Model-Checking Frameworks: Outline

- Theory (Part 1)
  - Notion of Abstraction
  - Aside: over- and under-approximation, simulation, bisimulation
  - Counter-example-based abstraction refinement
- Abstraction and abstraction refinement in program analysis (Part 2)
  - Kinds of abstraction:
    - Data, predicate
  - Building abstractions
    - Aside: weakest precondition
  - Counter-example-based abstraction refinement

Outline, cont'd

- 3-valued abstraction and abstraction-refinement (Part 3)
  - 3-valued logic
  - Theory of 3-valued abstractions: combining over- and under-approximation
  - 3-valued model-checking
  - Building 3-valued abstractions
  - Counter-example-based abstraction refinement

Acknowledgements

The following materials have been used in the preparation of this lecture:

- Edmund Clarke
  - "SAT-based abstraction/refinement in model-checking", a course lecture at CMU
- Corina Pasareanu
  - Conference presentations at TACAS'01 and ICSE'01
- John Hatcliff
  - Course materials from Specification and Verification in Reactive Systems

Many thanks for providing this material!

3

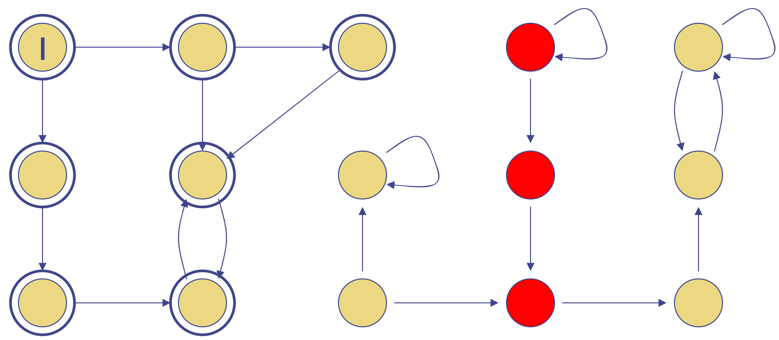
Model Checking

- Given a:
  - Finite transition system $M(S, s_0, R, L)$
  - A temporal property φ
- The model checking problem:
  - Does M satisfy φ?

$$M \overset{?}{\models} \varphi$$

4

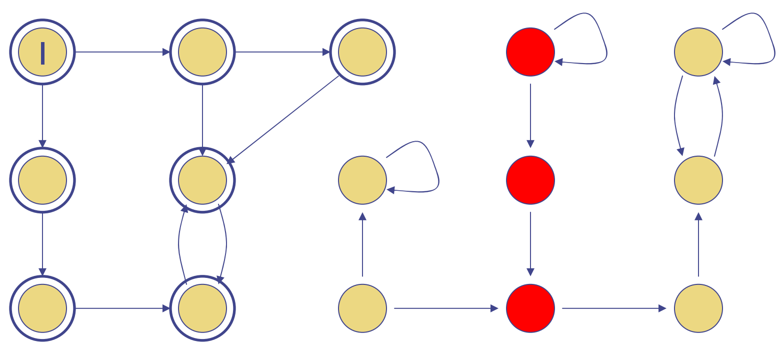
Model Checking (safety)



Add reachable states until reaching a fixed-point

= bad state

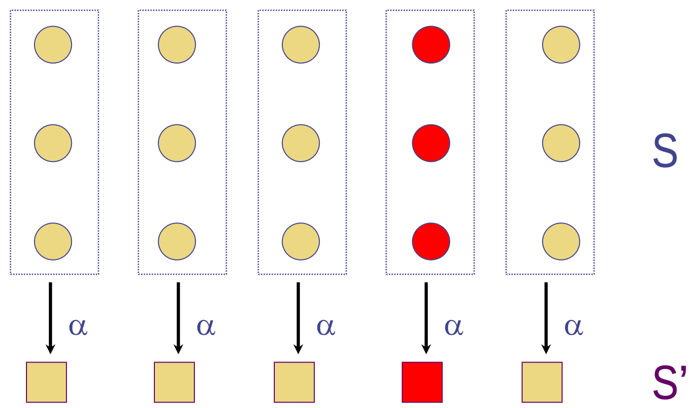
Model Checking (safety)



Too many states to handle !

