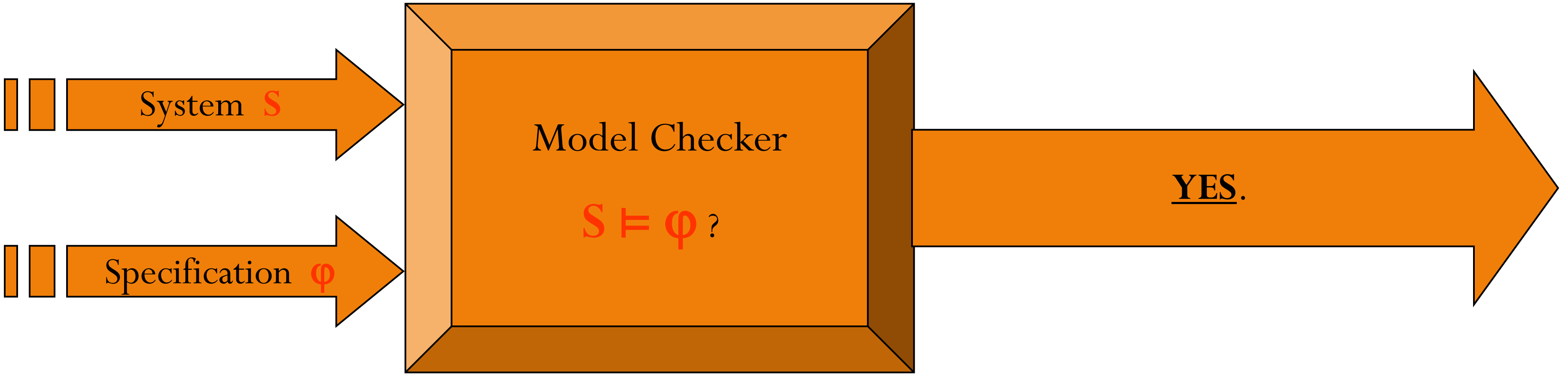
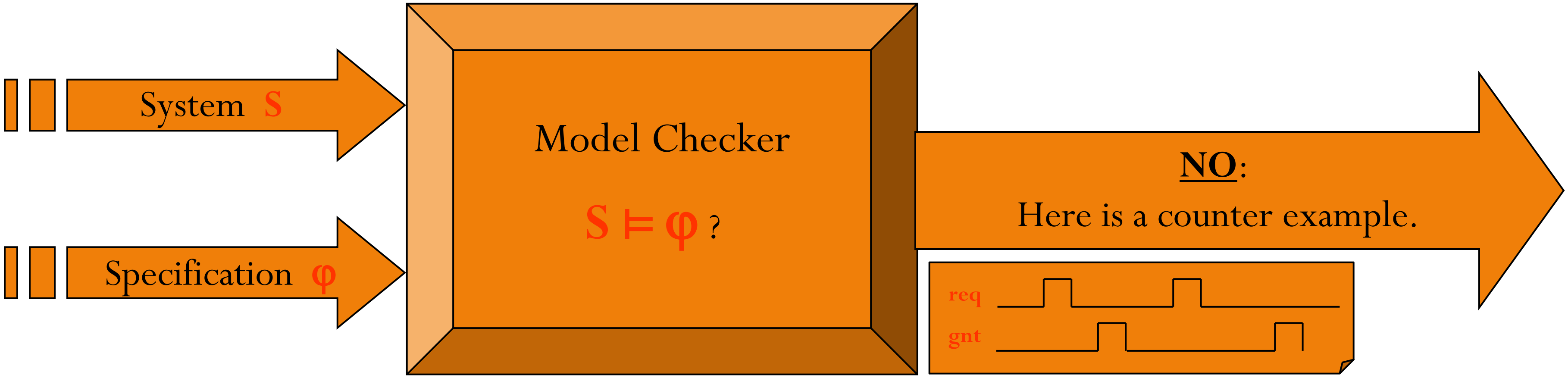
Bounded Model Checking using SAT Solving

