

# Bounded Model Checking

Hao Zheng

zheng@cse.usf.edu

Computer Science and Engineering  
University of South Florida

# Introduction

- Model Checking
  - Exhaustive verification.
  - Difficult to scale (**state explosion**).
- Bounded model checking (BMC) is targeted to find bugs, not to achieve the complete correctness proof.
  - Finds bugs in a bounded number of executions.
  - Can discover shallow bugs quickly.
  - Based on the latest advances in Boolean satisfiability solving.
  - State explosion is alleviated, but runtime may be a serious problem.

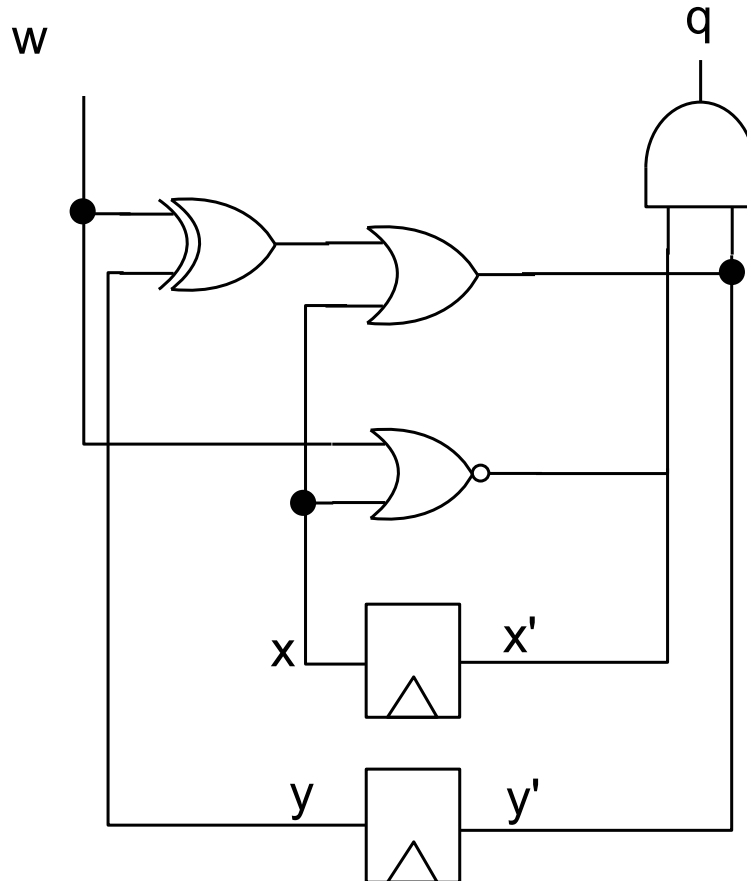
# BMC vs SMC

- Both operate on Boolean manipulations.
- BMC uses SAT solving while SMC uses OBDDs.
  - Both are exponential procedures (time or space).
- BMC can better solve some problems that cannot be solved by SMC, and vice versa.
- BMC cannot prove the absence of errors in large cases.
  - May require a very large bound  $k$ .

