

# SAT and Model Checking

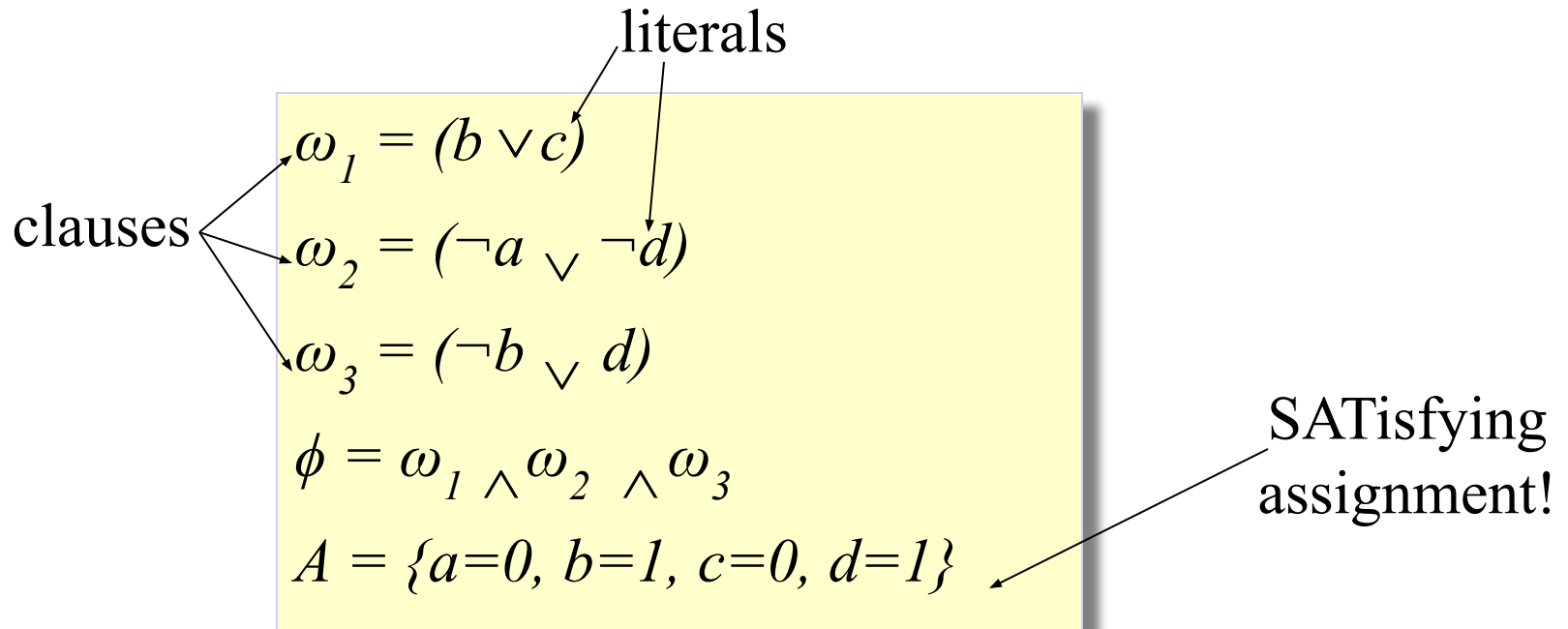
# Bounded Model Checking (BMC)

Biere, Cimatti, Clarke, Zhu, 1999

- A.I. Planning problems: *can we reach a desired state in  $k$  steps?*
- Verification of *safety* properties: *can we find a bad state in  $k$  steps?*
- Verification: *can we find a counterexample in  $k$  steps ?*

# What is SAT?

Given a propositional formula in CNF, find if there exists an assignment to Boolean variables :that makes the formula true



# BMC idea

Given: transition system  $M$ , temporal logic formula  $f$ , and user-supplied time bound  $k$

Construct propositional formula  $\Omega(k)$  that is *satisfiable* iff  $f$  is valid along a path of length  $k$

Path of length  $k$ : 
$$I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1})$$

Say  $f = \mathbf{EF} p$  and  $k = 2$ , then

$$\Omega(2) = I(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2) \wedge (p_0 \vee p_1 \vee p_2)$$

What if  $f = \mathbf{AG} p$  ?

