

Algorithms for Symbolic Model Checking

– *Symbolic Model Checking* –

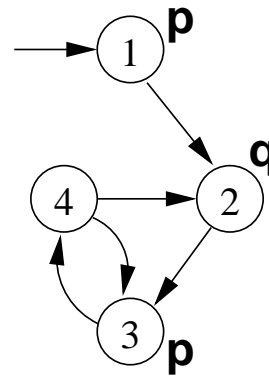
A. Cimatti and M. Pistore

ESSLLI, July 5-9, 2002, Trento (Italy)

Model Checking

Model Checking is a formal verification technique where...

- ...the system is represented as Finite State Machine



- ...the properties are expressed as temporal logic formulae

LTL: **G(p → Fq)**

CTL: **AG(p → AFq)**

- ...the model checking algorithm checks whether all the executions of the model satisfy the formula.

The Main Problem: State Space Explosion

The bottleneck:

- Exhaustive analysis may require to store all the states of the Kripke structure
- The state space may be exponential in the number of components
- State Space Explosion: too much memory required

Symbolic Model Checking:

- Symbolic representation
- Different search algorithms

Symbolic Model Checking

Symbolic representation:

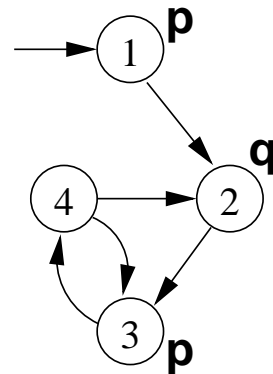
- manipulation of *sets of states* (rather than single states);
- sets of states represented by formulae in propositional logic;
 - set cardinality not directly correlated to size
- expansion of *sets of transitions* (rather than single transitions);
- two main symbolic techniques:
 - Binary Decision Diagrams (BDDs)
 - Propositional Satisfiability Checkers (SAT solvers)

Different model checking algorithms:

- Fix-point model checking (historically, for CTL)
- Bounded Model Checking (historically, for LTL)
- Invariant Checking

CTL Model Checking: Example

Consider a simple system and a specification:

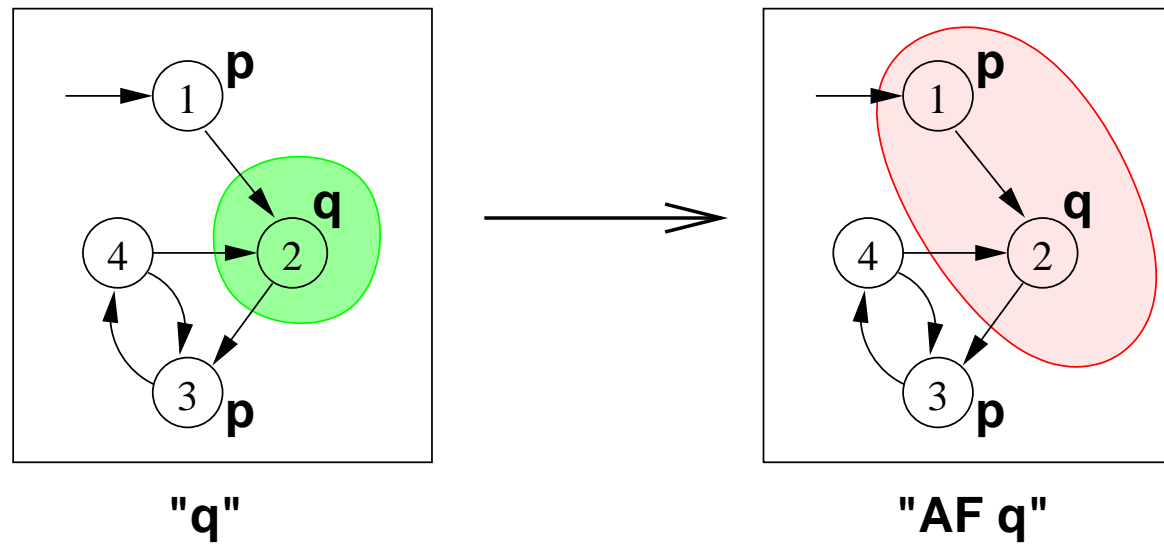


$AG(p \rightarrow AFq)$

Idea:

