

Decision Procedures

An Algorithmic Point of View

Lecture 4

SAT Internals

Part I

✓ ■ Reminders -

- ✓ What is Logic
- ✓ Proofs by deduction
- ✓ Proofs by enumeration
- ✓ Decidability, Soundness and Completeness
- ✓ some notes on Propositional Logic

■ Deciding Propositional Logic

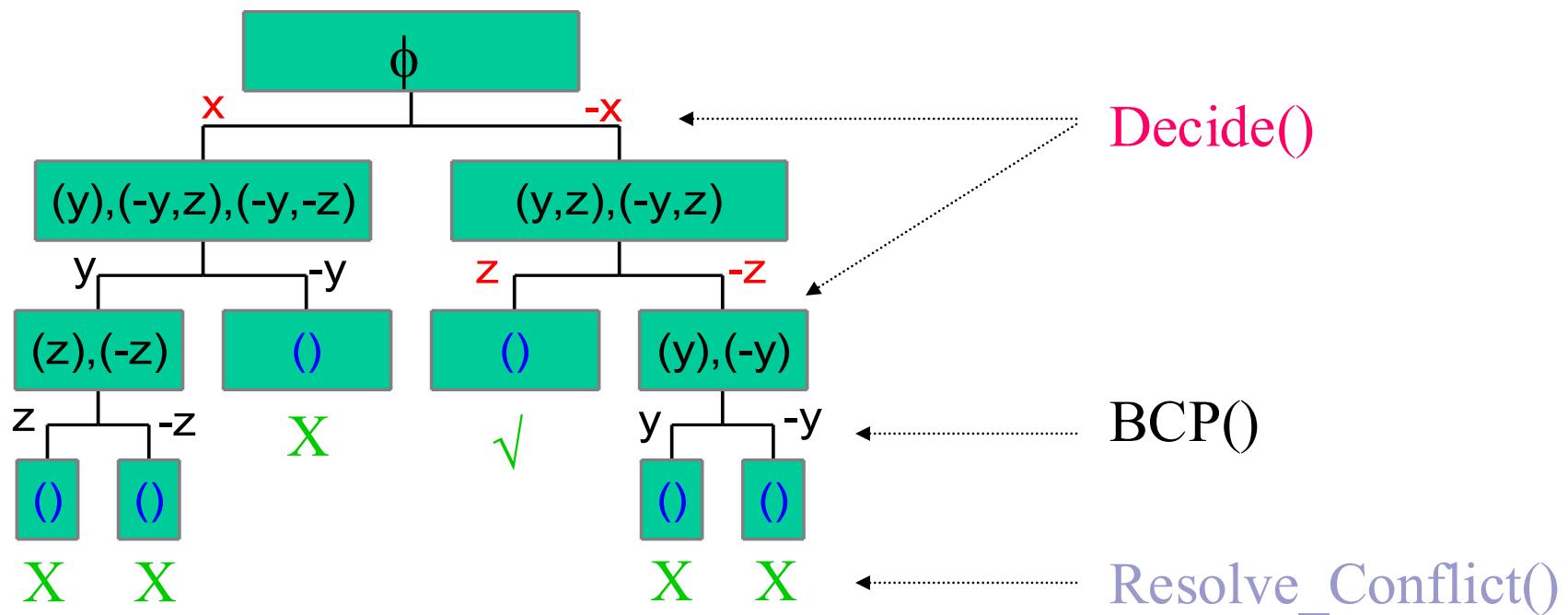
- SAT tools
- BDDs

Next: Deciding Propositional Formulas

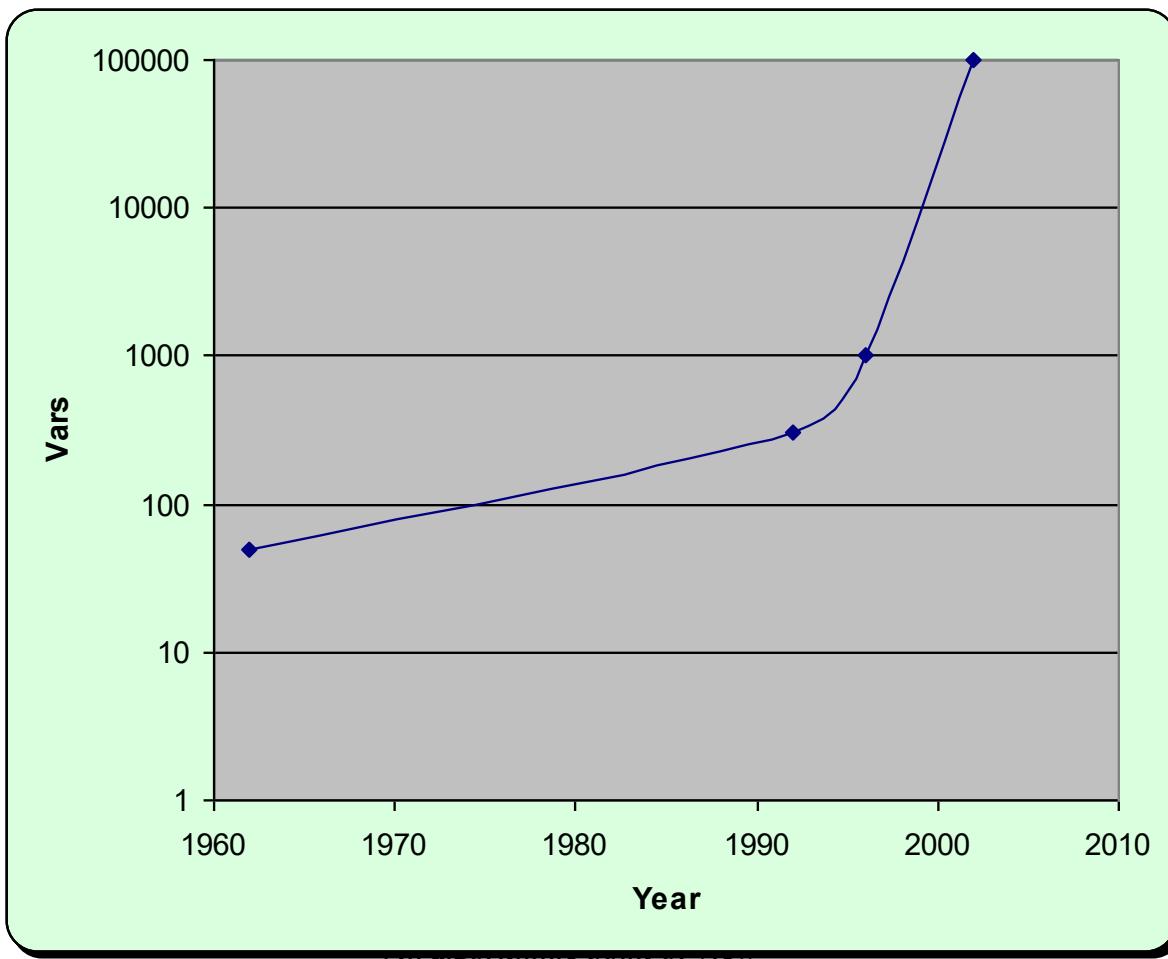
- SAT solvers
- Binary Decision Diagrams