# BMC idea (cont'd)

**AG**  $p$  means  $p$  must hold in every state along any path of length  $k$

We take

$$\neg\Omega(k) = (I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1})) \rightarrow \bigwedge_{i=0}^k p_i$$

So

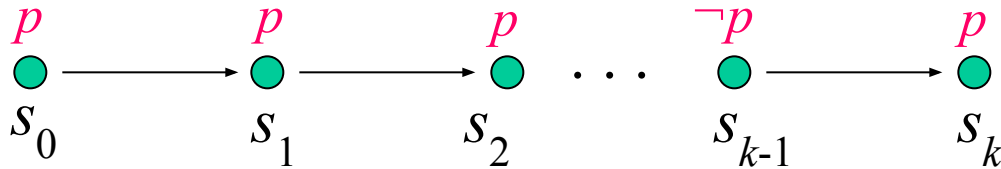
$$\Omega(k) = I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^k \neg p_i$$

That means we look for counterexamples

# Safety-checking as BMC

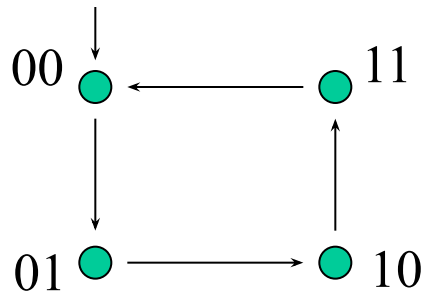
$p$  is preserved up to  $k$ -th transition iff  $\Omega(k)$  is unsatisfiable:

$$\Omega(k) = I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^k \neg p$$



If satisfiable, satisfying assignment gives counterexample to the safety property.

# Example: a two bit counter



Initial state:  $I : \neg l \wedge \neg r$

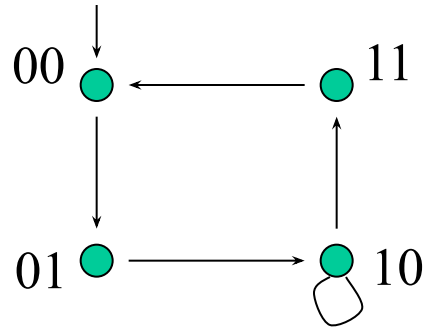
Transition:  $R: \begin{pmatrix} l' = (l \neq r) \wedge \\ r' = \neg r \end{pmatrix}$

**Safety** property:  $\mathbf{AG} (\neg l \vee \neg r)$

$$\Omega(2) : (\neg l_0 \wedge \neg r_0) \wedge \begin{pmatrix} l_1 = (l_0 \neq r_0) \wedge r_1 = \neg r_0 \wedge \\ l_2 = (l_1 \neq r_1) \wedge r_2 = \neg r_1 \end{pmatrix} \wedge \begin{pmatrix} (l_0 \wedge r_0) \vee \\ (l_1 \wedge r_1) \vee \\ (l_2 \wedge r_2) \end{pmatrix}$$

$\Omega(2)$  is unsatisfiable.  $\Omega(3)$  is satisfiable.

# Example: another counter



$$I : \neg l \wedge \neg r \quad R: \left( \begin{array}{l} l' = (l \neq r) \wedge \\ r' = \neg r \end{array} \right) \vee \left( \begin{array}{l} l' = l \wedge \\ r' = r \wedge \\ l \wedge \neg r \end{array} \right)$$

**Liveness** property: **AF**  $(l \wedge r)$

Check: **EG**  $(\neg l \vee \neg r)$

$$\Omega(2) = I(s_0) \wedge \bigwedge_{i=0}^1 R(s_i, s_{i+1}) \wedge \bigvee_{i=0}^2 (\neg l_i \vee \neg r_i) \wedge loop$$

where

$$loop = R(s_2, s_3) \wedge (s_3 = s_0 \vee s_3 = s_1 \vee s_3 = s_2)$$

$\Omega(2)$  is satisfiable

Satisfying assignment gives counterexample to the liveness property



# What BMC with SAT Can Do

- All LTL
- ACTL and ECTL
- In principle, all CTL and even mu-calculus
  - efficient universal quantifier elimination or fixpoint computation is an active area of research