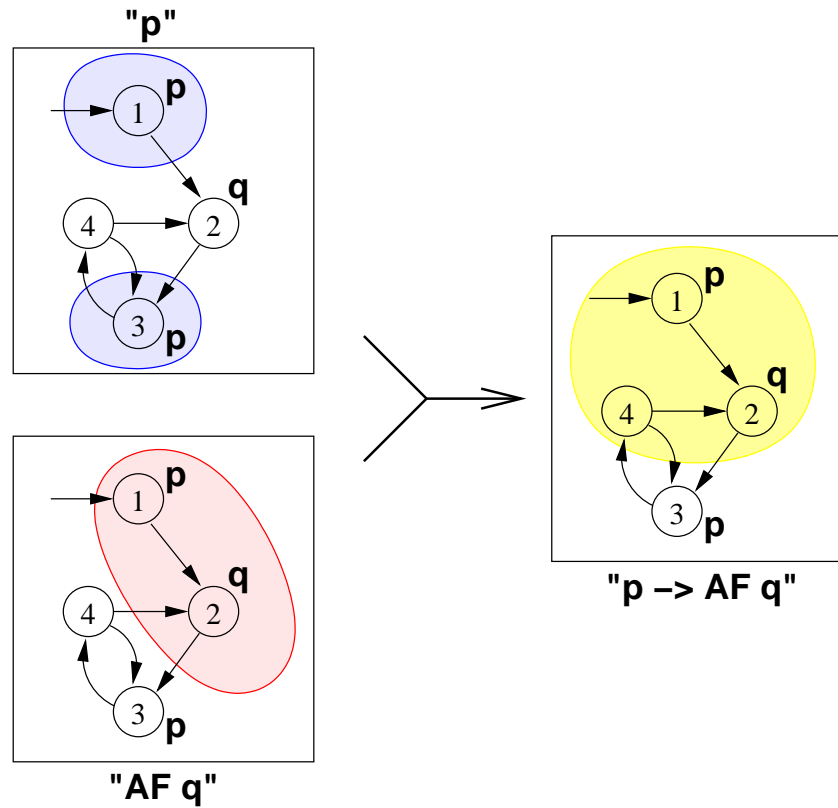
- construct the set of states where the formula holds
- proceeding “bottom-up” on the structure of the formula
- $q, AFq, p, p \rightarrow AF q, AG(p \rightarrow AF q)$

CTL Model Checking: Example

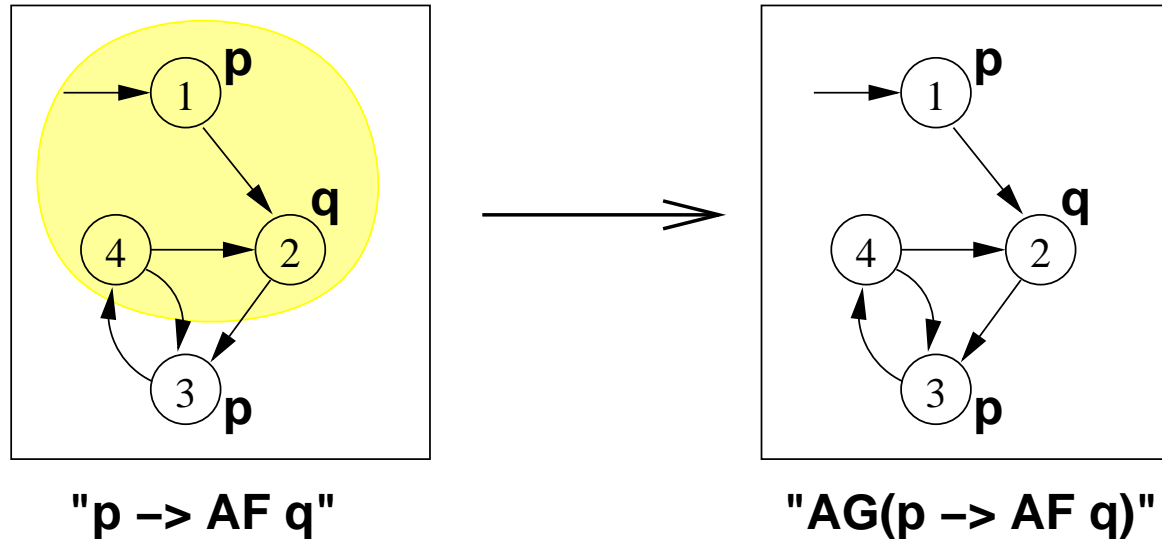


$AF q$ is the union of q , $AX q$, $AX AX q$, ...