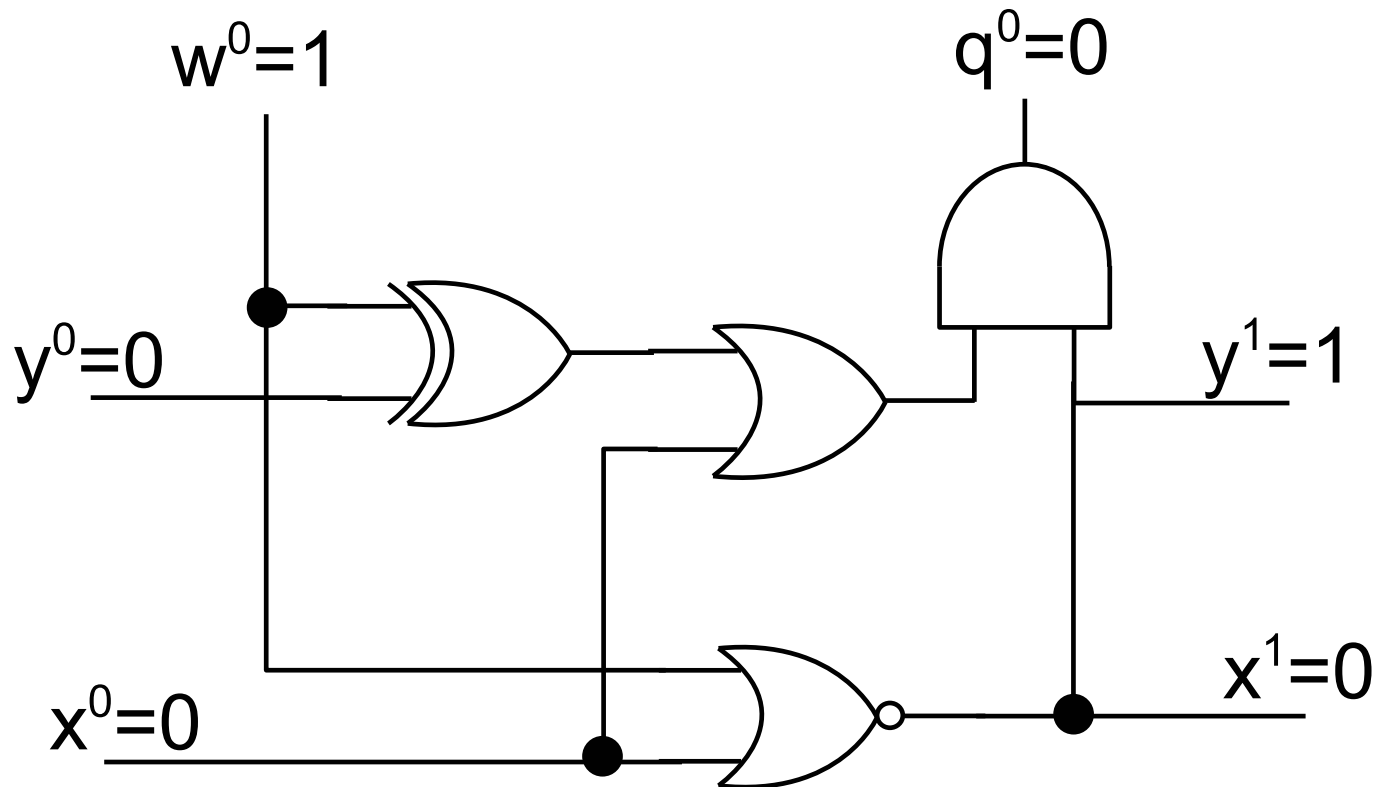
# An Illustrating Example

- Check if the circuit satisfies  $AG\neg q$ .

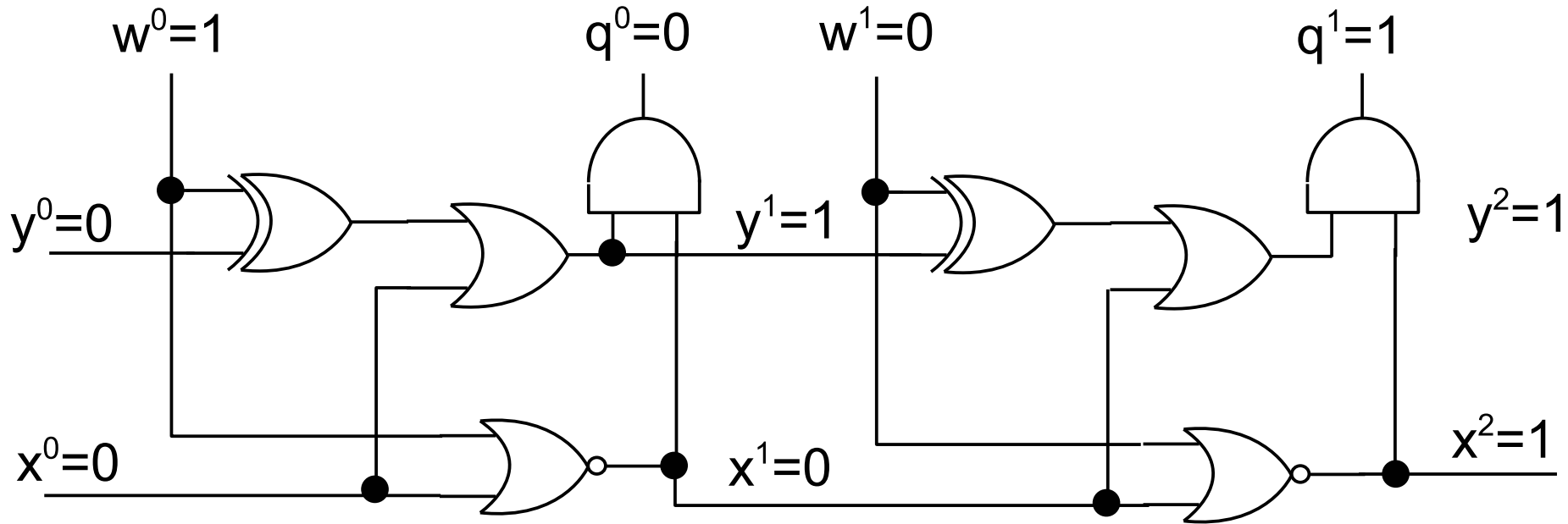
Initial state:  $x=0, y=0$ .