A Basic SAT algorithm

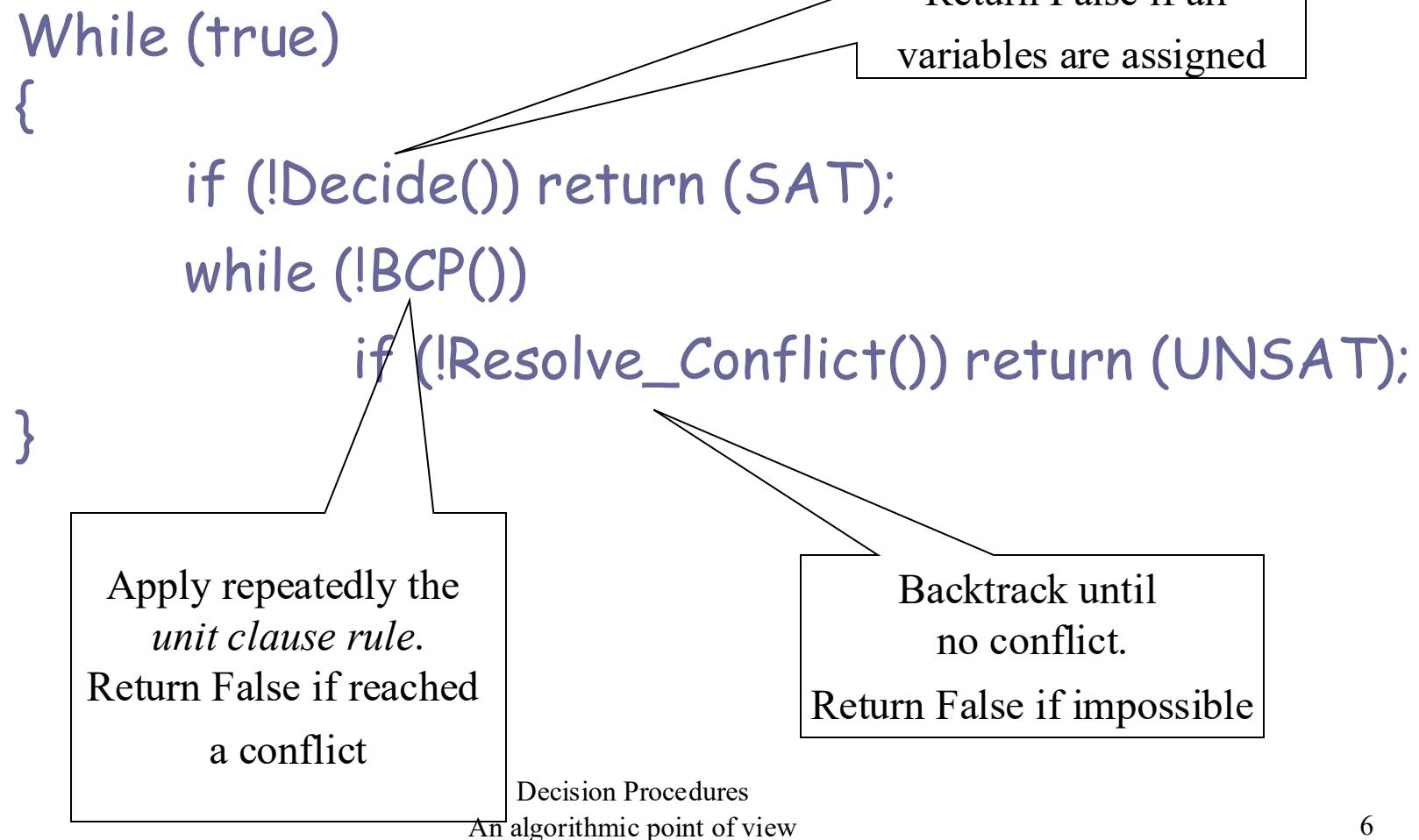
- Given ϕ in CNF: $(x,y,z),(-x,y),(-y,z),(-x,-y,-z)$



SAT made some progress...