= bad state

Abstraction



Abstraction Function $\alpha : S \rightarrow S'$

7

Abstraction Function: A Simple Example

- Partition variables into visible($\mathcal{V}$) and invisible($\mathcal{I}$) variables.
- The abstract model consists of $\mathcal{V}$ variables. $\mathcal{I}$ variables are made inputs.
- The abstraction function maps each state to its projection over $\mathcal{V}$.

8

Abstraction Function: Example

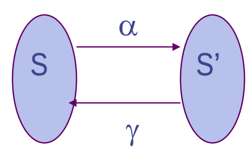
$x1$	$x2$	$x3$	$x4$
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1

$\xrightarrow{\alpha}$

$x1$	$x2$
0	0

Group concrete states with identical visible part to a single abstract state.

Computing Abstractions



- S – concrete state space
- S' – abstract state space
- α: S $\rightarrow$ S' - abstraction function
- γ: S' $\rightarrow$ S - concretization function
- Properties of α and γ:
  - $\alpha(\gamma(A)) = A$, for A in S'
  - $\gamma(\alpha(C)) \supseteq C$, for C in S
- The above properties mean that α and γ are Galois-connected

Aside: simulations

$M = (s_0, S, R, L)$

$M' = (t_0, S', R', L')$

Definition: p is a simulation between M and M' if

1. $(s_0, t_0) \in p$
2. $\forall (t, t_1) \in R'\ \exists (s, s_1) \in R$ s.t. $(s, t) \in p$ and $(s_1, t_1) \in p$

Intuitively, every transition in M' corresponds to some transition in M

11

Aside: bisimulation

$M = (s_0, S, R, L)$

$M' = (t_0, S', R', L')$

Definition: p is a bisimulation between M and M' if

1. p is a simulation between M and M' and
2. p is a simulation between M' and M

12

Computing Existential Transition Relation

- $R^{\exists\exists}$ [Dams'97]: $(t, t_1) \in R'$ iff $\exists\, s \in \gamma(t)$ s.t. $\exists\, s_1 \in \gamma(t_1)$ and $(s, s_1) \in R$
- This ensures that *M'* is the over-approximation if *M*, or *M'* simulates *M*.

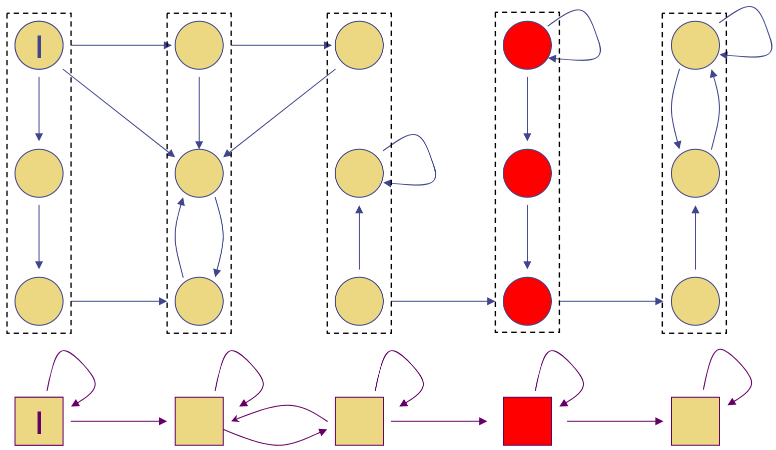
13

Abstract Kripke Structure

- Abstract interpretation of atomic propositions
  - $l'(a, p)$ = true iff forall s in $\gamma(a)$, $l(s, p)$ = true
  - $l'(a, p)$ = false iff forall s in $\gamma(a)$, $l(s, p)$ = false
- Abstract Transition Relation (2 choices)
  - Over-Approximation (Existential)
    - Make a transition from an abstract state if *at least one* corresponding concrete state has the transition.
  - Under-Approximation (Universal)
    - Make a transition from an abstract state if *all* the corresponding concrete states have the transition.