# Circuit State after Cycle 1

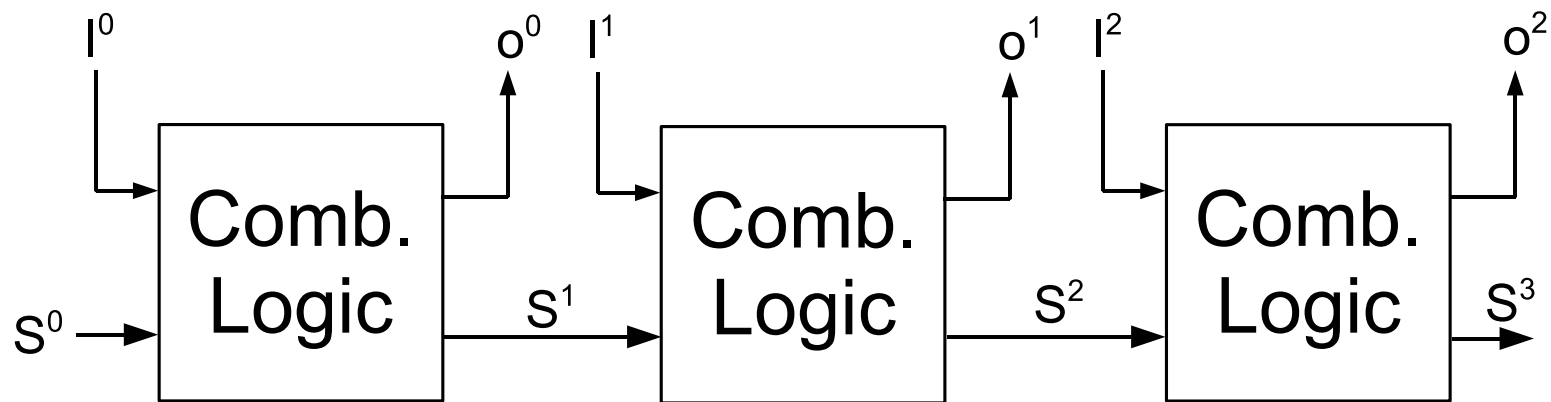
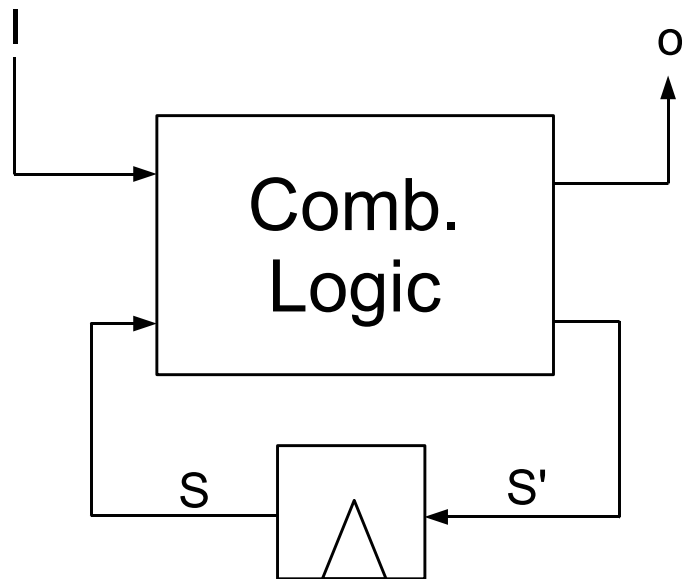


