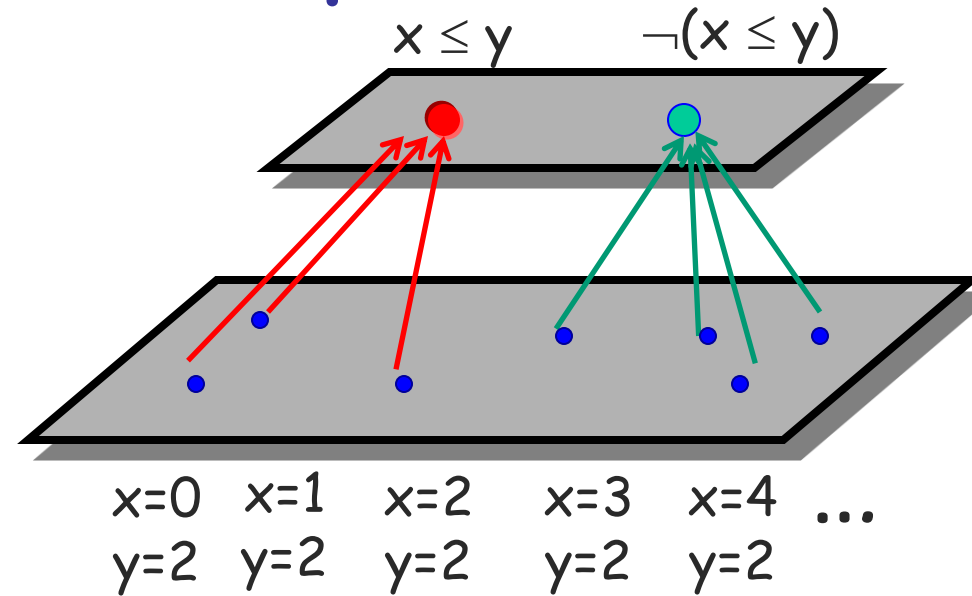
# Widely used Abstractions

- **Localization reduction / invisible variables**: each variable either keeps its concrete behavior or is fully abstracted (has free behavior) [Kurshan94]
  - Initially: unabstracted variables are those appearing in the **checked property**
- **Predicate abstraction**: concrete states are grouped together according to the set of predicates they satisfy [GS97, SS99]
  - Initially: predicates are extracted from the program's **control flow** and the **checked property**

# Abstraction Example

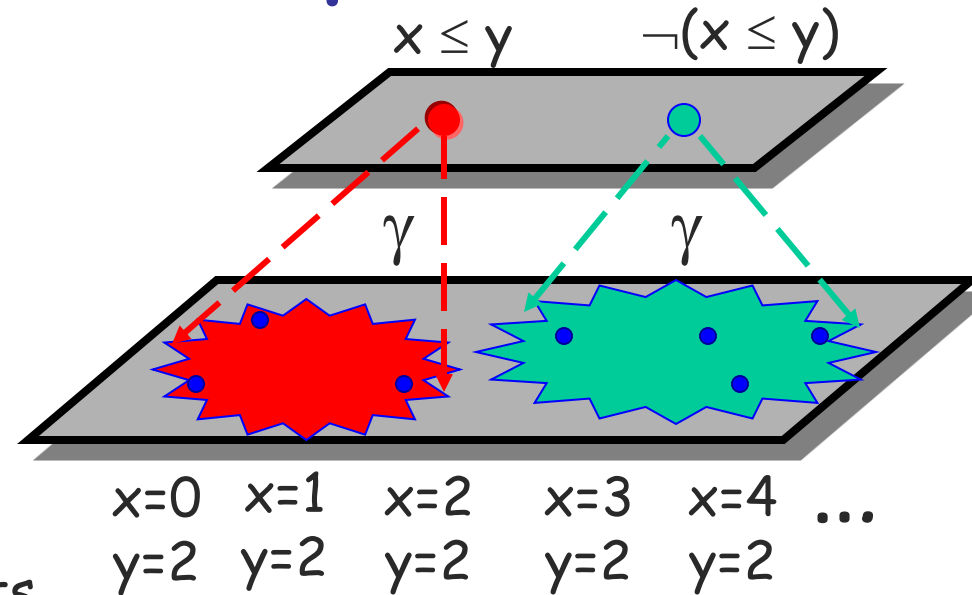
Abstract states

Concrete states



# Abstraction Example

- **Abstraction**  $(S_A, \gamma)$ :
  - Finite set of abstract states  $S_A$
  - Abstraction mapping  $\gamma : S_A \rightarrow 2^{Sc}$   
not necessarily disjoint sets



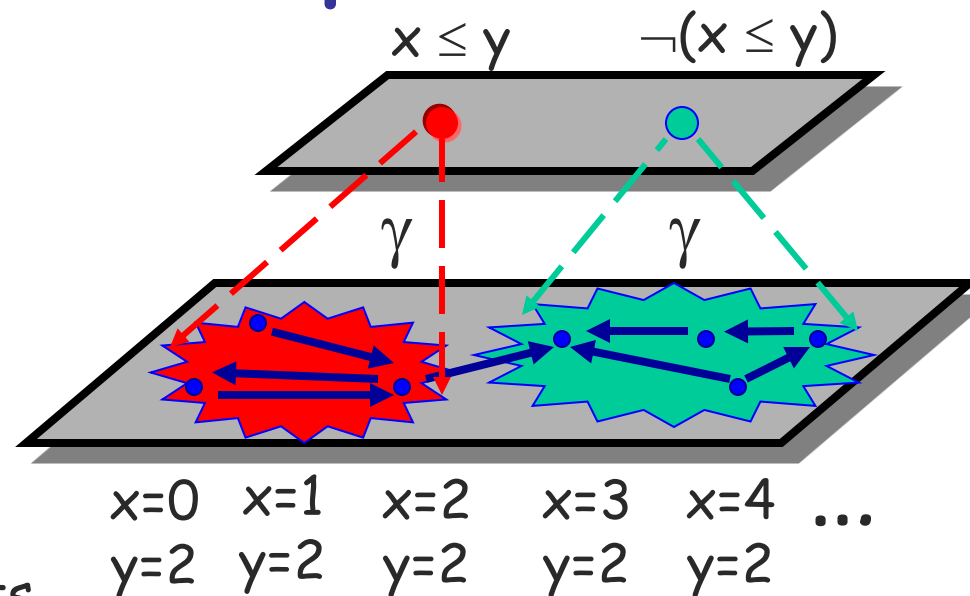
# Abstraction Example

- **Abstraction**  $(S_A, \gamma)$ :

- Finite set of abstract states  $S_A$
- Abstraction mapping

$$\gamma : S_A \rightarrow 2^{S_C}$$

not necessarily disjoint sets



- **Concrete** Kripke structure  $M_C = (S_C, R_C, L_C)$

➡ **Abstract model** over  $S_A$ : labeling, transitions

Need to be conservative



# Why Conservative?

Goal:

- Model check  $M_A$  instead of  $M_C$
- **Deduce** result over  $M_C$  from result over  $M_A$

What can we deduce?

- true ?
- false ?
- Both ?

For which properties?

Depends on abstract model and abstract semantics

# 2-valued CounterExample-Guided Abstraction Refinement (CEGAR)

For ACTL\*

[CGJLV00, JACM2003]

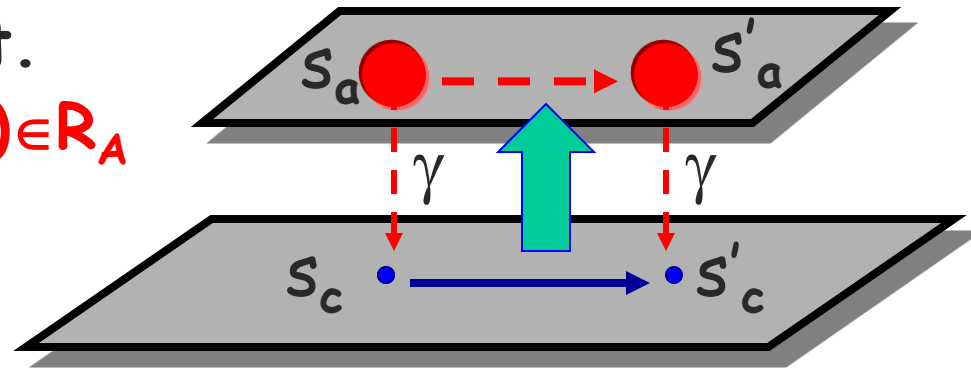
# Abstraction preserving ACTL/ACTL\*

## Existential Abstraction:

every concrete transition is represented by an abstract transition

If  $\exists s_c \in \gamma(s_a) \exists s'_c \in \gamma(s'_a)$  s.t.  
 $(s_c, s'_c) \in R_C$  then  $(s_a, s'_a) \in R_A$

Formally: simulation



The abstract model is an **over-approximation** of the concrete model:

- The abstract model has **more behaviors**
- no concrete behavior is lost

# Simulation Relation

$H \subseteq S_C \times S_A$  is a **simulation relation** from  $M_C$  to  $M_A$  if whenever  $(s_c, s_a) \in H$ :

- $L_C(s_c) \supseteq L_A(s_a)$
- If  $(s_c, s'_c) \in R_C$  for some  $s'_c$ , then there exists  $s'_a$  s.t.  $(s_a, s'_a) \in R_A$  and  $(s'_c, s'_a) \in H$

If there exists a simulation relation obeying the initial states, then  $M_C \leq_{\text{sim}} M_A$

# Existential Abstraction

- Abstract model is also a **Kripke structure**
  - **Same semantics** is used for abstract and concrete models
- Same model checking algorithms

Abstract model checking result is **true** or **false**  
(2-valued)

But... what can we **deduce**?

# Logic Preservation Theorem

**Theorem.** Let  $M_C \leq_{\text{sim}} M_A$ . Then:

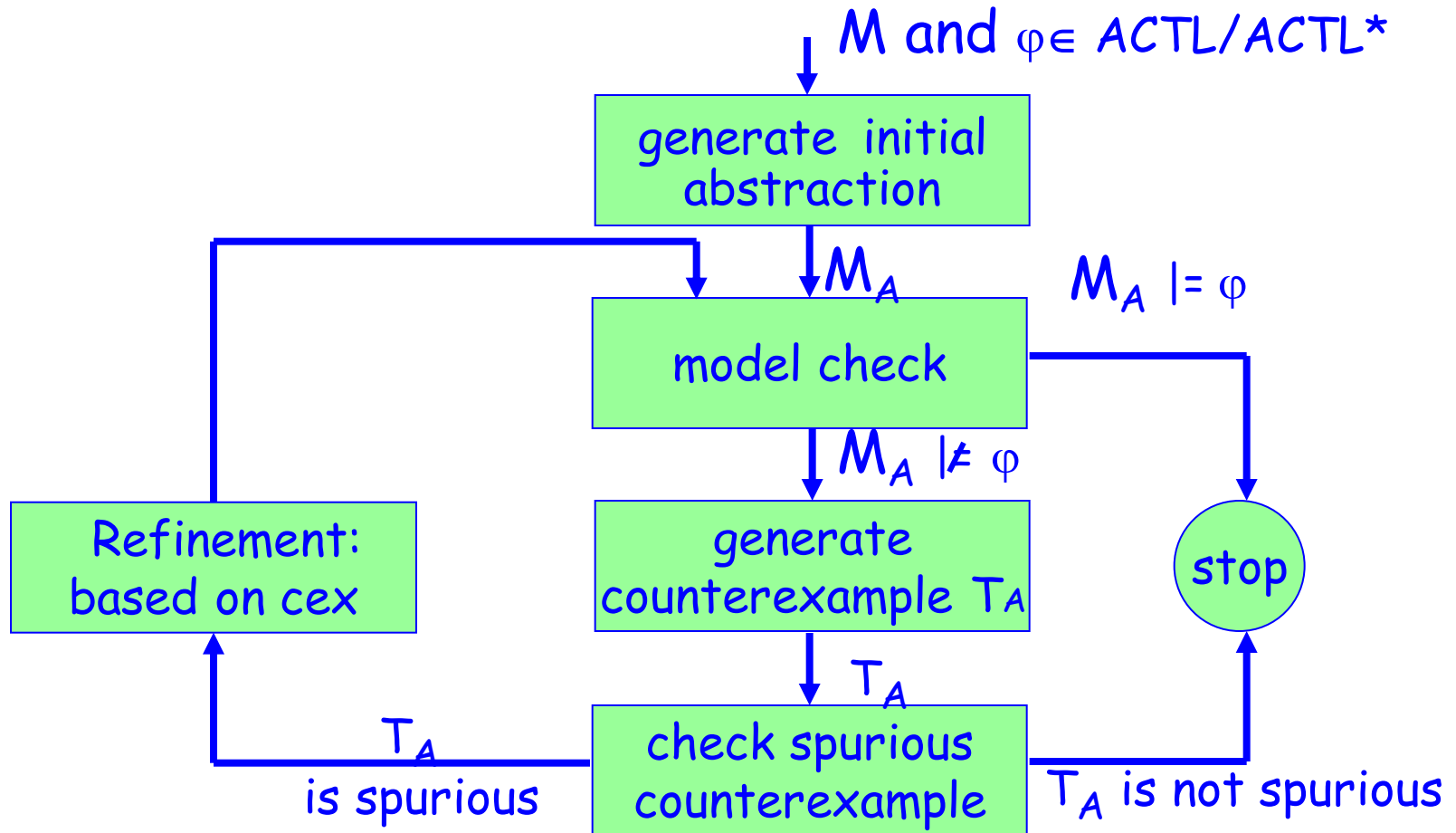
- Every ACTL/ACTL\* property **true** in the abstract model is also **true** in the concrete model:

$$M_A \models \varphi \Rightarrow M_C \models \varphi$$

However, the reverse may not be valid:

- If  $M_A \not\models \varphi$ , need to check further
  - Check if abstract counterexample is spurious

# CEX Guided Abstraction Refinement



# Three-Valued Abstraction Refinement (TVAR) for Full CTL\*

[SG03, GLLS05]