CTL Model Checking: Example



CTL Model Checking: Example



The set of states where the formula holds is empty!

Counterexample reconstruction is based on the intermediate sets.

Fix-Point Symbolic Model Checking

Model Checking Algorithm for CTL formulae based on fix-point computation:

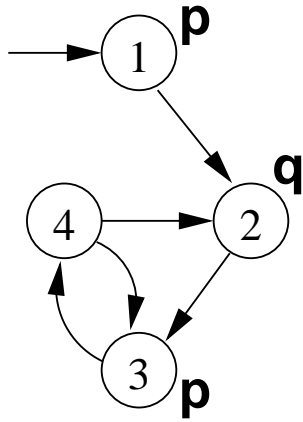
- traverse formula structure, for each subformula build set of satisfying states; compare result with initial set of states.
- boolean connectives: apply corresponding boolean operation;
- on $AX \Phi$, apply preimage computation
 - $\forall s'. (\mathcal{T}(s, s') \rightarrow \Phi(s'))$
- on $AF \Phi$, compute least fixpoint using
 - $AF \Phi \leftrightarrow (\Phi \vee AX AF \Phi)$
- on $AG \Phi$, compute greatest fixpoint using
 - $AG \Phi \leftrightarrow (\Phi \wedge AX AG \Phi)$

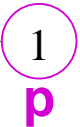
Bounded Model Checking

Key ideas:

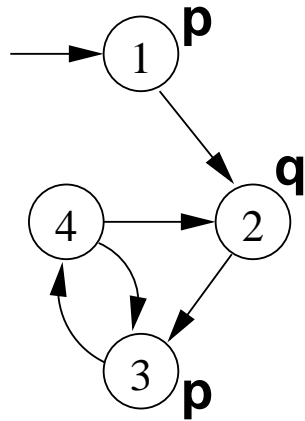
- looks for counter-example paths of increasing length k
 - oriented to finding bugs
- for each k , builds a boolean formula that is satisfiable iff there is a counter-example of length k
 - can be expressed using $k \cdot |s|$ variables
 - formula construction is not subject to state explosion
- satisfiability of the boolean formulas is checked using a *SAT procedure*
 - can manage complex formulae on several 100K variables
 - returns satisfying assignment (i.e., a counter-example)

Bounded Model Checking: Example

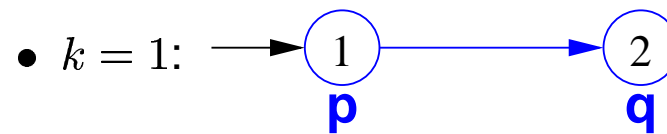


- Formula: $\mathbf{G}(p \rightarrow \mathbf{F}q)$
- Negated Formula (violation): $\mathbf{F}(p \ \& \ \mathbf{G} \ ! \ q)$
- $k = 0$: 
- No counter-example found.

Bounded Model Checking: Example

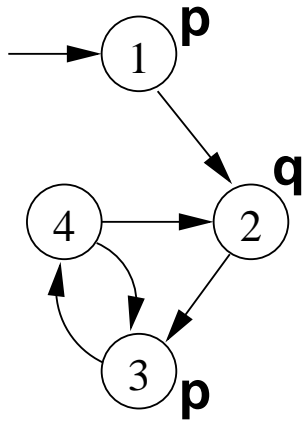


- Formula: $\mathbf{G}(p \rightarrow \mathbf{F}q)$

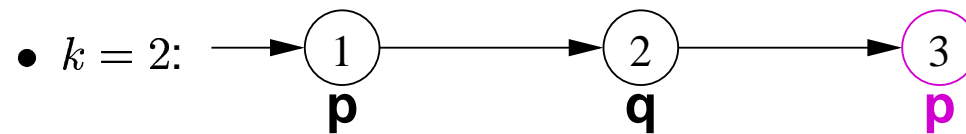


- $k = 1$: \rightarrow
- No counter-example found.