# Circuit State after Cycle 2

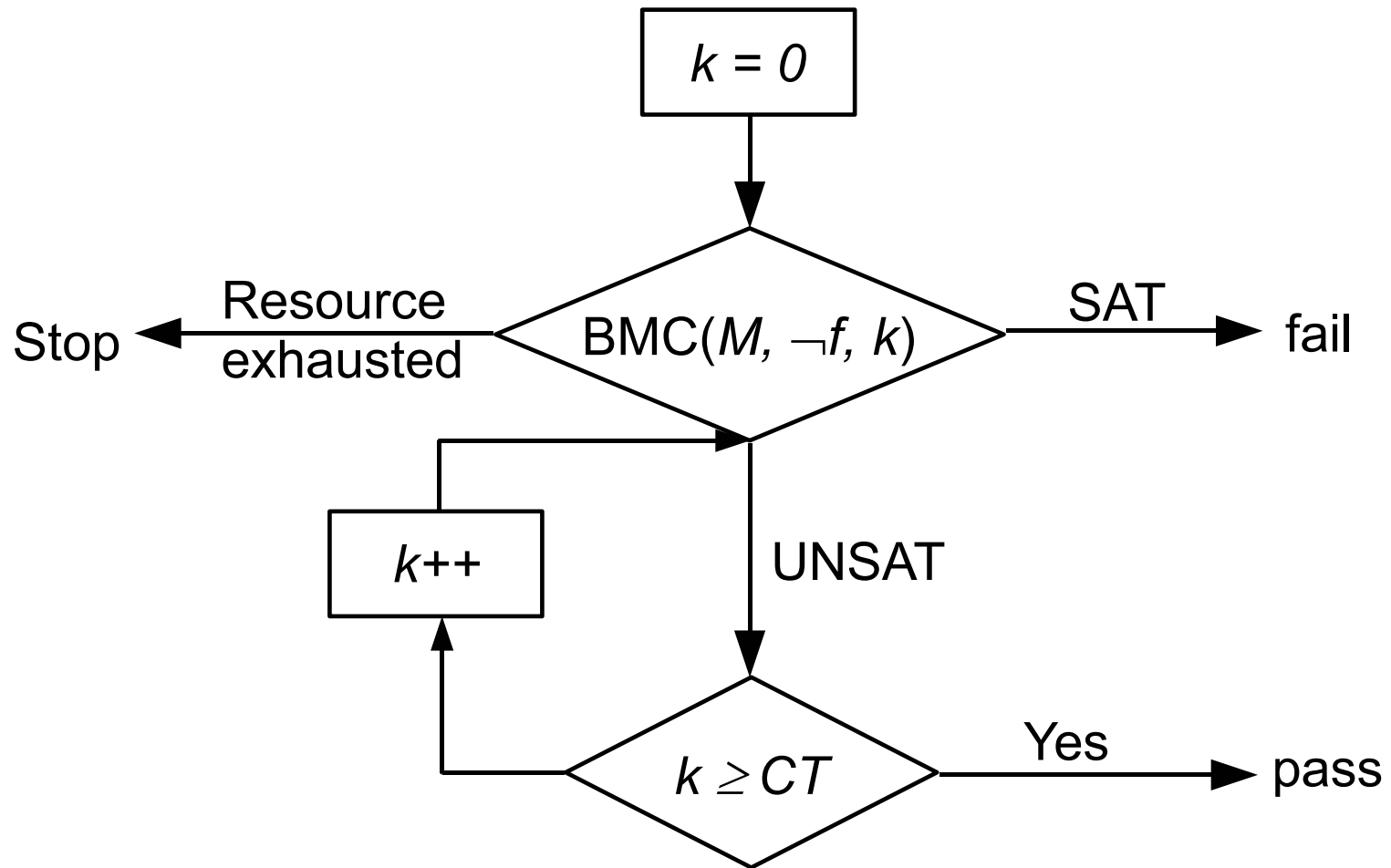


- $q = 1$  if  $w = 1$  in cycle 1 and  $w = 0$  in cycle 2.
- A counter-example to  $AG \neg q$  is a three state sequence.

# Big Picture of Bounded Model Checking

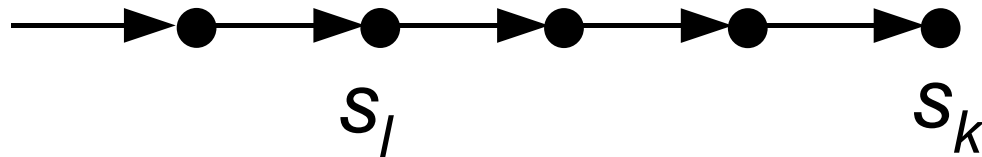


# How BMC Works

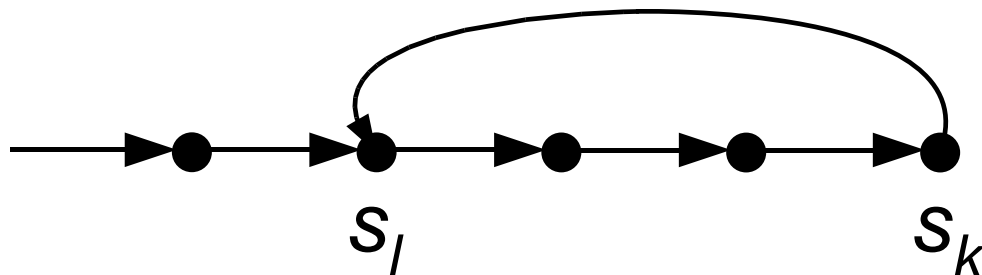


# $k$ -Bounded Path

- A  $k$ -bounded path is a sequence of  $k$  state transitions.



- A finite path is infinite if it has a back loop.



- A  $(k, l)$ -loop is a  $k$ -bounded path  $\rho$  such that  $R(s_k, s_l)$  holds.
- A path  $\rho$  is a  $k$ -loop if there exists  $0 \leq l \leq k$  such that  $\rho$  is a  $(k, l)$ -loop.