## 2-valued Approach is not Applicable

- Over-approximation (simulation) of the concrete model is not sound for verification of existential properties:

$$M_A \models E\psi \text{ does not imply } M_C \models E\psi$$

→ More complex abstract models (and relations) are needed to ensure logic preservation

# Abstract Models for CTL\*

**Branching-time** temporal logics combining **existential (E)** and **universal (A)** quantifiers:

→ two transition relations [LT88]

- **Rmay**: an **over-approximation**
- **Rmust**: an **under-approximation**

**Rmay** used to **verify**  $A\psi$  ... and **falsify**  $E\psi$

**Rmust** used to **verify**  $E\psi$  ... and **falsify**  $A\psi$

# Logic Preservation for CTL\*

If  $M_A$  is an abstraction of  $M_C$  then for every CTL\* formula  $\varphi$ ,

$$M_A \models \varphi \Rightarrow M_C \models \varphi$$

$$M_A \not\models \varphi \Rightarrow M_C \not\models \varphi$$

- But sometimes  $[M_A \models \varphi] = \text{don't know}$

## → 3-Valued Semantics

3 possible values: True, False,  $\perp$  (indefinite)

# Refinement

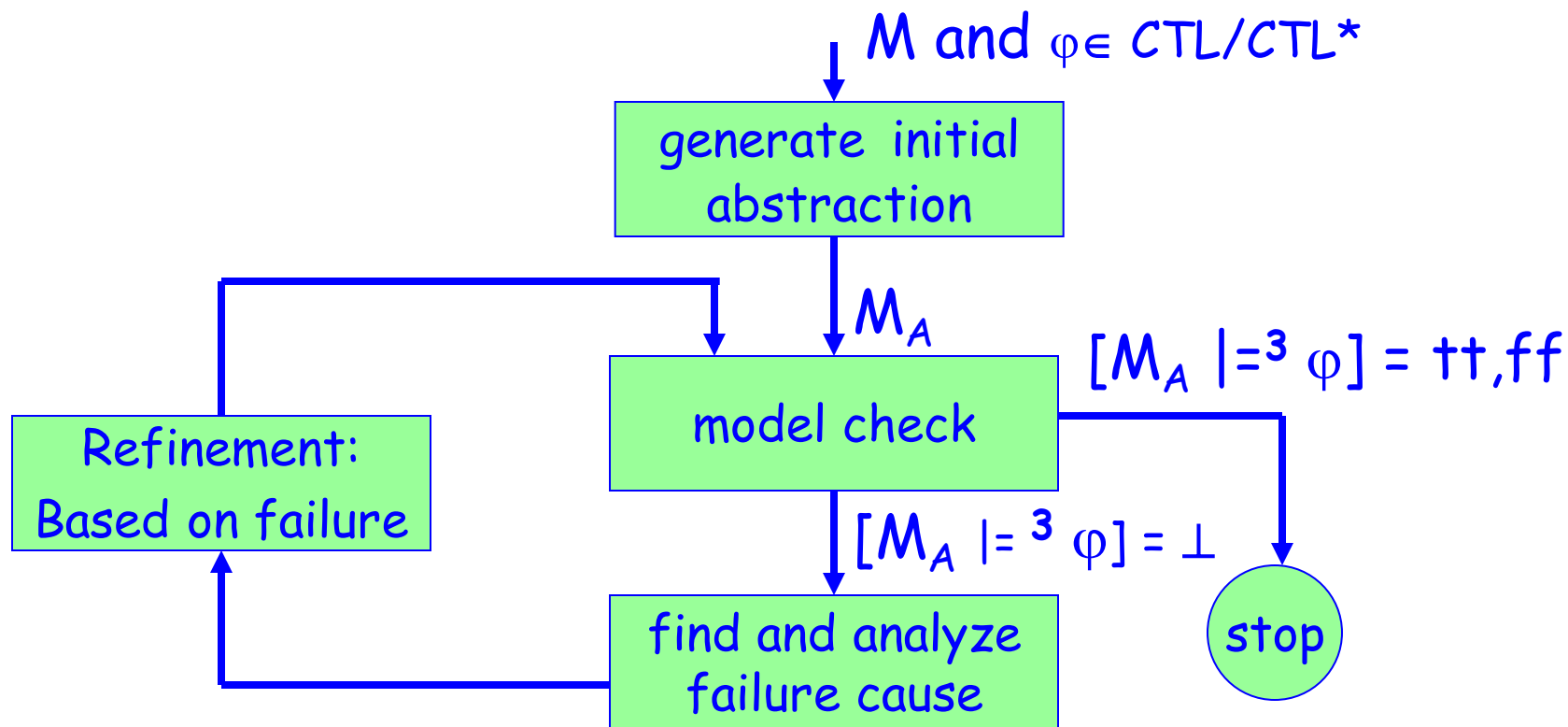
- Refinement is needed when result is  $\perp$

Traditional abstraction-refinement for universal properties **not** applicable:

- Refinement needed when result is **false**
- Based on a **counterexample**

## Three-Valued Abstraction-Refinement (TVAR)

# The TVAR Methodology



# Main Components

## 1. Abstract Models:

What formalism is suitable?

How to construct an abstract model in a conservative way?

## 2. Model Checking:

How to evaluate **branching-time** formulas over abstract models based on the **3-valued semantics**?

## 3. Refinement:

How to refine the abstract model?

# TVAR for CTL using Kripke Modal Transition Systems

[SG03]

# Abstract Models

## Kripke Modal Transition System (KMTS) [HJS01]

- $M = (AP, S, s^0, R_{\text{must}}, R_{\text{may}}, L)$ 
  - $R_{\text{must}} \subseteq S \times S$ : an **under-approximation**
  - $R_{\text{may}} \subseteq S \times S$ : an **over-approximation**
  - $R_{\text{must}} \subseteq R_{\text{may}}$



For simplicity.  
In MixTS, no such requirement



# Abstract Models

## Labeling function :

- $L: S \rightarrow 2^{\text{Literals}}$
- $\text{Literals} = AP \cup \{\neg p \mid p \in AP\}$
- **At most** one of  $p$  and  $\neg p$  is in  $L(s)$ .
  - Concrete: **exactly** one of  $p$  and  $\neg p$  is in  $L(s)$ .
  - KMTS: possibly **none** of them is in  $L(s)$ ,  
meaning that the value of  $p$  in  $s$  is **unknown**

## 3-Valued Semantics [BG99]

$[[\text{lit}]](s) = \text{tt}$  if  $\text{lit} \in L_A(s)$ ,  $\text{ff}$  if  $\neg \text{lit} \in L_A(s)$ ,  $\perp$  o.w.

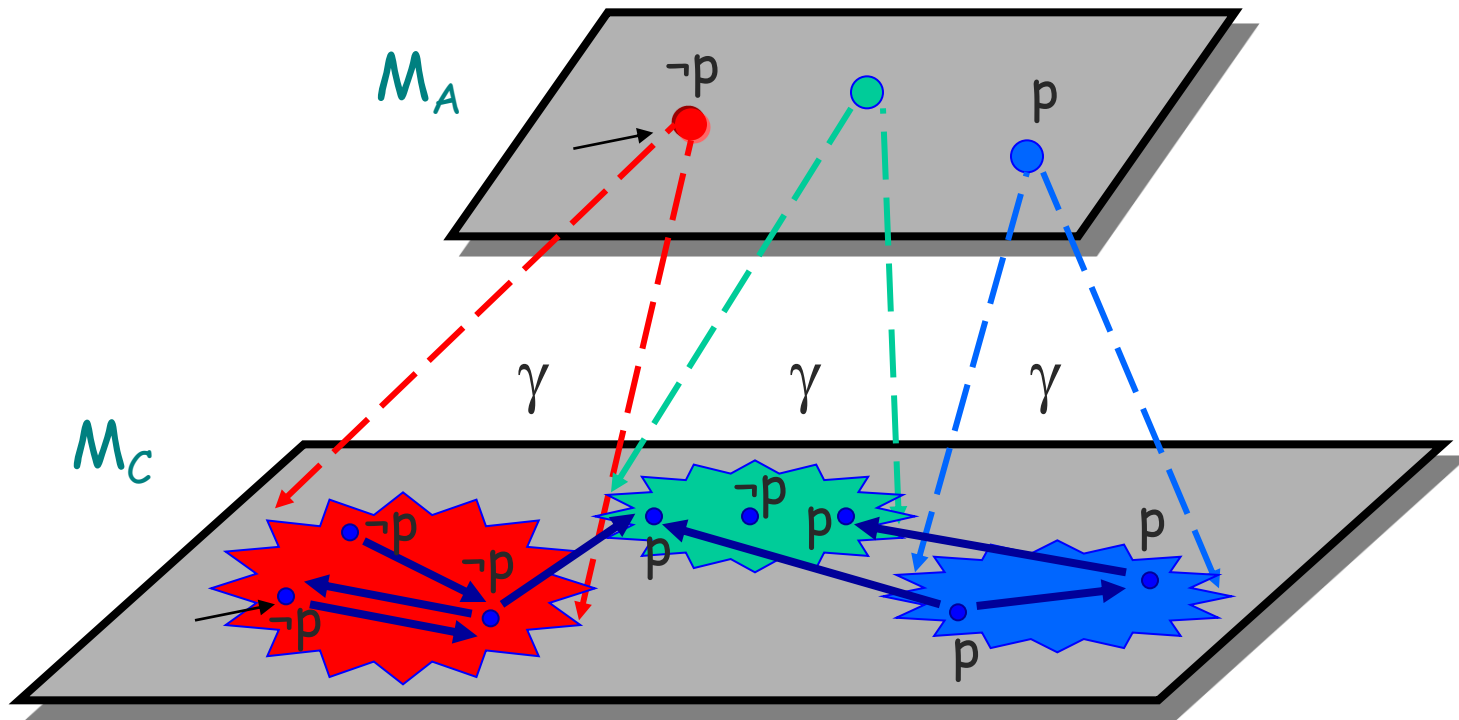
$[[AX\psi]](s) = \begin{cases} \text{tt} & \text{if forall } s', \text{ if } (s, s') \in \text{Rmay}, \\ & \text{then } [[\psi]](s') = \text{tt} \\ \text{ff} & \text{if exists } s' \text{ s.t. } (s, s') \in \text{Rmust} \\ & \text{and } [[\psi]](s') = \text{ff} \\ \perp & \text{otherwise} \end{cases}$   
 "all succ. satisfy  $\psi$ "

$[[EX\psi]](s)$  - dual

"exists succ. satisfying  $\psi$ "

# Construction of Abstract Model

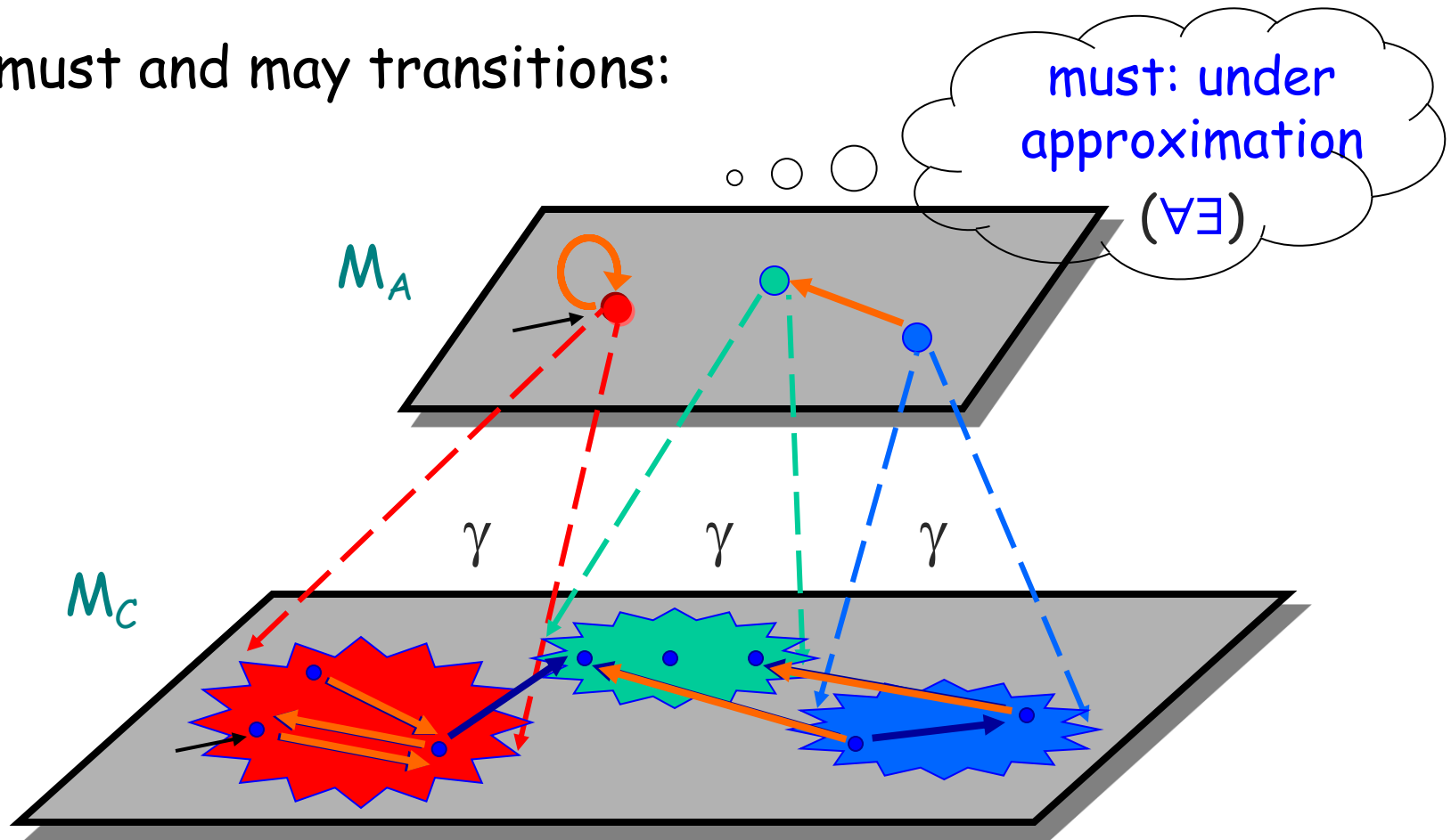
Labeling of abstract states



$$[ \forall s_c \in \gamma(s_a) \text{ lit} \in L_c(s_c) ] \Leftrightarrow \text{lit} \in L_A(s_a)$$