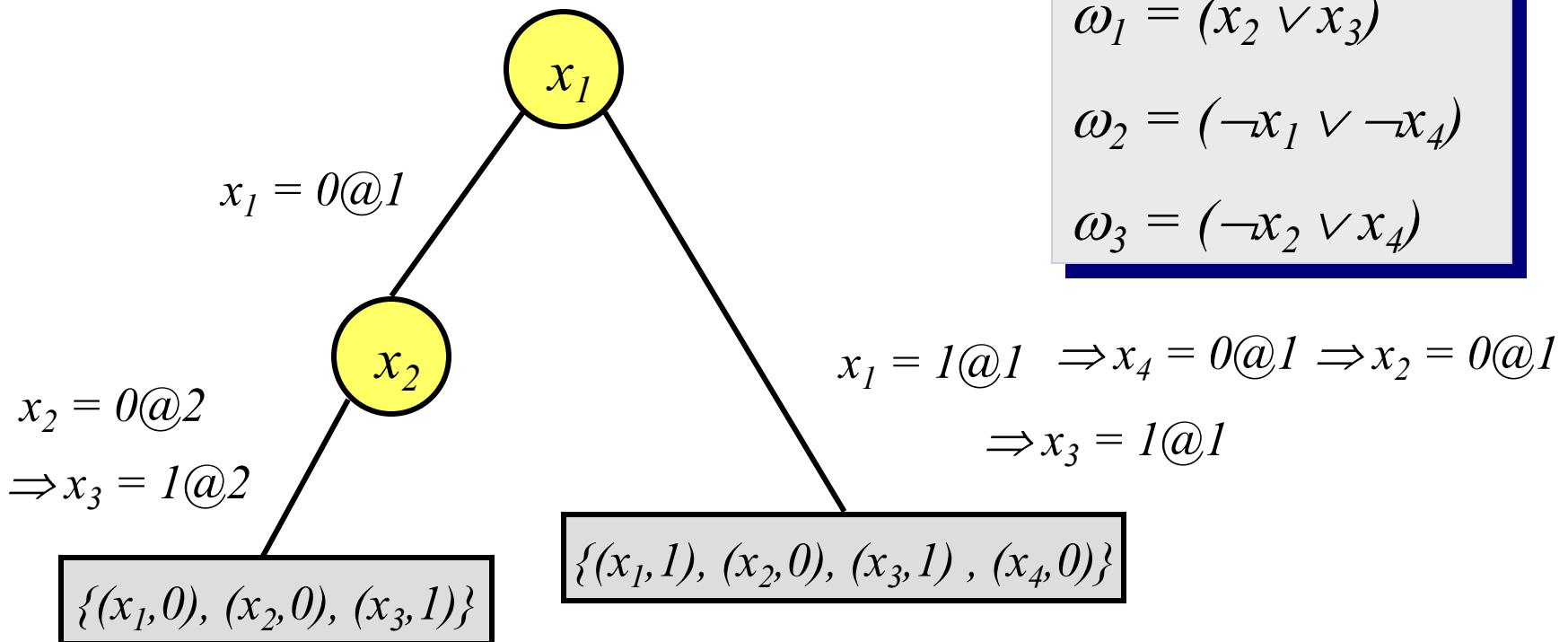
A Basic SAT algorithm



Basic Backtracking Search

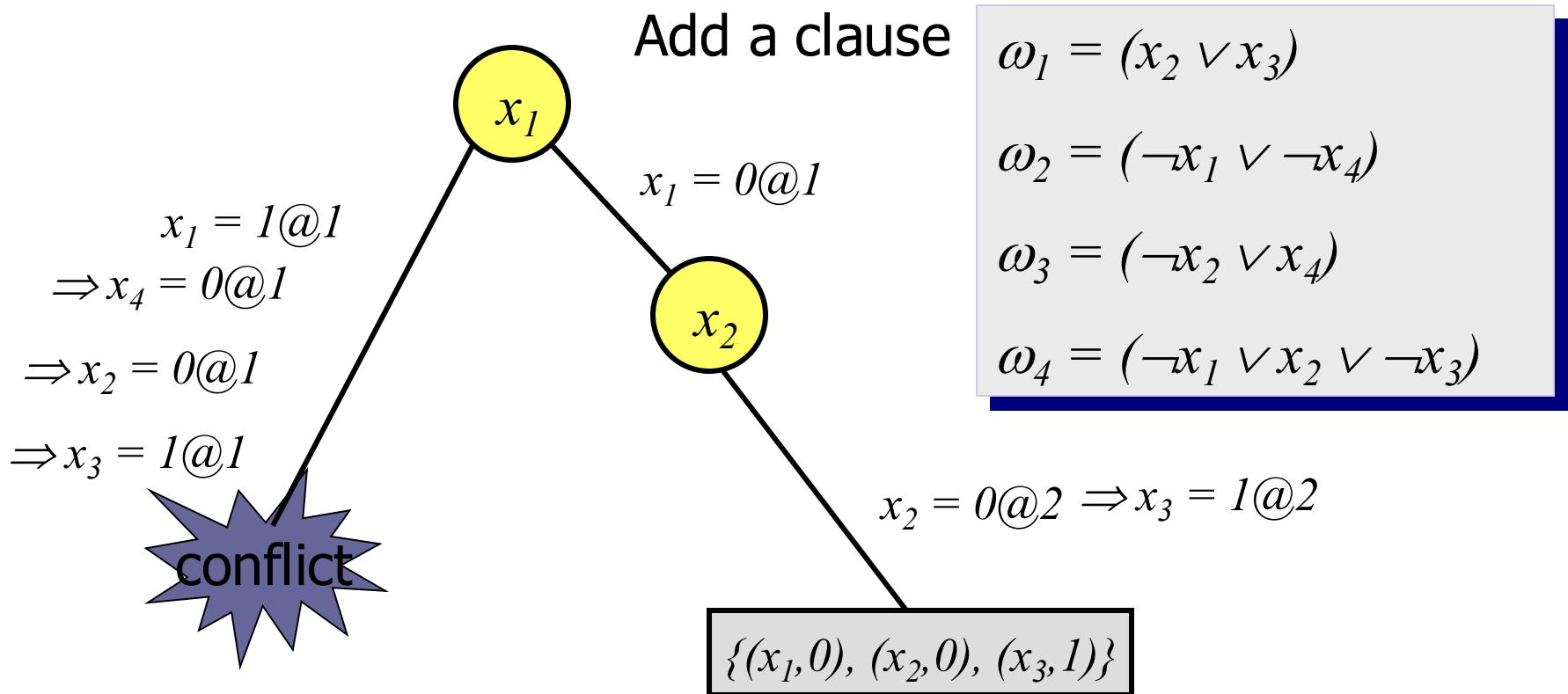
- Organize the search in the form of a decision tree
 - Each node corresponds to a decision
 - Definition: Decision Level (DL) is the depth of the node in the decision tree.
 - Notation: $x=v@d$
 $x \in \{0,1\}$ is assigned to v at decision level d

Backtracking Search in Action



No backtrack in this example,
regardless of the decision!

Backtracking Search in Action



Status of a clause

- A clause can be
 - Satisfied: at least one literal is satisfied
 - Unsatisfied: all literals are assigned but none are satisfied
 - Unit: all but one literals are assigned but none are satisfied
 - Unresolved: all other cases
- Example: $C = (x_1 \vee x_2 \vee x_3)$

x_1	x_2	x_3	C
1	0		Satisfied
0	0	0	Unsatisfied
0	0		Unit
	0		Unresolved

Decision heuristics - DLIS

DLIS (Dynamic Largest Individual Sum) – choose the assignment that increases the most the number of satisfied clauses

- For a given variable x :
 - C_{xp} – # unresolved clauses in which x appears positively
 - C_{xn} - # unresolved clauses in which x appears negatively
 - Let x be the literal for which C_{xp} is maximal
 - Let y be the literal for which C_{yn} is maximal
 - If $C_{xp} > C_{yn}$ choose x and assign it TRUE
 - Otherwise choose y and assign it FALSE
- Requires l (# literals) queries for each decision.