# How big should $k$ be?

- For every model  $M$  and LTL property  $\phi$  there exists  $k$  s.t.

$$M \models_k \phi \rightarrow M \models \phi$$

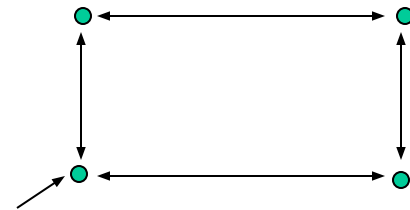
- The minimal such  $k$  is the *Completeness Threshold (CT)*

# How big should $k$ be?

- *Diameter*  $d$  = longest shortest path from an initial state to any other reachable state.
- *Recurrence Diameter*  $rd$  = longest loop-free path.
- $rd \leq d$

$$d = 2$$

$$rd = 3$$



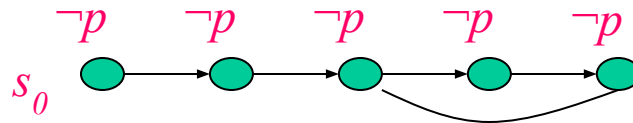
# How big should $k$ be?

- *Theorem:* for  $Gp$  properties  $CT = d$



# How big should $k$ be?

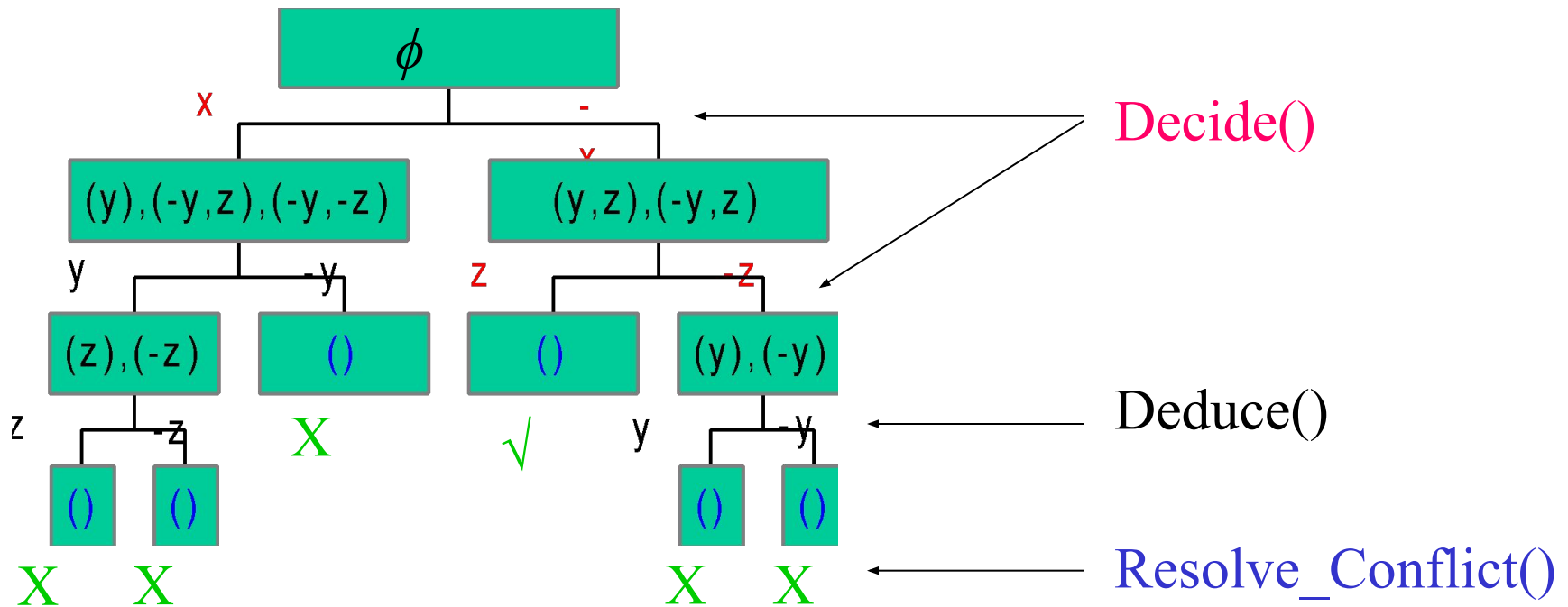
- *Theorem*: for **Fp** properties  $CT = rd$