# Construction of Abstract Model

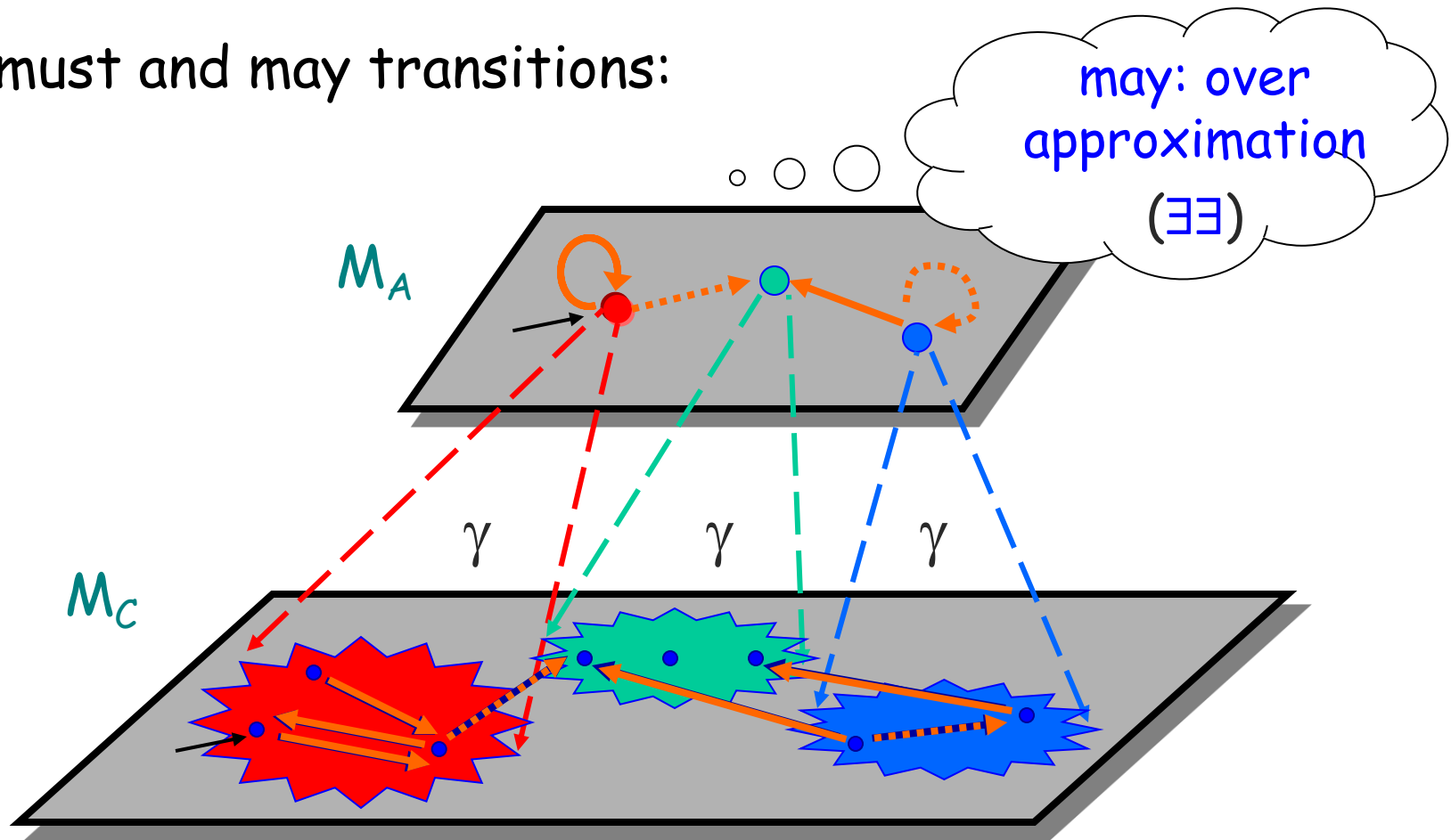
must and may transitions:



$$[\forall s_c \in \gamma(s_a) \exists s'_c \in \gamma(s'_a) \text{ s.t. } (s_c, s'_c) \in R_c] \Leftrightarrow (s_a, s'_a) \in R_{\text{must}}$$

# Construction of Abstract Model

must and may transitions:



$$[\exists s_c \in \gamma(s_a) \exists s'_c \in \gamma(s'_a) \text{ s.t. } (s_c, s'_c) \in R_c] \Leftrightarrow (s_a, s'_a) \in R_{\text{may}}$$

# Mixed Simulation

$H \subseteq S_C \times S_A$  is a **mixed simulation relation** from Kripke structure  $M_C$  to KMTS  $M_A$  if whenever  $(s_c, s_a) \in H$ :

- $L_C(s_c) \supseteq L_A(s_a)$
- If  $(s_c, s'_c) \in R_C$ , then there is  $s'_a$  s.t.  $(s_a, s'_a) \in R_{\text{may}}$  and  $(s'_c, s'_a) \in H$
- If  $(s_a, s'_a) \in R_{\text{must}}$ , then there is  $s'_c$  s.t.  $(s_c, s'_c) \in R_C$  and  $(s'_c, s'_a) \in H$

If there exists a mixed simulation relation obeying the initial states, then  $M_C \leq_{\text{mix}} M_A$

# Logic Preservation

## Theorem.

Let  $M_C \leq_{\text{mix}} M_A$ . Then:

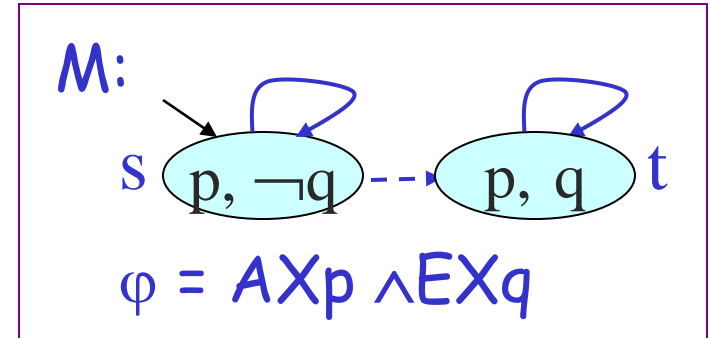
For every CTL\* formula  $\varphi$ ,

$$M_A \models \varphi \Rightarrow M_C \models \varphi$$

$$M_A \not\models \varphi \Rightarrow M_C \not\models \varphi$$

But if  $[M_A \models \varphi] = \perp$ , the value in  $M_C$  is unknown

# 3-Valued Model Checking Example



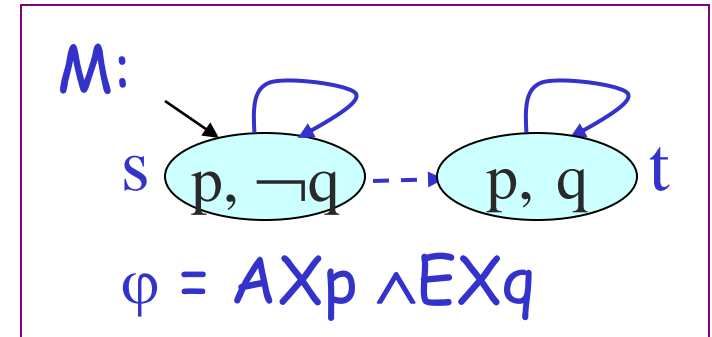


# 3-Valued Model Checking Example



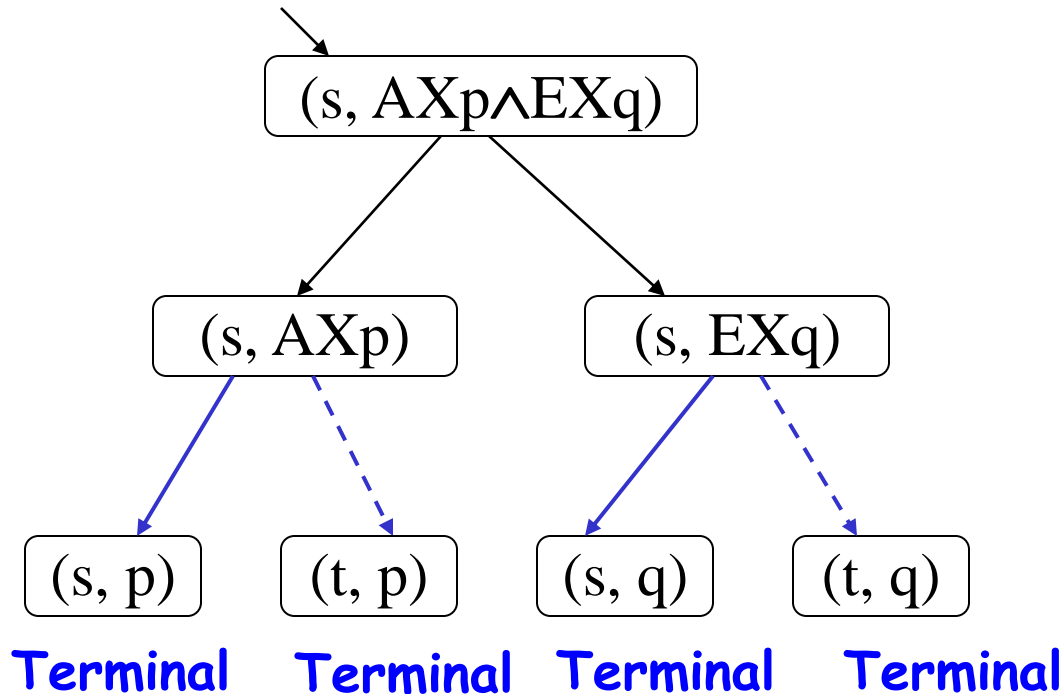
state of  
the model

formula that  
we want to  
evaluate in  $s_0$

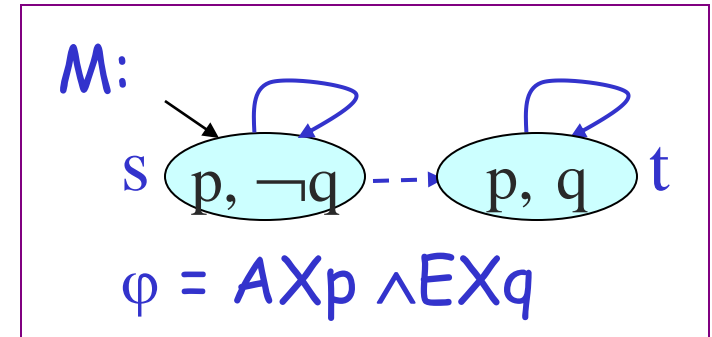


MC graph

# 3-Valued Model Checking Example

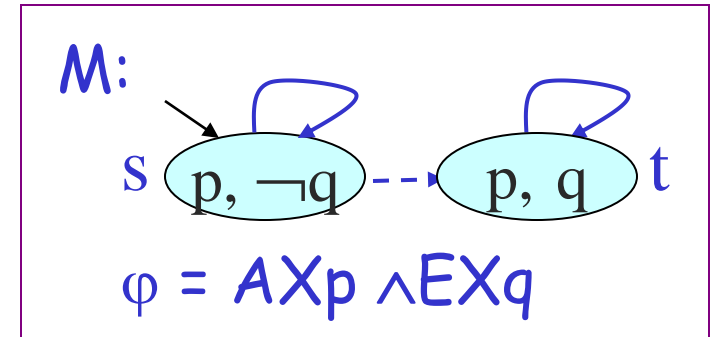
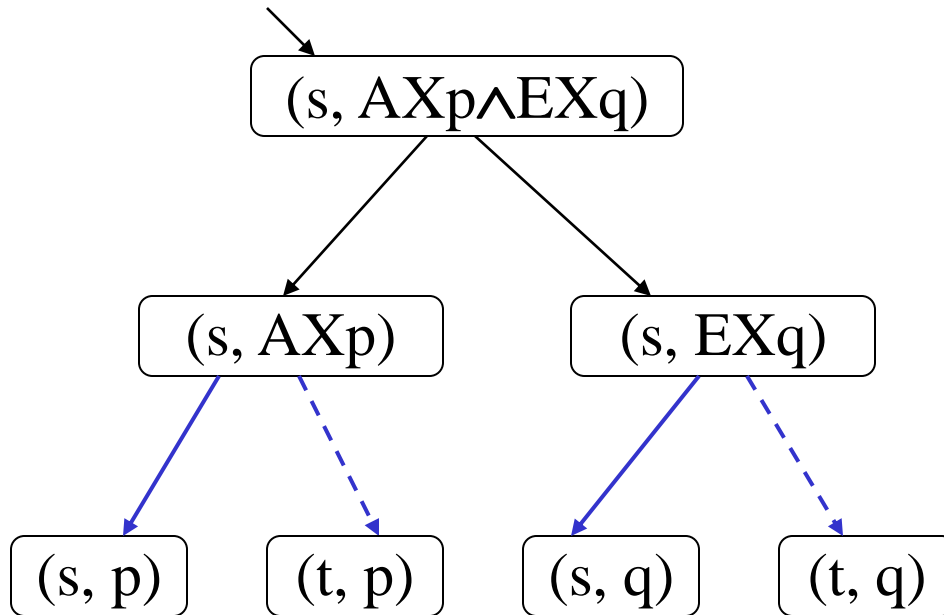