# Bounded Semantics of LTL Formulas

- Let  $\rho \models_k f$  denote the truth of the LTL formula  $f$  over the  $k$ -bounded path  $\rho$ .
  - Evaluate  $f$  only in the first  $k + 1$  states on  $\rho$ .
- Let  $\rho(i)$  denote the  $i^{\text{th}}$  state on  $\rho$ .
- Let  $\rho \models_k^i f$  denote the truth of  $f$  over the path from state  $\rho(i)$  to  $\rho(k)$ .
- If a path  $\rho$  is a  $k$ -loop,

$$\rho \models_k f \Leftrightarrow \rho \models f$$

.

# Bounded Semantics of LTL Formulas (2)

•  $\rho \models_k f \Leftrightarrow \rho \models_k^0$  where

$$\rho \models_k^i p \Leftrightarrow p \in L(\rho(i))$$

$$\rho \models_k^i \neg p \Leftrightarrow p \notin L(\rho(i))$$

$$\rho \models_k^i f \wedge g \Leftrightarrow \rho \models_k^i f \text{ and } \rho \models_k^i g$$

$$\rho \models_k^i f \vee g \Leftrightarrow \rho \models_k^i f \text{ or } \rho \models_k^i g$$

$$\rho \models_k^i \mathbf{G}f \quad \text{false}$$

$$\rho \models_k^i \mathbf{F}f \Leftrightarrow \exists i \leq j \leq k. \rho \models_k^j f$$

$$\rho \models_k^i \mathbf{X}f \Leftrightarrow i < k \text{ and } \rho \models_k^{i+1} f$$

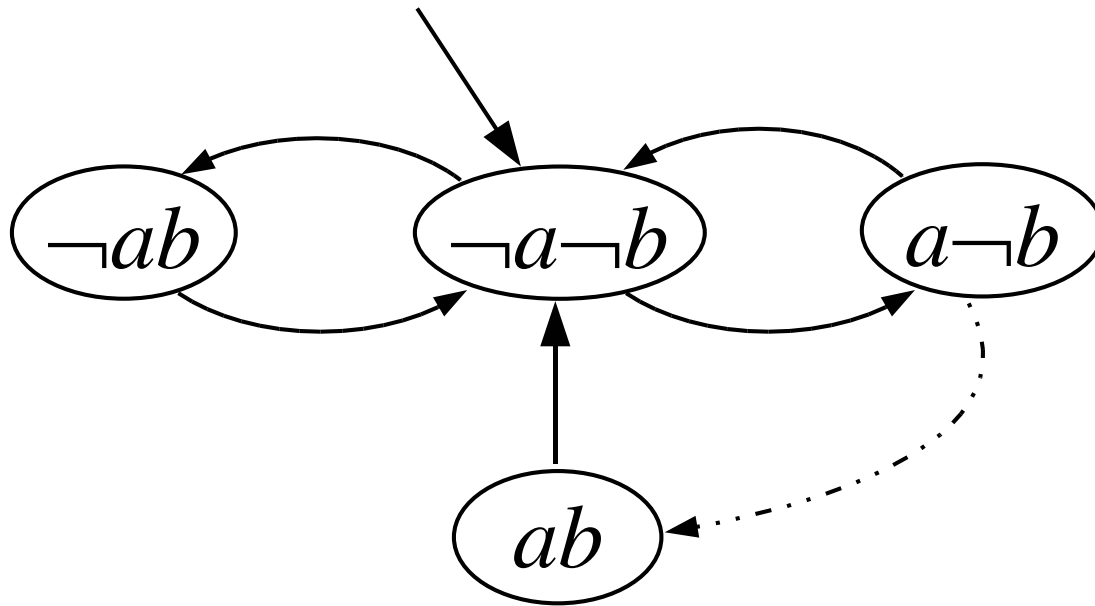
$$\rho \models_k^i f \mathbf{U}g \Leftrightarrow \exists i \leq j \leq k. \rho \models_k^j f \text{ and } \forall i \leq n \leq j. \rho \models_k^n g$$

# Bounded Model Checking of LTL

- Let  $M \models_k f$  denote a  $k$ -bounded model checking problem for the LTL formula  $f$ .
  - Formula  $f$  is evaluated on all  $k$ -bounded path.
- Let  $f$  be a LTL formula and  $\rho$  a path.  $\rho \models_k f \Rightarrow \rho \models f$ .
  - If for each  $\rho$  in  $M$  such that  $\rho \models_k f$ , then  $M \models f$  holds.
  - If there is a  $\rho$  in  $M$  such that  $\rho \models_k f$ , then  $M \models \neg f$  does not hold.
- $M \models f \Leftrightarrow \exists k \geq 0. M \models_k f$ .
  - There always exists a  $k$  such that the result of bounded model checking is equivalent to that of the complete one.