Shoham Ben-David

Model Checking

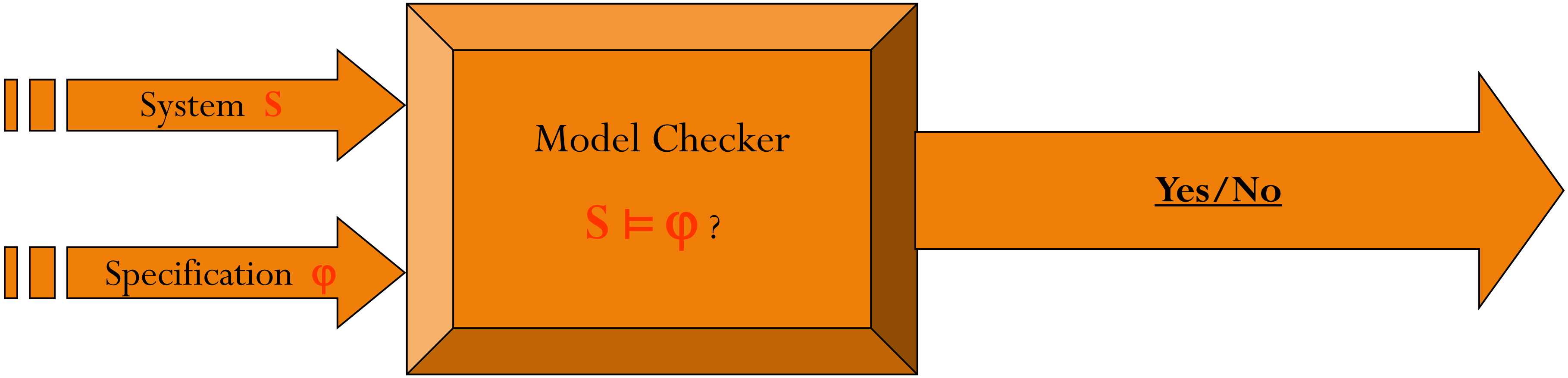


Model Checking



Model Checking

- What is **S**? What is **φ**? How is model checking performed?



- **S** is a program, software or hardware
- **φ** is a temporal logic specification
- The model checking method depends on **S** and **φ**.

Explicit Vs. Symbolic Model Checking

- Roughly speaking, model checking algorithms are divided into
  - **Explicit** methods
    - applied mainly to software programs
  - **Symbolic** methods
    - applied mainly to hardware
- In the context of model checking, **Symbolic** means manipulating sets of states

- The two branches of model checking use different sets of methods
  - Both use SAT solving
- In this talk: Symbolic Model Checking