MC graph



■  $\dagger\dagger$   
■  $ff$   
■  $\perp$

## Coloring the graph



← Model's transitions  
 need to consider:  
 • may vs. must

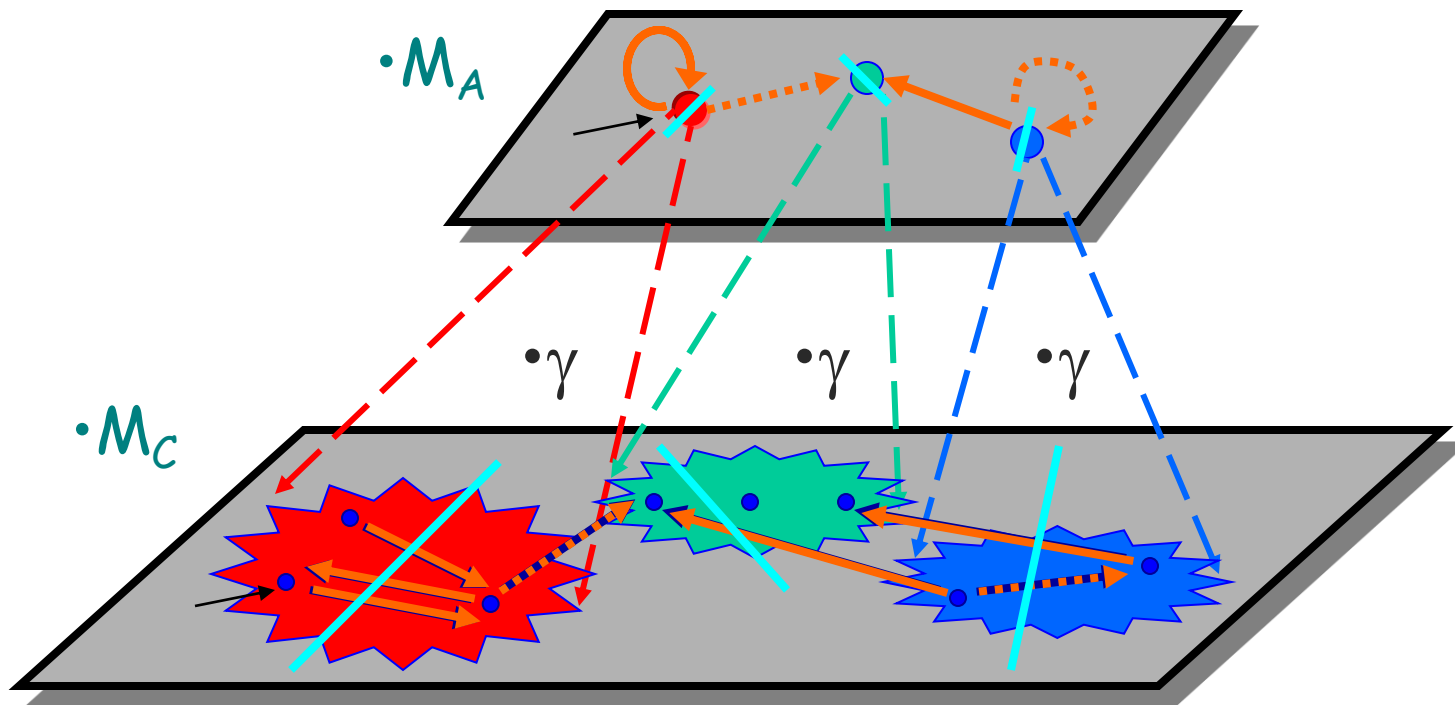
**Terminal nodes:** based on states labeling  
 **$\wedge, \vee, AX, EX$ :** according to sons, based on semantics

## 3-Valued Model Checking Results

- $\mathbf{tt}$  and  $\mathbf{ff}$  are **definite**: hold in the concrete model as well.
- $\perp$  is **indefinite**  
 $\Rightarrow$  Refinement is needed.

# Refinement

- done by **splitting** abstract states  
(as for the case of 2-values)

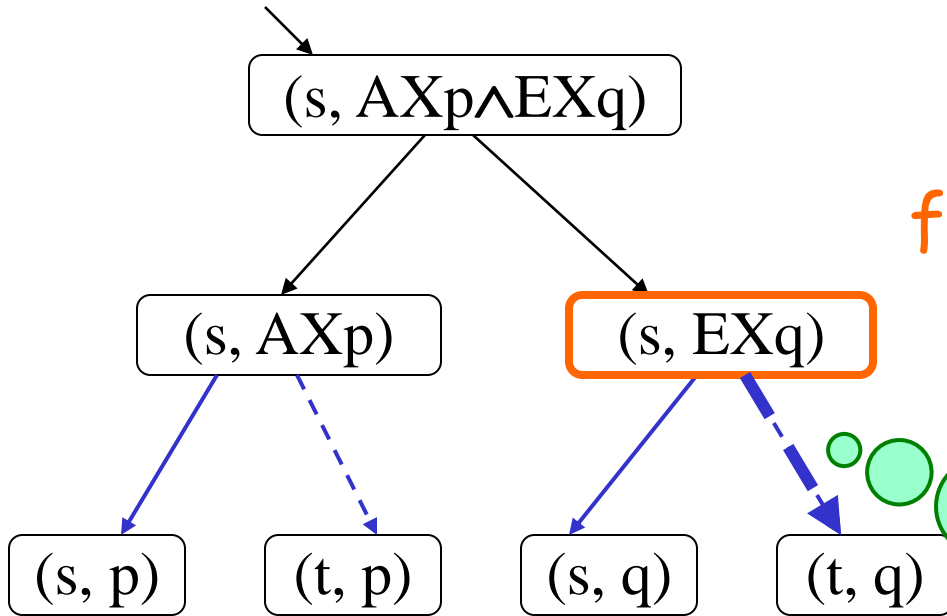


# Refinement

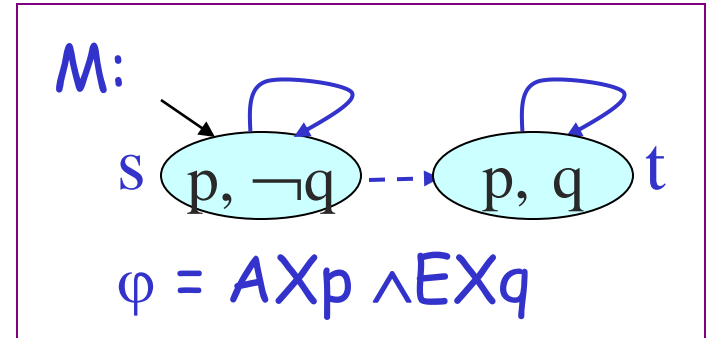
- Uses the colored MC graph
- Find a **failure node**  $n_f$ :
  - a node colored **⊥** whereas none of its sons was colored **⊥** at the time it got colored.
  - the point where certainty was lost
- purpose: change the **⊥** color of  $n_f$ .

# Example

■  $\dagger\dagger$   
■  $ff$   
■  $\perp$



failure



reason for failure:  
may-son

- not enough to verify
- prevents refutation



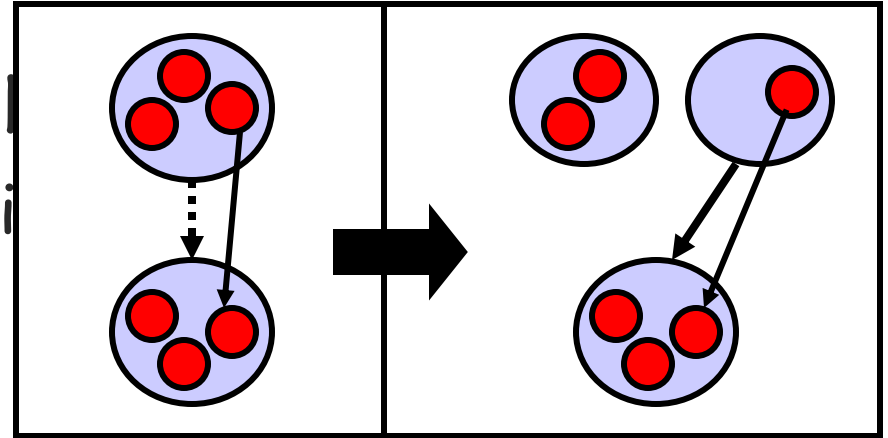
# Failure Reason

- Failure reason is either:
  - A **may-edge** which is **not** a must-edge.
  - A **⊥**-terminal node
- Back in the model:
  - Either a transition  $(s, s') \in R_{\text{may}} \setminus R_{\text{must}}$ :
    - Split **s** to get a must-transition or none.



# Failure Reason

- Failure reason is either
  - A **may-edge** which is
  - A **⊥**-terminal node



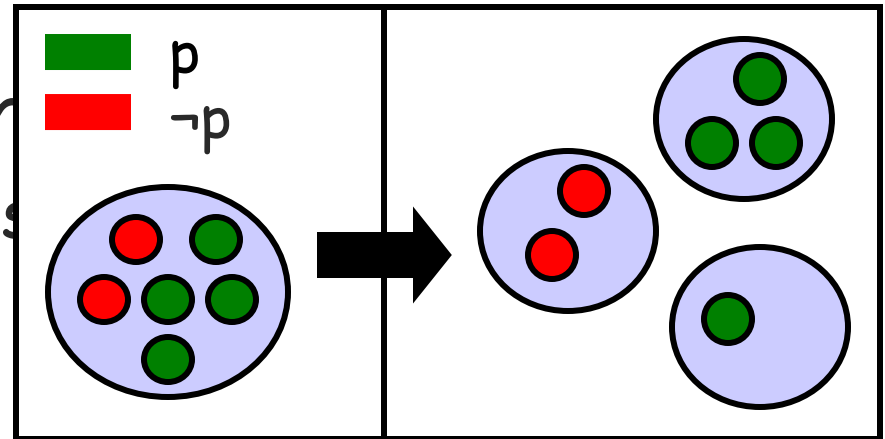
- Back in the model:
  - Either a transition  $(s, s') \in R_{\text{may}} \setminus R_{\text{must}}$ :
    - Split **s** to get a must-transition or none.

# Failure Reason

- **Failure reason** is either:
  - A **may-edge** which is **not** a must-edge.
  - A  **$\perp$** -terminal node
- **Back in the model:**
  - Either a transition  $(s, s') \in R_{\text{may}} \setminus R_{\text{must}}$ :
    - Split **s** to get a must-transition or none.
  - Or  $(s, \text{lit})$  where  $\text{lit} \notin L(s)$ ,  $\neg \text{lit} \notin L(s)$ 
    - Split **s** according to lit.

# Failure Reason

- Failure reason is either
  - A **may-edge** which is
  - A  **$\perp$ -terminal node**



- Back in the model:
  - Either a transition  $(s, s') \in R_{\text{may}} \setminus R_{\text{must}}$ :
    - Split **s** to get a must-transition or none.
  - Or  $(s, \text{lit})$  where  $\text{lit} \notin L(s)$ ,  $\neg \text{lit} \notin L(s)$ 
    - Split **s** according to lit.

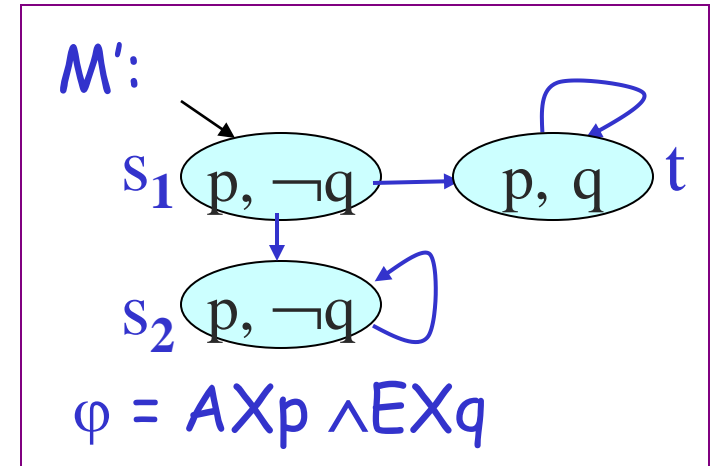
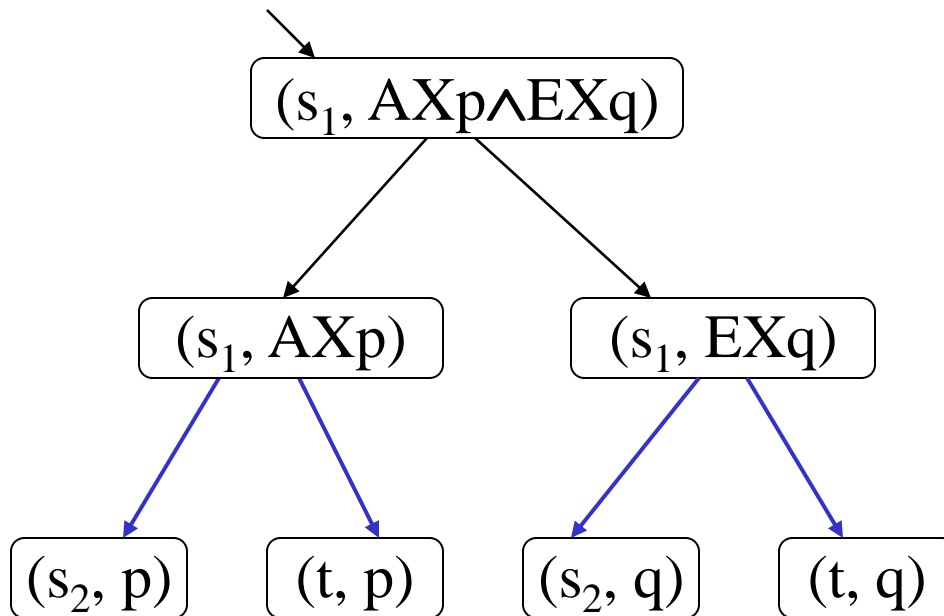
# Split

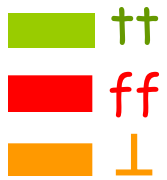
- Refinement is reduced to **separating sets of concrete states**.
  - done by known techniques [CGJLV00,CGKS02]