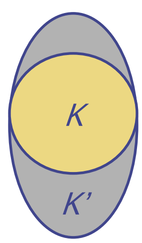
14

Existential Abstraction (Over-Approximation)



Preservation via Over-Approximation

- Let φ be a universal temporal formula (ACTL, LTL)
- Let *K'* be an over-approximating abstraction of *K*



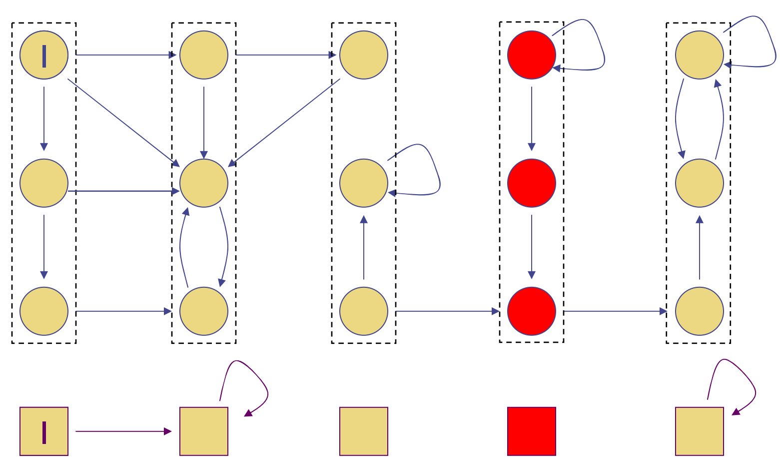
- Preservation Theorem
  - $K' \models \varphi$ implies $K \models \varphi$
- Converse does not hold
  - $K' \not\models \varphi$ does not imply $K \not\models \varphi$!!!

Computing Transition Relation

- $R^{\forall\exists}$ [Dams'97]: $(t, t_1) \in R'$ iff $\forall\, s \in \gamma(t)$ $\exists\, s_1 \in \gamma(t')$ and $(s, s_1) \in R$
- This ensures that *M'* is the under-approximation if *M*, or *M* simulates *M'*.

17

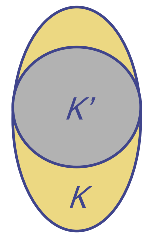
Universal Abstraction (Under-Approximation)



18

Preservation via Under-Approximation

- Let φ be an existential temporal formula (ECTL)
- Let K' be an under-approximating abstraction of K



- Preservation Theorem
  - $K' \models \varphi$ implies $K \models \varphi$
- Converse does not hold
  - $K' \not\models \varphi$ does not imply $K \not\models \varphi$!!!

19

Which abstraction to use?

Property Type	Expected Result	Abstraction to use
Universal (ACTL, LTL)	True	Over-
	False	Under-
Existential (ECTL)	True	Under-
	False	Over-

But what about mixed properties?!

20

Our specific problem

- Let φ be a universally-quantified property (i.e., expressed in LTL or ACTL) and *M'* simulates *M*
- Preservation Theorem

$$M' \models \varphi \rightarrow M \models \varphi$$

- Converse does not hold

$$M' \not\models \varphi \not\rightarrow M \not\models \varphi$$

- The counterexample may be spurious

Checking the Counterexample

- Counterexample : $(c_1, \ldots, c_m)$
  - Each c_i is an assignment to V.
- Simulate the counterexample on the concrete model.