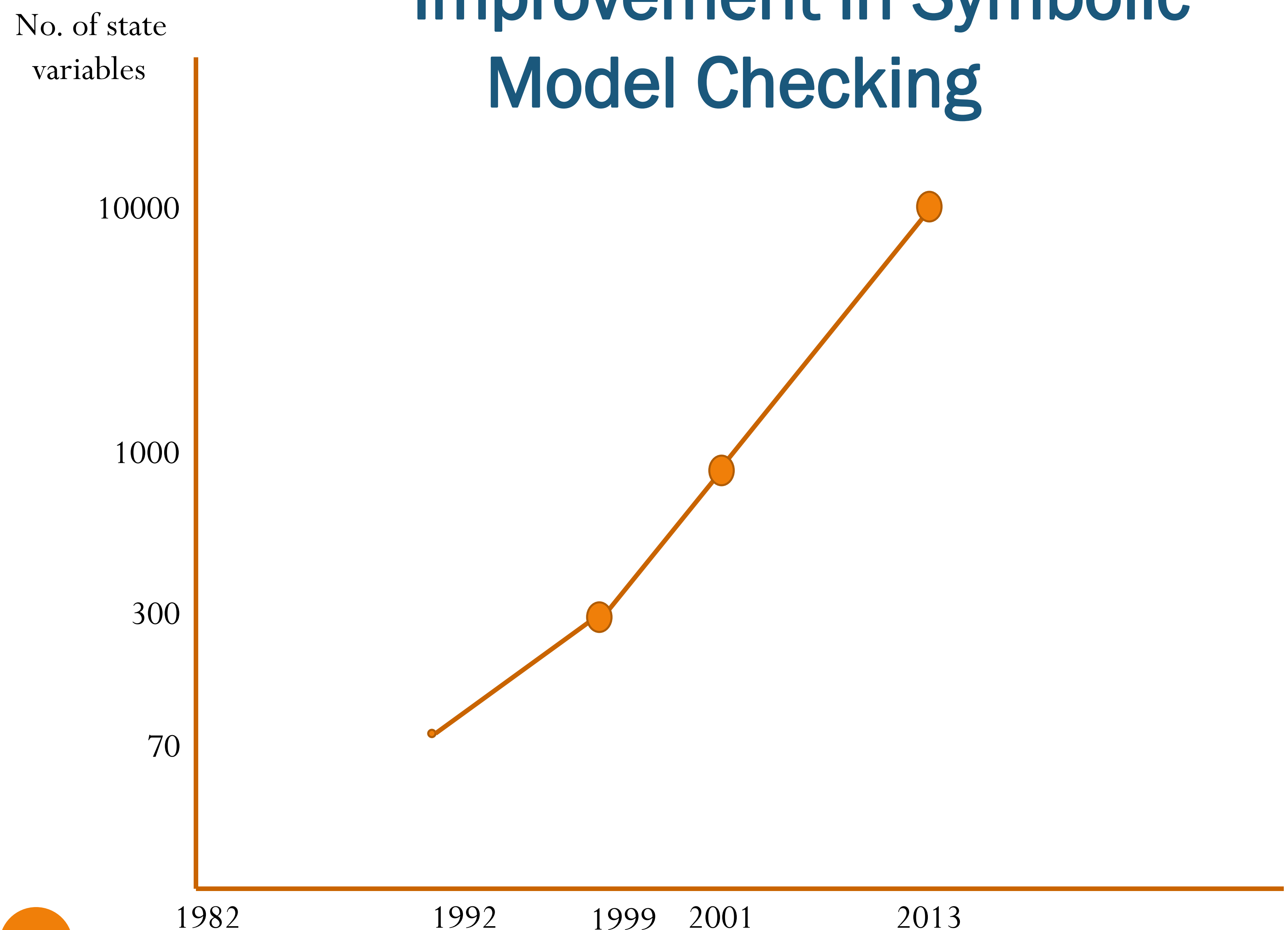
Temporal Logic Specifications

- Linear Temporal Logic (LTL)
  - Allows specifying events over time
    - **G**(req → **F**(ack))
    - **G**(ack → **X**(¬ack))
- Other languages exist: CTL, PSL and more
- For model checking purposes:
  - Translated into automata + a very simple formula
    - Invariant : **G**(p) type formula, with p being a Boolean formula
    - Most of the specifications are translated in this way
- In this talk: symbolic model checking of **G**(p) formulas
  - OR: Symbolic reachability analysis :
    - is p invariant?
    - is ¬p reachable?