→ **Refined abstraction mapping.**

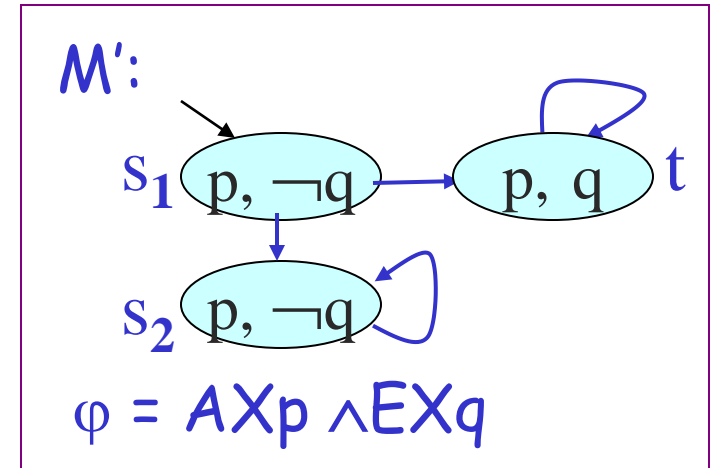
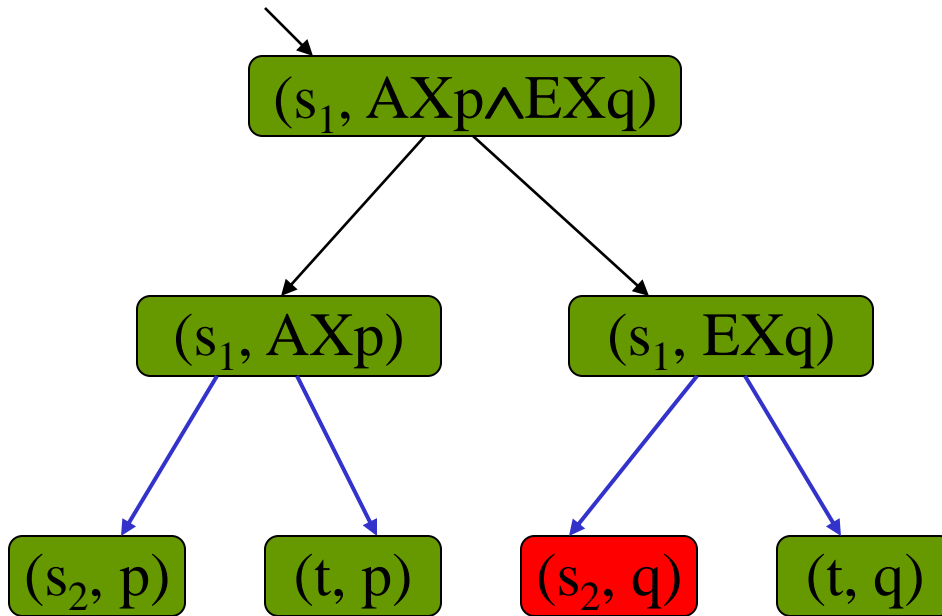
- Build **refined abstract model** and **refined MC-graph** accordingly.

## Example (cont.)





## Example (cont.)

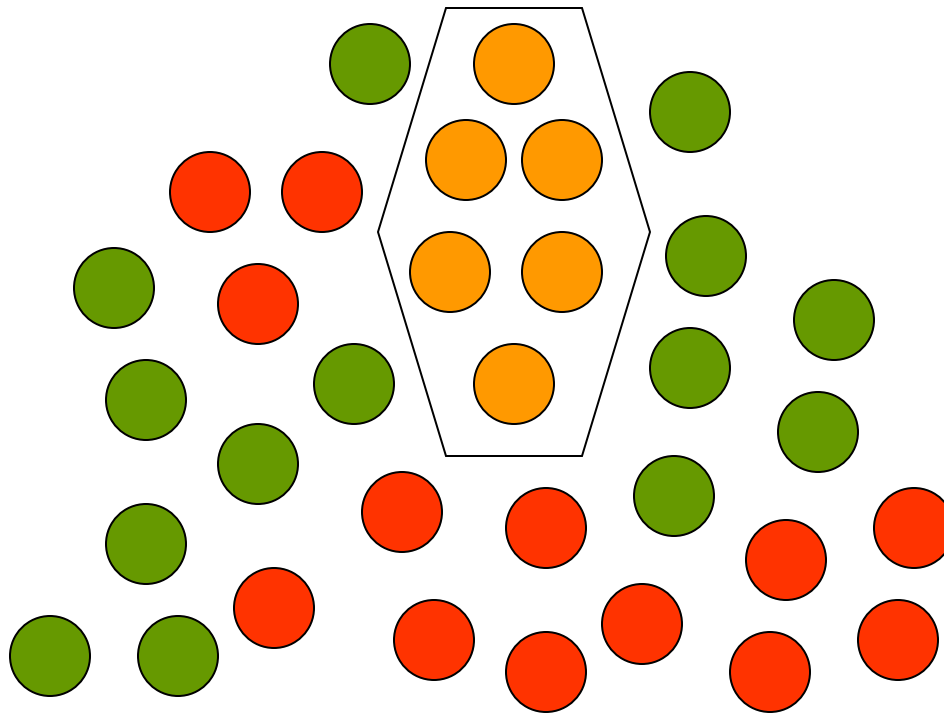


# Incremental Abstraction-Refinement

No reason to split states for which MC results are definite during refinement.

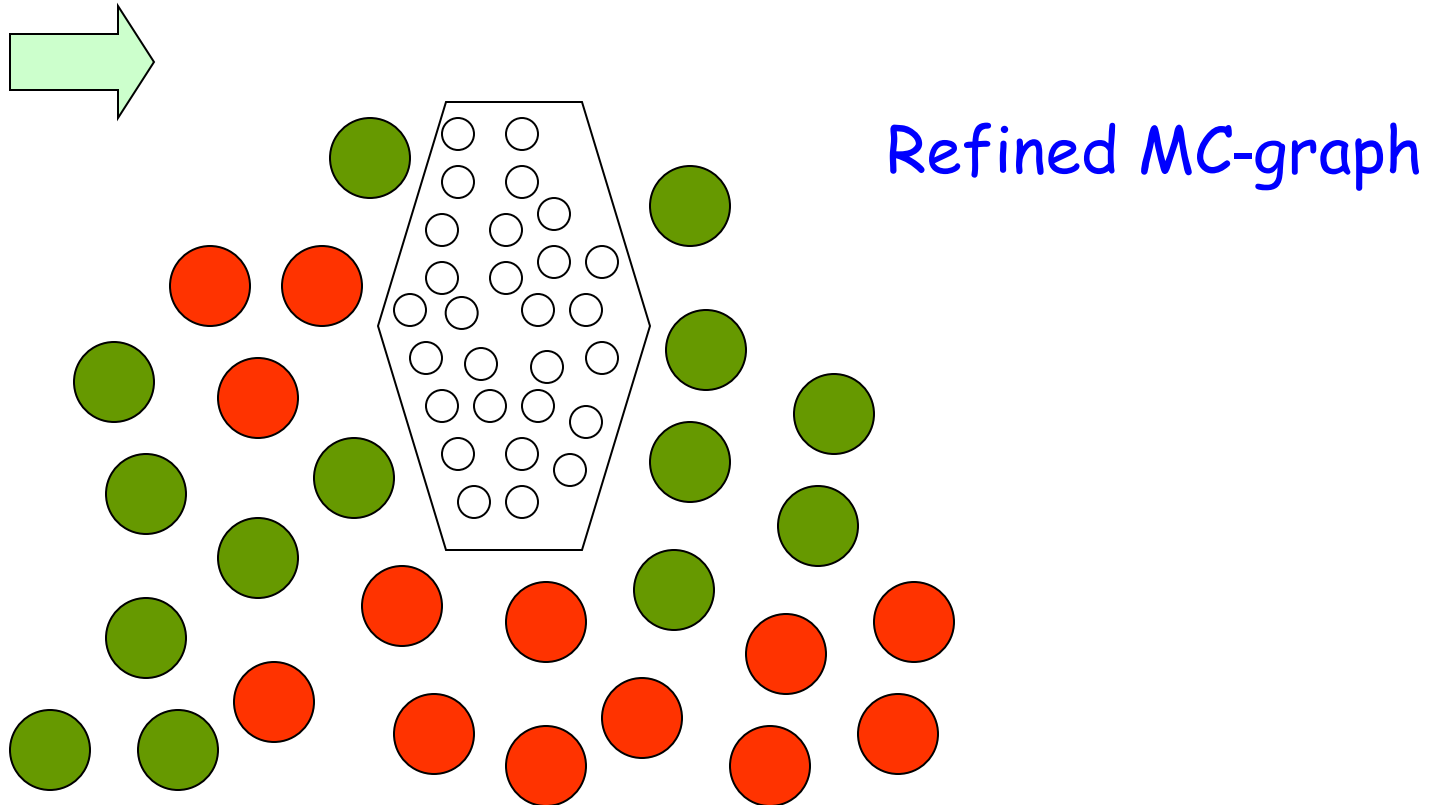
- After each iteration **remember** the nodes colored by **definite** colors.
- **Prune** the refined MC graph in **sub-nodes** of remembered nodes.  
[  $(s_a, \varphi)$  is a **sub-node** of  $(s_a', \varphi')$  if  $\varphi = \varphi'$  and  $\gamma(s_a) \subseteq \gamma'(s_a')$  ]
- Color such nodes by their **previous** colors.

# Example

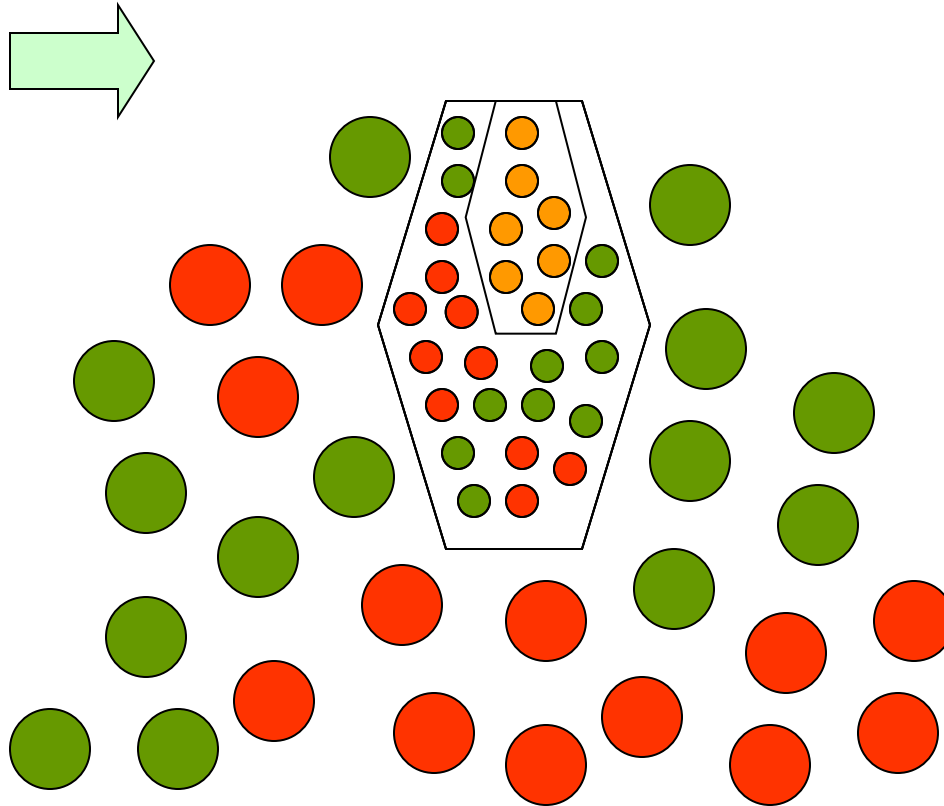




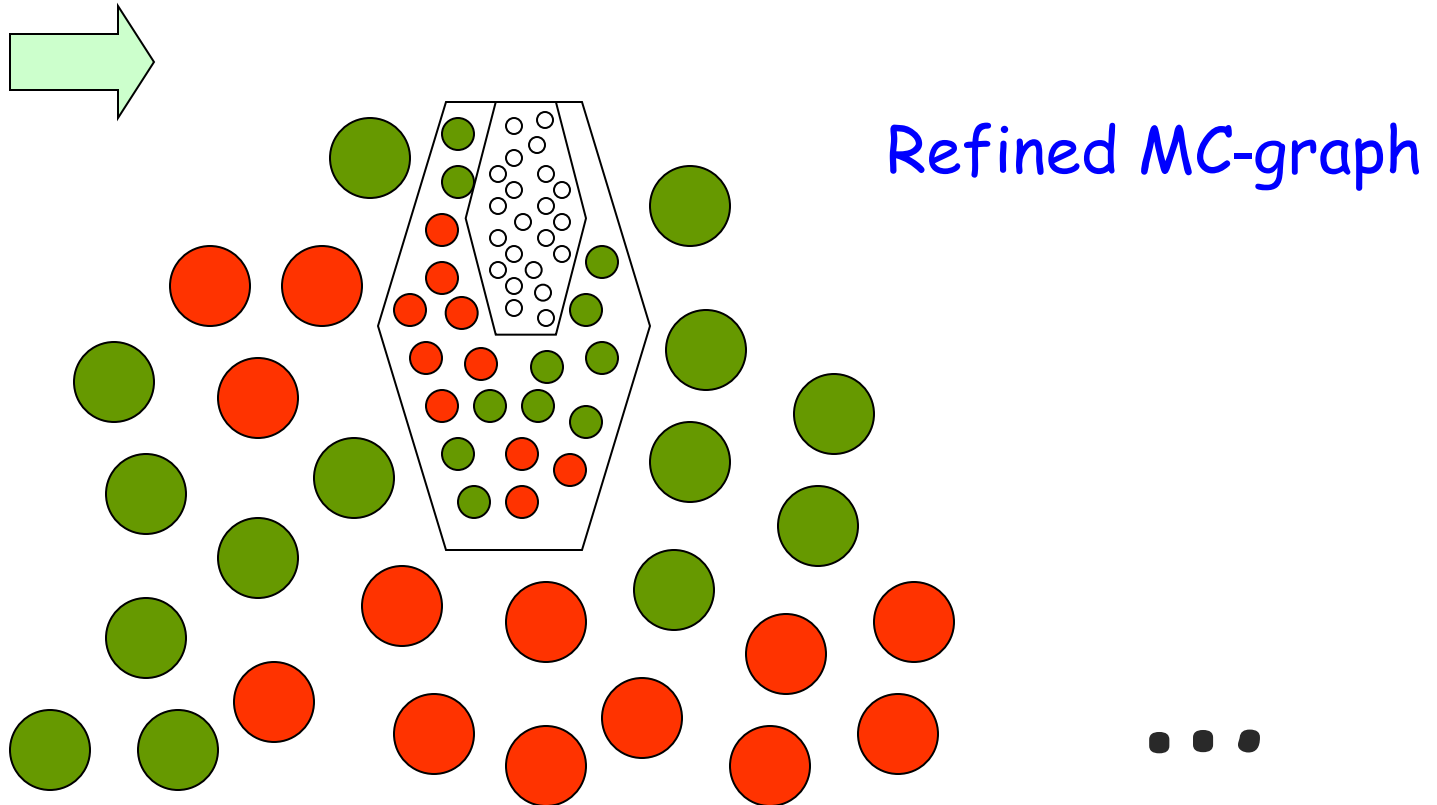
## Example (cont.)