- ◆ Open Problem: The value of  $CT$  for general Linear Temporal Logic properties is unknown

# A basic SAT solver

Given  $\phi$  in CNF:  $(x,y,z),(-x,y),(-y,z),(-x,-y,-z)$



# Basic Algorithm

While (true)

{

if (!Decide()) return (SAT);

while (!Deduce())

if (!Resolve\_Conflict()) return (UNSAT);

}

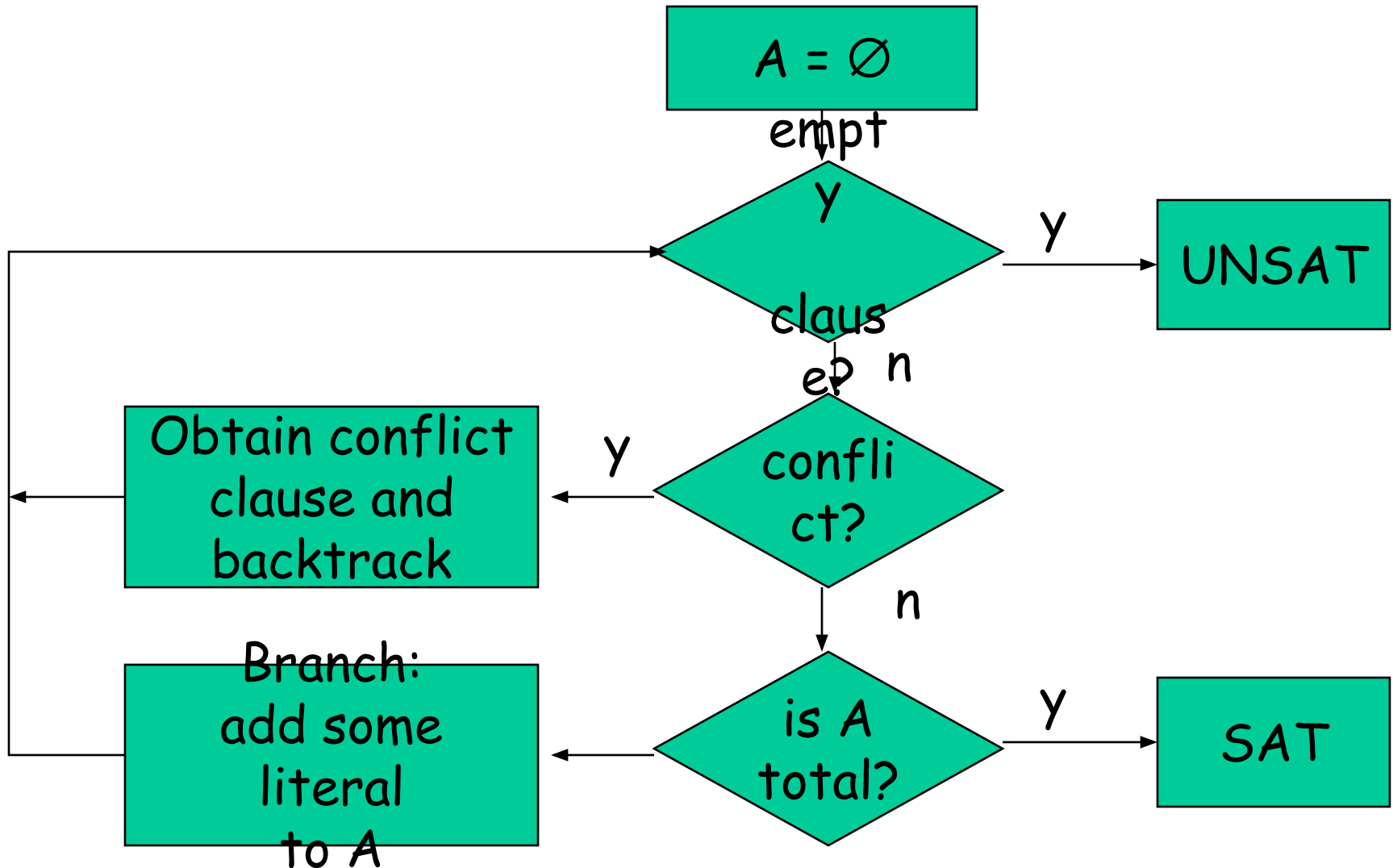
Choose the next  
variable and value.  
Return False if all  
variables are assigned