History of symbolic Model Checking

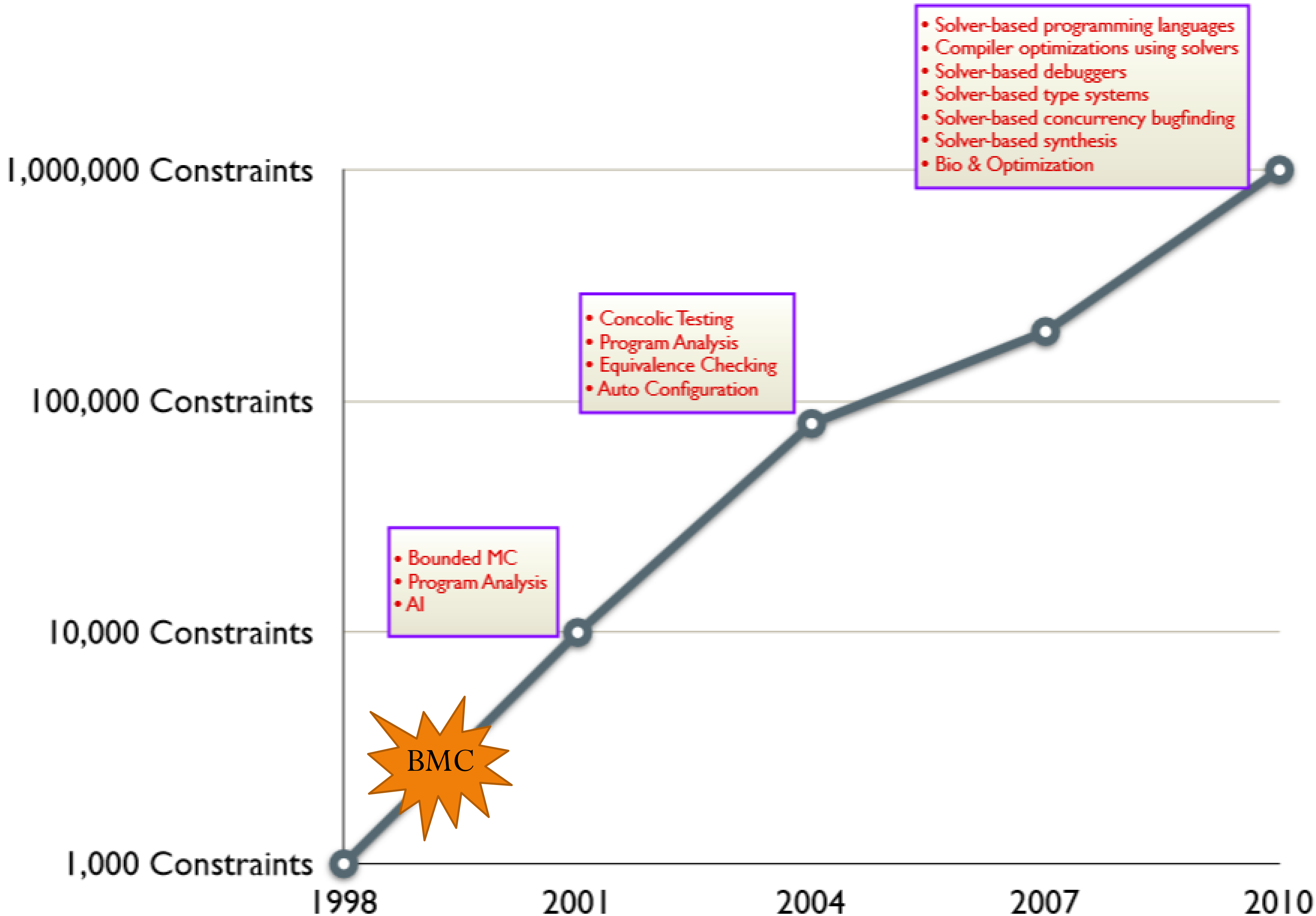
- 1977: Temporal Logic
  - Pnueli, "The temporal logic of programs"
- 1981/1982: Symbolic Model Checking
  - Clarke, Emerson: "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic"
  - Queille, Sifakis, "Specification and verification of concurrent systems in CESAR"
- 1992: Symbolic Model Checking using BDDs
  - Burch, Clarke, McMillan, Dill, Hwang: "Symbolic Model Checking: 10^20 States and Beyond".
    - McMillan also wrote the first symbolic model checker SMV
- 1999: Bounded Model Checking using SAT
  - Biere, Cimatti, Clarke, Zhu: "Symbolic Model Checking without BDDs"

Improvement in Symbolic Model Checking



SAT/SMT Solver Research Story

A 1000x Improvement



Example: a Simple Model

Three Boolean variables: V_1, V_2, V_3

INITIAL ASSIGNMENT

$init(V_1) := 1;\ init(V_2) := 1;\ init(V_3) := 0;$

NEXT STATE ASSIGNMENT

```
next(V1) :=  case
                 V1 & V2 :  0;
                 V3:  1;
                 else : {0,1};
             esac;
```