Decision heuristics - JW

Jeroslow-Wang method

Compute for every clause ω and every variable l (in each phase):

- $$J(l) := \sum_{l \in \omega, \omega \in \varphi} 2^{-|\omega|}$$
- Choose a variable l that maximizes $J(l)$.
- This gives an exponentially higher weight to literals in shorter clauses.

Pause... ||

- We will see other (more advanced) decision Heuristics soon.
- These heuristics are integrated with a mechanism called Learning with Conflict-Clauses, which we will learn next.

Implication graphs and learning: option #1

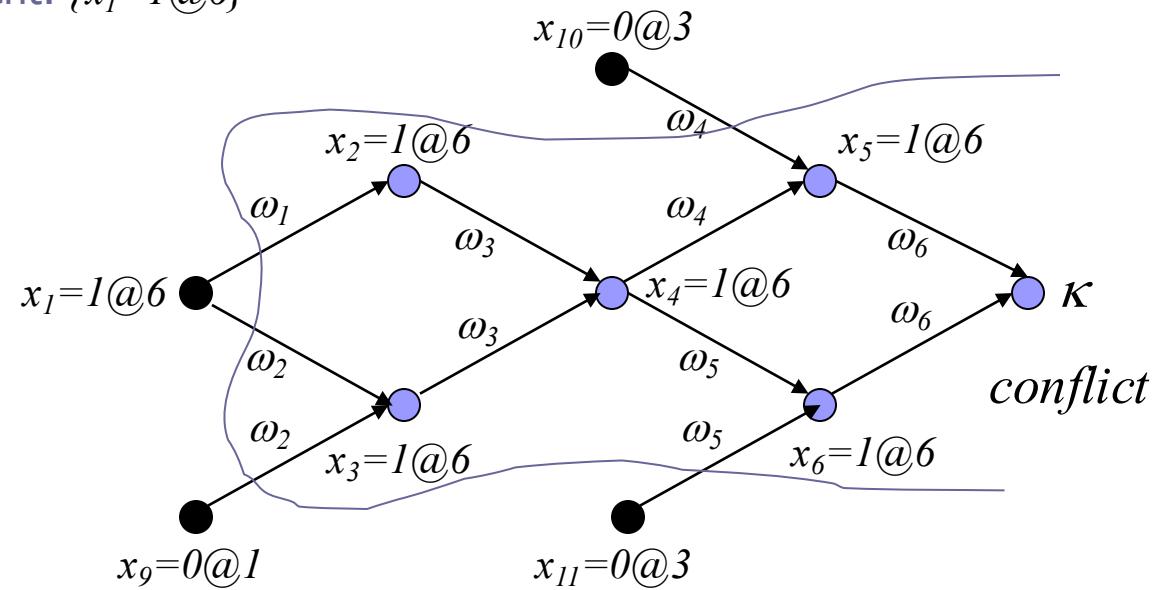
Current truth assignment: $\{x_9=0@1, x_{10}=0@3, x_{11}=0@3, x_{12}=1@2, x_{13}=1@2\}$

Current decision assignment: $\{x_1=1@6\}$

```

 $\omega_1 = (\neg x_1 \vee x_2)$ 
 $\omega_2 = (\neg x_1 \vee x_3 \vee x_9)$ 
 $\omega_3 = (\neg x_2 \vee \neg x_3 \vee x_4)$ 
 $\omega_4 = (\neg x_4 \vee x_5 \vee x_{10})$ 
 $\omega_5 = (\neg x_4 \vee x_6 \vee x_{11})$ 
 $\omega_6 = (\neg x_5 \vee \neg x_6)$ 
 $\omega_7 = (x_1 \vee x_7 \vee \neg x_{12})$ 
 $\omega_8 = (x_1 \vee x_8)$ 
 $\omega_9 = (\neg x_7 \vee \neg x_8 \vee \neg x_{13})$ 

```



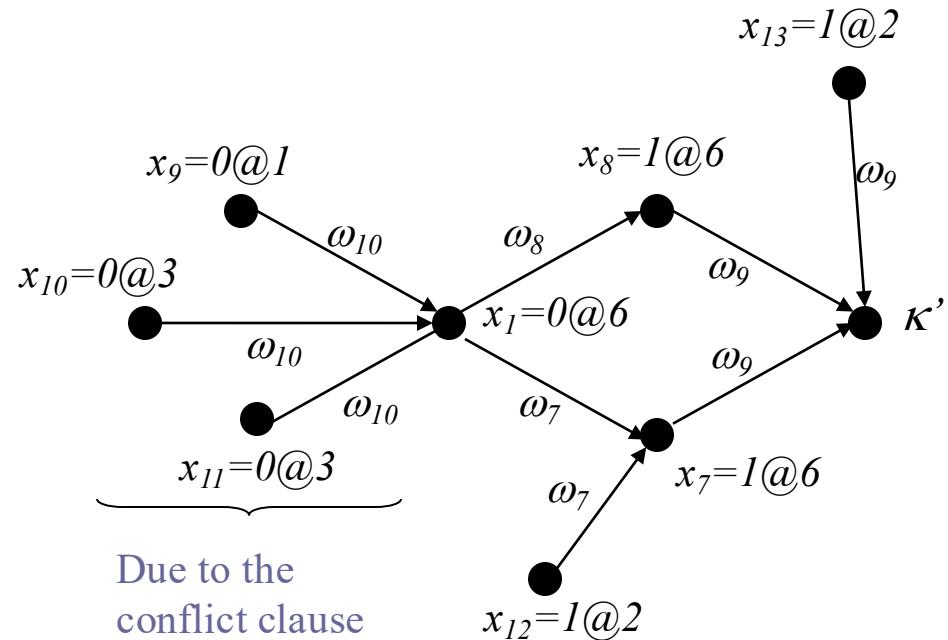
We learn the *conflict clause* $\omega_{10} : (\neg x_1 \vee x_9 \vee x_{11} \vee x_{10})$

Implication graph, flipped assignment option #1

```

 $\omega_1 = (\neg x_1 \vee x_2)$ 
 $\omega_2 = (\neg x_1 \vee x_3 \vee x_9)$ 
 $\omega_3 = (\neg x_2 \vee \neg x_3 \vee x_4)$ 
 $\omega_4 = (\neg x_4 \vee x_5 \vee x_{10})$ 
 $\omega_5 = (\neg x_4 \vee x_6 \vee x_{11})$ 
 $\omega_6 = (\neg x_5 \vee x_6)$ 
 $\omega_7 = (x_1 \vee x_7 \vee \neg x_{12})$ 
 $\omega_8 = (x_1 \vee x_8)$ 
 $\omega_9 = (\neg x_7 \vee \neg x_8 \vee \neg x_{13})$ 
 $\omega_{10} : (\neg x_1 \vee x_9 \vee x_{11} \vee x_{10})$ 