Bounded Model Checking: Example

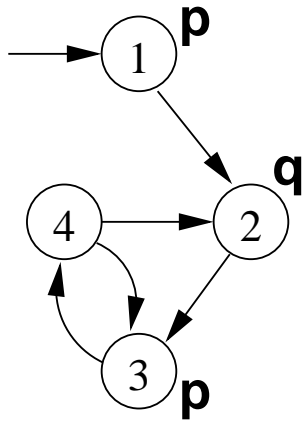


- Formula: $G(p \rightarrow Fq)$



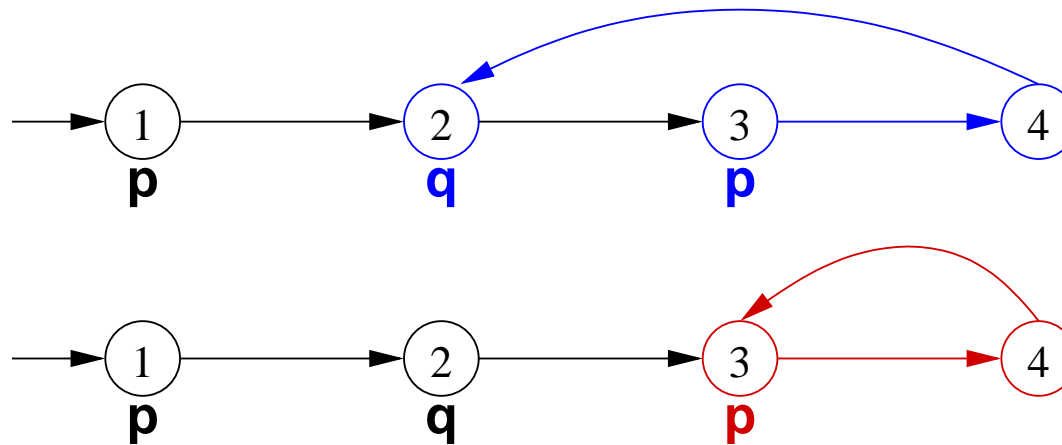
- No counter-example found.

Bounded Model Checking: Example



- Formula: $G(p \rightarrow Fq)$

- $k = 3$:



- The 2nd trace is a counter-example!

Bounded Model Checking

- *Bounded Model Checking:*

Given a FSM $\mathcal{M} = \langle \mathcal{S}, \mathcal{I}, \mathcal{T} \rangle$, an LTL property ϕ and a bound $k \geq 0$:

$$\mathcal{M} \models_k \phi$$

- This is equivalent to the satisfiability problem on formula:

$$[[\mathcal{M}, \phi]]_k \equiv [[\mathcal{M}]]_k \wedge [[\phi]]_k$$

where:

- $[[\mathcal{M}]]_k$ is a k -path compatible with \mathcal{I} and \mathcal{T} :

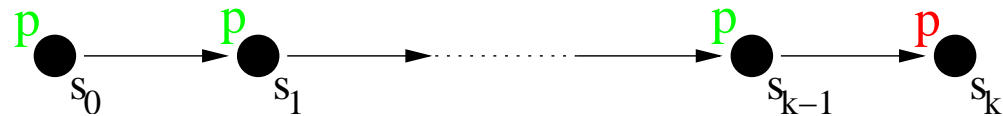
$$\mathcal{I}(s_0) \wedge \mathcal{T}(s_0, s_1) \wedge \dots \wedge \mathcal{T}(s_{k-1}, s_k)$$