## Example (cont.)



## Example (cont.)



Are KMTSs good enough for TVAR?

# Investigation of Abstract Models

- Monotonicity of Refinement
- Precision
- Completeness
- Efficiency

# (1) Monotonicity of Refinement

Is a **refined** abstract model **at least as precise** as the **unrefined** one?

# Example

$P ::$

input  $x > 0$

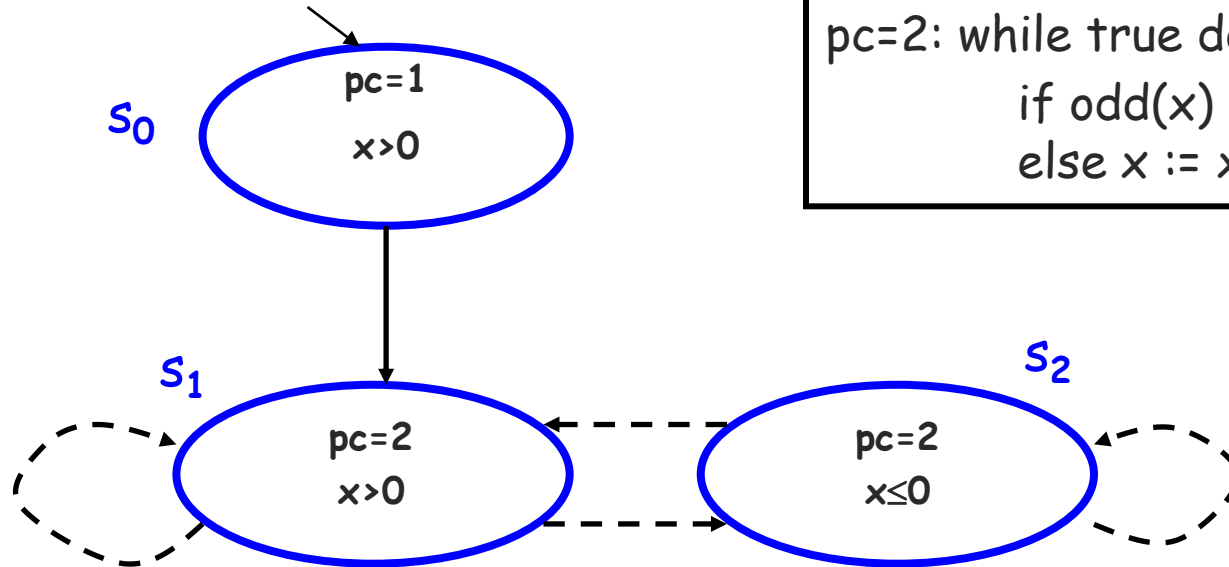
pc=1: if  $x > 5$  then  $x := x+1$  else  $x := x+2$

pc=2: while true do

    if odd( $x$ ) then  $x := -1$  else  $x := x+1$

$\phi = EF (x \leq 0)$

# An Abstract Model $M$

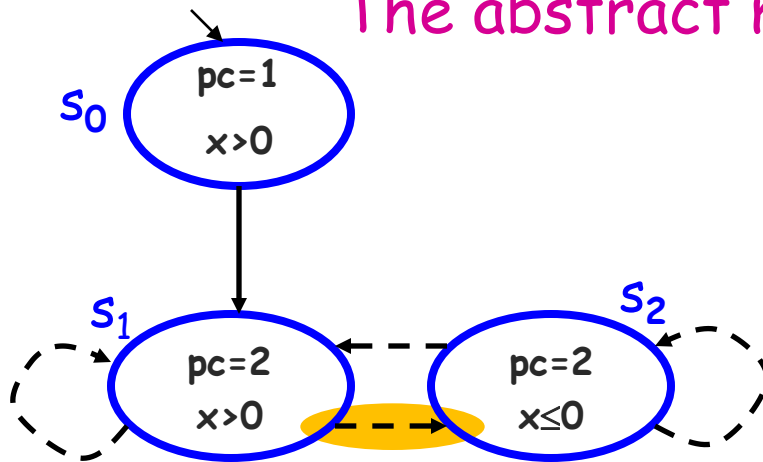


$P ::$   
input  $x > 0$   
 $pc=1$ : if  $x > 5$  then  $x := x+1$   
          else  $x := x+2$   
 $pc=2$ : while true do  
          if  $\text{odd}(x)$  then  $x := -1$   
          else  $x := x+1$

$$[ \text{EF } (x \leq 0) ] (M) = \perp$$

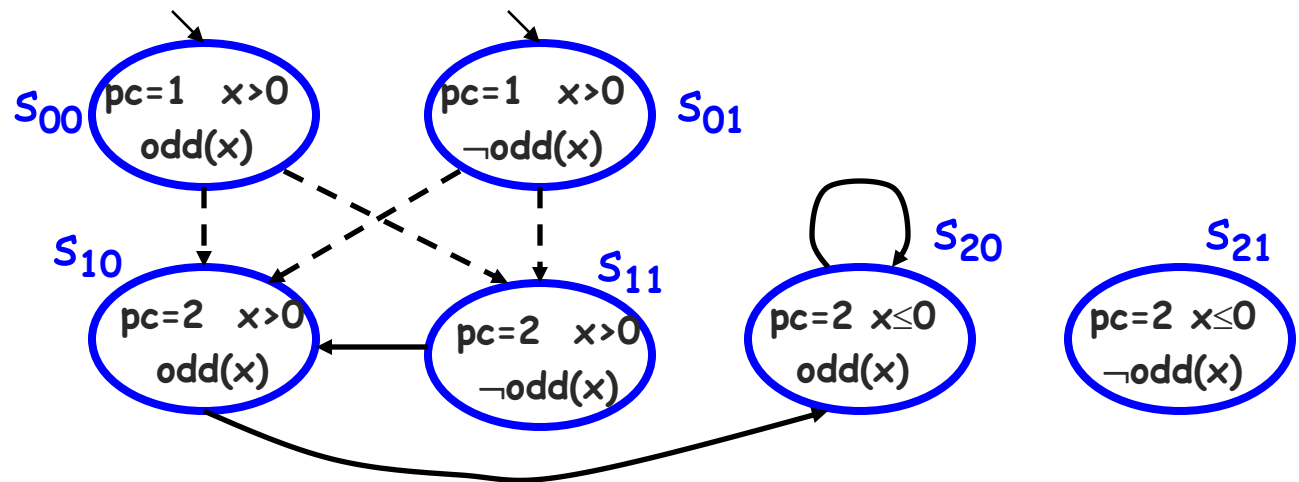


## The abstract model $M$

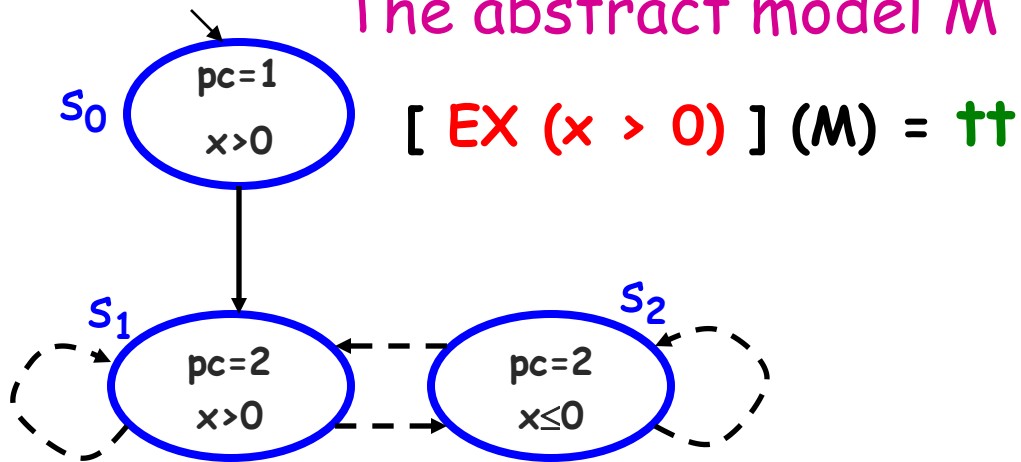


$P ::$   
input  $x > 0$   
 $pc=1$ : if  $x > 5$  then  $x := x+1$   
    else  $x := x+2$   
 $pc=2$ : while true do  
    if  $\text{odd}(x)$  then  $x := -1$   
    else  $x := x+1$

## A refinement $M'$ of $M$



## The abstract model $M$



$P ::$

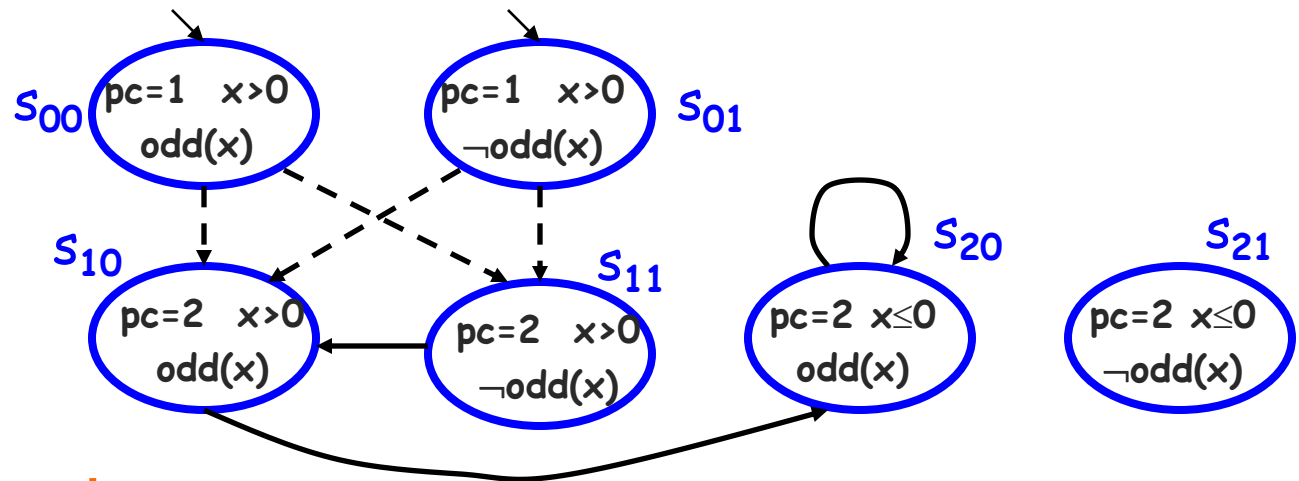
input  $x > 0$

$pc=1$ : if  $x>5$  then  $x := x+1$   
else  $x := x+2$

$pc=2$ : while true do  
if  $\text{odd}(x)$  then  $x := -1$   
else  $x := x+1$

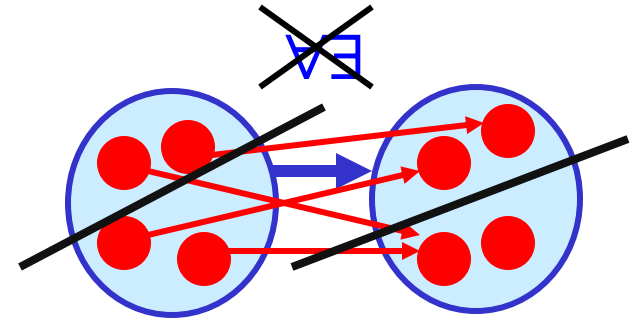
## A refinement $M'$ of $M$

$M' \not\leq_{CTL} M$



$[ EX (x > 0) ] (M') = \perp$

## Problem



- When splitting states during refinement we may **lose** must transitions
  - Existential formulas that were true before may become **indefinite !**  
(also universal formulas that were false)
  - Thus, the refined model is not necessarily **more precise**
- refinement is **not monotonic**