```



No decision here

Another conflict clause: $\omega_{11} : (\neg x_{13} \vee \neg x_{12} \vee x_{11} \vee x_{10} \vee x_9)$

where should we backtrack to now ?
Decision Procedures

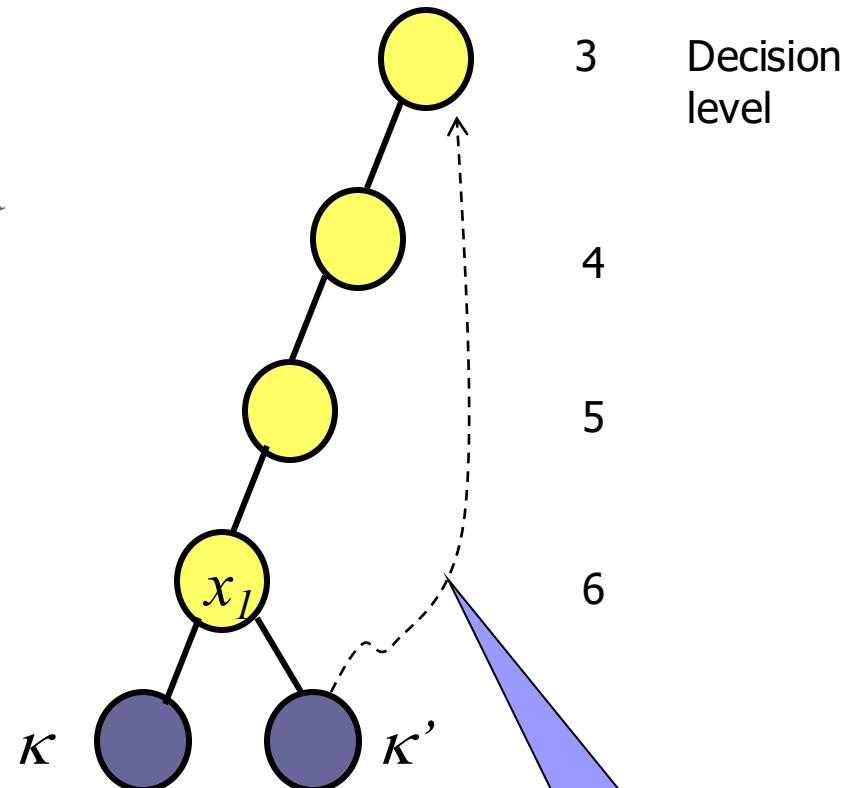
An algorithmic point of view

Non-chronological backtracking

Which assignments caused the conflicts ?

$x_9 = 0@1$
 $x_{10} = 0@3$
 $x_{11} = 0@3$
 $x_{12} = 1@2$
 $x_{13} = 1@2$

*These assignments
Are sufficient for
Causing a conflict.*



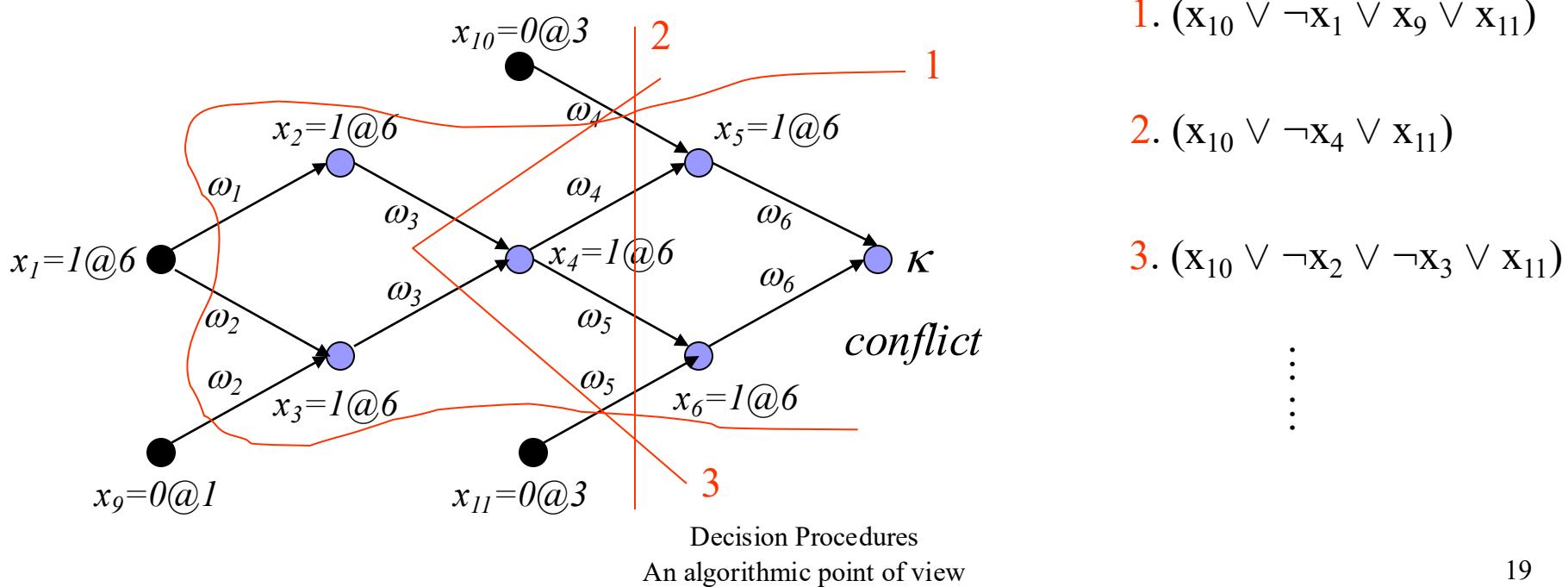
Backtrack to DL = 3

Non-chronological backtracking

- So the rule is: backtrack to the largest decision level in the conflict clause.
- This works for both the initial conflict and the conflict after the flip.
- Q: What if the flipped assignment works?
A: Change the decision retroactively.