$next(V_3) := V_1;$

```
next(V2) :=  case
                 V   : {0,1};
                 else :  0;
             esac;
```

SPECIFICATION

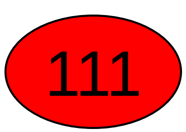
G(!(V1 & V2 & V3))

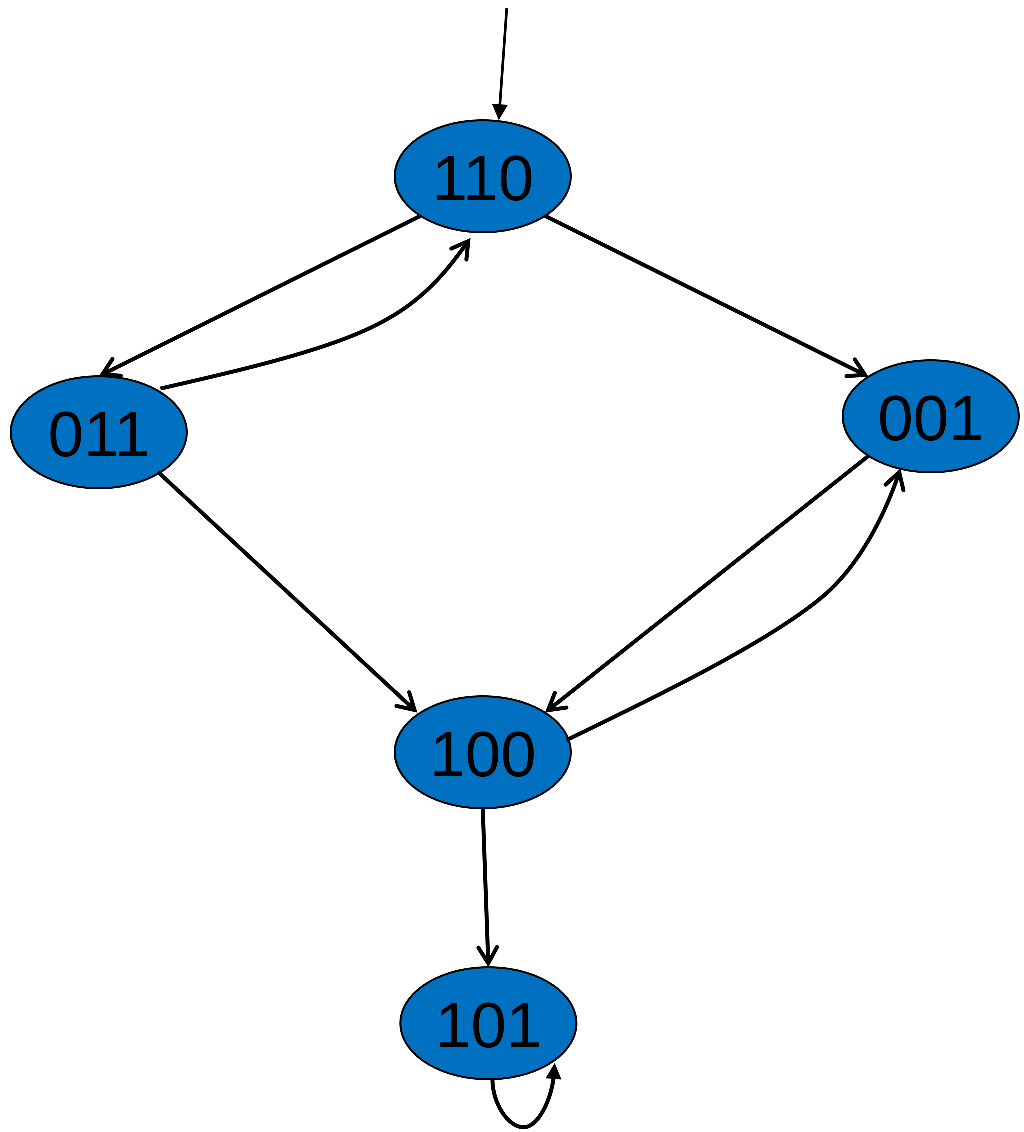
Model Checking : Reachability

$init(V_1) := 1;$ $init(V_2) := 1;$ $init(V_3) := 0;$

```
next(V1) :=  case
               V1 & V2 :  0;
               V3:  1;
               else : {0,1};
             esac;
next(V2) :=  case
               V    : {0,1};
               else :   0;
             esac;
next(V3) := V1;
```