Apply unit clause rule.  
Return False if reached  
a conflict

Backtrack until  
no conflict.  
Return False if  
impossible

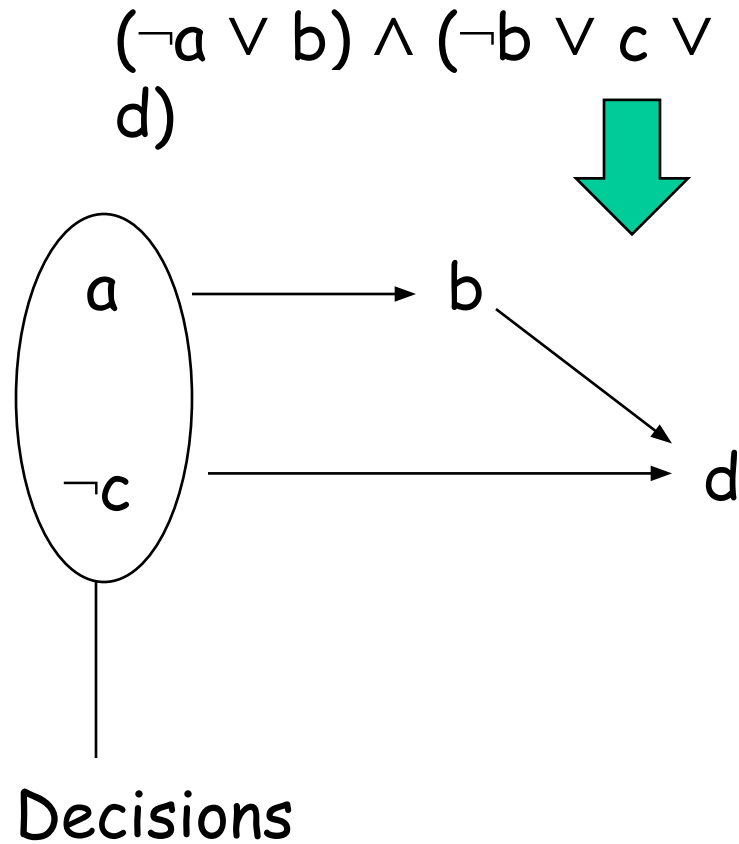
# DPLL-style SAT solvers

SATO, GRASP, CHAFF, BERKMIN



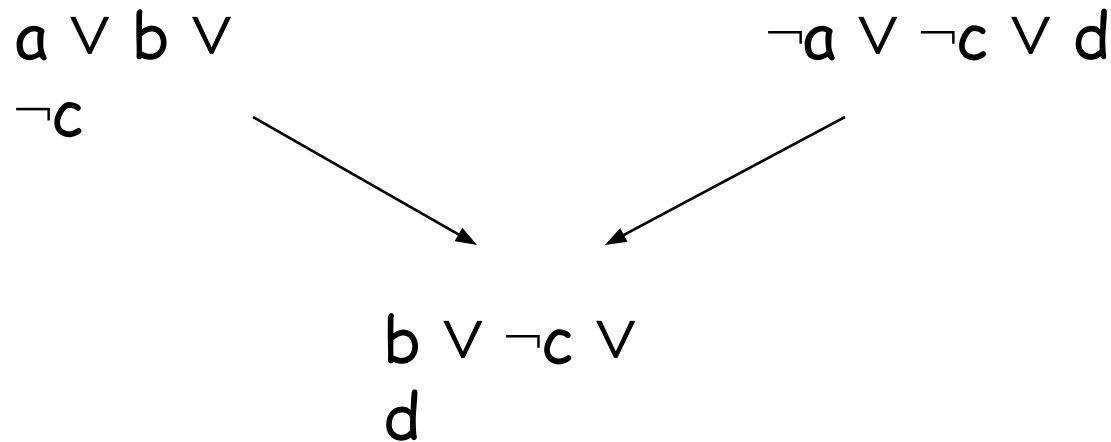