More Conflict Clauses

- Def: A Conflict Clause is any clause implied by the formula
- Let L be a set of literals labeling nodes that form a cut in the implication graph, separating the conflict node from the roots.
- Claim: $\bigvee_{l \in L} \neg l$ is a Conflict Clause.



Conflict clauses

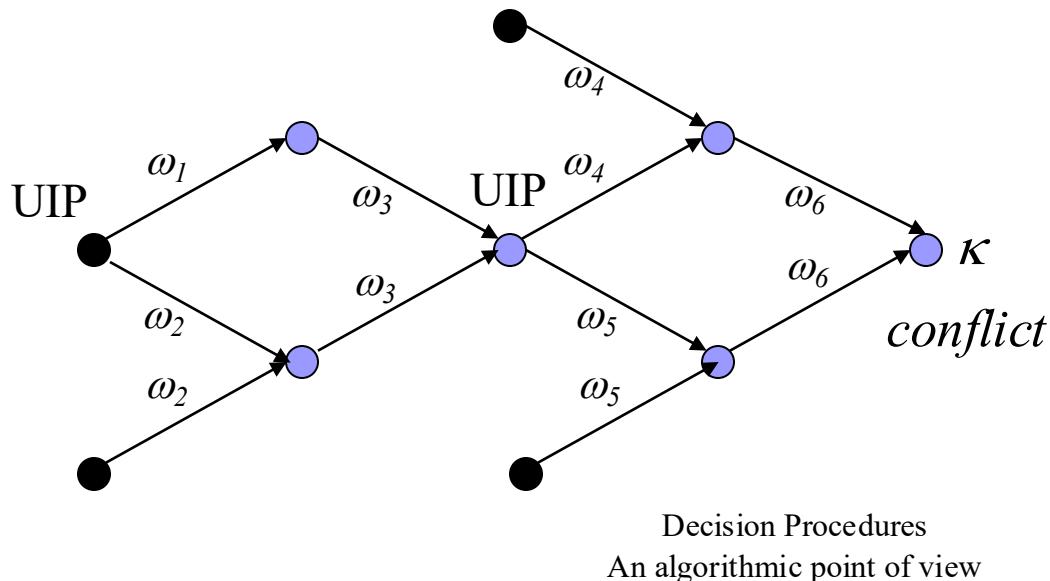
- How many clauses should we add ?
- If not all, then which ones ?
 - Shorter ones ?
 - Check their influence on the backtracking level ?
 - The most “influential” ?

Conflict clauses

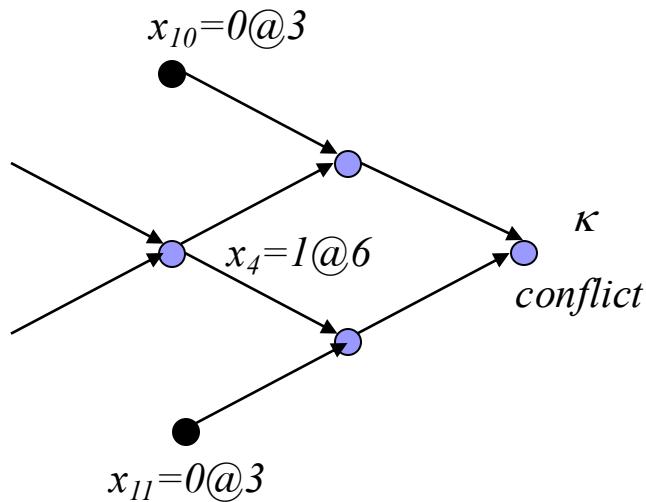
- Def: An **Asserting Clause** is a Conflict Clause with a single literal from the current decision level.
Backtracking (to the right level) makes it a Unit clause.
- Asserting clauses are those that force an immediate change in the search path.
- Modern solvers only consider Asserting Clauses.

Unique Implication Points (UIP's)

- Definition: A Unique Implication Point (UIP) is an internal node in the Implication Graph that all paths from the decision to the conflict node go through it.
- The First-UIP is the closest UIP to the conflict.