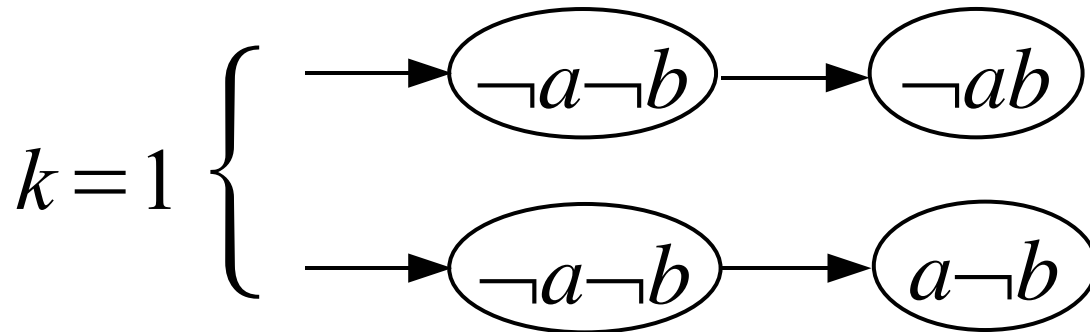
# An BMC Example

- $M \models G\neg(a \wedge b)$ .
- BMC checks if there is a bounded path on which  $F(a \wedge b)$  holds.