# The Implication Graph



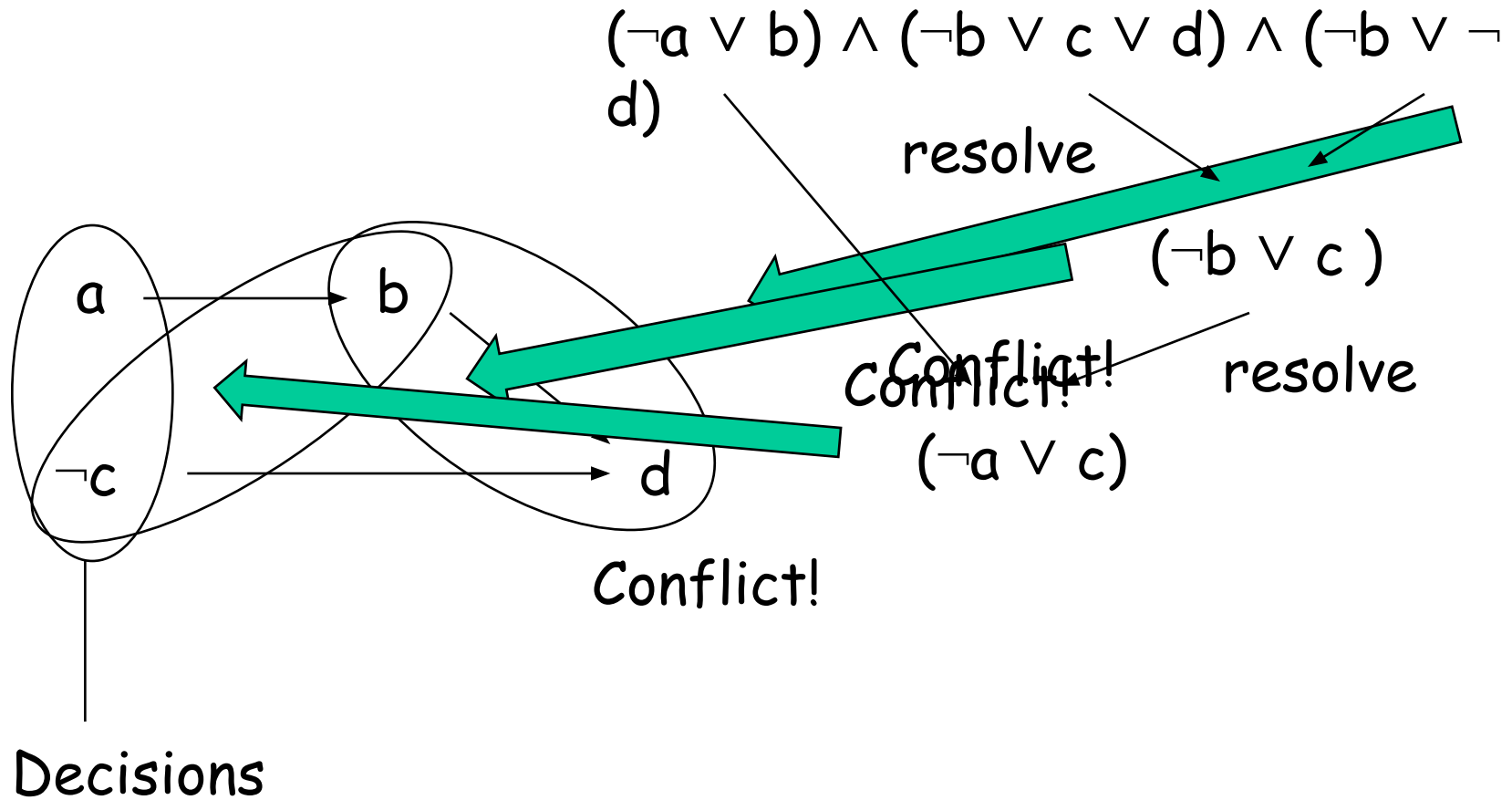
Assignment:  $a \wedge b \wedge \neg c \wedge d$

# Resolution



When a conflict occurs, the implication graph is used to guide the resolution of clauses, so that the same conflict will not occur again.