Checking the Counterexample

Concrete traces corresponding to the counterexample:

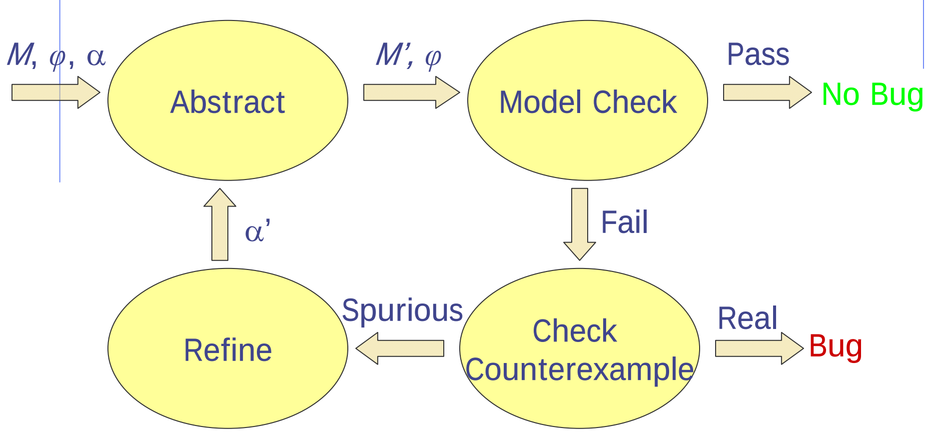
$\phi \quad = \quad I(s_1) \quad \wedge$ (Initial State <- s_0 in our case)

$\bigwedge_{i=1}^{m-1} R(s_i, s_{i+1}) \quad \wedge$ (Unrolled Transition Relation)

$\bigwedge_{i=1}^{m} \text{visible}(s_i) = c_i$ (Restriction of $\mathcal{V}$ to Counterexample)

23

Abstraction-Refinement Loop

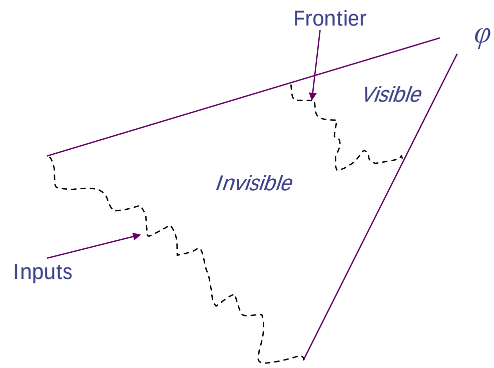


24

Refinement methods…

Localization

(R. Kurshan, 80's)



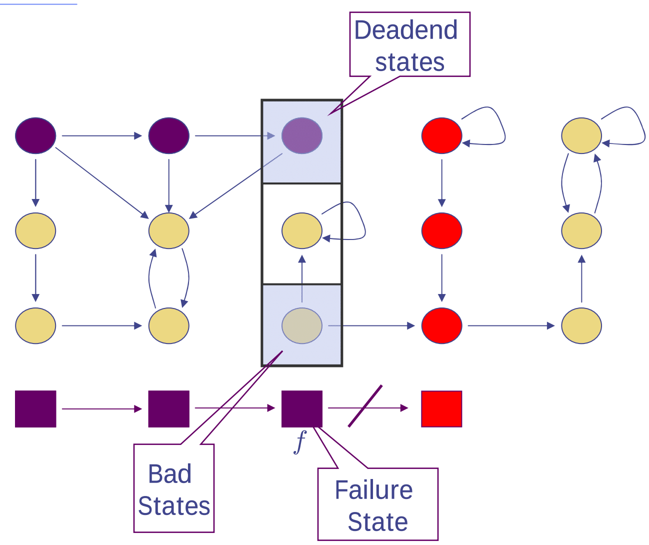
Refinement methods…

Abstraction/refinement with conflict analysis

(Chauhan, Clarke, Kukula, Sapra, Veith, Wang, FMCAD 2002)

- Simulate counterexample on concrete model with SAT
- If the instance is unsatisfiable, analyze conflict
- Make visible one of the variables in the clauses that lead to the conflict