- $[[\phi]]_k$ says that the k -path satisfies ϕ

Bounded Model Checking: Examples

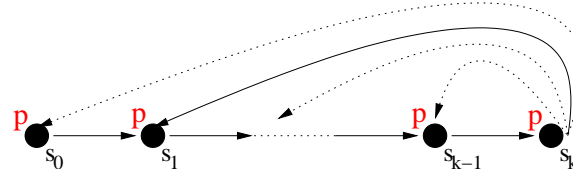
- $\phi = F p$

$$[[F p]]_k = \bigvee_{i=0}^k p(s_i)$$



- $\phi = G p$

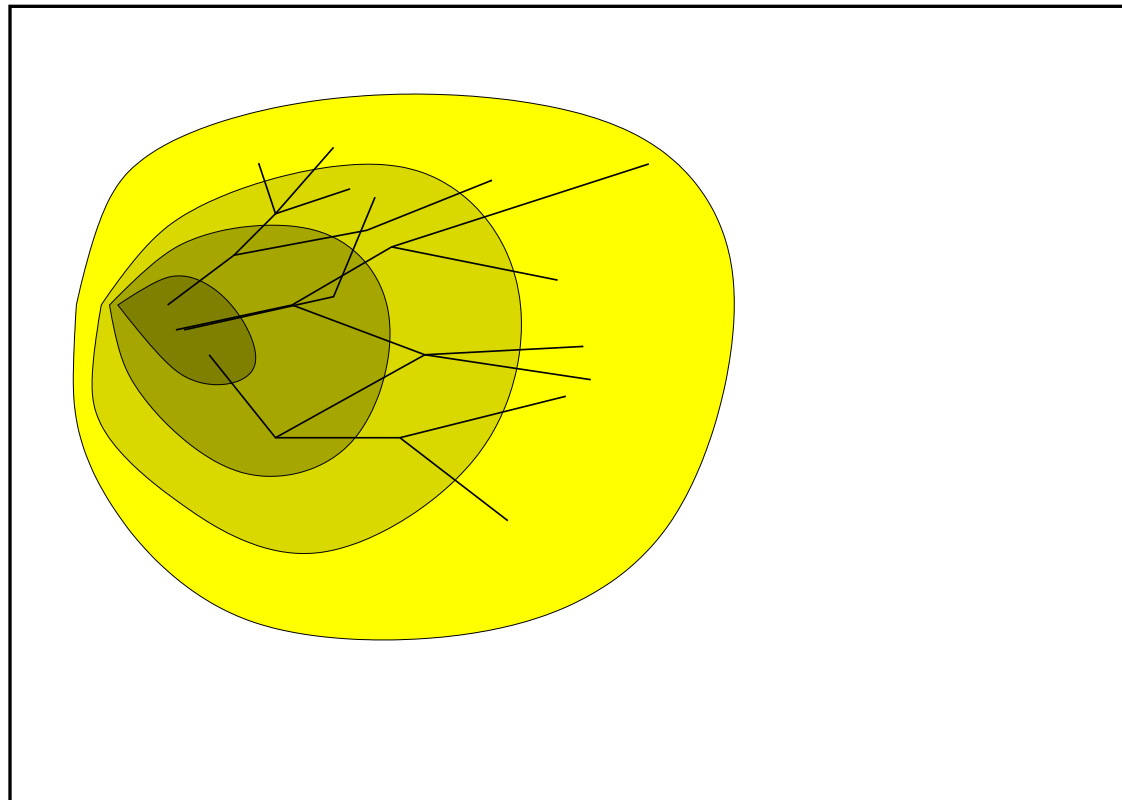
$$[[G p]]_k = \bigvee_{i=0}^k \left(\mathcal{T}(s_k, s_i) \wedge \bigwedge_{i=0}^k p(s_i) \right)$$



Symbolic Model Checking of Invariants

Checking invariant properties (e.g. **AG ! bad** is a reachability problem):

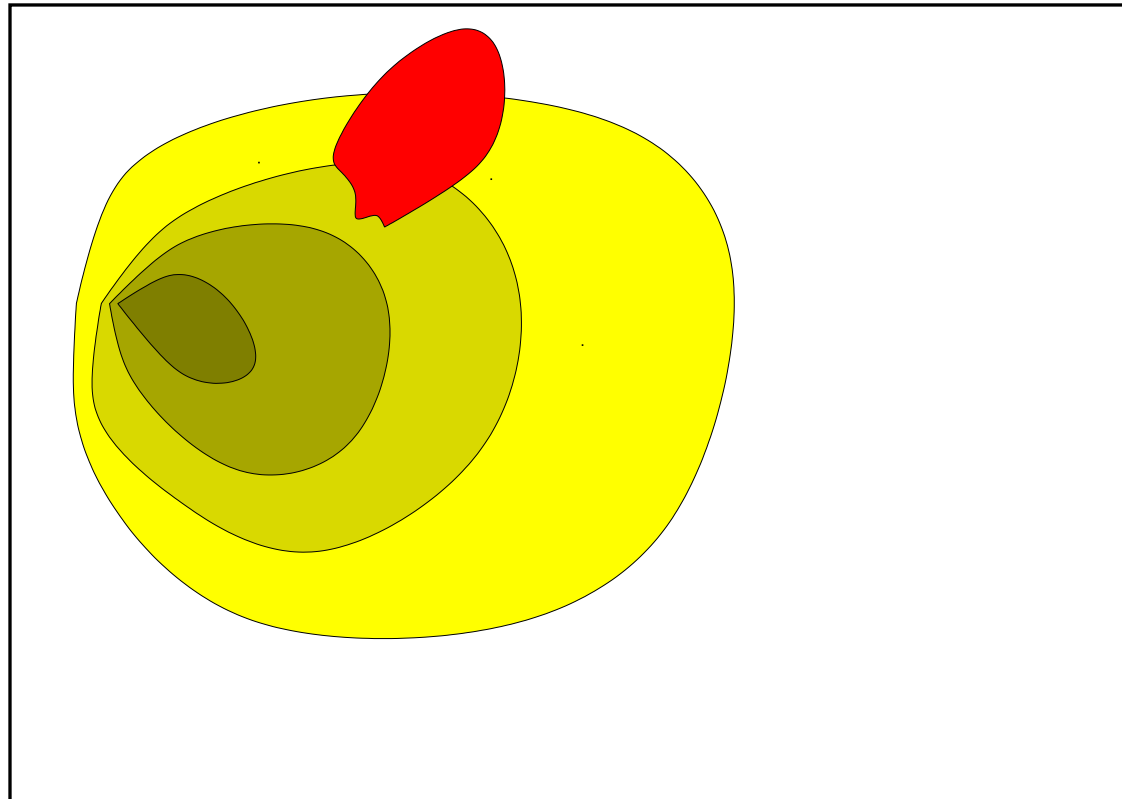
- is there a reachable state that is also a bad state?