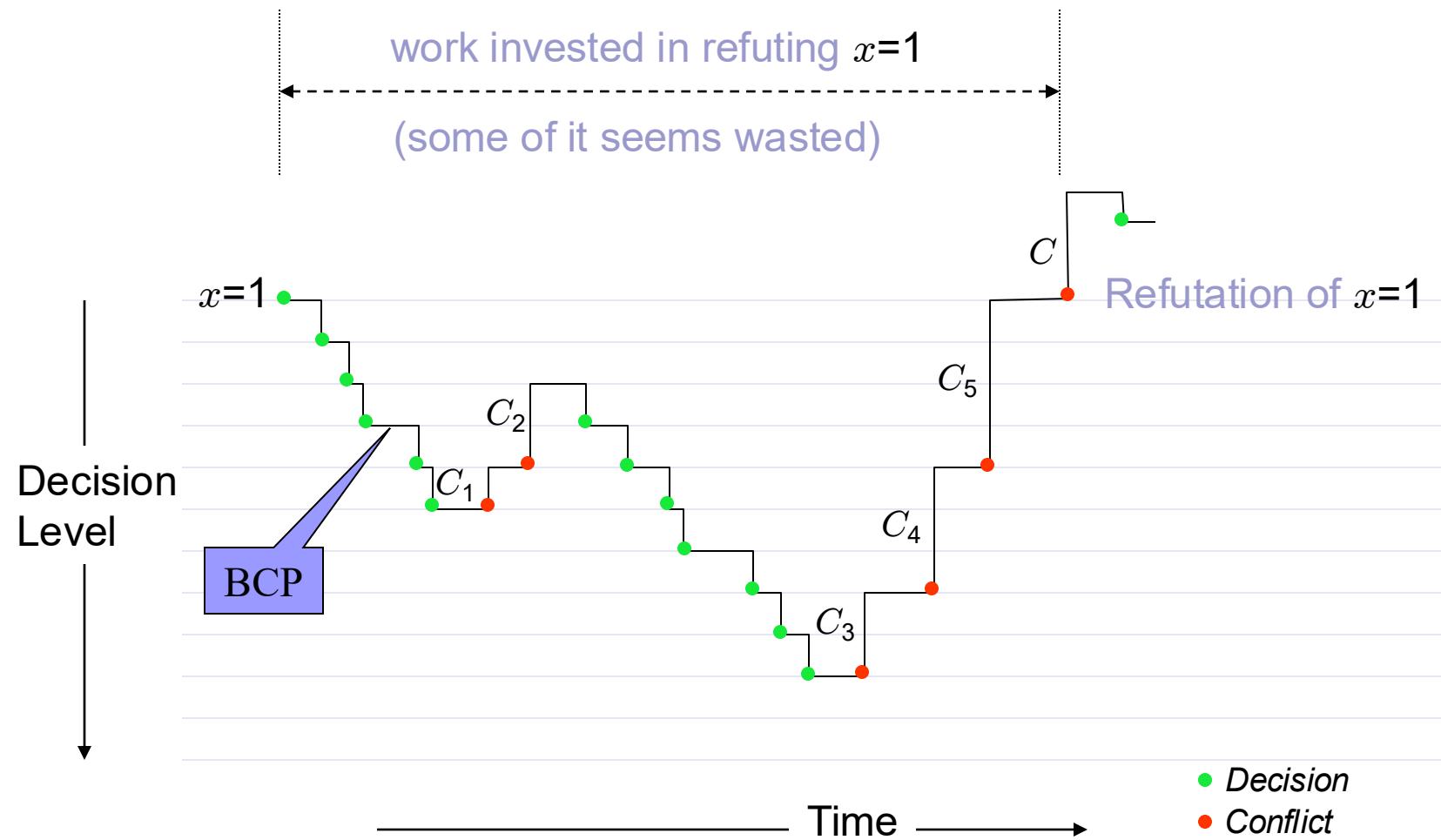


Conflict-driven backtracking (option #2)



- Conflict clause: $(x_{10} \vee \neg x_4 \vee x_{11})$
- With standard Non-Chronological Backtracking we backtracked to DL = 6.
- Conflict-driven Backtrack: backtrack to the second highest decision level in the clause (without erasing it).
- In this case, to DL = 3.
- Q: why?

Progress of a SAT solver



Conflict clauses and Resolution

- The Binary-resolution is a sound inference rule:

$$\frac{(a_1 \vee \dots \vee a_n \vee \beta) \quad (b_1 \vee \dots \vee b_m \vee (\neg \beta))}{(a_1 \vee \dots \vee a_n \vee b_1 \vee \dots \vee b_m)} \text{ (Binary Resolution)}$$

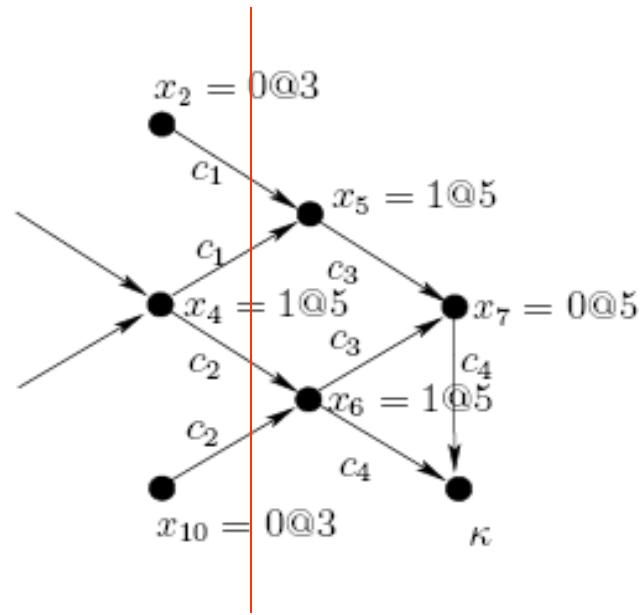
- Example:

$$\frac{(x_1 \vee x_2) \quad (\neg x_1 \vee x_3 \vee x_4)}{(x_2 \vee x_3 \vee x_4)}$$

Conflict clauses and resolution

- Consider the following example:

$$\begin{aligned}c_1 &= (\neg x_4 \vee x_2 \vee x_5) \\c_2 &= (\neg x_4 \vee x_{10} \vee x_6) \\c_3 &= (\neg x_5 \vee \neg x_6 \vee \neg x_7) \\c_4 &= (\neg x_6 \vee x_7) \\&\vdots \quad \vdots\end{aligned}$$

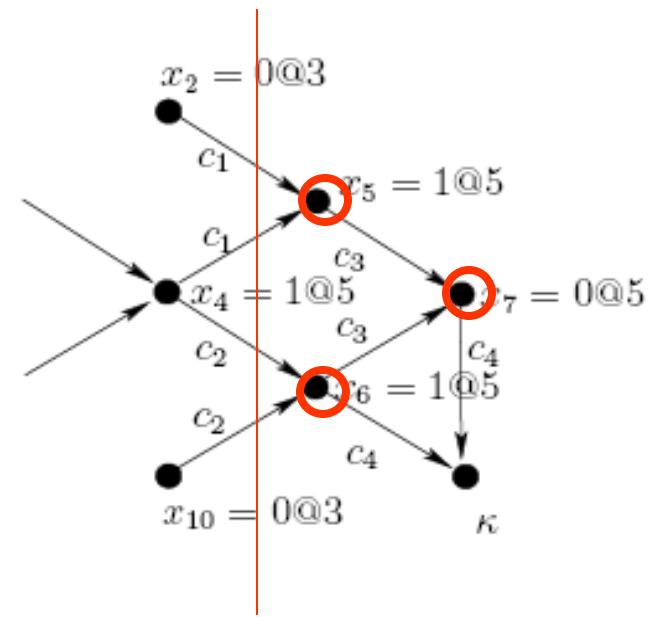


- Conflict clause: $c_5: (x_2 \vee \neg x_4 \vee x_{10})$

Conflict clauses and resolution

■ Conflict clause: $c_5: (x_2 \vee \neg x_4 \vee x_{10})$

$$\begin{array}{ll} c_1 & = (\neg x_4 \vee x_2 \vee x_5) \\ c_2 & = (\neg x_4 \vee x_{10} \vee x_6) \\ c_3 & = (\neg x_5 \vee \neg x_6 \vee \neg x_7) \\ c_4 & = (\neg x_6 \vee x_7) \\ \vdots & \vdots \end{array}$$



■ Resolution order: x_4, x_5, x_6, x_7

- $T1 = \text{Res}(c_4, c_3, x_7) = (\neg x_5 \vee \neg x_6)$
- $T2 = \text{Res}(T1, c_2, x_6) = (\neg x_4 \vee \neg x_5 \vee x_{10})$
- $T3 = \text{Res}(T2, c_1, x_5) = (x_2 \vee \neg x_4 \vee x_{10})$