**Goal:** define a refinement that **adds** under-approximated **must** transitions

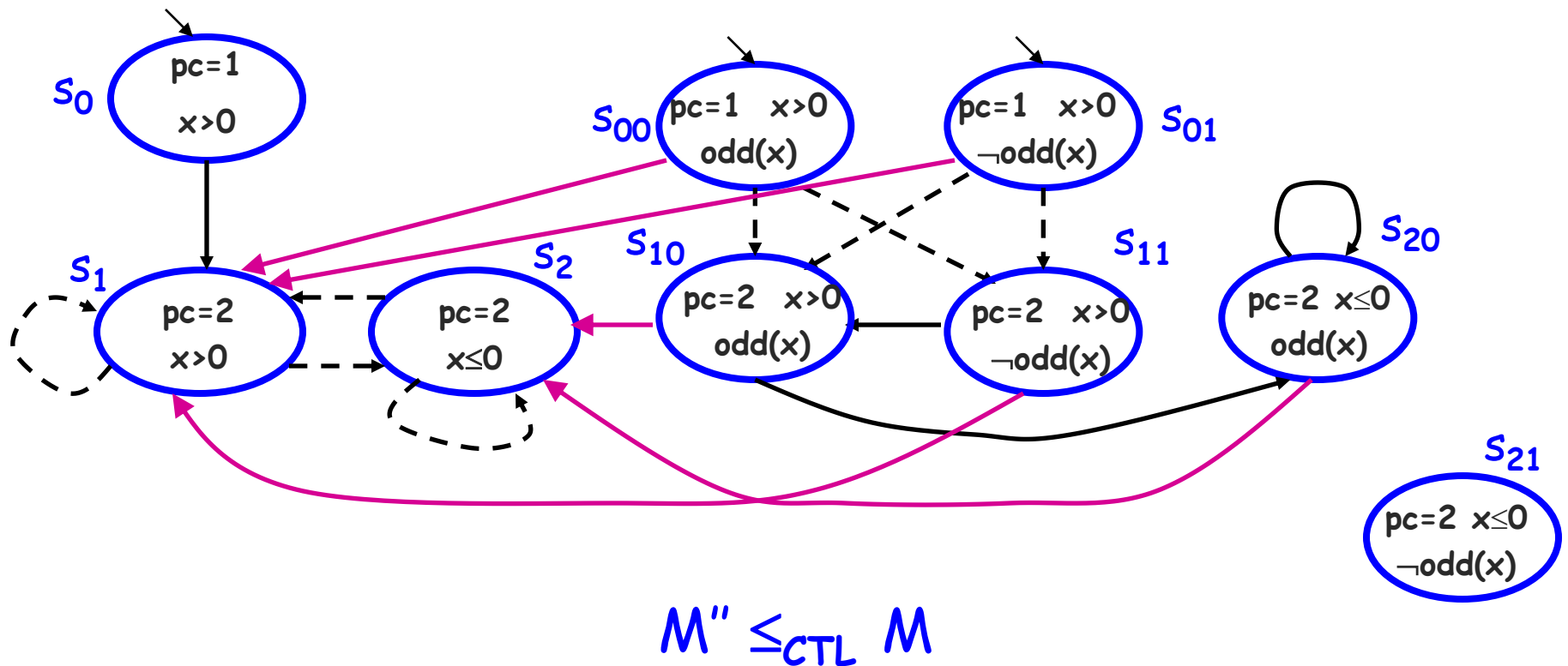
[current refinements **remove** over-approximated **may** transitions ]

**Result:** refined model will be **more precise**,  
i.e. more formulas will be definite (**tt** or **ff**)  
in it:

## Monotonic Refinement

**Notation:**  $M' \leq_{\text{CTL}} M$  :  $M'$  is more precise than  $M$

Refinement  $M''$  of  $M$ , according to Godefroid et. al.

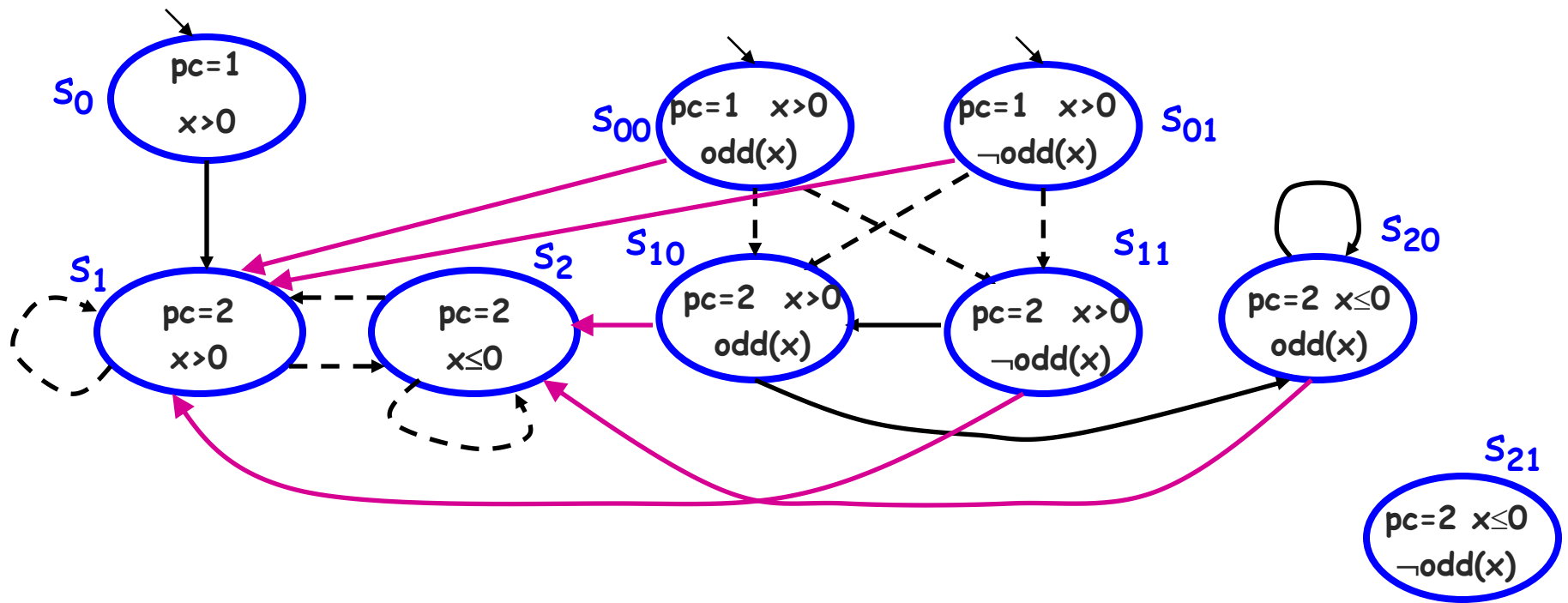


## (2) Precision

**Given** a state abstraction  $(S_A, \gamma)$

- “How many” formulas can be verified or falsified on the abstract model?

Refinement  $M''$  of  $M$ , according to Godefroid et. al.



$$M'' \leq_{CTL} M$$

$$[EF(x \leq 0)](M'') = \perp$$

## Another Solution [SG'04]

Use hyper-transitions as must transitions

**Hyper-transition** from a state  $s \in S$  is

- $(s, A)$  where  $A \subseteq S$  is nonempty



# Generalized KMTS (GTS)

$$M = (S, S_0, R_{\text{may}}, R_{\text{must}}, L)$$

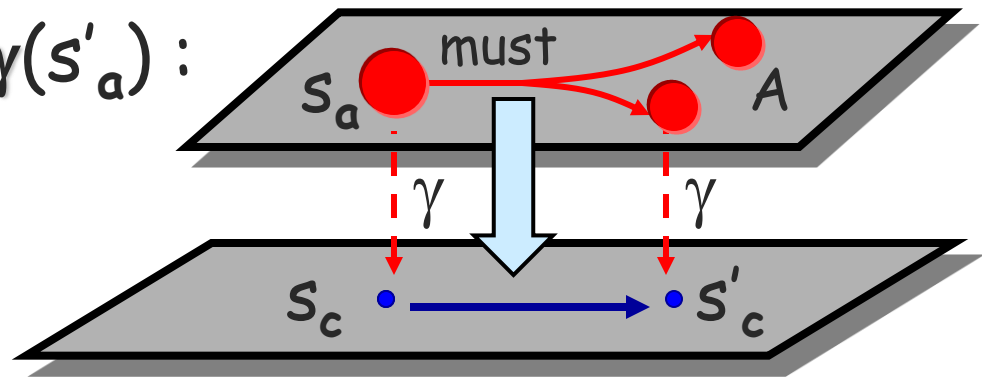
- $S, S_0, R_{\text{may}}, L$  as before
- $R_{\text{must}} \subseteq S \times 2^S$

# Constructing an Abstract GTS

Given  $M_C$ ,  $S_A$ , and  $\gamma : S_A \rightarrow 2^{S_C}$

- $(s_a, A) \in R_{\text{must}}$  only if  **$\forall \exists \exists$ -condition holds**:

$$\forall s_c \in \gamma(s_a) \exists s'_a \in A \exists s'_c \in \gamma(s'_a) : (s_c, s'_c) \in R_c$$



every state in  $\gamma(s_a)$  has a corresponding transition

Reminder: in **KMTS**:

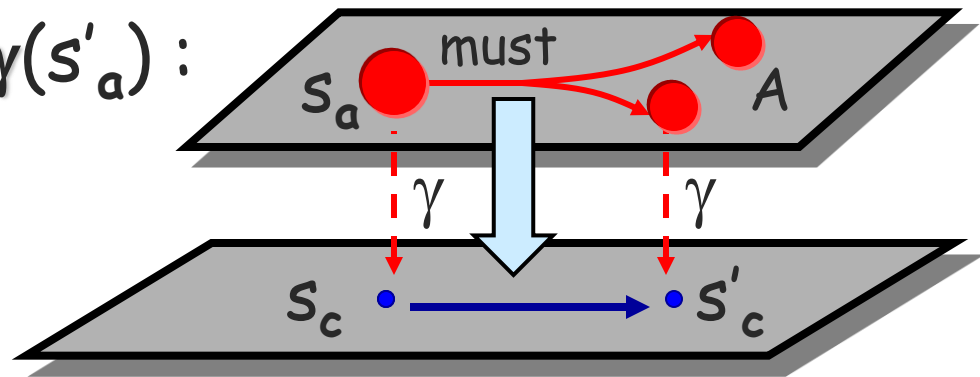
$(s_a, s'_a) \in R_{\text{must}}$  only if  **$\forall\exists$ -condition holds**:

$$\forall s_c \in \gamma(s_a) \exists s'_c \in \gamma(s'_a) : (s_c, s'_c) \in R_c$$

Given **M**

- $(s_a, A) \in R_{\text{must}}$  only if  **$\forall\exists\exists$ -condition holds**:

$$\forall s_c \in \gamma(s_a) \exists s'_a \in A \exists s'_c \in \gamma(s'_a) : (s_c, s'_c) \in R_c$$



every state in  $\gamma(s_a)$  has a corresponding transition

## 3-Valued Semantics over GTS

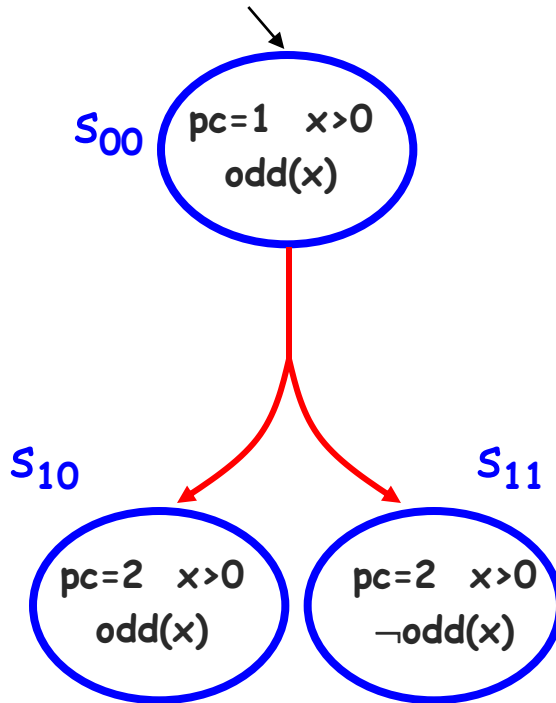
$[[\text{lit}]](s) = \text{tt}$  if  $\text{lit} \in L_A(s)$ ,  $\text{ff}$  if  $\neg \text{lit} \in L_A(s)$ ,  $\perp$  o.w.

$[[AX\psi]](s) = \begin{cases} \text{tt} & \text{if forall } s', \text{ if } (s, s') \in \text{Rmay}, \\ & \text{then } [[\psi]](s') = \text{tt} \\ \text{ff} & \text{if exists } A \subseteq S_A \text{ s.t. } (s, A) \in \text{Rmust} \\ & \text{and } [[\psi]](s') = \text{ff forall } s' \in A \\ \perp & \text{otherwise} \end{cases}$   
 "all succ. satisfy  $\psi$ "

$[[EX\psi]](s)$  - dual

"exists succ. satisfying  $\psi$ "

# Must Hyper-transition ( $\forall\exists\exists$ )

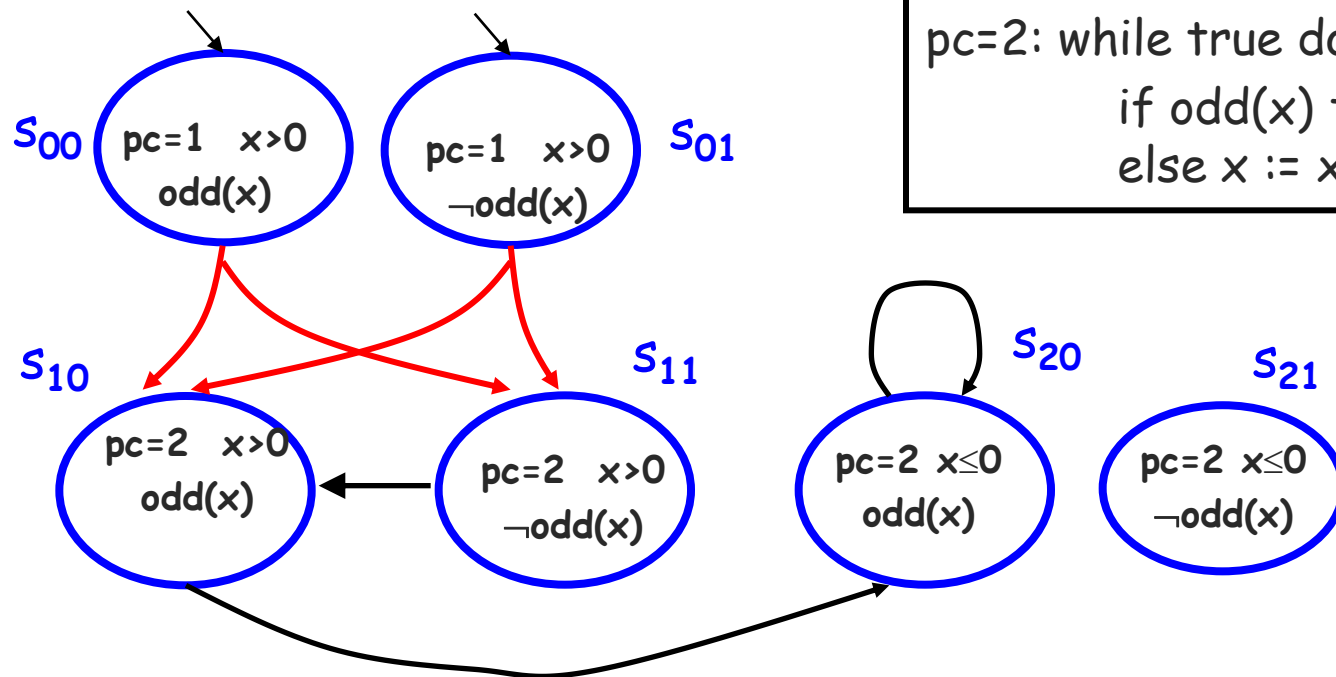


pc=1: if  $x > 5$  then  $x := x+1$   
else  $x := x+2$

Pc=2: ...