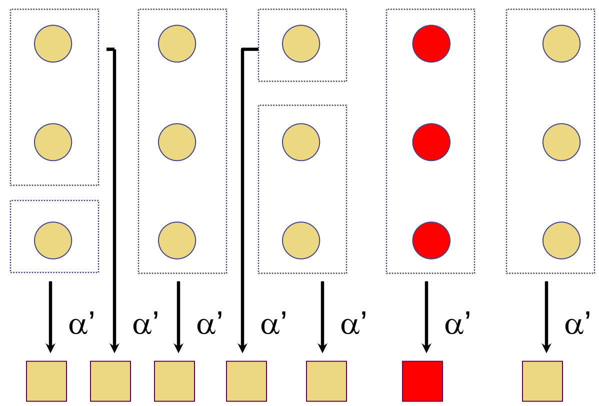
Why spurious counterexample?



Refinement

- Problem: Deadend and Bad States are in the same abstract state.
- Solution: Refine abstraction function.
- The sets of Deadend and Bad states should be separated into different abstract states.

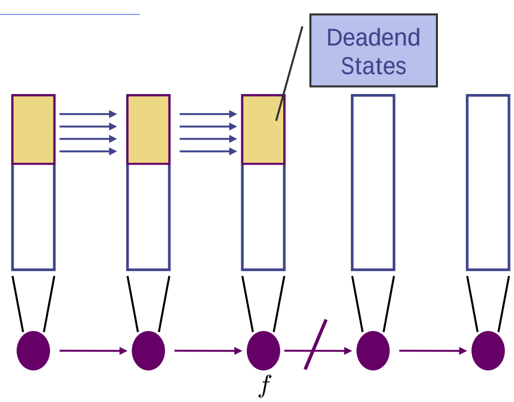
Refinement



Refinement : α’

29

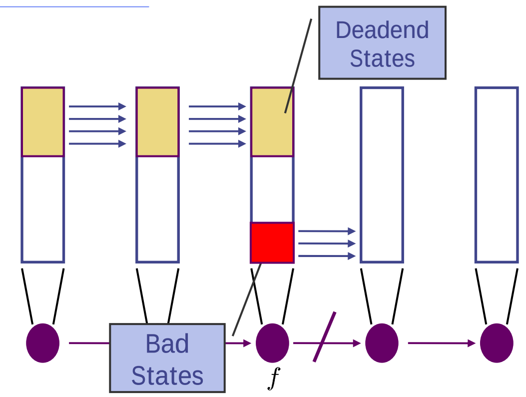
Refinement



$$\phi_D = I(s_1) \wedge \bigwedge_{i=1}^{f-1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=1}^{f} \text{visible}(s_i) = c_i$$

30

Refinement



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

Refinement as Separation

d_1	0	1	0	0	1	0	1
b_1	0	1	0	0	0	1	0
b_2	0	1	0	0	1	1	1

$\mathcal{I}$

$\mathcal{V}$

Refinement : Find subset $\mathcal{U}$ of $\mathcal{I}$ that separates between all pairs of deadend and bad states. Make them visible.

Keep $\mathcal{U}$ small !

Refinement as Separation

d_1	0	1	0	0	1	0	1
b_1	0	1	0	0	0	1	0
b_2	0	1	0	0	1	1	1

$\mathcal{I}$

$\mathcal{V}$

Refinement : Find subset $\mathcal{U}$ of $\mathcal{I}$ that separates between all pairs of deadend and bad states. Make them visible.

Keep $\mathcal{U}$ small !

33

Refinement as Separation

The state separation problem

Input: Sets D, B

Output: Minimal $\mathcal{U} \in \mathcal{I}$ s.t.:

$$\forall\ d \in \mathrm{D},\ \forall\ b \in \mathrm{B},\ \exists u \in \mathcal{U}.\quad d(u) \neq b(u)$$

The refinement α' is obtained by adding $\mathcal{U}$ to $\mathcal{V}$.

34

Two separation methods

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

We will not talk about these in class