Finding the conflict clause:

```
1: procedure Analyze-Conflict
2:   if current-decision-level = 0 then return -1;
3:   cl := current-conflicting-clause;
4:   while ( $\neg$ Stop-criterion-met(cl)) do
5:     lit := Last-assigned-literal(cl);
6:     var := Variable-of-literal(lit);
7:     ante := Antecedent(var);
8:     cl := Resolve(cl, ante, var);
9:     add-clause-to-database(cl);
```

cl is asserting
the first UIP

Applied to our example:

	<i>name</i>	<i>cl</i>	<i>lit</i>	<i>var</i>	<i>ante</i>
	<i>c</i> ₄	$(\neg x_6 \vee x_7)$	x_7	x_7	c_3
		$(\neg x_5 \vee \neg x_6)$	$\neg x_6$	x_6	c_2
		$(\neg x_4 \vee x_{10} \vee \neg x_5)$	$\neg x_5$	x_5	c_1
	<i>c</i> ₅	$(\neg x_4 \vee x_2 \vee x_{10})$			

Decision heuristics – VSIDS (cont'd)

- Chaff holds a list of unassigned variables sorted by the counter value.
- Updates are needed only when adding conflict clauses.
- Thus - decision is made in constant time.

Decision heuristics

VSIDS (cont'd)

VSIDS is a ‘quasi-static’ strategy:

- *static* because it doesn’t depend on current assignment
- *dynamic* because it gradually changes. Variables that appear in recent conflicts have higher priority.

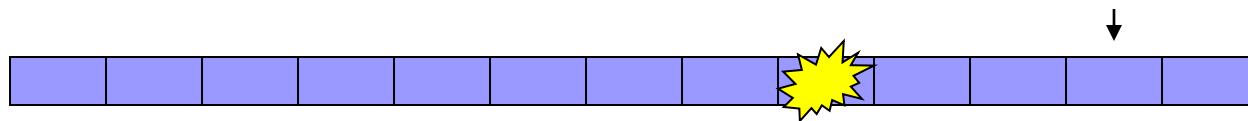
This strategy is a *conflict-driven* decision strategy.

“..employing this strategy dramatically (i.e. an order of magnitude) improved performance ... “

Decision Heuristics - Berkmin

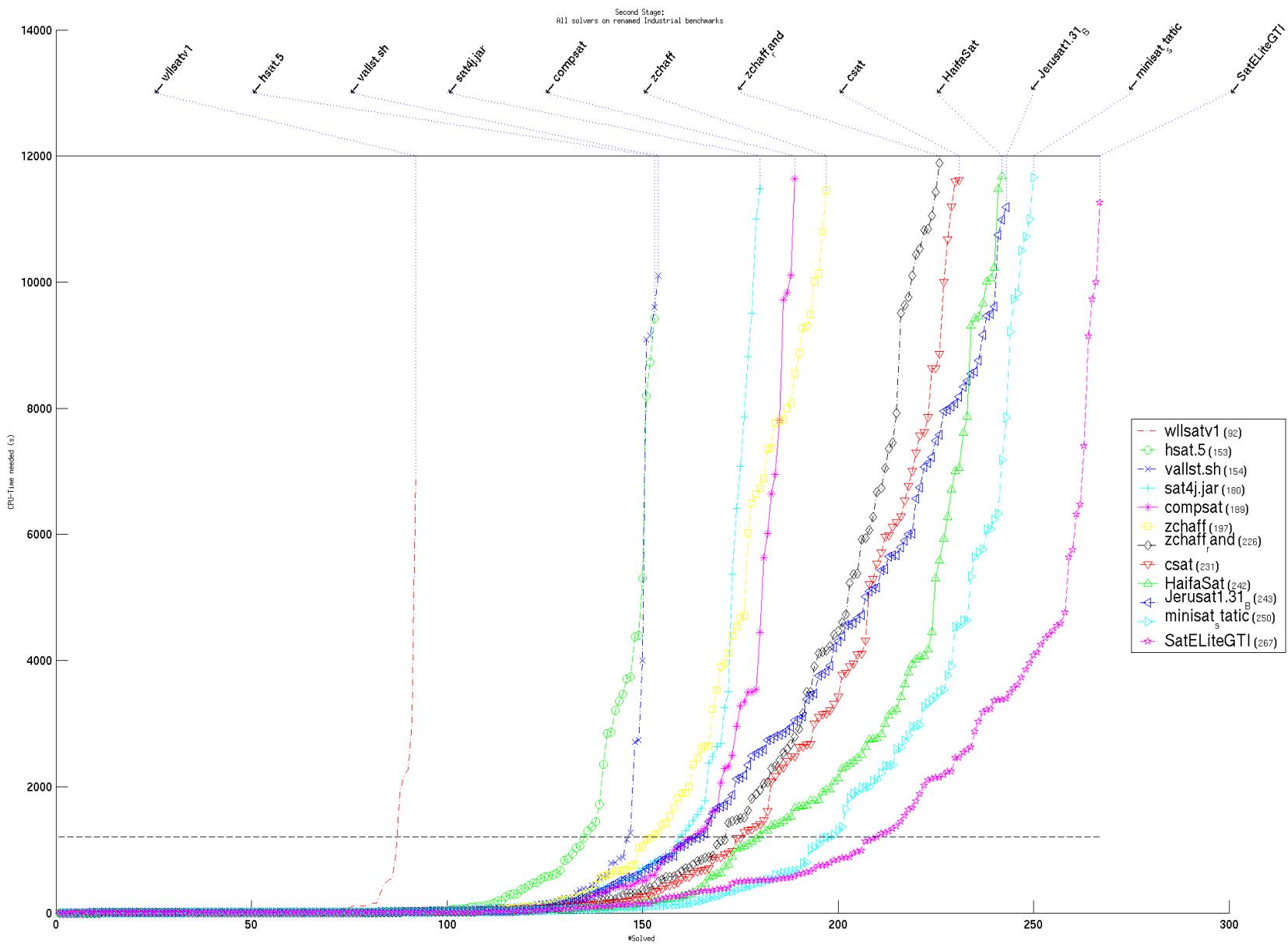
- Keep conflict clauses in a stack
- Choose the first unresolved clause in the stack
 - If there is no such clause, use VSIDS
- Choose from this clause a variable + value according to some scoring (e.g. VSIDS)
- This gives absolute priority to conflicts.

Berkmin heuristic



tail-
first conflict clause

The SAT competitions



- End of SAT (for now)
- Beginning of Binary Decision Diagrams