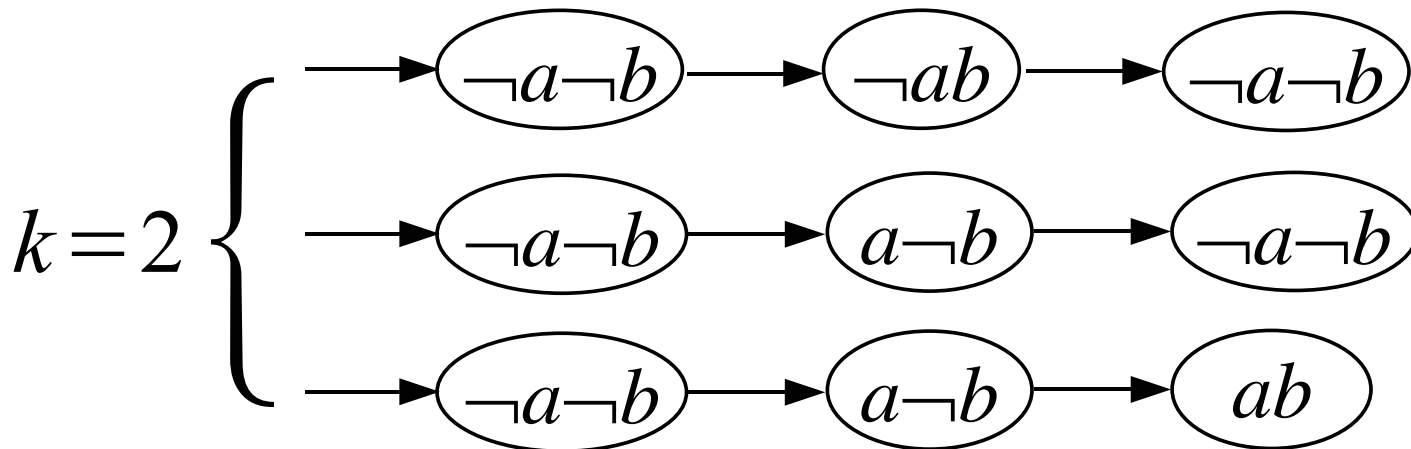


# An BMC Example (2)

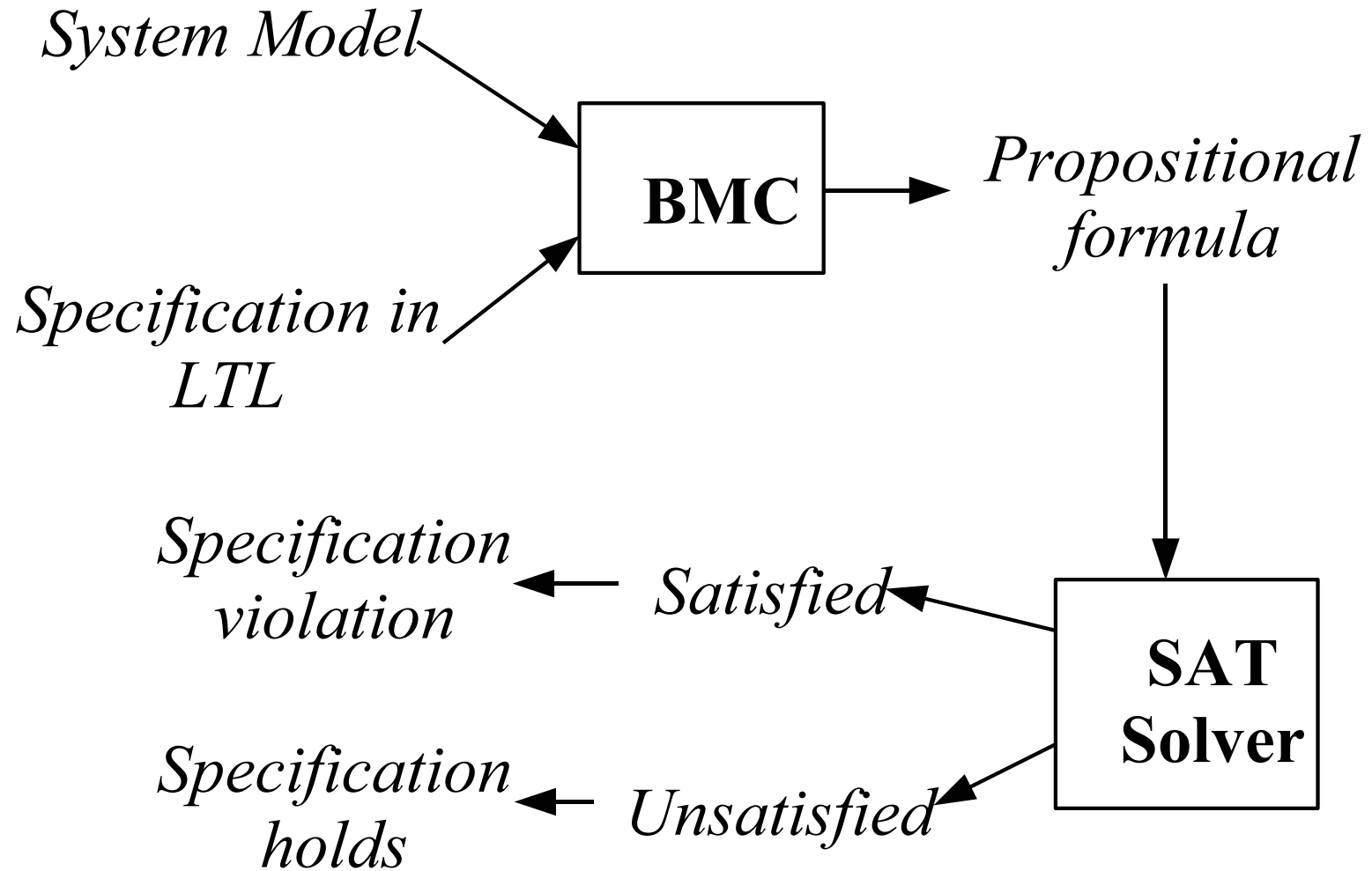
- $M \models_{k=1} \mathbf{G}\neg(a \wedge b)$ .



- $M \models_{k=2} \mathbf{G}\neg(a \wedge b)$ .



# Bounded Model Checking: Overview



# Boolean Encoding of Bounded Model Check

- Given a  $M$ , an LTL formula  $f$  and a bound  $k$ , generate a Boolean formula  $\llbracket M, f \rrbracket_k$ .
  - Encoding the constraints on  $k$ -paths in  $M$  such that these  $k$ -paths, if satisfiable, are witnesses of  $f$ .
- Three components of  $\llbracket M, f \rrbracket_k$ :
  - $\llbracket M \rrbracket_k$ : all  $k$ -paths in  $M$ .
  - $\llbracket f \rrbracket_k$  :: encoding of  $f$  on  $k$ -paths.
  - $l\llbracket f \rrbracket_k$  :: encoding of  $f$  on  $k$ -loops.

# Encoding of $\llbracket M \rrbracket_k$

- Unfolding of the transition relation

$$\llbracket M \rrbracket_k = I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}).$$

$\bigwedge_{i=0}^{k-1} R(s_i, s_{i+1})$  : encoding all  $k$ -paths of  $M$ .

$I(s_0)$  : constraints on the  $k$ -paths from the initial states.