Every concrete state in  $\gamma(s_{00})$  has a transition  
to a concrete state in either  $\gamma(s_{10})$  or  $\gamma(s_{11})$

# Generalized KMTS $M_G$



$P ::$   
input  $x > 0$   
 $pc=1$ : if  $x > 5$  then  $x := x+1$   
          else  $x := x+2$   
 $pc=2$ : while true do  
          if  $odd(x)$  then  $x := -1$   
          else  $x := x+1$

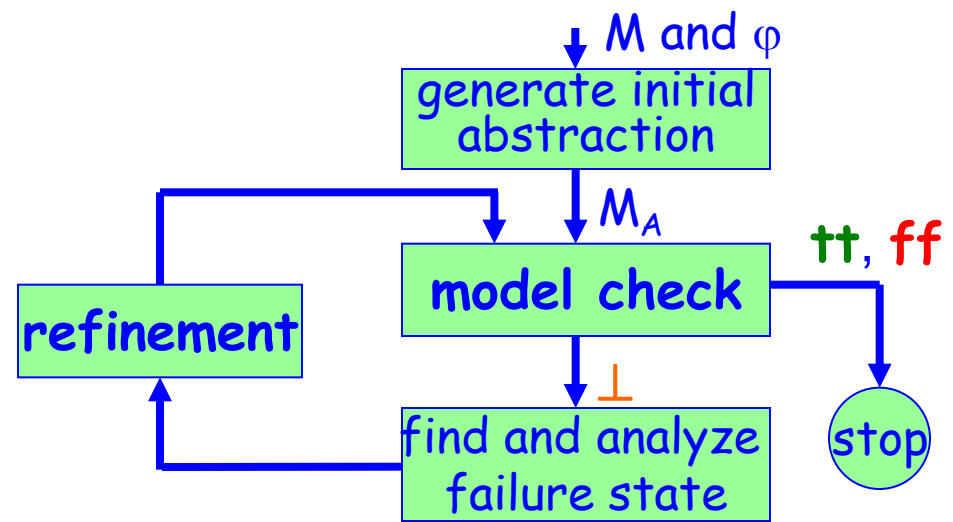
$$M_G \leq_{CTL} M'' \leq_{CTL} M \quad \text{and} \quad [EF(x \leq 0)](M_G) = \text{tt}$$

## Monotonicity Theorem:

Let  $M_A$  and  $M'_A$  be two abstract **GTSs** of  $M_c$  such that

- $M'_A$  is obtained from  $M_A$  by **splitting states**
- Both  $M_A$  and  $M'_A$  are **exact**

Then  $M'_A$  is **more precise** than  $M_A$



To complete the picture...

- Extension of the game-based 3-Valued **Model Checking** and **Failure Analysis** to GTSs



# Investigation of Abstract Models

- Monotonicity of Refinement
- Precision
- **Completeness**
- Efficiency

### (3) Completeness

- Suppose  $M_C \models \varphi$
- Does there **exist** a **finite abstraction**  $(S_A, \gamma)$  such that  $[M_A \models \varphi] = \text{tt}$ ?

# Monotonicity vs. Completeness vs. Precision

- **Monotonicity of refinement:**

Given **two** abstractions, where one is a **split** of the other, **is refined** abstraction **more precise** than **unrefined** one?

- **Precision:**

How many formulas can be verified on the abstract model, with a **given abstraction**  $(S_A, \gamma)$ ?

- **Completeness:**

Does there **exist** an **abstraction**  $(S_A, \gamma)$  for which we can verify the formula on the abstract model?

# Are KMTSs complete?

- No fairness constraints  
→ incomplete for **liveness** properties

What about **Safety**? (no least fixpoint)

**No** [Dams & Namjoshi, 2004]

But **GTSs are!** [de Alfaro et al, 2004]

# Investigation of Abstract Models

- Monotonicity of Refinement
- Precision
- Completeness
- **Efficiency**

## (4) Efficiency

### Cost:

- Size of the abstract model w.r.t.  $|S_A|$
- Efficiency of Model Checking

## Drawback of GTS

The number of must hyper transitions might be exponential in the number of abstract states  $|S_A|$

### Optimization:

including only  $(s, A)$  such that  $A$  is **minimal**

- Does not change precision of the abstract model

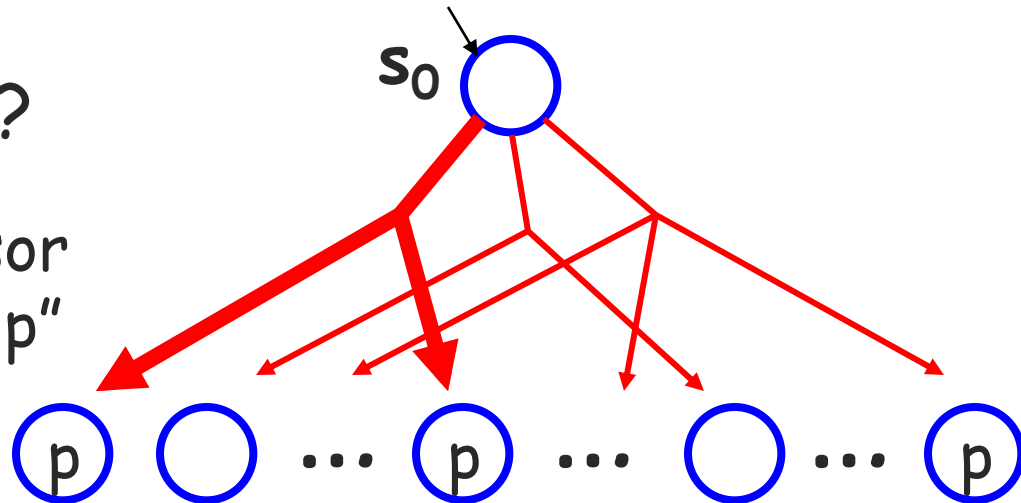
But, might still be too large

## In Practice

- Not all hyper-transitions are relevant for specific model checking problem

$[[EXp]](s_0) = ?$

"exists a successor  
that satisfies p"



→ Need to find **designated** hyper-transitions



## Alternative Approach [SG06]

- Compute hyper-transitions **during** Model Checking, **by need**

→ **Game-based Model Checking**

# Our Algorithm

Ordinary  
transitions

- Compute **over approximation** of concrete transition relation

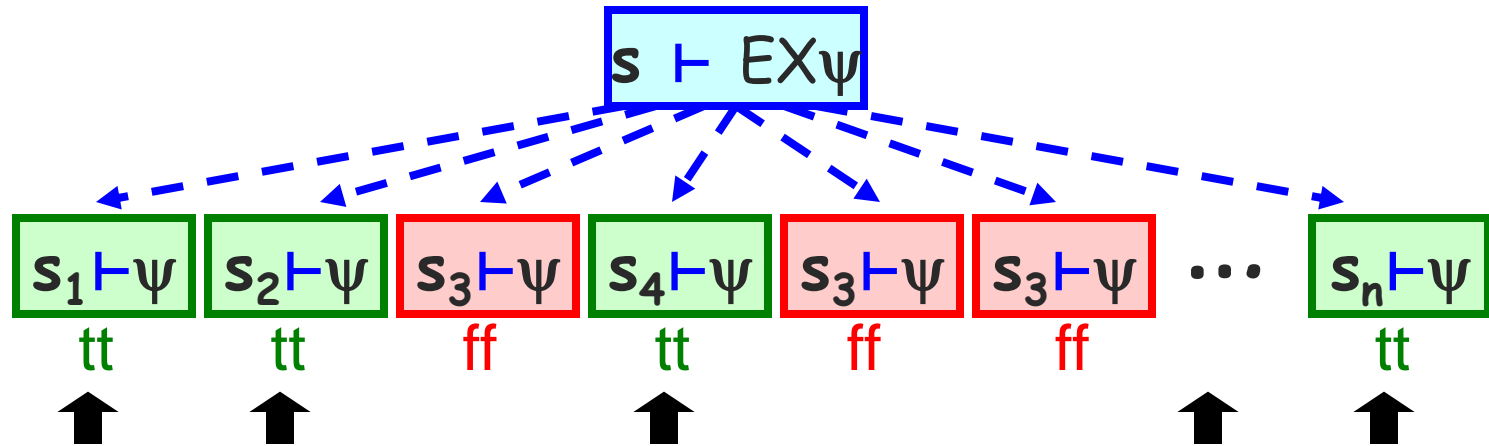
$(s_a, s'_a) \in R_A$  iff

$$\exists s_c \in \gamma(s_a) \exists s'_c \in \gamma(s'_a) : (s_c, s'_c) \in R_c$$

All reachable states are considered

- Construct **MC graph** based on  $R_A$
- Apply bottom up **coloring**

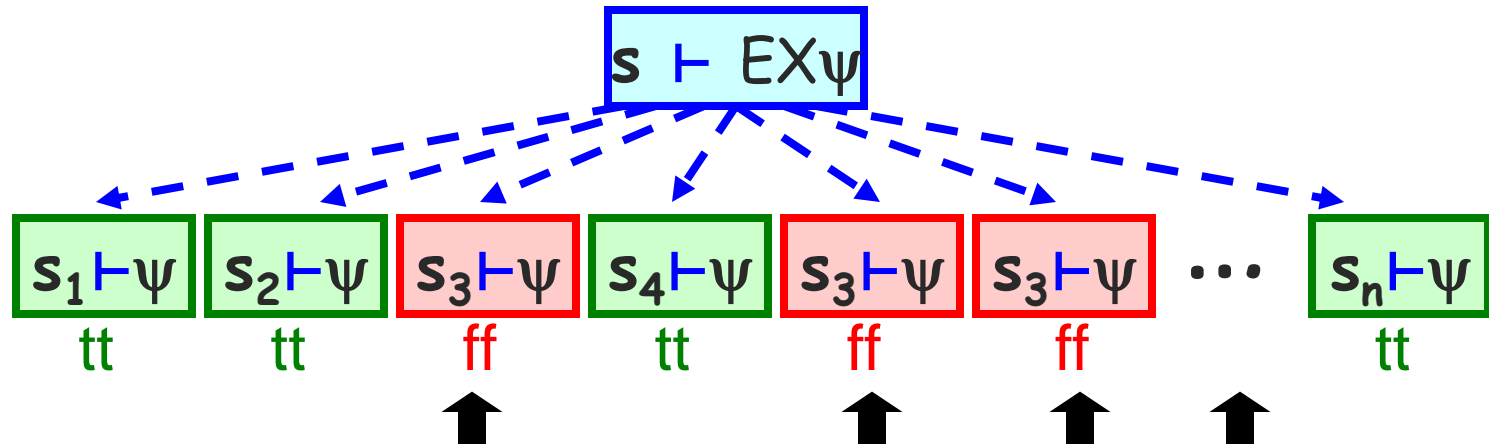
# During Coloring



$A_{tt}$ : all states in which the value of  $\psi$  is  $tt$

$(s, A_{tt})$  meets  $\forall\exists$ -condition  $[must]$ ? **yes**:  $[[EX\psi]](s_0) = tt$

# During Coloring



$A_{ff}$ : all states in which the value of  $\psi$  is  $ff$

$(s, A_{tt})$  meets  $\forall\exists\exists$ -condition [must]?    yes:  $[[EX\psi]](s_0) = tt$

All may transitions reach  $A_{ff}$ ?    yes:  $[[EX\psi]](s_0) = ff$

otherwise:  $[[EX\psi]](s_0) = \perp$

# Abstract Model Checking

- **Loops**: slight complication

Comparable to  
the complexity  
without hyper-  
transitions

In the paper [SG06]:

- Abstract MC for the alternation-free  $\mu$ -calculus
- Complexity:  $O(|S_A|^2 \times |\varphi|)$
- **In particular: num of  $\forall\exists\exists$  checks,  
num of hyper transitions**

As precise as constructing the full GTS

# Abstraction-Refinement

- If  $[[\varphi]](s_0) = \perp$ , apply **refinement** by splitting abstract states, as in [SG03]
- **Refinement is monotonic:**  
refined model is **more precise**, i.e. more  $\mu$ -calculus formulas are definite (**tt** or **ff**) in it

➔ **Abstraction-refinement loop**

# Summary

We presented the **TVAR** framework for **3-valued abstraction-refinement** in model checking:

- Properties preserved:
  - CEGAR: **truth** of **ACTL\***
  - TVAR: both **truth** and **falsity** of **Full CTL\***
- Refinement eliminates
  - CEGAR: **Counterexamples**
  - TVAR: **indefinite results** ( $\perp$ )

# Summary

The TVAR framework requires

1. Different abstract models (Rmust, Rmay)
  - Rmust is harder to compute, and problematic in terms of monotonicity, precision, completeness, and efficiency
  - KMTS, GTS, HTS
2. Adapted Model checking for new models:
  - 3-valued Coloring of MC-graph



# Summary

The TVAR framework requires

3. Refinement eliminating indefinite results
  - Identify failure state and cause
  - Incremental abstraction-refinement (similar to lazy abstraction in 2-valued MC)

Gives benefits in preciseness and in the properties preserved