On the fly Checking of Invariants

Anticipate bug detection:

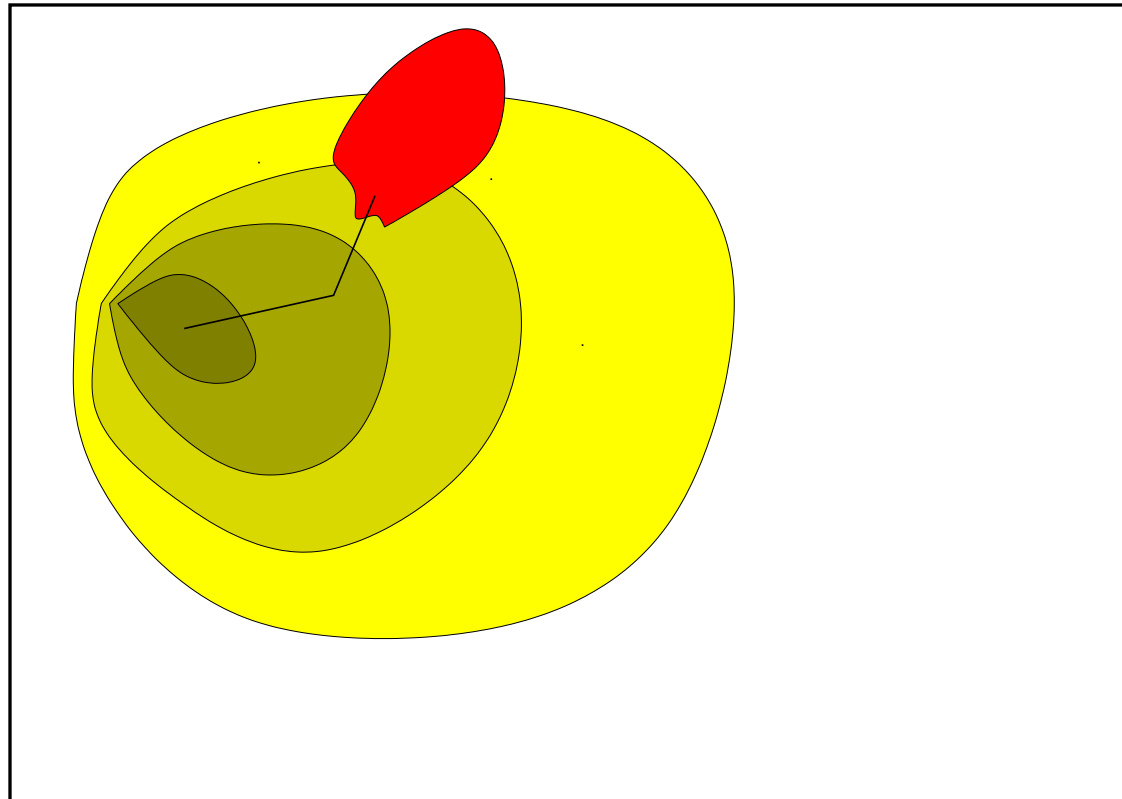
- at each layer, check if a new state is a bug



On the fly Checking of Invariants: Counterexamples

If a bug is found,

- a counterexample can be reconstructed proceeding backwards



Inductive Reasoning on Invariants

1. If all the initial states are good,
 2. and if from any good state we only go to good states
- ⇒ then we can conclude that the system is correct for all reachable states.