# Encoding of $f$ on $k$ -Paths $\llbracket f \rrbracket_k$

- Inductive case:  $\forall i \leq k$

$$\llbracket p \rrbracket_k^i \equiv p(s_i)$$

$$\llbracket \neg p \rrbracket_k^i \equiv \neg p(s_i)$$

$$\llbracket f \wedge g \rrbracket_k^i \equiv \llbracket f \rrbracket_k^i \wedge \llbracket g \rrbracket_k^i$$

$$\llbracket f \vee g \rrbracket_k^i \equiv \llbracket f \rrbracket_k^i \vee \llbracket g \rrbracket_k^i$$

$$\llbracket \mathbf{G}f \rrbracket_k^i \equiv \llbracket f \rrbracket_k^i \wedge \llbracket \mathbf{G}f \rrbracket_k^{i+1}$$

$$\llbracket \mathbf{F}f \rrbracket_k^i \equiv \llbracket f \rrbracket_k^i \vee \llbracket \mathbf{F}f \rrbracket_k^{i+1}$$

$$\llbracket f \mathbf{U}g \rrbracket_k^i \equiv \llbracket g \rrbracket_k^i \vee (\llbracket f \rrbracket_k^i \wedge \llbracket f \mathbf{U}g \rrbracket_k^{i+1})$$

$$\llbracket \mathbf{X}f \rrbracket_k^i \equiv \llbracket f \rrbracket_k^{i+1}$$

- Base case:

$$\llbracket f \rrbracket_k^{k+1} \equiv \text{false}$$

# Encoding of $f$ on $k$ -Loops ${}_l \llbracket f \rrbracket_k$

- For  $k, l, i \geq 0$  and  $l, i \leq k$

$${}_l \llbracket p \rrbracket_k^i \equiv p(s_i)$$

$${}_l \llbracket \neg p \rrbracket_k^i \equiv \neg p(s_i)$$

$${}_l \llbracket f \wedge g \rrbracket_k^i \equiv {}_l \llbracket f \rrbracket_k^i \wedge {}_l \llbracket g \rrbracket_k^i$$

$${}_l \llbracket f \vee g \rrbracket_k^i \equiv {}_l \llbracket f \rrbracket_k^i \vee {}_l \llbracket g \rrbracket_k^i$$

$${}_l \llbracket \mathbf{G}f \rrbracket_k^i \equiv {}_l \llbracket f \rrbracket_k^i \wedge {}_l \llbracket \mathbf{G}g \rrbracket_k^{\text{succ}(i)}$$

$${}_l \llbracket \mathbf{F}f \rrbracket_k^i \equiv {}_l \llbracket f \rrbracket_k^i \vee {}_l \llbracket \mathbf{F}f \rrbracket_k^{\text{succ}(i)}$$

$${}_l \llbracket f \mathbf{U}g \rrbracket_k^i \equiv {}_l \llbracket g \rrbracket_k^i \vee ({}_l \llbracket f \rrbracket_k^i \wedge {}_l \llbracket f \mathbf{U}g \rrbracket_k^{\text{succ}(i)})$$

$${}_l \llbracket \mathbf{X}f \rrbracket_k^i \equiv {}_l \llbracket f \rrbracket_k^{\text{succ}(i)}$$

where  $\text{succ}(i)$  is defined as follows.

$$\text{succ}(i) = \begin{cases} i + 1 & \text{if } i < k \\ l & \text{if } i = k \end{cases}$$

# Encoding of the Looping Conditions

- A loop forms if there is a transition from  $s_k$  back to  $s_i$  for  $i \leq k$ .
- There are  $k$  states  $s_i$ , the looping condition of  $k$ -path is a disjunction of  $k$  different looping conditions.

$$L_k = \bigvee_{i=0}^k {}_iL_k$$

where

$${}_iL_k = R(s_k, s_i).$$

# General Translation

- Let  $M$  be a Kripke structure,  $f$  an LTL formula, and  $k \geq 0$  a bound.

$$\llbracket M, f \rrbracket_k = \llbracket M \rrbracket_k \wedge \left( (\neg L_k \wedge \llbracket f \rrbracket_k^0) \wedge \bigvee_{i=0}^k ({}_i L_k \wedge {}_i \llbracket f \rrbracket_k^0) \right)$$

- $\llbracket M, f \rrbracket_k$  is satisfiable if, and only if,  $M \models_k \mathbf{E}(f)$ .
- $\llbracket M, f \rrbracket_k$  is satisfiable if, and only if,  $M \models_k \neg f$  does not hold.

# An BMC Example: Translation

- $M \models \mathbf{G}\neg(a \wedge b)$  for  $k = 2$ .
- $M = (I, R)$  where

$$I = \neg a \wedge \neg b$$

$$R = (\neg a \wedge \neg b \wedge a' \wedge \neg b') \vee (\neg a \wedge \neg b \wedge \neg a' \wedge b') \vee$$
$$(\neg a \wedge b \wedge \neg a' \wedge \neg b') \vee (a \wedge \neg b \wedge \neg a' \wedge \neg b') \vee$$
$$(a \wedge \neg b \wedge a' \wedge b') \vee (a \wedge b \wedge \neg a' \wedge \neg b')$$