# Conflict clauses



Assignment:  $a \wedge b \wedge \neg c \wedge d$

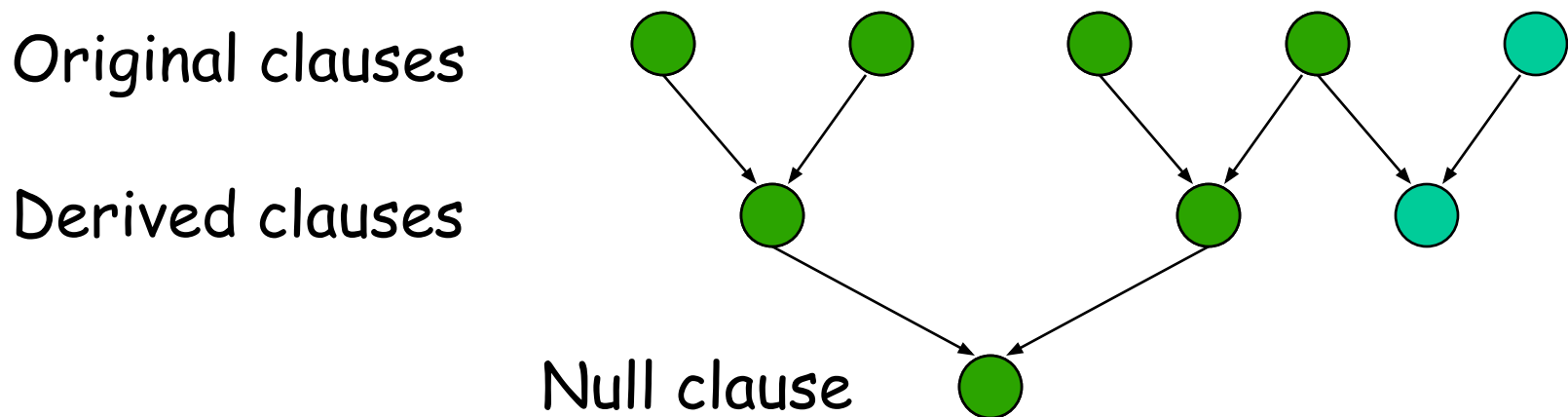
# Conflict Clauses (cont.)

- Conflict clauses:
  - Are generated by resolution
  - Are implied by existing clauses
  - Are in conflict with the current assignment
  - Are safely added to the clause set

Many heuristics are available for determining when to terminate the resolution process.

# Generating refutations

- Refutation = a proof of the null clause
  - Record a DAG containing all resolution steps performed during conflict clause generation.
  - When null clause is generated, we can extract a proof of the null clause as a resolution DAG.



# Unbounded Model Checking

- A variety of methods to exploit SAT and BMC for unbounded model checking:
  - **Completeness Threshold**
  - $k$  - induction
  - Abstraction (refutation proofs useful here)
  - Exact and over-approximate image computations (refutation proofs useful here)
  - Use of Craig interpolation

# Conclusions: BDDs vs. SAT

- Many models that cannot be solved by BDD symbolic model checkers, can be solved with an optimized SAT Bounded Model Checker.
- The reverse is true as well.
- BMC with SAT is faster at finding shallow errors and giving short counterexamples.
- BDD-based procedures are better at proving absence of errors.

# Acknowledgements

“Exploiting SAT Solvers in Unbounded Model Checking” by K. McMillan, tutorial presented at CAV’03

“Tuning SAT-checkers for Bounded Model Checking” and “Heuristics for Efficient SAT solving” by O. Strichman

Slides originally prepared for 2108 by Mihaela Gheorghiu.