G(!(V1 & V2 & V3))



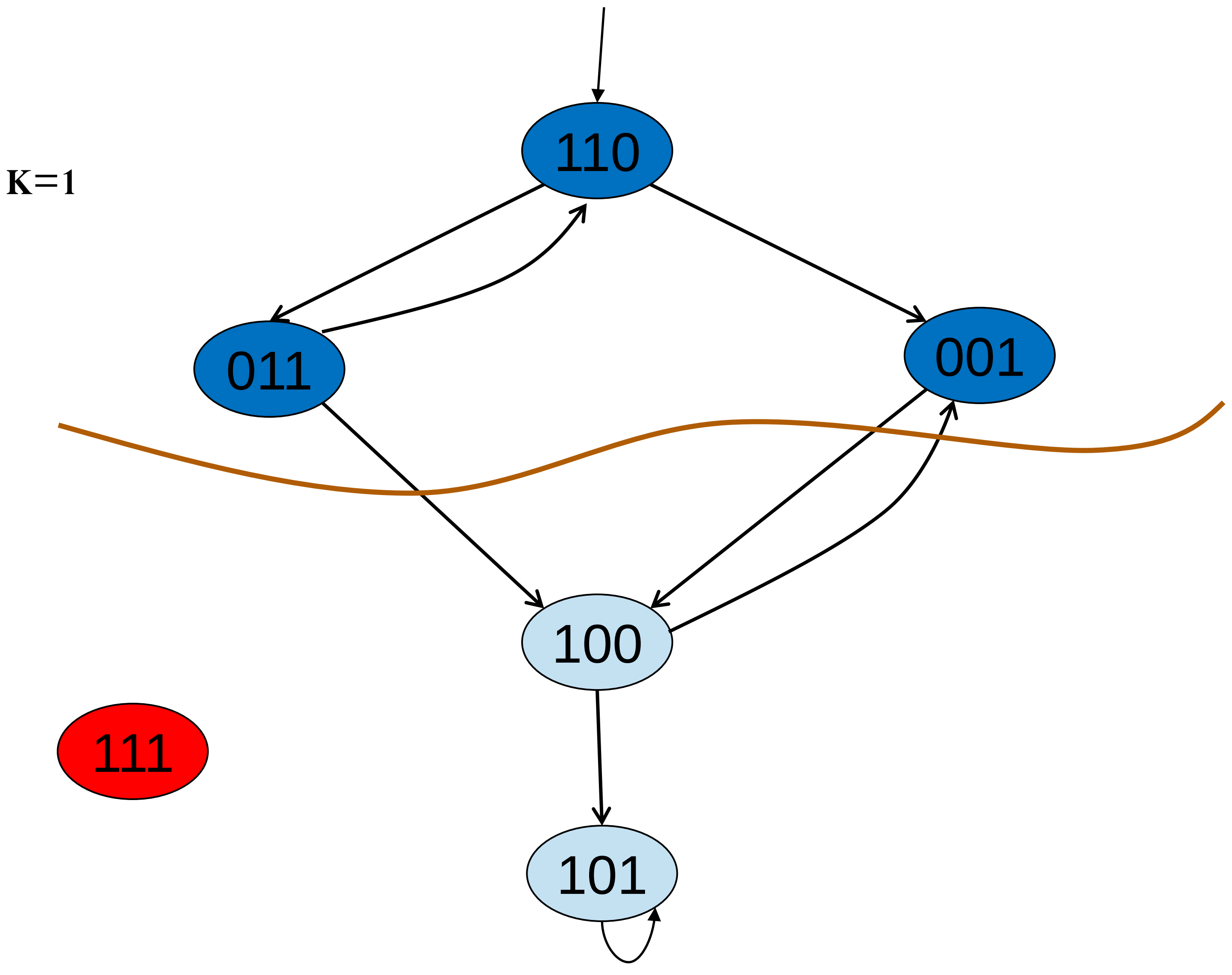


Bounded Model Checking using SAT

- Biere, Cimatti, Clarke, Zhu: **Symbolic Model Checking without BDDs.** TACAS 1999

- Bounded reachability: main idea
  - Let S be a model , G(p) a specification, and k a natural number
  - Build a Boolean formula B(S,p,k), such that
    - if B(S,p,k) is satisfiable, then S $\nvDash$ G(p), and the satisfying assignment is a counterexample
    - Otherwise (B(S,p,k) is not satisfiable), no counterexample of length k or less exists in the model

Bounded Model Checking (BMC)



Bounded Model Checking using SAT

- Let **I** be a Boolean formula representing the set of initial states
- Introduce k new sets of variables $\mathbf{V^1, \ldots, V^k}$
  - Use $\mathbf{V^0}$ for the original set of variables
- Let $\mathbf{T(V^i,V^{i+1})}$ represent the transition relation, in terms of the variables $\mathbf{V^i,V^{i+1}}$
- Let $\mathbf{p^i}$ represent p written in terms of $\mathbf{V^i}$
- Define B(S,p,k) to be

$$\mathbf{I \wedge T(V^0,V^1) \wedge T(V^1,V^2) \wedge \ldots \wedge T(V^{k-1},V^k) \wedge (\neg p^0 \vee \neg p^1 \vee \ldots \vee \neg p^k)}$$

Bounded Model Checking using SAT

- B(S,p,k) =

  $$I \wedge T(V^0,V^1) \wedge T(V^1,V^2) \wedge \ldots \wedge T(V^{k-1},V^k) \wedge (\neg p \vee \neg p^1 \vee \ldots \vee \neg p^k)$$

- What if B(S,p,k) is satisfiable?
- What if B(S,p,k) is unsatisfiable?

BMC: Example

K=1

init(V_1) := 1; init(V_2) := 1; init(V_3) := 0;

```
next(V1) :=  case
               V1 & V2 :  0;
               V3   :  1;
               else : {0,1};
             esac;
next(V2) :=  case
               V    : {0,1};
               else :   0;
             esac;
next(V3) := V1;
```

G(!(V1 & V2 & V3))

I = (1)(2)(-3)

P = (-1,-2,-3)

¬P = (1)(2)(3)

Introduce variables
1`,2`,3`

T=((1)(2) → (-1`)) ((-1,-2)(3)→(1`))
(-2)→(-2`)((1)→(3`)) ((-1)→(-3`))

Converting to CNF:

T = (-1,3`)(1,-3`),(2,-2`)(-1,-2,-1`)
(1,-3,1`)(2,-3,1`)

$B(S,p,1) = I \wedge T(V^0,V^1) \wedge (\neg p^0 \vee \neg p^1)$

$(\neg p^0 \vee \neg p^1)$=(1)(2)(3) ∨(1`)(2`)(3`)

Converting to CNF:

(4,5)(-4,1)(-4,2)(-4,3)(-5,1`)(-5,2`)(-5,3`)

B(S,p,1)= (1)(2)(-3) (-1,3`)(1,-3`),(2,-2`)(-1,-2,-1`) (1,-3,1`)(2,-3,1`)
(4,5)(-4,1)(-4,2)(-4,3)(-5,1`)(-5,2`)(-5,3`)

BMC – Summary

- Algorithm
  - Pick an initial k and increasing integer i
  - Loop:
    1. Build B(S,p,k)
    2. Check: is B(S,p,k) satisfiable?
       - If it is: return $S \nvDash G(p)$ + counterexample
    3. Set k := k+i

Unbounding BMC

- Many solutions exist, notably
  - Induction
    - Sheeran, Singh, Stålmarck 2000: "Checking Safety Properties Using Induction and a SAT-Solver"
  - Interpolation
    - McMillan 2003: "Interpolation and SAT-Based Model Checking